

---

**UNITED STATES COURT OF APPEALS  
FOR THE FEDERAL CIRCUIT**

---

VIRNETX INC., LEIDOS, INC., fka Science Applications International Corporation,  
*Plaintiffs-Appellees,*

v.  
APPLE INC.,  
*Defendant-Appellant.*

---

VIRNETX INC.,  
*Plaintiff-Appellee,*

v.  
APPLE INC.,  
*Defendant-Appellant.*

---

On Appeal from the United States District Court for the Eastern District of Texas,  
Nos. 6:12-cv-00855-RWS and 6:11-cv-00563-RWS, Judge Robert Schroeder, III

---

**NON-CONFIDENTIAL BRIEF FOR  
DEFENDANT-APPELLANT APPLE INC.**

---

BRITTANY BLUEITT AMADI  
WILMER CUTLER PICKERING  
HALE AND DORR LLP  
1875 Pennsylvania Avenue NW  
Washington, DC 20006  
(202) 663-6000

THOMAS G. SPRANKLING  
WILMER CUTLER PICKERING  
HALE AND DORR LLP  
950 Page Mill Road  
Palo Alto, CA 94304  
(650) 858-6000

January 24, 2019

WILLIAM F. LEE  
MARK C. FLEMING  
LAUREN B. FLETCHER  
WILMER CUTLER PICKERING  
HALE AND DORR LLP  
60 State Street  
Boston, MA 02109  
(617) 526-5000

*Attorneys for Defendant-Appellant  
Apple Inc.*

---

## CERTIFICATE OF INTEREST

Counsel for Defendant-Appellant Apple Inc. certifies the following:

1. The full name of every party or *amicus* represented by me is:

Apple Inc.

2. The name of the real party in interest represented by me is:

Not applicable.

3. All parent corporations and any publicly held companies that own 10 percent or more of the stock of the party or *amicus curiae* represented by me are:

None.

4. The names of all law firms and the partners or associates that appeared for the party or *amicus* now represented by me in the trial court or agency or are expected to appear in this court (and who have not or will not enter an appearance in this case) are:

WILMER CUTLER PICKERING HALE AND DORR LLP: Dominic E. Massa, Elizabeth Reilly, Rebecca Bact

KIRKLAND & ELLIS, LLP: Robert A. Appleby, Gregory S. Arovas, Akshay S. Deoras, David N. Draper, Jeanne M. Heffernan, Mangesh A. Kulkarni, Joseph A. Loy, Thomas V. Matthew, F. Christopher Mizzo, Steve Papazian, John C. O'Quinn, Aaron D. Resetarits, Leslie M. Schmidt, Diwei Zhang

NELSON BUMGARDNER ALBRITTON PC: Eric M. Albritton, Shawn A. Latchford

DESMARAIS LLP: John M. Desmarais, Edward B. Geist, Ameet A. Modi, Jennifer Przybylski, Michael P. Stadnick, Wesley White

JACKSON WALKER LLP: Brian K. Buss, Christopher N. Cravey, Danny L. Williams

POTTER MINTON, A PROFESSIONAL CORPORATION: John F. Bufe, Michael E. Jones

WILLIAMS MORGAN, PC: Ruben S. Bains, Terry D. Morgan, Matthew R. Rodgers

ALLEN GARDNER LAW, PLLC: Allen Franklin Gardner

AHMAD ZAVITSANOS ANAIPAKOS ALAVI & MENSING PC: Kyung Tai Kim

5. The title and number of any case known to counsel to be pending in this or any other court or agency that will directly affect or be directly affected by this court's decision in the pending appeal:

*VirnetX Inc. v. The Mangrove Partners Master Fund, Ltd.*, Nos. 17-1368, -1383 (Fed. Cir.)

*VirnetX Inc. v. Apple Inc.*, Nos. 17-1591, -1592, -1593 (Fed. Cir.)

*VirnetX Inc. v. Black Swamp, IP, LLC*, Nos. 17-2593, -2594 (Fed. Cir.)

*VirnetX Inc. v. Cisco Systems, Inc.*, No. 18-1197 (Fed. Cir.)

*VirnetX Inc. v. Cisco Systems, Inc.*, No. 18-1751 (Fed. Cir.)

Inter Partes Reexamination Control No. 95/001,679 (USPTO)

Inter Partes Reexamination Control No. 95/001,682 (USPTO)

Inter Partes Reexamination Control No. 95/001,714 (USPTO)

Inter Partes Reexamination Control No. 95/001,697 (USPTO)

Dated: January 24, 2019

/s/ William F. Lee  
WILLIAM F. LEE  
WILMER CUTLER PICKERING  
HALE AND DORR LLP  
60 State Street  
Boston, MA 02109  
(617) 526-6000

## TABLE OF CONTENTS

	Page
CERTIFICATE OF INTEREST .....	i
TABLE OF AUTHORITIES .....	vii
STATEMENT OF RELATED CASES .....	1
JURISDICTIONAL STATEMENT .....	4
INTRODUCTION .....	4
STATEMENT OF ISSUES .....	6
STATEMENT OF CASE .....	7
A.    Background Regarding VPN On Demand .....	7
1.    VirnetX's '135 and '151 patents.....	7
2.    VPN On Demand .....	9
a.    Original VPN On Demand .....	10
b.    Redesigned VPN On Demand .....	12
B.    Background Regarding FaceTime.....	13
1.    VirnetX's '504 and '211 patents.....	13
2.    FaceTime.....	15
a.    Original FaceTime .....	15
b.    Redesigned FaceTime.....	17
C.    Litigation History .....	18
1.    First trial and appeal in 417 Action .....	18
2.    Consolidated trial in 417 and 855 Actions.....	19



3.	Retrial and second appeal in 417 Action .....	20
4.	Retrial in 855 Action.....	21
5.	Post-trial rulings in 855 Action.....	24
	SUMMARY OF ARGUMENT .....	27
	STANDARD OF REVIEW .....	31
	ARGUMENT .....	32
I.	THE INFRINGEMENT JUDGMENT FOR THE '135 AND '151 PATENTS SHOULD BE REVERSED. ....	32
A.	Redesigned VPN On Demand Does Not “Automatically Initiate” A VPN In Response To “Determining” That A DNS Request Is For A Secure Server. ....	32
1.	The “If Needed” mode initiates a VPN based on the requesting device’s location, not—as the claims require—whether the DNS request is for a secure server. ....	33
2.	The district court’s reasons for denying JMOL are inconsistent with <i>VirnetX I</i> and unsupported by substantial evidence. ....	36
B.	VirnetX Failed To Prove Direct Or Induced Infringement.....	40
1.	VirnetX did not prove direct or induced infringement of the '135 patent. ....	40
2.	VirnetX did not prove direct or induced infringement of the '151 patent. ....	43
3.	The damages should be reduced to reflect the limited scope of infringement (if any) based on use of the <i>optional</i> probe. ....	46

II.	THE INFRINGEMENT JUDGMENT FOR THE '504 AND '211 PATENTS SHOULD BE REVERSED. ....	46
A.	The District Court Erroneously Instructed The Jury That The Claimed “DNS System” Does Not Incorporate The “DNS” Construction. ....	46
1.	The district court’s claim construction instruction was erroneous. ....	47
2.	Under the correct claim construction, the infringement judgment for redesigned FaceTime cannot stand. ....	49
3.	At a minimum, the district court’s instruction was prejudicial error that warrants a new trial. ....	50
B.	Redesigned FaceTime Does Not Provide The Claimed “Indication.” ....	51
1.	The Accept Push message does not “indicate” support for a “direct” communication link. ....	52
2.	The district court’s reasons for denying JMOL are legally incorrect and unsupported by substantial evidence. ....	54
III.	THE DISTRICT COURT ERRED IN APPLYING ISSUE PRECLUSION TO APPLE’S INVALIDITY DEFENSES AND COUNTERCLAIMS. ....	55
IV.	THE DISTRICT COURT ERRED IN DENYING JUDGMENT OF NON-INFRINGEMENT AS TO iMESSAGE. ....	58
V.	THE JUDGMENT SHOULD BE VACATED IF THIS COURT CONCLUDES THAT EITHER SET OF PATENTS-IN-SUIT IS UNPATENTABLE AND/OR NOT INFRINGED. ....	61
VI.	APPLE PRESERVES ITS ARGUMENTS ON THE ISSUES DECIDED AGAINST IT IN THE 417 ACTION. ....	62
A.	Redesigned FaceTime Does Not Provide “Anonymity.” ....	62

B.	The Damages And Interest Awards Should Be Reversed Or Vacated. ....	63
1.	Mr. Weinstein’s testimony should have been excluded under <i>Daubert</i> . ....	63
2.	Even if admissible, Mr. Weinstein’s testimony did not provide substantial evidence supporting the verdict.....	66
3.	The district court abused its discretion in awarding \$93 million in prejudgment interest.....	67
CONCLUSION .....		68
ADDENDUM		
CERTIFICATE OF SERVICE		
CERTIFICATE OF COMPLIANCE		

**CONFIDENTIAL MATERIAL OMITTED**

The material omitted from page 24 contains confidential information regarding the amount of royalties received by VirnetX and the number of units sold pursuant to certain third-party license agreements.

## TABLE OF AUTHORITIES

### CASES

	Page(s)
<i>ACCO Brands, Inc. v. ABA Locks Manufacturer Co.</i> , 501 F.3d 1307 (Fed. Cir. 2007) .....	31, 42
<i>Akamai Technologies, Inc. v. Limelight Networks, Inc.</i> , No. 06-cv-11109 (D. Mass. June 24, 2016).....	68
<i>Alcon Research Ltd. v. Barr Laboratories, Inc.</i> , 745 F.3d 1180 (Fed. Cir. 2014) .....	60
<i>Ball Aerosol &amp; Specialty Container, Inc. v. Limited Brands, Inc.</i> , 555 F.3d 984 (Fed. Cir. 2009) .....	44, 57
<i>Blonder-Tongue Laboratories, Inc. v. University of Illinois Foundation</i> , 402 U.S. 313 (1971).....	57
<i>Branch-Hines v. Herbert</i> , 939 F.2d 1311 (5th Cir. 1991) .....	61
<i>Commonwealth Scientific &amp; Industrial Research Organisation v.</i> <i>Cisco Systems, Inc.</i> , 809 F.3d 1295 (Fed. Cir. 2015) .....	64
<i>Connell v. Sears, Roebuck &amp; Co.</i> , 722 F.2d 1542 (Fed. Cir. 1983) .....	60
<i>Copeland v. Merrill Lynch &amp; Co.</i> , 47 F.3d 1415 (5th Cir. 1995) .....	56
<i>Datascope Corp. v. SMEC, Inc.</i> , 776 F.2d 320 (Fed. Cir. 1985) .....	60
<i>Daubert v. Merrell Dow Pharmaceuticals, Inc.</i> , 509 U.S. 579 (1993).....	66
<i>E-Pass Technologies, Inc. v. 3Com Corp.</i> , 473 F.3d 1213 (Fed. Cir. 2007) .....	41, 43

<i>Ericsson, Inc. v. D-Link Systems, Inc.</i> , 773 F.3d 1201 (Fed. Cir. 2014) .....	45, 64, 66
<i>Exmark Manufacturing Co. v. Briggs &amp; Stratton Power Products Group, LLC</i> , 879 F.3d 1332 (Fed. Cir. 2018) .....	66
<i>Fresenius USA, Inc. v. Baxter International, Inc.</i> , 721 F.3d 1330 (Fed. Cir. 2013) .....	61
<i>Fujitsu Ltd. v. Netgear Inc.</i> , 620 F.3d 1321 (Fed. Cir. 2010) .....	44, 45
<i>Garretson v. Clark</i> , 111 U.S. 120 (1884).....	63
<i>General Motors Corp. v. Devex Corp.</i> , 461 U.S. 648 (1983).....	67
<i>GLF Construction Corp. v. LAN/STV</i> , 414 F.3d 553 (5th Cir. 2005) .....	56
<i>Global-Tech Appliances, Inc. v. SEB S.A.</i> , 563 U.S. 754 (2011).....	43
<i>Group One, Ltd. v. Hallmark Cards, Inc.</i> , 407 F.3d 1297 (Fed. Cir. 2005) .....	32
<i>Hallco Manufacturing Co. v. Foster</i> , 256 F.3d 1290 (Fed. Cir. 2001) .....	55-56
<i>Hewlett-Packard Co. v. Acceleron LLC</i> , 587 F.3d 1358 (Fed. Cir. 2009) .....	32, 59
<i>Highmark Inc. v. Allcare Health Management System, Inc.</i> , 572 U.S. 559 (2014).....	31
<i>Intel Corp. v. International Trade Commission</i> , 946 F.2d 821 (Fed. Cir. 1991) .....	45
<i>Intellectual Ventures I LLC v. Motorola Mobility LLC</i> , 870 F.3d 1320 (Fed. Cir. 2017) .....	31

<i>Knight v. Kirby Inland Marine Inc.</i> , 482 F.3d 347 (5th Cir. 2007) .....	31
<i>Lear, Inc. v. Adkins</i> , 395 U.S. 653 (1969).....	57
<i>Lenoir v. C.O. Porter Machinery Co.</i> , 672 F.2d 1240 (5th Cir. 1982) .....	51
<i>Lucent Technologies, Inc. v. Gateway, Inc.</i> , 580 F.3d 1301 (Fed. Cir. 2009) .....	31, 40, 46, 67
<i>MasterMine Software, Inc. v. Microsoft Corp.</i> , 874 F.3d 1307 (Fed. Cir. 2017) .....	45
<i>Medtronic, Inc. v. Mirowski Family Ventures, LLC</i> , 571 U.S. 191 (2014).....	57, 59
<i>Microprocessor Enhancement Corp. v. Texas Instruments, Inc.</i> , 520 F.3d 1367 (Fed. Cir. 2008) .....	45
<i>Mirror Worlds, LLC v. Apple Inc.</i> , 692 F.3d 1351 (Fed. Cir. 2012) .....	42
<i>Moleculon Research Corp. v. CBS, Inc.</i> , 793 F.2d 1261 (Fed. Cir. 1986) .....	43
<i>Northern Telecom, Inc. v. Datapoint Corp.</i> , 908 F.2d 931 (Fed. Cir. 1990) .....	55
<i>Novo Nordisk Pharmaceuticals, Inc. v. Bio-Technology General Corp.</i> , 424 F.3d 1347 (Fed. Cir. 2005) .....	60
<i>Oiness v. Walgreen Co.</i> , 88 F.3d 1025 (Fed. Cir. 1996) .....	68
<i>Phonometrics, Inc. v. Northern Telecom Inc.</i> , 133 F.3d 1459 (Fed. Cir. 1998) .....	47
<i>Poly-America, L.P. v. API Industries, Inc.</i> , 839 F.3d 1131 (Fed. Cir. 2016) .....	31

<i>Revolution Eyewear, Inc. v. Aspex Eyewear, Inc.</i> , 563 F.3d 1358 (Fed. Cir. 2009) .....	45
<i>Rude v. Westcott</i> , 130 U.S. 152 (1889).....	65
<i>Streck, Inc. v. Research &amp; Diagnostic Systems, Inc.</i> , 665 F.3d 1269 (Fed. Cir. 2012) .....	60
<i>TASER International, Inc. v. Karbon Arms, LLC</i> , 6 F. Supp. 3d 510 (D. Del. 2013).....	57
<i>Translogic Technology, Inc. v. Hitachi, Ltd.</i> , 250 F. App'x 988 (Fed. Cir. 2007) .....	61
<i>Typhoon Touch Technologies, Inc. v. Dell, Inc.</i> , 659 F.3d 1376 (Fed. Cir. 2011) .....	44, 45
<i>UltimatePointer, L.L.C. v. Nintendo Co.</i> , 816 F.3d 816 (Fed. Cir. 2016) .....	45
<i>Uniloc USA, Inc. v. Microsoft Corp.</i> , 632 F.3d 1292 (Fed. Cir. 2011) .....	65
<i>United States v. CITGO Petroleum Corp.</i> , 801 F.3d 477 (5th Cir. 2015) .....	32
<i>United States v. Lewis</i> , 796 F.3d 543 (5th Cir. 2015) .....	31
<i>Verizon Services Corp. v. Cox Fibernet Virginia, Inc.</i> , 602 F.3d 1325 (Fed. Cir. 2010) .....	31
<i>VirnetX Inc. v. Apple Inc.</i> , 665 F. App'x 880 (Fed. Cir. 2016) .....	3, 7
<i>VirnetX Inc. v. Apple Inc.</i> , 671 F. App'x 786 (Fed. Cir. 2016) .....	3
<i>VirnetX Inc. v. Apple Inc.</i> , 671 F. App'x 789 (Fed. Cir. 2016) .....	3

<i>VirnetX Inc. v. Apple Inc.</i> , 715 F. App'x 1024 (Fed. Cir. 2018) (per curiam) .....	3
<i>VirnetX Inc. v. Apple Inc.</i> , 909 F.3d 1375 (Fed. Cir. 2018) .....	3
<i>VirnetX Inc. v. Cisco Systems, Inc.</i> , ___ F. App'x ___, No. 18-1197, 2019 WL 190518 (Fed. Cir. Jan. 15, 2019) (per curiam) .....	1, 20
<i>VirnetX, Inc. v. Cisco Systems, Inc.</i> , 767 F.3d 1308 (Fed. Cir. 2014) .....	<i>passim</i>
<i>Voter Verified, Inc. v. Election Systems &amp; Software LLC</i> , 887 F.3d 1376 (Fed. Cir. 2018) .....	29, 56, 57
<i>Wills v. Arizon Structures Worldwide, L.L.C.</i> , 824 F.3d 541 (5th Cir. 2016) .....	32

## DOCKETED CASES

<i>VirnetX, Inc. v. Cisco Systems, Inc.</i> , No. 18-1197 (Fed. Cir.) .....	62, 63
---	--------

## STATUTES

28 U.S.C.	
§ 1295(a)(1) .....	4
§ 1338(a) .....	4
35 U.S.C.	
§ 102.....	29, 56, 57
§ 103.....	29, 56, 58
§ 271(a) .....	40
§ 284.....	67



## STATEMENT OF RELATED CASES

The patents at issue in this appeal, U.S. Patent Nos. 6,502,135 (“the ’135 patent”), 7,490,151 (“the ’151 patent”), 7,418,504 (“the ’504 patent”), and 7,921,211 (“the ’211 patent”), were also the subject of a related proceeding in the United States District Court for the Eastern District of Texas. A first trial in *VirnetX Inc. v. Cisco Systems, Inc.*, No. 6:10-cv-00417 (“the 417 Action”), resulted in a jury verdict and judgment that Apple appealed to this Court (No. 13-1489). This Court affirmed-in-part, reversed-in-part, vacated-in-part, and remanded for further proceedings involving the four patents. *VirnetX, Inc. v. Cisco Sys., Inc.*, 767 F.3d 1308 (Fed. Cir. 2014) (Prost, C.J., joined by Chen, J.) (“*VirnetX I*”). On remand, a subsequent trial resulted in a jury verdict and judgment that Apple again appealed to this Court (No. 18-1197). A panel of this Court affirmed without opinion. *VirnetX Inc. v. Cisco Sys., Inc.*, \_\_ F. App’x \_\_, 2019 WL 190518 (Fed. Cir. Jan. 15, 2019) (Prost, C.J., Moore & Reyna, JJ.) (per curiam).

Each asserted claim of the four patents at issue in this appeal has been declared unpatentable in at least one Patent and Trademark Office (“PTO”) proceeding, beginning with proceedings filed by Apple and Cisco Systems, Inc. in 2011:

Patent	Date filed	Type of Proceeding	PTO Docket No.	Date of PTO Final Decision
'504	Oct. 18, 2011	Reexamination	95/001,788	Sept. 12, 2016
'211	Oct. 18, 2011	Reexamination	95/001,789	Sept. 12, 2016
'211	Dec. 12, 2011	Reexamination	95/001,856	Sept. 12, 2016
'504	Dec. 13, 2011	Reexamination	95/001,851	Sept. 18, 2017
'135	April 14, 2015	IPR	IPR2015-01046	Sept. 9, 2016
'151	April 14, 2015	IPR	IPR2015-01047	Sept. 9, 2016
'504	Feb. 29, 2016	IPR	IPR2016-00693	July 24, 2017
'211	April 27, 2016	IPR	IPR2016-00957	July 24, 2017

As of the filing of this brief, appeals of all these proceedings were pending before this Court:

Patent	PTO Docket No.	Federal Circuit Docket No.	Date of Federal Circuit Docketing
'135	IPR2015-01046	17-1368	December 16, 2016
'151	IPR2015-01047	17-1383	December 20, 2016
'504	95/001,788	17-1591	February 7, 2017
'211	95/001,789	17-1592	February 7, 2017
'211	95/001,856	17-1593	February 7, 2017
'504	IPR2016-00693	17-2593	September 25, 2017
'211	IPR2016-00957	17-2594	September 25, 2017
'504	95/001,851	18-1751	March 30, 2018

A panel of this Court (Prost, C.J., Moore & Reyna, JJ.) heard oral argument in *VirnetX Inc. v. The Mangrove Partners Master Fund, Ltd., et al.*, Nos. 17-1368, -1383, and *VirnetX Inc. v. Apple Inc., et al.*, Nos. 17-1591, -1592, and -1593 on January 8, 2019. *VirnetX Inc. v. Cisco Systems, Inc.*, No. 18-1751, is fully briefed and awaiting an argument date. Meanwhile, the Court has stayed briefing in *VirnetX Inc. v. Black Swamp IP, LLC*, Nos. 17-2593 and -2594, pending resolution of Appeal Nos. 17-1591, -1592, -1593.

Additional *inter partes* reexaminations filed by Apple and Cisco pertaining to the patents at issue in this appeal are still pending before the PTO. *See* Reexamination Nos. 95/001,679 ('135 patent), 95/001,682 ('135 patent), 95/001,697 ('151 patent), and 95/001,714 ('151 patent).

This Court has also decided numerous appeals regarding closely related VirnetX patents that have nearly identical specifications as the patents-in-suit, in each case affirming the Patent Trial and Appeal Board's ("the Board's") findings that all challenged claims are unpatentable. *See VirnetX Inc. v. Apple Inc.*, Nos. 17-2490, -2494, 909 F.3d 1375 (Fed. Cir. 2018) (O'Malley, J., joined by Newman & Chen, JJ.) (U.S. Patent No. 8,504,696); *VirnetX Inc. v. Apple Inc.*, Nos. 17-1131, -1132, -1186, -1274, -1275, -1276, -1291, 715 F. App'x 1024 (Fed. Cir. 2018) (per curiam) (Newman, Mayer, & Lourie, JJ.) (U.S. Patent Nos. 8,868,705, 8,850,009, 8,458,341, 8,516,131, and 8,560,705); *VirnetX Inc. v. Apple Inc.*, Nos. 16-1211, -1213, -1279, -1281, 671 F. App'x 786 (Fed. Cir. 2016) (O'Malley, J., joined by Mayer & Wallach, JJ.) (U.S. Patent Nos. 7,188,180 and 7,987,274); *VirnetX Inc. v. Apple Inc.*, No. 16-1480, 671 F. App'x 789 (Fed. Cir. 2016) (O'Malley, J., joined by Mayer & Wallach, JJ.) (U.S. Patent No. 8,051,181); *VirnetX Inc. v. Apple Inc.*, Nos. 15-1934, -1935, 665 F. App'x 880 (Fed. Cir. 2016) (Wallach, J., joined by Mayer, J.; dissent by O'Malley, J.) (U.S. Patent No. 8,504,697).

## **JURISDICTIONAL STATEMENT**

The district court had jurisdiction under 28 U.S.C. § 1338(a) and entered final judgment. Appx119. Apple timely appealed. Appx16324-16325. This Court has jurisdiction under 28 U.S.C. § 1295(a)(1).

## **INTRODUCTION**

After VirnetX accused prior versions of two Apple features of infringement, Apple redesigned them to avoid VirnetX's patents. VirnetX then pivoted to claim construction and infringement theories that are unsupported by, and indeed inconsistent with, the patents and the evidence. The district court nonetheless allowed VirnetX to parlay its improper theories into a judgment of nearly \$600 million, while preventing Apple from fairly presenting its defenses to the jury.

For instance, Apple redesigned VPN On Demand so that it no longer “automatically initiates” a VPN in response to a determination that the requested server is secure. VirnetX responded by shifting its sights to a narrow, specific implementation using an optional probe—a configuration that it did not show has ever been used in the United States. And even in that implementation, whether a VPN is created hinges on the requesting device's location—which VirnetX previously recognized does not satisfy the claims. Despite this, the district court allowed VirnetX to demand—and the jury to award—damages on every Apple

device running redesigned VPN On Demand, even though VirnetX conceded that most implementations did not infringe.

Apple also redesigned FaceTime such that it no longer returns an IP address, which was required by the construction of “domain name service” (“DNS”) that Judge Davis entered years ago. VirnetX dodged again, asking a new judge (Judge Schroeder) to rule that the claimed “DNS system” need not include a “DNS,” contrary to the claims’ and patents’ obvious meaning. The district court not only obliged, but also instructed the jury about this point unnecessarily and prejudicially. VirnetX further argued that the required “indication” of support for a “direct” communication between devices could be met by a message that remains the same whether the resulting communication is direct or indirect—an expansion of the claims that the district court again erroneously accepted.

There were other problems with this trial as well. The district court erroneously invoked issue preclusion to forbid Apple from presenting invalidity issues never previously adjudicated. And it refused to enter judgment of non-infringement regarding an accused feature (iMessage) on which VirnetX offered no evidence.

Finally, in separate proceedings brought by Apple, Cisco, and others beginning in 2011, the PTO has held several times over that every asserted claim of every patent-in-suit is unpatentable. If this Court affirms those unpatentability

determinations and/or reverses the infringement findings for some or all of the four patents-in-suit, the nearly \$600 million judgment should be vacated and remanded for a determination of damages on any remaining patents and infringement findings.

### **STATEMENT OF ISSUES**

1. Whether the infringement judgment for the '135 and '151 patents should be reversed or vacated because: (a) redesigned VPN On Demand does not “automatically initiate” a VPN based on a determination that the DNS request is for a secure server, as the claims require; and (b) VirnetX failed to prove any act of direct infringement through configuration and use of the *optional* HTTPS probe, or that Apple induced such action.

2. Whether the infringement judgment for the '504 and '211 patents should be reversed or vacated because: (a) the district court incorrectly instructed the jury that the claimed “DNS system” does not include a “DNS” as the court construed it, and the correct construction forecloses infringement; and (b) redesigned FaceTime’s servers do not provide an “indication” that the system supports establishing a direct communication link, as the claims require.

3. Whether the district court erred in concluding that issue preclusion barred Apple from raising its invalidity defenses and counterclaims that were not actually litigated in the prior case.

4. Whether the district court erred in refusing to enter judgment of non-infringement on Apple's counterclaim regarding iMessage, where VirnetX presented no evidence regarding iMessage at trial.

5. Whether the judgment should be vacated if this Court affirms the PTO's determinations that the asserted claims of some or all patents-in-suit are unpatentable.

## **STATEMENT OF CASE**

### **A. Background Regarding VPN On Demand**

#### **1. VirnetX's '135 and '151 patents**

Every computer on the Internet is typically identified by a unique Internet Protocol ("IP") address (*e.g.*, 123.45.678.9). *VirnetX Inc. v. Apple Inc.*, 665 F. App'x 880, 882 (Fed. Cir. 2016). IP addresses are often associated with domain names (*e.g.*, www.apple.com). *Id.* When one computer seeks to communicate with another, it sends a DNS request to a domain name server, requesting the IP address that corresponds to the domain name. *Id.*; Appx320-321(36:61-37:6). The domain name server looks up the IP address corresponding to the requested domain name and returns it to the requesting computer. Appx320-321(36:64-37:10).

The common specification of the '135 and '151 patents describes systems and methods for "automatic[ally] creati[ng]" a "virtual private network (VPN) in

response to a domain name server look-up function.” Appx320(36:57-59). When a user sends a DNS request as in conventional Internet communications, a DNS proxy “intercepts all DNS lookup functions” and “determines whether access to a secure site has been requested” by, for example, referencing “an internal table of such sites.” Appx321(37:60-66); Appx296(Fig. 26); *see VirnetX I*, 767 F.3d at 1315.

If “access to a secure site has been requested,” a VPN is automatically initiated between the requesting computer and the secure site. Appx321(37:60-38:2, 38:59-62); Appx322(39:10-19). However, if the DNS request is for “a non-secure web site,” the DNS proxy “merely pass[es]” the DNS request to a “conventional DNS server ... , which would be handled in a conventional manner, returning the IP address of [the] non-secure web site.” Appx321(38:12-16); *see* Appx297(Fig. 27); *VirnetX I*, 767 F.3d at 1315.

VirnetX asserted claims 1 and 7 of the ’135 patent and claim 13 of the ’151 patent. Appx2755. Claim 1 of the ’135 patent, from which claim 7 depends, reads:

1. A method of transparently creating a virtual private network (VPN) between a client computer and a target computer, comprising the steps of:

- (1) generating from the client computer a Domain Name Service (DNS) request that requests an IP address corresponding to a domain name associated with the target computer;



(2) determining whether the DNS request transmitted in step (1) is requesting access to a secure web site; and

*(3) in response to determining that the DNS request in step (2) is requesting access to a secure target web site, automatically initiating the VPN between the client computer and the target computer.*

Appx180(47:20-32).<sup>1</sup>

Claim 13 of the '151 patent similarly recites:

13. A computer readable medium storing a domain name server (DNS) module comprised of computer readable instructions that, when executed, cause a data processing device to perform the steps of:

(i) determining whether a DNS request sent by a client corresponds to a secure server;

(ii) when the DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer; and

*(iii) when the intercepted DNS request corresponds to a secure server, automatically creating a secure channel between the client and the secure server.*

Appx326(48:18-29).

## **2. VPN On Demand**

VPN On Demand is a feature that allows Apple's iPhones, iPads, and iPod touches to connect through a VPN to user-selected websites or domain names

---

<sup>1</sup> Emphases are added unless otherwise noted.

behind a firewall. Appx2198.<sup>2</sup> This Court previously affirmed a finding that an earlier version of VPN On Demand in versions 3 through 6 of Apple’s iOS operating system (“original VPN On Demand”) infringed the ’135 and ’151 patents. *VirnetX I*, 767 F.3d at 1320-1321. Apple modified VPN On Demand in later iOS versions by removing the functionality found to infringe in *VirnetX I* (“redesigned VPN On Demand”).

***a. Original VPN On Demand***

As relevant here, original VPN On Demand included two modes of operation: “Always” and “If Needed.” Appx2200-2201; Appx5090. Only the “Always” mode was at issue in *VirnetX I*, as VirnetX admitted “the If Needed functionality did not infringe.” Appx1331-1332(77:22-78:2); *see* Appx78; Appx16281.

When using the “Always” mode, a user populated a configuration file with a list of domain names to which the user wished to connect using a VPN. Appx2200-2201; Appx5089-5090. The “Always” mode compared a requested domain name to the configuration file. Appx2200-2204. If the requested domain name matched a name in the configuration file, the “Always” mode initiated a VPN. *Id.*; *VirnetX I*, 767 F.3d at 1315. In the “Always” mode, a VPN would

---

<sup>2</sup> A firewall is a server through which all communications with computers in a private network must pass. It protects the private network from being accessed by unauthorized persons. Appx303(2:60-63); Appx2334-2335; *see* Appx2196-2197.

*always* be established if the requested domain name was in the configuration file, regardless of whether the requesting device was located inside or outside the requested site's firewall. Appx2203-2204; Appx1450-1451; Appx1463-1464; Appx1514-1515.

By contrast, as the district court here recognized, the "If Needed" mode was "location-based." Appx78. In deciding whether to initiate a VPN, the "If Needed" mode performed an added "location check" to determine whether the requesting device was located inside or outside the requested site's firewall. Appx2201; Appx2205-2207. The "If Needed" mode initiated a VPN if the domain name was listed in the configuration file *and* the requesting device was outside the firewall. Appx2206-2207. But if the requesting device was inside the firewall, the "If Needed" mode did *not* create a VPN—even if the requested domain name was listed in the configuration file. *Id.*; *see* Appx2201.

In *VirnetX I*, VirnetX alleged that the "Always" mode "determined whether" the user was requesting access to a secure site by checking the requested domain name against the configuration file, which was "designed and intended to be used only for accessing secure private networks." 767 F.3d at 1320. This Court agreed, concluding that the "Always" mode was "consistent with how the claimed functionality is described in the specification," which explains that "the proxy

identifies a request for ‘access to a secure site ... by reference to an internal table of such sites.’” *Id.* (quoting Appx175(38:23-30)).

***b. Redesigned VPN On Demand***

In September 2013, Apple removed the “Always” mode from VPN On Demand in iOS versions 7 and later. Appx2209-2210; Appx2223-2224 (“The code for Always was removed.”); Appx2227; Appx5042; *see* Appx1332; Appx1969. The “If Needed” mode remained and was updated to include additional location checks to ensure that a VPN is initiated only when the requesting device is outside the secure site’s firewall. Appx2214-2215.<sup>3</sup>

For redesigned VPN On Demand, VirnetX’s infringement theory accused only a specific implementation of the “If Needed” mode, namely with an optional “HTTPS probe,” which allows the requesting device “to better determine whether [it is] inside the firewall or outside the firewall.” Appx2215; Appx2217-2218; Appx2349. The probe—which is turned off by default and requires special configuration to install (Appx2218, Appx1432)—was designed for the rare situation in which a requested server inside a firewall has the same domain name as a server outside the firewall. Appx2219-2220; Appx2354. In such circumstances, the optional probe—when configured—sends a test message to a

---

<sup>3</sup> At trial, witnesses also referred to the accused functionality in redesigned VPN On Demand as “Evaluate Connection,” which includes the “If Needed” mode. *E.g.*, Appx1431(176:6-9); Appx2226(176:10-14); Appx2391(83:5-12); *see* Appx5041-5043.

separate probe server to determine whether the requesting device is inside or outside the firewall. Appx2220-2222; Appx1439.

If the optional probe determines that the requesting device is outside the secure site's firewall (because the test message fails to reach the probe server), the "If Needed" mode initiates a VPN. Appx2220-2221; Appx2344-2345; Appx1439-1441. But if the probe determines that the requesting device is inside the firewall (because the test message succeeds), the "If Needed" mode does *not* initiate a VPN. Appx2221-2222; Appx2345. Thus, as in original VPN On Demand, the "If Needed" mode in redesigned VPN On Demand "uses only the location of [the] device and whether it's outside the firewall" to decide whether to initiate a VPN, even when the requested domain name is listed in the configuration file. Appx2224; *see* Appx2253-2254; Appx2345-2347; Appx1444-1448; Appx1460-1462; Appx1515-1517.

## **B. Background Regarding FaceTime**

### **1. VirnetX's '504 and '211 patents**

The common specification of the '504 and '211 patents discloses methods and systems for "establishing a secure communication link between a first computer and a second computer over a computer network, such as the Internet." Appx237(6:40-43); *see* Appx259(49:4-6); *VirnetX I*, 767 F.3d at 1314. The

described system is “built on top of the existing Internet protocol (IP).” Appx237(6:21-24); *see* Appx238(7:41-45); Appx261(53:18-23).

According to the specification, a DNS request is first sent to the domain name server as in conventional Internet communications. Appx259(49:21-25). The domain name server, which contains a “database of secure domain names and corresponding secure network addresses” (Appx260(51:11-15)), returns the requested server’s IP address, allowing the web page associated with the requested domain name to be displayed “in a well-known manner” (Appx259-260(49:32-36, 51:43-46)). In one embodiment, the web page also displays “a hyperlink, or an icon representing a hyperlink” (*e.g.*, “‘go secure’ hyperlink”), indicating that the system supports establishing a secure communication link with the server corresponding to the requested domain name. Appx259-261(49:36-44, 51:62-67, 52:9-14); *see* Appx230-231(Figs. 33-34).

VirnetX asserted claims 1, 2, 5, and 27 of the ’504 patent and claims 36, 47, and 51 of the ’211 patent. Appx2755. Claim 1 of the ’504 patent is representative:

1. A system for providing a domain name service for establishing a secure communication link, the system comprising:

a ***domain name service system*** configured to be connected to a communication network, to store a plurality of domain names and corresponding network addresses, to receive a query for a network address, and to comprise ***an indication that the domain name service system supports establishing a secure communication link.***

Appx262(55:49-56). The district court construed “domain name service” as “a lookup service that returns an *IP address* for a requested domain name to the requester.” Appx22214; Appx15064; Appx15066.

The district court interpreted the “indication” limitation to mean “an indication other than merely returning of requested DNS records, such as an IP address or key certificate, that the domain name service system supports establishing a secure communication link.” Appx15049-15051; *see* Appx15051-15052 (similar construction for “indicate” limitation in ’211 patent). This Court previously construed “secure communication link” as “a direct communication link that provides data security and anonymity.” *VirnetX I*, 767 F.3d at 1317-1319.

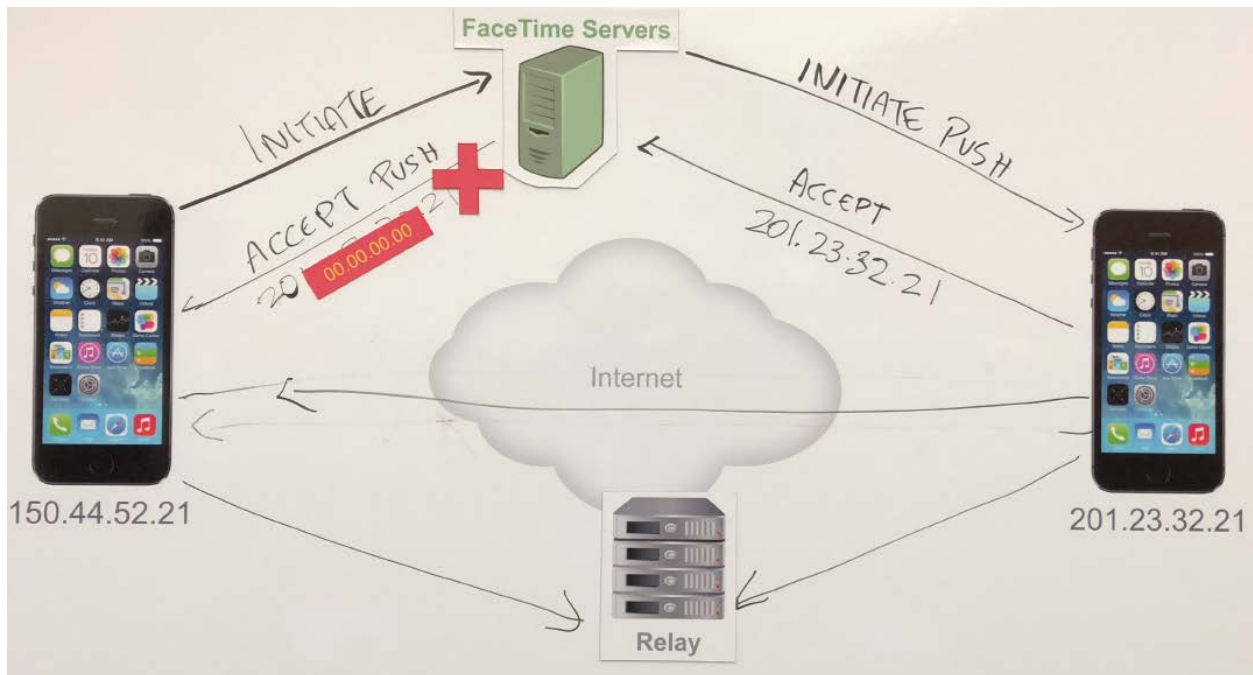
## **2. FaceTime**

FaceTime is a video-calling application for Apple’s iPhones, iPads, iPod touches, and Mac computers. Appx2107. In 2012, a jury found that an earlier version of FaceTime (“original FaceTime”) infringed the ’504 and ’211 patents. Appx1315. Apple modified FaceTime in 2013 to remove the functionality found to infringe (“redesigned FaceTime”).

### ***a. Original FaceTime***

A caller initiates a FaceTime call by selecting the intended recipient’s phone number or email address. Appx2112; Appx2295-2296. The caller’s device then sends an “Initiate” message to Apple’s FaceTime server. Appx2112; Appx2296.

The FaceTime server responds by sending an “Initiate Push” message to the receiving device, which causes the receiving device to ring. Appx2112; Appx2296. When the recipient accepts the incoming call, the receiving device sends an “Accept” message to the FaceTime server, as shown below. Appx2112-2113; Appx2296.



Appx5140.

After receiving the “Accept” message, the FaceTime server sends a fourth message—the “Accept Push” message—to the caller’s device. Appx2113; Appx2296; *see* Appx1361-1362; Appx1410-1411. In original FaceTime, the Accept Push message included the receiving device’s IP address, which the caller’s device could use to establish a direct connection with the receiving device. Appx2113; Appx2297-2298. After the FaceTime server transmits the Accept Push



message, its role in establishing a call comes to an end, and the two communicating devices exchange audio/video information over the Internet without passing through the FaceTime server. Appx2296; *VirnetX I*, 767 F.3d at 1314.

***b. Redesigned FaceTime***

In April 2013, Apple modified its FaceTime servers by “zero[ing] out” or removing the receiving device’s IP address from the Accept Push message. Appx1422; Appx2115-2116; Appx2297-2298. The Accept Push message instead contained the IP address of a “relay” server. Appx1425; Appx1502-1503; Appx2133. With that modification, all FaceTime calls were routed “indirectly” through a relay server. Appx2115-2117; Appx2298-2299. VirnetX agreed this redesign did not infringe. *See* Appx1417(Jones) (relay connection is not “direct” and therefore is non-infringing); Appx1423(Jones) (modified Accept Push message was not claimed “indication”).

In September 2013, Apple released a software update for the operating systems on its mobile devices (iOS version 7) and Mac computers (OS X version 10.9). Appx1313; Appx1420; Appx2108. That software update provided a way (called “the ICE protocol”) for the two communicating devices themselves to establish a non-relayed connection *after* the Accept Push message is sent by the FaceTime servers and received by the caller’s device. Appx2319; Appx2325-

2326. VirnetX's infringement accusations in this case are directed to the September 2013 version of FaceTime. Appx1315.

### **C. Litigation History**

#### **1. First trial and appeal in 417 Action**

VirnetX initially sued Apple in August 2010 ("the 417 Action"), alleging infringement by original VPN On Demand and original FaceTime. Appx1. In November 2012, a jury found all four asserted patents infringed and not invalid as anticipated by the single prior-art reference presented at trial ("Kiuchi"), and awarded \$368 million. Appx2.

This Court affirmed that original VPN On Demand infringed the '135 and '151 patents and that the asserted claims were not invalid as anticipated by Kiuchi, but vacated the original FaceTime infringement judgment. *VirnetX I*, 767 F.3d at 1313-1314, 1320-1324. The Court ruled that the claimed "secure communication link" in the '504 and '211 patents requires "anonymity," and remanded to determine whether FaceTime provided anonymity. *Id.* at 1317-1319. The Court also vacated the damages award, because VirnetX's expert Roy Weinstein failed "to apportion the royalty down to a reasonable estimate of the value of its claimed technology." *Id.* at 1329.

## **2. Consolidated trial in 417 and 855 Actions**

On the same day the verdict was announced in the 417 Action, VirnetX filed this suit (“the 855 Action”) asserting the same four patents against the redesigned versions of VPN On Demand and FaceTime, as well as a separate application called iMessage. Appx2; Appx66.

In the new 855 Action, Apple sought to challenge the patents’ validity on grounds not adjudicated in the 417 Action. The district court recognized that claim preclusion did not prevent Apple from doing so, because Apple’s redesigned products were not “essentially the same” as the original accused products. Appx5. However, the court barred Apple’s invalidity defenses and counterclaims based on issue preclusion. Appx5-9. The court reasoned that “patent invalidity is a single ‘issue’ for preclusion purposes” and that—because Apple raised and lost on one invalidity ground (anticipation by Kiuchi) in the 417 Action—it was barred from raising *any* invalidity ground in the 855 Action. Appx6-7.

The district court consolidated the 417 and 855 Actions over Apple’s objection. Appx67. In February 2016, the jury in the consolidated trial found infringement by redesigned VPN On Demand, original and redesigned FaceTime, and iMessage. Appx15620-15621. That jury awarded \$625 million in damages. Appx15619-15622.

The district court granted Apple a new trial, explaining that the consolidation and VirnetX's repeated statements about the prior verdict had created "the potential for juror confusion and unfairly prejudiced Apple's right to a fair trial." Appx15625. The court accordingly deconsolidated the 417 and 855 Actions for separate trials. Appx15639.

### **3. Retrial and second appeal in 417 Action**

The third trial in the 417 Action took place in September 2016. The sole infringement issue was whether original FaceTime provided "anonymity." Appx22404. VirnetX again presented its damages case through Mr. Weinstein, who urged the jury to award \$1.20 per unit—a rate he derived from six VirnetX license agreements entered to settle litigation. Appx1811-1812; Appx1868.

The jury found that original FaceTime infringed, and awarded \$302 million for infringement of all four patents by original VPN On Demand and original FaceTime. Appx68. The district court denied Apple's post-trial motions, and awarded VirnetX \$137 million in enhanced damages, prejudgment interest, attorney's fees, and costs. Appx22453-22456. In January 2019, a panel of this Court summarily affirmed the \$439 million judgment. *VirnetX Inc. v. Cisco Sys., Inc.*, No. 2018-1197, 2019 WL 190518 (Fed. Cir. Jan. 15, 2019) (per curiam).

#### 4.     **Retrial in 855 Action**

The second trial in the 855 Action, from which this appeal arises, was held in April 2018.

***Redesigned VPN On Demand.*** Although VirnetX conceded that original VPN On Demand’s “If Needed” mode did not infringe (Appx1331-1332), VirnetX contended that redesigned VPN On Demand’s “If Needed” mode, when operated with the optional HTTPS probe, performed the claimed steps of “determining whether” a DNS request corresponds to a secure server and “automatically initiating” a VPN in response to that determination. Appx1432-1434. Yet VirnetX’s expert Mark Jones, Apple’s engineer Simon Patience, and Apple’s expert Matthew Blaze all testified that the “If Needed” mode may or may not initiate a VPN when the requested domain name is listed in the configuration file—and thus corresponds to a secure server as this Court held in *VirnetX I*, 767 F.3d at 1320. *E.g.*, Appx1447-1448(Jones). Instead, the ultimate decision to initiate a VPN is ***location-based***: the “If Needed” mode initiates a VPN only if it determines the requesting device is located outside the secure server’s firewall. Appx1440-1441, Appx1447-1448(Jones); Appx2220-2224(Patience); Appx2344-2345(Blaze).

VirnetX offered no evidence that Apple or any of its customers actually used the narrow, accused implementation of redesigned VPN On Demand in the United States (by configuring the optional probe), or that Apple induced any such use.

***Redesigned FaceTime—“DNS System.”*** Earlier in the 855 Action, Apple had moved to exclude Dr. Jones’s infringement opinion for the ’504 and ’211 patents because his report concededly did not address how redesigned FaceTime satisfied the court’s construction of “DNS,” which requires return of an IP address. Appx15147-15149; Appx15539-15540; see Appx15607(146:23-147:7); Appx15217-15218 & n.8. The district court denied Apple’s motion, stating that “[a]lthough Apple presents valid criticisms of Dr. Jones’s opinions, they go to the weight of the evidence rather than admissibility.” Appx13.

Shortly before the 2018 trial, the district court—at VirnetX’s urging—ruled that the construction of “DNS” did not apply to the claimed “DNS system.” Appx19. Because Apple’s non-infringement defense regarding this limitation turned on the fact that redesigned FaceTime does not return an IP address (and thus does not use a “DNS” as construed), that order meant that Apple could not present a non-infringement defense or cross-examine Dr. Jones based on the term “DNS system.” Nevertheless, VirnetX convinced the court to instruct the jury, over Apple’s objections (Appx2638-2648, Appx2746-2747), that “[DNS] system’ ...

does *not* incorporate or include the [c]ourt’s construction for the term “[DNS].” Appx2758.

***Redesigned FaceTime—“Indication.”*** VirnetX contended that the Accept Push message sent by the FaceTime server to the caller’s device was the claimed “indication,” even though Apple had removed the receiving device’s IP address from that message—a change Dr. Jones previously conceded was non-infringing. Appx1422-1424. At trial, Dr. Jones was unable to explain how the Accept Push message “indicates” that the accused FaceTime servers support establishing a *direct* communication link. See Appx1362; Appx1376-1378. Meanwhile, Dr. Blaze explained that nothing in the Accept Push message provides such an “indication.” Appx2328-2329.

***iMessage.*** Despite maintaining throughout the 855 Action that Apple’s iMessage feature also infringed the ’504 and ’211 patents, VirnetX presented no evidence relating to iMessage at trial. Appx89.

***Damages.*** Mr. Weinstein again calculated a \$1.20 per-unit rate, as in the 417 Action. He arrived at that rate based on six VirnetX license agreements that had been entered to settle litigation. Appx1897. Although each license covered many more patents than the four asserted here and was negotiated as either a lump sum or a percentage of the entire market value of the covered products, Mr.

Weinstein derived a per-unit rate for each license by dividing the royalties paid by the number of products sold or expected to be sold. Appx1809; Appx1812.

<b>Licensee</b>	<b>Royalties Paid</b>	<b>Number of Unit Sales</b>	<b>Per-Unit Royalty Rate</b>
Microsoft	\$200,000,000		\$0.19
Avaya			\$0.34
Siemens			\$1.21
Mitel			\$1.43
Aastra			\$1.80
NEC			\$2.26

Appx5155.

Despite the wide variation in total royalties, unit sales, and per-unit rates, Mr. Weinstein gave all licenses equal weight by taking the simple average of their implied per-unit rates, producing a per-unit rate of \$1.20. Appx1809; Appx1812; Appx1852. Applying \$1.20 to 419 million accused Apple units, Mr. Weinstein calculated a royalty of \$502 million. Appx1855-1856; Appx5183.

**Verdict.** The jury found that redesigned VPN On Demand and redesigned FaceTime infringed, and awarded the \$502 million that Mr. Weinstein proposed. Appx50-52.

## **5. Post-trial rulings in 855 Action**

The district court denied Apple's JMOL and new trial motions. Appx71-102.



***Redesigned VPN On Demand.*** For the '135 and '151 patents, the district court concluded that “the fact that the HTTPS probe is location-based is not fatal to VirnetX’s claims.” Appx82. The court also believed there was sufficient evidence that Apple or “some subset of [its] customers” directly infringed by using redesigned VPN On Demand in the accused configuration (with the optional probe). Appx85-88. The court further concluded there was sufficient evidence for the jury to infer that Apple induced some customers to infringe, although it could not say how many or which ones. Appx90-91.

***Redesigned FaceTime.*** For the '504 and '211 patents, the district court concluded there was substantial evidence that redesigned FaceTime’s “Accept Push” message provided the claimed “indication” that the DNS system supports establishing a “direct” communication link. Appx71-76. According to the court, “[t]hat the accept push message can also be used to establish a relayed [*i.e.*, indirect] FaceTime call does not change the result because ‘[t]he addition of features does not avoid infringement[.]’” Appx75-76 (citation omitted).

Separately, the court “decline[d] to rule on” Apple’s argument that redesigned FaceTime does not provide the required “anonymity,” explaining that the same issue was resolved in VirnetX’s favor during the 417 Action and thus “issue preclusion attaches.” Appx76.

***“DNS System.”*** The court also denied Apple’s motions for JMOL or a new trial based on the “DNS system” limitation, stating that it would not reconsider claim construction rulings and that its jury instruction that the claimed “DNS system” did not include a “DNS” did not prejudice Apple. Appx76-77; Appx98-99.

***iMessage.*** The district court denied Apple’s request for JMOL as to iMessage. Appx89. Although neither VirnetX’s infringement claim nor Apple’s non-infringement counterclaim had been dismissed, and VirnetX had asserted that iMessage infringed “up to the time of trial,” the court concluded there was no active case or controversy because iMessage was “not presented for consideration to the jury.” *Id.*

***Damages.*** The district court largely reiterated its prior damages rulings. Appx91; *see* Appx14 (*Daubert* ruling noting Apple’s “valid criticisms of Mr. Weinstein’s opinions,” but saying they go to weight of the evidence rather than admissibility). The court held that apportionment of the six VirnetX licenses on which Mr. Weinstein relied “[wa]s not necessary,” because it believed the licenses’ rates were already apportioned. Appx93. Nor was the court troubled “that Mr. Weinstein could not describe how the apportionment ‘was done,’” as that gap went only to Mr. Weinstein’s “credibility” and “d[id] not render his opinion unreliable.” *Id.* As for Mr. Weinstein’s failure to account for the differences between the

VirnetX licenses and the hypothetical license, the court concluded it was enough that “Mr. Weinstein explained the factual circumstances surrounding each license[.]” Appx92.

The district court awarded VirnetX supplemental damages and ongoing royalties at the jury’s implied rate of \$1.20 per unit, \$173,549 in costs, \$93,177,592 in prejudgment interest, and post-judgment interest. Appx111-117; Appx16317-13618.<sup>4</sup>

### SUMMARY OF ARGUMENT

1. The infringement judgment for the ’135 and ’151 patents should be reversed. Apple redesigned VPN On Demand so that it does *not* “automatically initiate” a VPN if the requested domain name appears in the configuration file (and is thus a “secure server” as this Court held in *VirnetX I*, 767 F.3d at 1320). Instead—just like original VPN On Demand’s “If Needed” mode that VirnetX conceded was non-infringing—redesigned VPN On Demand undisputedly decides whether to initiate a VPN based on the requesting device’s *location*. Accordingly, no reasonable jury could find that redesigned VPN On Demand “automatically initiates” a VPN in response to “determining” that the DNS request is for a secure server, as the asserted claims require.

---

<sup>4</sup> The jury found willfulness (Appx64-65), but the district court declined to enhance damages and thereafter denied Apple’s post-trial motions regarding willfulness as moot. Appx96-97; Appx103-110. The court also denied VirnetX’s request for fees. Appx111. VirnetX has not cross-appealed those rulings.

VirnetX also failed to prove direct or induced infringement of the '135 and '151 patents. Because VirnetX accused only a narrow implementation of redesigned VPN On Demand—the “If Needed” mode with the optional HTTPS probe configured—it was required to prove actual infringing use of that implementation. Yet VirnetX adduced no non-speculative evidence that Apple or its customers used, or that Apple induced its customers to use, redesigned VPN On Demand in that configuration in the United States. And even if isolated uses could be inferred, they cannot support the full scope of the damages award—which treated as infringing every iPhone, iPad, and iPod touch running iOS version 7 or later.

2. The infringement judgment for the '504 and '211 patents cannot stand either. Apple redesigned FaceTime so that its servers do not “return[] an IP address,” as the construction of “DNS” requires. The district court’s ruling that the term “DNS system” does not include a “DNS” makes no sense as a matter of claim construction and is contrary to how the parties litigated the case. Once that legal error is corrected, redesigned FaceTime undisputedly does not infringe.

Separate from the claim construction issue, redesigned FaceTime still cannot infringe because the Accept Push message does not “indicat[e]” that FaceTime’s servers support a *direct* communication link, as the claim construction requires. VirnetX’s expert conceded that the Accept Push message in the April 2013

redesign “would not satisfy [the] requirements of the claims to be an indication of support.” Appx1423. The Accept Push message in the September 2013 redesign fails to provide the claimed “indication” for the same reasons.

3. The district court erred by barring Apple’s invalidity defenses and counterclaims. Issue preclusion does not apply, because Apple’s invalidity challenges in this case are not “identical” to the single invalidity issue actually litigated in the 417 Action (anticipation by Kiuchi under 35 U.S.C. § 102). This Court’s intervening ruling in *Voter Verified, Inc. v. Election Systems & Software LLC*, 887 F.3d 1376, 1382-1383 (Fed. Cir. 2018), confirms that the district court’s application of issue preclusion—which treated all invalidity theories as a single “issue”—was legal error. At a minimum, the case should be remanded so that Apple may present its different invalidity theories under 35 U.S.C. § 103.

4. The district court erred in refusing to enter judgment that iMessage does not infringe the ’504 and ’211 patents. The court plainly had jurisdiction to decide the issue: neither VirnetX’s infringement claim nor Apple’s non-infringement counterclaim had been dismissed, both parties identified iMessage in the governing pretrial order, and VirnetX refused to stipulate to dismissal with prejudice or provide Apple with a covenant not to sue. Thus, after years of litigating iMessage, there remained a genuine dispute of sufficient immediacy to allow the court to resolve it. Because VirnetX presented no

iMessage evidence at trial, Apple is entitled to judgment that iMessage does not infringe.

5. The PTO has found each asserted claim of the four patents-in-suit unpatentable in *inter partes* review and reexamination proceedings. If this Court affirms those unpatentability determinations and/or reverses the infringement findings for some or all patents-in-suit, the district court's judgment should be vacated and remanded for further proceedings (if necessary) to determine the applicable damages (if any) for the patents and infringement findings that remain.

\* \* \*

Apple also preserves the following arguments, which a panel of this Court summarily decided in VirnetX's favor in the 417 Action:

Redesigned FaceTime's servers do not provide "anonymity" for the same reasons that original FaceTime's servers do not, which is an additional basis for reversing the infringement judgment for the '504 and '211 patents.

The damages judgment should be reversed, or vacated and remanded for a new trial. As in the 417 Action, Mr. Weinstein's testimony should have been excluded because he did not apportion his demand to the value of the claimed invention in Apple's products, he failed to account for the differences between VirnetX's settlement licenses and the hypothetical license, and his \$1.20 per-unit

royalty rate was arbitrary. Additionally, the district court abused its discretion by awarding \$93 million in prejudgment interest.

### **STANDARD OF REVIEW**

This Court reviews JMOL, new trial, and evidentiary rulings under regional circuit law. *Verizon Servs. Corp. v. Cox Fibernet Va., Inc.*, 602 F.3d 1325, 1331 (Fed. Cir. 2010). The Fifth Circuit reviews the denial of JMOL *de novo*, determining whether the jury's verdict is supported by substantial evidence. *ACCO Brands, Inc. v. ABA Locks Mfr. Co.*, 501 F.3d 1307, 1311-1312 (Fed. Cir. 2007). Evidentiary rulings are reviewed for abuse of discretion. *Knight v. Kirby Inland Marine Inc.*, 482 F.3d 347, 351 (5th Cir. 2007); *United States v. Lewis*, 796 F.3d 543, 545-546 (5th Cir. 2015). An "erroneous view of the law" is "necessarily" an abuse of discretion. *Highmark Inc. v. Allcare Health Mgmt. Sys., Inc.*, 572 U.S. 559, 563 n.2 (2014) (internal quotation marks omitted).

Claim construction relying only on intrinsic evidence is reviewed *de novo*. *Poly-Am., L.P. v. API Indus., Inc.*, 839 F.3d 1131, 1135-1136 (Fed. Cir. 2016). Infringement is reviewed for substantial evidence. *Intellectual Ventures I LLC v. Motorola Mobility LLC*, 870 F.3d 1320, 1331 (Fed. Cir. 2017).

A jury's damages award is reviewed for substantial evidence. *Lucent Techs., Inc. v. Gateway, Inc.*, 580 F.3d 1301, 1310, 1324 (Fed. Cir. 2009). A

prejudgment interest award is reviewed for abuse of discretion. *Group One, Ltd. v. Hallmark Cards, Inc.*, 407 F.3d 1297, 1307 (Fed. Cir. 2005).

The Fifth Circuit reviews *de novo* whether issue preclusion applies. *Wills v. Arizon Structures Worldwide, L.L.C.*, 824 F.3d 541, 545 (5th Cir. 2016).

This Court reviews whether the district court lacked subject matter jurisdiction *de novo*, but reviews underlying factual findings for clear error. *Hewlett-Packard Co. v. Acceleron LLC*, 587 F.3d 1358, 1361 (Fed. Cir. 2009).

Jury instructions are reviewed for abuse of discretion, but the underlying “legal conclusions” are reviewed *de novo*. *United States v. CITGO Petroleum Corp.*, 801 F.3d 477, 481 (5th Cir. 2015).

## **ARGUMENT**

### **I. THE INFRINGEMENT JUDGMENT FOR THE ’135 AND ’151 PATENTS SHOULD BE REVERSED.**

#### **A. Redesigned VPN On Demand Does Not “Automatically Initiate” A VPN In Response To “Determining” That A DNS Request Is For A Secure Server.**

Each asserted claim of the ’135 and ’151 patents requires “determining” whether a DNS request corresponds to a “secure server” (or “secure web site”). Appx180(47:27-28); Appx326(48:22-23). If the DNS request is for a secure server, the claimed invention requires “automatically initiating [or creating]” a VPN or secure channel. Appx180(47:29-32) (“in response to determining that the DNS request in step (2) is requesting access to a secure target web site,



automatically initiating the VPN”); Appx326(48:27-29) (“when the intercepted DNS request corresponds to a secure server, automatically creating a secure channel”).

VirnetX conceded that redesigned VPN On Demand does not infringe when, as by default, the optional HTTPS probe is not enabled. Appx1432-1433; Appx2714-2715. VirnetX asserted infringement only in the “If Needed” mode with the probe enabled. Appx1432-1433. But even in that optional implementation, the “If Needed” mode undisputedly decides whether to initiate a VPN based on *the requesting device’s location*—*i.e.*, whether the requesting device is inside or outside the secure server’s firewall. Accordingly, no reasonable jury could find that redesigned VPN On Demand “automatically initiates” a VPN in response to “determining” that the DNS request is for a secure server.

**1. The “If Needed” mode initiates a VPN based on the requesting device’s location, not—as the claims require—whether the DNS request is for a secure server.**

In *VirnetX I*, this Court held that the claimed step of “determining whether” a DNS request is for a secure server was satisfied by checking the requested domain name against the configuration file. 767 F.3d at 1320. And in original VPN On Demand, the “Always” mode automatically initiated a VPN in response to that determination. *Id.* at 1315, 1320 (“If the entered domain name matches a

domain name in the configuration file, VPN On Demand ... automatically establishes a VPN between the user's browser and the target computer[.]”).

In redesigned VPN On Demand, by contrast, the “If Needed” mode does *not* automatically initiate a VPN if the requested domain name appears in the configuration file. VirnetX accordingly conceded that, in redesigned VPN On Demand, checking the configuration file alone does not infringe. Appx2712-2714(82:3-84:6)(Jones). As in original VPN On Demand's “If Needed” mode, which VirnetX also concedes is non-infringing (*see supra* p. 10), the “If Needed” mode in redesigned VPN On Demand performs a *location* check and *only* initiates a VPN if it determines that the requesting device is located outside the secure server's firewall. Appx2220-2222, Appx2224(Patience); Appx2344-2345(Blaze); Appx1440-1441(Jones). Thus, even when the requested domain name appears in the configuration file—and is therefore a “secure server” as this Court held in *VirnetX I*, 767 F.3d at 1320—the “If Needed” mode does *not* initiate a VPN if the requesting device is inside the secure server's firewall. Appx2224(Patience).

These facts were undisputed. VirnetX's expert Dr. Jones admitted that, “[e]ven if the domain name is on the list” in the configuration file, whether the “If Needed” mode initiates a VPN depends on the requesting device's *location*:

QUESTION: So as we just saw with the If Needed mode for 7 and greater, depending on where you were, you may or may not get a VPN. Right? Even if the domain name is on the list. Right?

ANSWER: Yes.

QUESTION: So it's location dependent. Right?

ANSWER: Effectively, yes.

Appx1448 (internal quotation marks omitted); *see* Appx1447-1448(Jones) (agreeing that “for iOS 7, depending on where you are, you may or may not get VPN even if the domain name is on the list”); Appx82 (district court acknowledging “the HTTPS probe is location-based”).

Dr. Jones thus conceded that users attempting to access the *same secure server* would achieve different results (VPN or no VPN) depending solely on whether they were inside or outside the secure server's firewall. Appx1446-1448; Appx1515-1516 (“Q. ... VPN On Demand did not create a VPN when the [same] user was inside but did create a VPN when it was outside. Right? A. Absolutely. Q. Different results based on the different location. Right? A. Yes.”); Appx1516-1517 (“Q. And so you could have the same server with the same website, right, both of the people authorized by the same company to access that server, and this person inside will not get a VPN and this person outside will. Right? A. Yes. Q. And the only difference between those people is that one is inside and one's outside. That's why you get a different result. Right? A. Correct.”).

The record thus demonstrates that—even when the requested domain name is listed in the configuration file and thus corresponds to a secure server, *see*

*VirnetX I*, 767 F.3d at 1320—the “If Needed” mode does not “automatically initiat[e]” a VPN. It initiates a VPN based on a separate determination that the requesting device is located outside the secure server’s firewall. Accordingly, no reasonable jury could find that redesigned VPN On Demand “automatically initiat[es or creates]” a VPN in response to “determining” that a DNS request corresponds to a secure server.

**2. The district court’s reasons for denying JMOL are inconsistent with *VirnetX I* and unsupported by substantial evidence.**

The district court nonetheless concluded that “the fact that the HTTPS probe is location-based is not fatal to VirnetX’s claims.” Appx82. That is contrary to the claim language, the uncontroverted evidence, and this Court’s decision in *VirnetX I*.

The asserted claims require that a VPN be automatically created when the DNS request is determined to correspond to a “secure server,” and this Court previously held—at VirnetX’s urging—that checking the configuration file *by itself* “determines whether” the DNS request corresponds to a secure server. *VirnetX I*, 767 F.3d at 1320. But redesigned VPN On Demand does not automatically create a VPN based on the configuration file check or any determination that a secure server is requested. Rather, after *that determination* is made, redesigned VPN On Demand consults the result of the HTTPS probe’s

location check; and only then—in response to determining that the requesting device is outside the firewall—does it initiate a VPN. Appx5055; Appx2258(Patience). The subsequent decision based on the location check means that redesigned VPN On Demand does not “automatically initiate” a VPN in response to determining that the DNS request is for a secure server (*i.e.*, that the requested domain name is in the configuration file), but only in response to a determination that the requesting device is outside the firewall. *See supra* pp. 33-36. The district court did not address this point in its JMOL opinion. Appx81-82; *see* Appx16221 (Apple’s JMOL argument).

The district court believed that the HTTPS probe’s location check somehow determines whether a DNS request is for access to a secure server because, in its view, “whether the requesting device is inside or outside the private network affects whether a server *requires authorization for access* (which is a requirement of the [c]ourt’s construction of ‘secure server’).” Appx81-82 (emphasis in original).<sup>5</sup> The district court’s only support for that assertion—that a server can be both secure and non-secure, depending on the requesting device’s location—was Dr. Jones’s testimony that a requesting device located “outside of a private network” requires “authorization” to access a server behind a firewall. Appx82

---

<sup>5</sup> The district court construed “secure server” as “a server that requires authorization for access and that can communicate in an encrypted channel[.]” Appx15046; *see* Appx15065 (similar for “secure web site”).

(citing Appx1485(32:3-14)); *see* Appx1340-1341(86:14-87:22). But that is beside the point; what matters is whether a requesting device *inside* the firewall would *also* require authorization to access the server, such that it should receive a VPN according to the patent claims (but would not receive one through redesigned VPN On Demand). The record on *that* key issue is unequivocal: Dr. Jones never denied that a requesting device inside the firewall *also requires authorization* to access the same server. Indeed, he agreed that redesigned VPN On Demand could have two people “*authorized by the same company to access [the same] server*, and th[e] person inside [the private network] will not get a VPN and th[e] person outside will.” Appx1516-1517. Apple engineer Simon Patience also testified, without contradiction, that a server behind a firewall in a private network “require[s] authorization for access” *no matter where* the requesting device is located. Appx2254(204:11-25); *see* Appx2346(38:8-23) (Apple’s expert Dr. Blaze explaining that, even if the requesting device is “internal to the company network, [it would] need some authorization” to access the secure server).

The district court also tried to invoke Mr. Patience’s testimony, but it misstated the record. Mr. Patience never testified that “the location of a requesting device can bear on whether a target server is a ‘secure server.’” Appx82. He testified that a *server* located behind a firewall is typically secure. Appx2248(198:11-17); *see* Appx2264(214:8-14). But he never said that a server’s

security somehow turns on the “location of a requesting device.” Likewise, the fact that Mr. Patience “confirmed that servers behind firewalls require authorization for access” (Appx82 (citing Appx2254(204:11-15)) in no way suggests that such servers do *not* “require authorization for access” if the requesting device is inside the firewall. Rather, Mr. Patience explained that authorization is required *regardless* of where the requesting device is located. Appx2254(204:11-25) (“Q. When the device is outside of the firewall, does it require authorization for access to communicate with a server within the firewall? A. Yes.... Q. And when a device is inside the firewall, would it require authorization to access the server in the same network? A. Yes....”).

In short, none of the testimony cited by the district court—indeed, no testimony at all—supports a finding that the HTTPS probe’s check regarding the *requesting device’s* location determines whether the *target server* is secure. The record was clear that it does not. Thus, no reasonable jury could find that making the creation of a VPN turn on the requesting device’s location—as the accused implementation of redesigned VPN On Demand does—satisfies the requirement of “automatically initiating [or creating]” a VPN based on a determination that the requested server is secure.

**B. VirnetX Failed To Prove Direct Or Induced Infringement.**

Separately, VirnetX gave the jury no non-speculative basis for finding that the narrow, optional implementation it accused was ever actually used in the United States. Without proof of either direct infringement or inducement, the infringement judgment cannot stand; at least, the damages award must be sharply reduced so that it sweeps no further than proven acts of infringement.

**1. VirnetX did not prove direct or induced infringement of the '135 patent.**

Because both asserted '135 patent claims are method claims, infringement depends on proof that “a person ... practiced all steps of the claimed method” in the United States. *Lucent*, 580 F.3d at 1317; *see* 35 U.S.C. § 271(a). VirnetX admitted that redesigned VPN On Demand can be used in non-infringing ways, including in the default setting with the HTTPS probe disabled. Appx1432-1433(Jones). VirnetX proved no instance in which anyone employed the specific accused implementation—the “If Needed” mode with the optional HTTPS probe configured—within the United States.

The district court believed the jury could infer such use through three pieces of circumstantial evidence, but none advanced beyond speculation. *First*, the court noted Dr. Jones’s assertion that Apple itself infringed “by testing VPN On Demand” (Appx86 (citing Appx1386(131:10-12))), but even Dr. Jones did not say



any such testing occurred in the United States. The court cited no evidence supporting such a finding.

*Second*, the district court cited an internal Apple email with a “draft of the VPN On Demand test plan” containing three possible “cases.” Appx5149 (cited at Appx86). The email states: “Ideally, these options would be presented to the customer, and based on the response, one of the three cases would be used as a test plan.” *Id.* Only one of the cases (“Case 3”) mentions the optional probe that is essential to VirnetX’s infringement theory. Appx5152 (“Optionally add the key *RequiredURLStringProbe* with a URL to an internal HTTPS host[.]”). VirnetX offered no evidence that: (1) this “draft” was ever finalized; (2) the final version still contained “Case 3”; (3) it was ever “presented” to a customer; (4) the customer chose “Case 3”; (5) the customer chose to “[o]ptionally add” the probe; (6) such a test plan was actually carried out; and (7) it was performed in the United States. It would have been pure speculation for the jury to infer that *any* of those steps occurred, yet *all seven* were required to prove direct infringement under VirnetX’s theory. While a jury may make reasonable inferences based on evidence, it may not make unsupported leaps where the party with the burden of proof merely raises a bare possibility that something *could* have happened. *See E-Pass Techs., Inc. v. 3Com Corp.*, 473 F.3d 1213, 1222 (Fed. Cir. 2007) (affirming summary judgment of non-infringement because evidence of “proposed

... protocols fail[ed] to show that any such protocol was ever actually deployed or that, if deployed, it would infringe”).

*Third*, the district court found “circumstantial evidence that some subset of Apple’s customers” used the accused implementation of redesigned VPN On Demand. Appx87. The “evidence” it cited does not support that conclusion. The court relied on Dr. Jones’s testimony that a user “can” “replicate the Always functionality” from original VPN On Demand by misconfiguring the HTTPS probe. Appx1388-1389(133:2-134:14); *see* Appx5142-5144; Appx5145-5148.<sup>6</sup> But the mere fact that users “can” hypothetically infringe is not, by itself, substantial evidence that they did. *See, e.g., Mirror Worlds, LLC v. Apple Inc.*, 692 F.3d 1351, 1361 (Fed. Cir. 2012) (although accused products “*could* infringe[,] ... such testimony alone is not sufficient to find inducement of infringement of a method patent. Evidence of actual use of each limitation is required.”); *ACCO Brands*, 501 F.3d at 1313 (instructions describing infringing method did not constitute “evidence of actual users having operated the lock in an infringing

---

<sup>6</sup> Dr. Jones conceded on cross-examination that he was unaware of Apple telling anyone to misconfigure the probe such that it always failed, as would be needed to “replicate” the original “Always” functionality. Appx1519(66:3-11).

manner”); *E-Pass*, 473 F.3d at 1222 (“[I]t requires too speculative a leap to conclude that any customer actually performed the claimed method.”).<sup>7</sup>

Moreover, liability based on third-party actions would require proof that Apple induced such a third party to infringe. *Global-Tech Appliances, Inc. v. SEB S.A.*, 563 U.S. 754, 765-766 (2011). Whatever “subset” of Apple’s customers the district court believed configured the optional probe in the United States, it identified no evidence that they did so with Apple’s encouragement. The court’s reference to the draft test plan and the suggestion that the proposed options “[i]deally” “would be presented” to an unidentified customer (Appx90; Appx5149) again comes nowhere close to proving that such a presentation ever happened, that the customer chose an option including the optional probe, or that the claimed steps were performed in the United States.

**2. VirnetX did not prove direct or induced infringement of the ’151 patent.**

Although claim 13 of the ’151 patent is not a method claim, the analysis is similar because it requires “computer readable instructions that, when executed, cause a data processing device” to perform specified steps. Appx326(48:18-29). It

---

<sup>7</sup> The district court (Appx87) cited *Moleculon Research Corp. v. CBS, Inc.*, 793 F.2d 1261 (Fed. Cir. 1986), but as this Court later explained, “the device at issue in *Moleculon* was intended to be used in only one way—to practice the infringing method—and that method was explicitly taught by the proffered instructions” accompanying the product. *E-Pass*, 473 F.3d at 1222. That is not the case here.

is undisputed that redesigned VPN On Demand software as sold on Apple's accused devices **does not** perform the claimed steps, because the optional HTTPS probe is disabled by default and cannot be turned on without special configuration. Appx2218; Appx1432.

The district court waved off this fact by misinterpreting claim 13 as “drawn to structure that has a specified capability.” Appx87. That was incorrect for two reasons. *First*, the claim does not recite mere capability; it requires that execution of the computer readable instructions actually “**cause** a data processing device” to perform the claimed steps, including “automatically creating a secure channel between the client and the secure server.” Appx326(48:28-29). The specification likewise describes not mere capability, but performance of the recited functions: for example, the claimed invention “**automatically sets up** a [VPN].” Appx321(37:36-38); *see Typhoon Touch Techs., Inc. v. Dell, Inc.*, 659 F.3d 1376, 1381 (Fed. Cir. 2011) (similar claim and specification language “requires that the memory is actually programmed or configured” to perform claimed function). Where, as here, claim language does not “specif[y] that the claim is drawn to capability,” the possibility that a product is “reasonably capable of being put into the claimed configuration is insufficient for ... infringement.” *Ball Aerosol & Specialty Container, Inc. v. Limited Brands, Inc.*, 555 F.3d 984, 994-995 (Fed. Cir. 2009); *see Fujitsu Ltd. v. Netgear Inc.*, 620 F.3d 1321, 1329 (Fed. Cir. 2010)

(“Unless the claim language *only requires the capacity* to perform a particular claim element, we have held that it is not enough to simply show that a product is capable of infringement; the patent owner must show evidence of specific instances of direct infringement.”).<sup>8</sup>

*Second*, even if claim 13 of the ’151 patent were directed to mere capability, VirnetX was still required to prove that “the *unmodified* accused devices” were reasonably capable of performing the claimed functions “without significant alterations.” *Ericsson, Inc. v. D-Link Sys., Inc.*, 773 F.3d 1201, 1217 (Fed. Cir. 2014). Here, it is undisputed that the accused devices would *need to be modified* to perform the claimed functions. Appx1432 (Dr. Jones admitting that “unless someone configures it, [the optional probe is] not going to be used”); Appx2218

---

<sup>8</sup> By contrast, the line of cases on which the district court relied (Appx87-88) involved claims specifically drawn to capability. *See, e.g., Fujitsu*, 620 F.3d at 1329 (distinguishing *Intel Corp. v. ITC*, 946 F.2d 821 (Fed. Cir. 1991), as involving term that “only required that the infringing product be capable of infringing”); *Typhoon Touch*, 659 F.3d at 1380 (similarly distinguishing *Microprocessor Enhancement Corp. v. Texas Instruments, Inc.*, 520 F.3d 1367, 1375 (Fed. Cir. 2008)); *Revolution Eyewear, Inc. v. Aspex Eyewear, Inc.*, 563 F.3d 1358, 1369 (Fed. Cir. 2009) (claim recited “capable of engaging”). Two cited cases did not even decide infringement, but merely that the presence of functional language in apparatus claims did not render them indefinite. *MasterMine Software, Inc. v. Microsoft Corp.*, 874 F.3d 1307, 1315-1316 (Fed. Cir. 2017); *UltimatePointer, L.L.C. v. Nintendo Co.*, 816 F.3d 816, 826 (Fed. Cir. 2016).

(uncontradicted testimony of Apple engineer that HTTPS probe is “rarely needed” and would need to be configured “by the IT person”).<sup>9</sup>

**3. The damages should be reduced to reflect the limited scope of infringement (if any) based on use of the *optional* probe.**

Even if VirnetX had proven some isolated “testing” by Apple or use by a “subset” of customers, the damages would have to be limited to “the extent to which the infringing method has been used.” *Lucent*, 580 F.3d at 1335. A finding that every Apple product containing redesigned VPN On Demand “performed the patented method ... in the United States” is simply not supported. *Id.* at 1334. At a minimum, therefore, the damages remedy should be vacated and remanded.

**II. THE INFRINGEMENT JUDGMENT FOR THE ’504 AND ’211 PATENTS SHOULD BE REVERSED.**

**A. The District Court Erroneously Instructed The Jury That The Claimed “DNS System” Does Not Incorporate The “DNS” Construction.**

Each asserted claim of the ’504 and ’211 patents requires a “domain name service system” (“DNS system”). Appx262(55:49-56); Appx402(57:38-46). During claim construction, both parties treated that term as incorporating the similar term “domain name service” (“DNS”), which the district court construed to

---

<sup>9</sup> As with the ’135 patent, the district court’s alternative finding of instances of “actual infringement” of the ’151 patent relied on the draft test plan, which the court believed allowed the jury to “infer ... that Apple carried out its test plan and followed through on its plan to present it to customers.” Appx88. As shown above (pp. 41-43), such a finding requires multiple speculative leaps.

mean “a lookup service that returns an IP address for a requested domain name to the requester.” Appx22214; Appx15064; Appx15066. But after Apple redesigned FaceTime to avoid that limitation (and the “indication” limitation, *see infra* pp. 51-55), VirnetX convinced the district court to instruct the jury—for the first time at the fourth trial—that the claimed “[DNS] system’ ... does *not* incorporate or include the [c]ourt’s construction for the term ‘[DNS].’” Appx2758. That instruction was erroneous and prejudicial to Apple, and requires reversal or at least a new trial.

**1. The district court’s claim construction instruction was erroneous.**

The claimed “DNS system” naturally includes the limitations of the claimed “DNS.” That is the ordinary meaning of the claim language, which uses identical words—“domain name service”—in both terms. The district court improperly stripped “DNS system” of its full meaning. *See, e.g., Phonometrics, Inc. v. Northern Telecom Inc.*, 133 F.3d 1459, 1465 (Fed. Cir. 1998) (“A word or phrase used consistently throughout a claim should be interpreted consistently.”).

VirnetX itself previously acknowledged that the term “DNS system” includes the claimed “DNS.” During claim construction proceedings before the prior district judge (Judge Davis), VirnetX proposed that “DNS system” required no construction or, alternatively, should be interpreted as “a computer system that includes a [DNS].” Appx20024; *see* Appx21108; Appx21231-21232; Appx22166;

Appx22183; Appx22218. VirnetX even said that “[t]his alternative, proposed construction is *a straightforward adaptation of the [c]ourt’s prior construction of ‘[DNS].’*” Appx20024. Apple likewise proposed that the “DNS system” included a DNS. Appx21303-21304.

The parties’ claim construction dispute instead focused on a different issue, namely whether the “DNS system” must *also* be “capable of differentiating between, and responding to, both standard and secure top-level domain names,” as Apple urged. Appx21303-21304; *see* Appx22218. Judge Davis rejected Apple’s proposal and concluded that “DNS system” “d[id] not require construction.” Appx22219; *see* Appx22364 n.3 (Judge Davis stating that he “did not construe ‘[DNS] system’ because the claim language itself provide[s] a description of the term, i.e. that it must ‘comprise an indication that [it] supports establishing a secure communication link’”).

Thus, although Judge Davis construed “DNS” and “DNS system” as “separate terms with different constructions” (Appx19), that was to address the additional limitation proposed by Apple. Judge Davis’s claim construction order did not—and could not—remove “DNS” from the term “DNS system.” Consistent with that understanding, VirnetX’s expert expressly applied the limitations of Judge Davis’s “DNS” construction—*i.e.*, “a lookup service that returns an IP address for a requested domain name to the requester”—to the “DNS system” term



in his infringement analysis for original FaceTime. *E.g.*, Appx15547 (2012 expert report); Appx15601(67:3-68:10) (2012 trial testimony).

Judge Davis’s 2013 JMOL decision further confirms that the claimed “DNS system” incorporated the “DNS” construction. After the first trial, Apple challenged the infringement verdict on the basis that VirnetX failed to prove that original FaceTime provided a “lookup service” as required by the “DNS” construction. Appx22251-22253. Although VirnetX argued in opposition that “the claim term ... is ‘[DNS] system’ not ‘[DNS]’” (Appx22309 n.7), Judge Davis did *not* adopt that position. Instead, he applied the “DNS” construction to “DNS system,” finding sufficient evidence that original FaceTime “meets the ‘lookup service’ limitation when Apple’s FaceTime servers return an IP address for the requested domain name[.]” Appx22364.

**2. Under the correct claim construction, the infringement judgment for redesigned FaceTime cannot stand.**

Properly construed, the claimed “DNS system” includes a “DNS” and therefore must “return[] an IP address for a requested domain name to the requester.” *See* Appx22214. VirnetX’s expert conceded that redesigned FaceTime’s servers do not return an IP address for a requested domain name. Appx1422(Jones) (Apple modified FaceTime “to zero out or effectively remove the callee IP address”); Appx1424(Jones). Indeed, his report offered no opinion that redesigned FaceTime satisfied the “DNS” construction. Appx15607(146:23-

147:7); *see* Appx15217-15218 & n.8 (VirnetX admitting “Dr. Jones’s 2014 Report d[id] not address DNS with respect to claim 1 of the ’504 Patent”). Thus, under the correct claim construction, the infringement judgment for redesigned FaceTime should be reversed.

**3. At a minimum, the district court’s instruction was prejudicial error that warrants a new trial.**

Separate from the claim construction error, the district court’s jury instruction was unjustified and unfairly prejudiced Apple. After VirnetX filed an “emergency motion” for clarification regarding claim scope, Judge Schroeder ruled that the claimed “DNS system” did not incorporate the “DNS” construction. Appx19. As a result, Apple did not present a non-infringement defense based on—or cross-examine VirnetX’s expert regarding—the “DNS” construction. There was accordingly no basis to instruct the jury about the issue at all, but the district court did so nonetheless.

The district court’s suggestion that Apple’s expert’s testimony somehow justified the instruction is simply wrong. The court mistakenly believed there was “one instance” where Dr. Blaze, “*in response to Apple’s examination*,” suggested that the construction for “[DNS]” applied to “[DNS] system.” Appx98-99. But it was *VirnetX’s counsel* who confusingly asked Dr. Blaze “[w]hich one of these claim constructions does not apply to the ’504 and ’211 patents”—and Apple objected to that line of questioning. Appx2412-2413. Immediately after the

exchange, Dr. Blaze testified that he “underst[ood] that the construction of ‘[DNS] system’ does not incorporate the construction of ‘[DNS].’” Appx2421-2422.

The district court’s irrelevant instruction unfairly prejudiced Apple. Not only did it likely confuse the jury, but it also improperly suggested that the failure to return an IP address could not be a basis for non-infringement, even though that was part of Apple’s argument for the separate “indication” limitation. *See infra* pp. 51-55. Accordingly, if the infringement judgment is not reversed, a new trial is warranted. *See, e.g., Lenoir v. C.O. Porter Mach. Co.*, 672 F.2d 1240, 1247-1248 (5th Cir. 1982) (remanding for new trial where jury instructions “unclear and confusing”).

**B. Redesigned FaceTime Does Not Provide The Claimed “Indication.”**

Each asserted claim of the ’504 and ’211 patents requires a DNS system that “*indicat[es]*” whether the system “supports establishing a secure communication link.” Appx262(55:54-56); Appx402(57:43-46). This Court construed “secure communication link” as “a *direct* communication link that provides data security and anonymity.” *VirnetX I*, 767 F.3d at 1319. Thus, the claimed “indication” must indicate whether the system supports a “direct” communication link.

VirnetX alleged that the claimed “indication” in redesigned FaceTime was the “Accept Push” message sent by the FaceTime servers to the caller’s device. Appx1376; Appx1501; Appx2710. But since Apple removed the receiving

device's IP address from the Accept Push message in 2013, the Accept Push message does not indicate support for a *direct* communication link. Accordingly, no reasonable jury could find that redesigned FaceTime provides the claimed "indication."

**1. The Accept Push message does not "indicate" support for a "direct" communication link.**

As both experts agreed, establishing a FaceTime call via a "direct" communication link requires the caller's device to know the receiving device's IP address. *E.g.*, Appx1424(Jones) ("Q. ... [F]or the actual direct call to happen ... the caller has to get somehow the callee's IP address. Right? A. Yes. In any communication like that it will, yes."); Appx2321-2322(Blaze) ("Q. Without the IP address of the receiving phone, can the calling phone set up a direct peer-to-peer connection with the recipient? A. No.").

When Apple redesigned FaceTime in April 2013, it "zero[ed] out or effectively remove[d]" the receiving device's IP address from the Accept Push message. Appx1422(Jones); *see* Appx2298(Blaze). VirnetX's expert thus conceded that the Accept Push message in the April 2013 redesign "would *not* satisfy [the] requirements of the claims to be an indication of support." Appx1423.

VirnetX nevertheless contended that the Accept Push message somehow provided the claimed "indication" after Apple modified FaceTime again in September 2013. But even for that accused version, Dr. Jones agreed that "still the

accept push message did not have the IP address of the callee.” Appx1424; *see* Appx1493(Jones); Appx2318, Appx2321-2322(Blaze). Instead, it contains “the IP address of a relay server,” which can only be used to establish an *indirect* call. Appx1425; Appx1502-1503; *see* Appx1417(Jones) (agreeing that “a relay communication is indirect”).

While two communicating devices may themselves subsequently establish a direct communication link via the separate “ICE protocol,” that has nothing to do with the accused FaceTime servers or Accept Push message. Appx1425-1426(Jones); Appx2319, Appx2325-2326(Blaze). Importantly, Dr. Jones never explained how any information in the Accept Push message “indicates” that the accused FaceTime servers support establishing a *direct* communication link. *See* Appx1362; Appx1376-1378. On the contrary, he admitted “the caller *can’t initiate a direct FaceTime call* to the callee” “based on the contents of the [A]ccept [Push] message alone[.]” Appx2710-2711; *see* Appx2328-2329(Blaze) (“Every one of those things [in the Accept Push message] is useful for relay communication or for either relayed or indirect, but nothing in there indicates support for direct communication.”).

In short, no reasonable jury could find that the Accept Push message “indicates” that the accused FaceTime servers support establishing a *direct* communication link, as the claims require.

**2. The district court’s reasons for denying JMOL are legally incorrect and unsupported by substantial evidence.**

In denying JMOL, the district court treated the fact that the Accept Push message no longer contains the receiving device’s IP address as irrelevant, though it is central. The court reasoned that its claim construction, which requires “an indication *other than* the mere return of an IP address,” meant that “return of an IP address is not required.” Appx74 (emphasis in original).<sup>10</sup> That is a non sequitur. The fact that the claimed “indication” must do *more* than “merely” return an IP address does not somehow render the return of an IP address (or the failure to do so) irrelevant to determining whether there is an indication of support for a communication link that is direct, secure, and anonymous, *see VirnetX I*, 767 F.3d at 1319—particularly where all parties agreed that providing the receiving device’s IP address accomplishes the “direct” part of that requirement.

The district court’s cited evidence again fails to support its conclusion. Dr. Jones’s testimony that the Accept Push message represents the “culmination of a provisioning process” and contains tokens, certificates, and other information (Appx75 (quoting Appx1376-1379)) in no way shows that the Accept Push message indicates support for a *direct* communication link. *See supra* pp. 52-53.

---

<sup>10</sup> The district court construed the “indication” limitation as “an indication other than merely returning of requested DNS records, such as an IP address or key certificate, that the [DNS] system supports establishing a secure communication link.” Appx15049-15051.

Indeed, as Dr. Jones admitted, the Accept Push message in redesigned FaceTime indicates support for an *indirect* communication link (via a “relay”) by returning the IP address of the relay server, rather than the IP address of the receiving device. Appx1417; Appx1425.<sup>11</sup>

The district court dismissed the fact “[t]hat the accept push message can also be used to establish a relayed FaceTime call” on the ground that “[t]he addition of features does not avoid infringement.” Appx76 (quoting *Northern Telecom, Inc. v. Datapoint Corp.*, 908 F.2d 931, 945 (Fed. Cir. 1990)). Again, that is beside the point. It is not as though the evidence showed that the Accept Push message indicates support for a direct communication link and something else. The Accept Push message does not “indicate” support for a direct communication link at all, which is what the claims require.

### **III. THE DISTRICT COURT ERRED IN APPLYING ISSUE PRECLUSION TO APPLE’S INVALIDITY DEFENSES AND COUNTERCLAIMS.**

The district court correctly ruled that claim preclusion did not bar Apple’s invalidity arguments, because this case involves redesigned products that are not “essentially the same” as those at issue in the 417 Action. Appx5 (citing *Hallco*

---

<sup>11</sup> The district court cited Dr. Jones’s testimony that “IP addresses from both the caller and callee are not necessary to set up a direct call,” but that is immaterial. Appx74 (emphasis omitted) (citing Appx1482-1483). Dr. Jones conceded that at least the callee’s IP address is required to establish a direct call. Appx1424 (agreeing that “for the actual direct call to happen ... the caller has to get somehow the callee’s IP address”); Appx1482 (“[I]t just needs the address of that [receiving] device.”).

*Mfg. Co. v. Foster*, 256 F.3d 1290, 1294 (Fed. Cir. 2001)). However, the court erred in holding that issue preclusion barred Apple’s invalidity arguments. Appx9.

This Court reviews issue preclusion under the law of the regional circuit, applying Federal Circuit precedent “for any aspects that may have special or unique application to patent cases.” *Voter Verified*, 887 F.3d at 1382. In the Fifth Circuit, the party seeking issue preclusion must prove (among other things) that “the issue under consideration is identical to that litigated in the prior action.” *Copeland v. Merrill Lynch & Co.*, 47 F.3d 1415, 1422 (5th Cir. 1995); *see GLF Const. Corp. v. LAN/STV*, 414 F.3d 553, 555 n.2 (5th Cir. 2005). That requirement was not met here.

The district court ruled that invalidity is always a single “issue” for preclusion purposes, while noting that this Court “ha[d] not yet explicitly addressed this matter.” Appx6-7.

Since then, however, this Court has rejected the district court’s approach of treating all invalidity defenses as a single, undifferentiated “issue,” by holding that issue preclusion did not bar a defendant who lost invalidity challenges under §§ 102 and 103 from pursuing a § 101 challenge in a second case. *Voter Verified*, 887 F.3d at 1382-1383. *Voter Verified* necessarily rejected the single-issue approach to invalidity in concluding that the mere fact that a defendant lost one type of invalidity challenge did not bar it from pursuing another in a different case.



*Id.*; see *TASER Int’l, Inc. v. Karbon Arms, LLC*, 6 F. Supp. 3d 510, 519 (D. Del. 2013) (failure to prove invalidity does not demonstrate “the patent is valid for all time, but only that the accused infringer has failed to prove the patent invalid on the specific grounds it asserted”).

This rule—that, at a minimum, there is no issue preclusion for invalidity challenges raised under different statutory sections than the challenges already rejected—makes good sense. After all, patents are not held “valid,” only “not invalid” based on the specific grounds presented. *TASER*, 6 F. Supp. 3d at 519; *Ball Aerosol*, 555 F.3d at 994. Allowing defendants to raise meritorious invalidity arguments safeguards the public’s “paramount interest in seeing that patent monopolies are kept within their legitimate scope.” *Medtronic, Inc. v. Mirowski Family Ventures, LLC*, 571 U.S. 191, 203 (2014); see *Lear, Inc. v. Adkins*, 395 U.S. 653, 672-673 (1969). The district court’s rule, by contrast, would contravene the Supreme Court’s instruction that a patentee “should not be insulated from the assertion of defenses and thus allowed to exact royalties for the use of an idea that is not in fact patentable.” *Blonder-Tongue Labs., Inc. v. University of Ill. Found.*, 402 U.S. 313, 349-350 (1971).

The only invalidity defense Apple raised at trial in the 417 Action was anticipation by Kiuchi under 35 U.S.C. § 102. See Appx2. Accordingly, under *Voter Verified* and application of standard issue preclusion principles, it was error

to bar Apple's invalidity defenses—at least those arising under other statutory provisions, including obviousness under § 103. *See* Appx2; *see generally* Appx15001-15020.<sup>12</sup> The court's summary judgment on Apple's invalidity arguments should be reversed, the no-invalidity judgment for all four patents vacated, and the case remanded.

#### **IV. THE DISTRICT COURT ERRED IN DENYING JUDGMENT OF NON-INFRINGEMENT AS TO iMESSAGE.**

Throughout this litigation, VirnetX asserted that Apple's iMessage feature infringed the '504 and '211 patents, while Apple denied infringement and maintained a non-infringement counterclaim. Appx15073-15090; Appx15093. In 2016, a jury found that iMessage infringed (Appx15621), but that verdict was vacated. *See supra* p. 20. Just before the 2018 retrial, VirnetX unexpectedly decided not to present evidence relating to iMessage. When Apple subsequently moved for JMOL of non-infringement as to iMessage, the court refused, citing a lack of jurisdiction and the general policy against discouraging parties from narrowing a case for trial. Appx89. That was error.

The district court had jurisdiction to rule on iMessage. Neither VirnetX's infringement claim nor Apple's non-infringement counterclaim had been dismissed

---

<sup>12</sup> For example, Apple contended that the asserted claims were obvious based on the Altiga, Aventail, Beser, Aziz, SIP, Schulzrinne Presentation, and H.323 references. Apple also argued that the patents were invalid because they failed to name Dr. Schulzrinne as a co-inventor.

at the time of trial or Apple's JMOL motion. The pretrial order identified infringement by iMessage as an "issue[] of fact that remain[ed] to be litigated." Appx15762; Appx15728. VirnetX stated that iMessage "infringement is still live" at the pretrial conference one week before trial. Appx15952. Even after trial, VirnetX refused to stipulate to dismissal of its iMessage infringement claim with prejudice or provide Apple with a covenant not to sue. Appx16225-16226. That was sufficient to create declaratory judgment jurisdiction for Apple's counterclaim, as VirnetX's behavior could "be reasonably inferred as demonstrating intent to enforce [its] patent[s]." *Hewlett-Packard*, 587 F.3d at 1363.

That VirnetX chose at the last minute not to present any iMessage evidence at trial does not alter the analysis. Even in the declaratory judgment context, the patentee bears the burden of proving infringement. *Medtronic*, 571 U.S. at 198. So long as there is a "genuine dispute of sufficient immediacy and reality, about the patent's validity or its application," the purported infringer can "force the patentee into full blown patent-infringement litigation." *Id.* at 202-203 (internal quotation marks omitted).

The district court apparently thought VirnetX could avoid judgment simply because its infringement claims "were not presented for consideration to the jury" (Appx89), but the cited cases do not support that proposition. In one decision, this Court merely held that the district court lacked jurisdiction to decide invalidity for

patent claims that were not asserted to be infringed. *Strech, Inc. v. Research & Diagnostic Sys., Inc.*, 665 F.3d 1269, 1276, 1283-1284 (Fed. Cir. 2012). Indeed, *Strech* observed that courts have jurisdiction to consider declaratory judgment counterclaims where, as here, there is a “continuing case or controversy with respect to ... unasserted claims.” *Id.* at 1283; *see Alcon Research Ltd. v. Barr Labs., Inc.*, 745 F.3d 1180, 1193 (Fed. Cir. 2014) (“If an accused infringer has filed a counterclaim, then the patentee has notice that, even if it drops its infringement claims, the issue of infringement remains to be litigated.”). The remaining cited decisions all involved specific patent claims that were not litigated and therefore could not be included in judgments declaring the patents invalid generally—in other words, the patent challengers never proved those specific claims invalid.<sup>13</sup>

Here, VirnetX asserted specific patent claims against iMessage to verdict in the prior consolidated trial, and defined the scope of this trial as including iMessage, stating in the pretrial order: “VirnetX alleges ... that ... iMessage ha[s] infringed and continue[s] to infringe claims 1-2, 5, and 27 of the ’504 patent and

---

<sup>13</sup> *See Connell v. Sears, Roebuck & Co.*, 722 F.2d 1542, 1552 (Fed. Cir. 1983) (claim 2 was “not before the trial court, by way of testimony or otherwise, and ... not before this [C]ourt”); *Datascope Corp. v. SMEC, Inc.*, 776 F.2d 320, 327 (Fed. Cir. 1985) (“validity of claims 16-23 was not litigated”); *Novo Nordisk Pharms., Inc. v. Bio-Tech. General Corp.*, 424 F.3d 1347, 1352, 1356 (Fed. Cir. 2005) (invalidity of claim 2 “not litigated at trial”).

claims 36, 47, and 51 of the '211 patent.” Appx15750. In the Fifth Circuit, “a joint pretrial order signed by both parties supersedes all pleadings and governs the issues and evidence to be presented at trial.” *Branch-Hines v. Herbert*, 939 F.2d 1311, 1319 (5th Cir. 1991). Neither the law nor the policy of encouraging case narrowing prevented the district court from granting JMOL of non-infringement regarding iMessage. The district court erred in refusing to do so.

**V. THE JUDGMENT SHOULD BE VACATED IF THIS COURT CONCLUDES THAT EITHER SET OF PATENTS-IN-SUIT IS UNPATENTABLE AND/OR NOT INFRINGED.**

The PTO has found each asserted claim of VirnetX’s four patents-in-suit unpatentable in *inter partes* review and/or reexamination proceedings brought by Apple, Cisco, and others. Some of those proceedings are pending before this Court; others are still before the PTO. *See supra* pp. 1-3.

If this Court affirms the PTO’s unpatentability determinations for all four patents-in-suit, the judgment should be vacated and remanded for dismissal. *See Fresenius USA, Inc. v. Baxter Int’l, Inc.*, 721 F.3d 1330, 1340 (Fed. Cir. 2013) (“[W]hen a claim is cancelled, the patentee loses any cause of action based on that claim, and any pending litigation in which the claims are asserted becomes moot.”); *Translogic Tech., Inc. v. Hitachi Ltd.*, 250 F. App’x 988 (Fed. Cir. 2007) (non-precedential) (“In light of this court’s decision in *In re Translogic Tech., Inc.* [affirming unpatentability determination from reexamination], this court vacates

the district court's decision and remands this case to the district court for dismissal.").

Similarly, if the Court affirms the PTO's unpatentability determinations and/or reverses the infringement findings for the '135 and '151 patents (asserted against redesigned VPN On Demand) or the '504 and '211 patents (asserted against redesigned FaceTime), the district court's judgment should be vacated and remanded to determine the applicable damages, prejudgment interest, and ongoing royalties for the patents and infringement findings that remain. That is because the jury's damages award assumed that Apple infringed four valid patents; it did not differentiate on a patent-by-patent basis. Appx50-52. And although VirnetX's damages theory was based on a \$1.20 per-unit royalty, different numbers of accused units were accused of infringing the two sets of patents. Appx1853-1855.

## **VI. APPLE PRESERVES ITS ARGUMENTS ON THE ISSUES DECIDED AGAINST IT IN THE 417 ACTION.**

### **A. Redesigned FaceTime Does Not Provide "Anonymity."**

Each asserted claim of the '504 and '211 patents requires a DNS system that supports establishing a "secure communication link," which must provide "data security and anonymity." *VirnetX I*, 767 F.3d at 1317-1319. In its second appeal in the 417 Action, Apple sought reversal of the infringement judgment because original FaceTime does not provide "anonymity." Opening Br. 8-15, 25-26, 30-40, Reply Br. 2-13, *VirnetX, Inc v. Cisco Sys., Inc.*, No. 2018-1197 (Fed. Cir.).

Redesigned FaceTime operates in the same way with respect to “anonymity.” Appx15958-15963. Thus, the district court ruled that “issue preclusion attaches to Apple’s [non-infringement] argument” regarding “anonymity.” Appx76. Accordingly, if further review of the infringement issue in No. 2018-1197 leads to a decision that original FaceTime does not infringe because it does not provide “anonymity,” the infringement judgment for the ’504 and ’211 patents should likewise be reversed here.

**B. The Damages And Interest Awards Should Be Reversed Or Vacated.**

In its second appeal in the 417 Action, Apple also challenged VirnetX’s damages expert’s reasonable royalty methodology and the district court’s prejudgment interest award. Opening Br. 19-22, 24-28, 40-51, 67-68, Reply Br. 14-23, 31, *VirnetX Inc. v. Cisco Sys., Inc.*, No. 2018-1197 (Fed. Cir.). Mr. Weinstein and the district court committed the same errors here. Accordingly, if further review in No. 2018-1197 leads to a decision vacating the damages and/or prejudgment interest awards, the same result should apply here.

**1. Mr. Weinstein’s testimony should have been excluded under *Daubert*.**

A patent “for an improvement, and not for an entirely new machine or contrivance,” entitles the patentee only to the benefit springing from the innovation. *Garretson v. Clark*, 111 U.S. 120, 121-122 (1884). Thus, a reasonable

royalty “must be based on the incremental value that the patented invention adds to the end product. In short, apportionment.” *Commonwealth Sci. & Indus. Research Org. v. Cisco Sys., Inc.*, 809 F.3d 1295, 1301 (Fed. Cir. 2015) (internal quotation marks and citation omitted). And to ensure that what is valued encompasses only the patented invention as used in the accused products, damages testimony that relies on comparable licenses “**must** account for ... distinguishing facts” of those licenses. *Ericsson*, 773 F.3d at 1227.

Mr. Weinstein’s methodology (*see supra* pp. 23-24) did not reliably ensure that his \$1.20 per-unit rate was apportioned to reflect only the incremental value of the claimed inventions to Apple’s products. Although Mr. Weinstein testified “that the rates that VirnetX has established in its written [licensing] policy have been designed to ... ‘apportion,’” he admitted that the policy merely set a 1-2% rate based on the *entire market value* of the licensed products. Appx1804; *see* Appx1812-1813. When asked what analysis was done to ensure the licenses were apportioned to the value that the claimed inventions added to the licensed products, he answered: “I’m not certain.” Appx1884.

The district court believed Mr. Weinstein was not required to apportion VirnetX’s licensing policy to the specific facts of this case. Appx93-94. That cannot be right. Each agreement Mr. Weinstein relied upon licensed “at least 16 patents.” Appx1894-1895; *see* Appx1873-1874. And Apple’s accused devices



included computers, smartphones, and tablets that were “far more complex” and included “many more features” than the simple VOIP phones covered by five of the licenses. Appx1892-1893. Yet Mr. Weinstein did nothing to apportion down the rates he derived from those licenses to reflect only the value of VirnetX’s four patents in Apple’s complex products. *See Uniloc USA, Inc. v. Microsoft Corp.*, 632 F.3d 1292, 1318 (Fed. Cir. 2011).

Mr. Weinstein also failed to account for the differences between the prior VirnetX licenses and the hypothetical Apple-VirnetX license. The VirnetX licenses granted rights to much more intellectual property for a much longer period than the four patents and approximately four years at issue here, and five licenses covered much simpler products than Apple’s. Appx1873-1874; Appx1894-1895. Moreover, the six licenses were settlement agreements entered to avoid litigation costs. Appx1897; *see Rude v. Westcott*, 130 U.S. 152, 164 (1889) (“[m]any considerations other than the value of the improvements patented” may lead to settlement). Yet Mr. Weinstein made *no adjustments* to the per-unit rates to account for these differences. Appx1811-1812.

The district court found Mr. Weinstein’s factual summary of the VirnetX licenses sufficient to satisfy *Daubert* (Appx91-92), but it was not. A reliable damages methodology must not only describe the facts of the prior licenses but also “account” for the differences as compared to the hypothetical license—which

Mr. Weinstein did not do. *See Ericsson*, 773 F.3d at 1227. Indeed, the wide variation in the per-unit rates he derived from the six licenses—which ranged from \$0.19 to \$2.26 per unit (*see supra* p. 24)—confirms his methodology for isolating the patented technology’s value was arbitrary. VirnetX’s ability to convince a few small companies to agree to nominal license fees to avoid litigation costs does not transform Mr. Weinstein’s methodology into a reliable approach for valuing the claimed inventions.

By deferring to the jury on these issues (Appx14, Appx92-94), the district court abdicated its “gatekeeping” responsibilities under *Daubert v. Merrell Dow Pharmaceuticals, Inc.*, 509 U.S. 579, 589, 597 (1993). A court’s obligation to ensure that damages evidence is reliable includes requiring “the expert [to] ... apportion damages and sufficiently tie the royalty rate to the facts of the case.” *Exmark Mfg. Co. v. Briggs & Stratton Power Prods. Grp., LLC*, 879 F.3d 1332, 1351 (Fed. Cir. 2018). Mr. Weinstein’s methodology failed that test.

**2. Even if admissible, Mr. Weinstein’s testimony did not provide substantial evidence supporting the verdict.**

Though Mr. Weinstein claimed his royalty figure was apportioned, he testified that he only “apportioned down” to the *products* covered by the licenses he relied upon (VOIP phones for five licenses), not the patented inventions. Appx1883. He then claimed “there wasn’t any further apportionment necessary”

because he relied on VirnetX's licenses, but admitted he was "not certain" what was done to apportion the licenses to begin with. Appx1883-1884.

While an expert need not "convey all his knowledge to the jury about each license agreement in evidence," Mr. Weinstein needed to present "particularized ... testimony" explaining how his \$1.20 rate was apportioned and how to account for the "various differences" between prior licenses and the hypothetical license. *See Lucent*, 580 F.3d at 1329. Because Mr. Weinstein did neither, the verdict—which awarded his full damages demand—cannot stand.

**3. The district court abused its discretion in awarding \$93 million in prejudgment interest.**

The district court's prejudgment interest award (Appx16317-16318) was inappropriate because the jury's award already gives VirnetX "complete compensation" under 35 U.S.C. § 284. *See General Motors Corp. v. Devex Corp.*, 461 U.S. 648, 655 (1983). Moreover, the court's method of calculating interest "compounded annually, beginning at the date of the hypothetical negotiation" (Appx117) erroneously computed interest as if the jury had awarded a lump sum payable at the time of the hypothetical negotiation, although VirnetX's damages theory was based on a per-unit royalty. Consequently, Apple owes interest going back to September 2013 (start of damages period) even for products not sold until March 2018 (end of damages period). Appx16180(¶5).

The interest period was also unfairly extended due to errors invited by VirnetX, including VirnetX's prejudicial statements in the consolidated trial that required a retrial. *See supra* p. 20. Interest should not have been awarded from February 4, 2016 (day after consolidated-trial verdict) to April 2, 2018 (first day of 855 Action retrial). *See Akamai Techs., Inc. v. Limelight Networks, Inc.*, No. 06-cv-11109 (D. Mass. June 24, 2016), ECF No. 518 at 5. Awarding interest for all five years is impermissibly punitive. *See Oiness v. Walgreen Co.*, 88 F.3d 1025, 1033 (Fed. Cir. 1996).

### CONCLUSION

The judgment should be reversed or, alternatively, vacated and remanded.

Respectfully submitted,

BRITTANY BLUEITT AMADI  
WILMER CUTLER PICKERING  
HALE AND DORR LLP  
1875 Pennsylvania Avenue NW  
Washington, DC 20006  
(202) 663-6000

THOMAS G. SPRANKLING  
WILMER CUTLER PICKERING  
HALE AND DORR LLP  
950 Page Mill Road  
Palo Alto, CA 94304  
(650) 858-6000

January 24, 2019

/s/ William F. Lee  
WILLIAM F. LEE  
MARK C. FLEMING  
LAUREN B. FLETCHER  
WILMER CUTLER PICKERING  
HALE AND DORR LLP  
60 State Street  
Boston, MA 02109  
(617) 526-5000

*Attorneys for Defendant-Appellee  
Apple Inc.*

# **ADDENDUM**

## TABLE OF CONTENTS

	<b>Page(s)</b>
Order Granting in Part VirnetX’s Motion For Partial Summary Judgment on Apple’s Invalidity Counterclaims Asserted In Prior Litigation, Dkt. No. 181 (Aug. 8, 2014) .....	Appx1-10
Order regarding Motions for Summary Judgment and Daubert Motions, Dkt. No. 362 (Jan. 11, 2016) .....	Appx11-15
Memorandum Opinion re Dkt. 362, Order regarding Motions for Summary Judgment and Daubert Motions, Dkt. No. 468 (Mar. 22, 2016) .....	Appx16-25
Final Jury Instructions - Phase I, Dkt. No. 721 (Apr. 12, 2018) .....	Appx26-49
Jury Verdict - Phase I, Dkt. No. 722 (Apr. 12, 2018).....	Appx50-52
Memorandum Opinion & Order on Post-Trial Motions, Dkt. No. 798 (Aug. 20, 2018) .....	Appx66-118
Final Judgment, Dkt. No. 801 (Aug. 30, 2018) .....	Appx119
U.S. Patent No. 6,502,135.....	Appx120-191
U.S. Patent No. 7,418,504.....	Appx192-264
U.S. Patent No. 7,490,151.....	Appx265-326
U.S. Patent No. 7,921,211.....	Appx327-403

**IN THE UNITED STATES DISTRICT COURT  
FOR THE EASTERN DISTRICT OF TEXAS  
TYLER DIVISION**

**VIRNETX INC. and SCIENCE APPLICATIONS INTERNATIONAL CORPORATION,**

**Plaintiffs,**

**VS.**

**APPLE, INC.,**

**Defendant.**



**CASE NO. 6:12-CV-855**

## ORDER

Before the Court is VirnetX, Inc. and Science Applications International Corp.’s (collectively, “VirnetX”) Motion for Partial Summary Judgment on Apple, Inc.’s (“Apple”) Invalidity Counterclaims Asserted in the Prior Litigation (Docket No. 149). The Court heard arguments regarding this Motion on May 20, 2014. Based on the parties’ briefings and arguments, the Motion is **GRANTED IN PART** and **DENIED IN PART**.

## BACKGROUND

This is the second case between VirnetX and Apple. The first case, “*Apple I*,” was filed on August 11, 2010. *VirnetX, Inc. v. Cisco Systems, Inc., et al.*, No. 6:10-cv-417 (E.D. Tex. Aug. 11, 2010).<sup>1</sup> In *Apple I*, VirnetX accused two Apple product features of infringement: FaceTime and VPN On Demand. VirnetX originally asserted ninety claims from four patents.<sup>2</sup> Apple

<sup>1</sup> Cisco Systems, Inc. was Apple's co-defendant in *Apple I*. The Court separated the defendants for trial. *Apple I*, Docket No. 542.

<sup>2</sup> VirnetX asserted U.S. Patent Nos. 6,502,135 (“the ‘135 Patent”), 7,418,504 (“the ‘504 Patent”), 7,490,151 (“the ‘151 Patent”), and 7,921,211 (“the ‘211 Patent”). The ‘135 and ‘151 Patents generally describe a method of transparently creating a virtual private network (“VPN”) between a client computer and a target computer, while the ‘504 and ‘211 Patents disclose a secure domain name service.

originally asserted several theories of invalidity, including anticipation, obviousness, failure to comply with the written description requirement, derivation, and non-joinder.

The *Apple I* case proceeded to trial on October 31, 2012. As it does in many complex patent cases, the Court in *Apple I* encouraged and required the parties to narrow their cases for trial. Accordingly, VirnetX only presented sixteen patent claims at trial.<sup>3</sup> Similarly, the only invalidity theory Apple presented was anticipation based on a 1996 publication by Takahiro Kiuchi (the “Kiuchi reference”). However, the narrowing of Apple’s case was only partially voluntary. Prior to trial, the Court disposed of Apple’s derivation and non-joinder invalidity theories when it granted a motion for summary judgment filed by VirnetX. *Apple I*, Docket No. 555.

Following a five-day trial, the jury in *Apple I* found that the four asserted patents were not invalid and that Apple infringed the sixteen asserted claims. It awarded VirnetX \$368,160,000 to compensate for Apple’s infringement. The Court entered judgment on the jury’s verdict. *Apple I*, Docket No. 732.

On the same day the jury reached a verdict in *Apple I*, VirnetX filed this action. In this case, VirnetX accuses the re-designed versions of the FaceTime and VPN on Demand features accused in *Apple I*, plus two features that were not at issue in the prior litigation: Per App VPN and iMessage. VirnetX originally asserted the same four patents as in *Apple I*, and later amended its complaint to assert two additional patents.<sup>4</sup> Docket Nos. 1, 58, 75. Apple contends that the asserted patents in this case are invalid, including the four patents asserted in *Apple I*. It no longer asserts anticipation based on the Kiuchi reference, which it presented at trial in *Apple I*,

---

<sup>3</sup> At the *Apple I* trial, VirnetX presented claims 1, 3, 7, 8 of the ’135 Patent; claims 1 and 13 of the ’151 Patent; claims 1, 2, 5, 16, 21, and 27 of the ’504 Patent; and claims 36, 37, 47 and 51 of the ’211 Patent.

<sup>4</sup> VirnetX added U.S. Patent Nos. 8,015,181 (“the ’181 Patent”) and 8,504,697 (“the ’697 Patent”). The ’181 Patent discloses a method of establishing a secure communication link, while the ’697 Patent discloses a method of communicating between network devices.



but reasserts other invalidity theories that it did not present at trial, including the derivation and non-joinder invalidity theories that the Court dismissed on summary judgment. VirnetX filed the current motion, requesting the Court to rule that issue and claim preclusion bar Apple's invalidity defenses in this case.

### **APPLICABLE LAW**

Summary judgment shall be rendered when the pleadings, depositions, answers to interrogatories, and admissions on file, together with the affidavits, if any, show that there is no genuine issue as to any material fact and that the moving party is entitled to judgment as a matter of law. FED. R. CIV. P. 56(c); *Celotex Corp. v. Catrett*, 477 U.S. 317, 323–25 (1986); *Ragas v. Tenn. Gas Pipeline Co.*, 136 F.3d 455, 458 (5th Cir. 1998). An issue of material fact is genuine if the evidence could lead a reasonable jury to find for the non-moving party. *Anderson v. Liberty Lobby, Inc.*, 477 U.S. 242, 248 (1986). In determining whether a genuine issue for trial exists, the court views all inferences drawn from the factual record in the light most favorable to the nonmoving party. *Id.*; *Matsushita Elec. Indus. Co. v. Zenith Radio*, 475 U.S. 574, 587 (1986).

If the moving party has made an initial showing that there is no evidence to support the nonmoving party's case, the party opposing the motion must assert competent summary judgment evidence of the existence of a genuine fact issue. *Matsushita*, 475 U.S. at 586. Mere conclusory allegations, unsubstantiated assertions, improbable inferences, and unsupported speculation are not competent summary judgment evidence. *See Eason v. Thaler*, 73 F.3d 1322, 1325 (5th Cir. 1996); *Forsyth v. Barr*, 19 F.3d 1527, 1533 (5th Cir. 1994). The party opposing summary judgment is required to identify evidence in the record and articulate the manner in which that evidence supports his claim. *Ragas*, 136 F.3d at 458. “Only disputes over facts that

might affect the outcome of the suit under the governing laws will properly preclude the entry of summary judgment.” *Anderson*, 477 U.S. at 248. Summary judgment must be granted if the nonmoving party fails to make a showing sufficient to establish the existence of an element essential to its case and on which it will bear the burden of proof at trial. *Celotex*, 477 U.S. at 322–23.

## ANALYSIS

### I. Claim Preclusion (*Res Judicata*)

VirnetX contends that claim preclusion bars Apple’s invalidity defenses to any of the claims previously asserted in the *Apple I* case, including the seventy-four claims not presented at trial. The Fifth Circuit applies claim preclusion where: “(1) the parties are identical in the two actions; (2) the prior judgment was rendered by a court of competent jurisdiction; (3) there was a final judgment on the merits; and (4) the same claim or cause of action is involved in both cases.” *Oreck Direct, LLC v. Dyson, Inc.*, 560 F.3d 398, 401 (5th Cir. 2009) (citation omitted).

The parties dispute mainly concerns the fourth element of claim preclusion, whether the same claim or cause of action is involved in both cases.<sup>5</sup> VirnetX argues that Apple is precluded from asserting invalidity in this case because Apple asserted invalidity in *Apple I*. Apple does not dispute that it raised an invalidity defense in *Apple I*. However, it contends that its invalidity defense is not precluded because VirnetX’s infringement claims are different in this case. The parties’ dispute centers on what qualifies as a “claim” to be precluded. VirnetX contends an invalidity defense qualifies as a separate “claim” for the purposes of claim preclusion; Apple argues the “claim” is based on infringement allegations.

---

<sup>5</sup> The parties also dispute the second element of claim preclusion, whether the prior judgment was rendered by a court of competent jurisdiction. However, for the reasons discussed herein, the Court does not need to reach the parties’ dispute with respect to the second element.

Although regional circuit law governs claim preclusion generally, the question of whether an invalidity defense qualifies as a separate “claim” for the purposes of claim preclusion is “particular to patent law” and is to be decided based on the law of the U.S. Court of Appeals for the Federal Circuit. *Hallco Mfg. Co., Inc. v. Foster*, 256 F.3d 1290, 1294 (Fed. Cir. 2001). The Federal Circuit has held that “[a]n assertion of invalidity of a patent by an alleged infringer is not a ‘claim’ but a defense to the patent owner’s ‘claim.’” *Foster v. Hallco Mfg. Co., Inc.*, 947 F.2d 469, 479 (Fed. Cir. 1991). “[T]he right to pursue the invalidity defense in later litigation between the parties . . . depends on whether the underlying cause of action is different from the one brought earlier, which in turn depends on whether the [ ] devices [in the two cases] are essentially the same, or if any differences between them are merely colorable.” *Hallco v. Foster*, 256 F.3d at 1297.

The accused devices in this case are Per App VPN, iMessage, FaceTime, and VPN on Demand. Per App VPN and iMessage were not at issue in *Apple I*. FaceTime and VPN on Demand have been re-designed since the prior case. Thus, the accused features in this and the prior case are not “essentially the same.” Therefore, this case does not present the same “claim” as in *Apple I*.

Accordingly, claim preclusion does **NOT PRECLUDE** Apple from raising invalidity, under any theory, as a defense against any of the patent claims asserted in *Apple I*.

## **II. Issue Preclusion (*Collateral Estoppel*)**

Additionally, VirnetX asserts that issue preclusion bars Apple’s invalidity defenses to the sixteen claims presented during the *Apple I* trial. A party is estopped from relitigating an issue when “(1) the issue under consideration is identical to that litigated in the prior action; (2) the issue was fully and vigorously litigated in the prior action; (3) the issue was necessary to support

the judgment in the prior case; and (4) there is no special circumstance that would make it unfair to apply the doctrine.” *Copeland v. Merrill Lynch & Co., Inc.*, 47 F.3d 1415, 1422 (5th Cir. 1995). The judgment must be final to preclude relitigation of an issue. *Harvey Specialty & Supply, Inc. v. Anson Flowline Equip. Inc.*, 434 F.3d 320, 323 (5th Cir. 2005).

**a. Invalidity Defenses Against Previously Tried Claims**

The parties dispute the first element of issue preclusion, whether an identical issue exists. VirnetX argues that Apple is precluded from asserting invalidity in this case because that issue was decided in *Apple I*. As noted in the previous section, Apple does not dispute that it raised an anticipation defense at trial in *Apple I*. Docket No. 155 at 3. However, it claims that the invalidity theories it asserts in this case are not precluded because they are different from the invalidity theory it tried before the *Apple I* jury. The parties’ dispute centers on the “issue” to be given preclusive effect. VirnetX contends patent invalidity is a single “issue” for preclusion purposes; Apple claims each invalidity theory is a separate “issue.”

Although regional circuit law governs issue preclusion generally, the question of whether invalidity is a single “issue” for preclusion purposes is “particular to patent law” and is to be decided based on the law of the U.S. Court of Appeals for the Federal Circuit. *Applied Med. Res. Corp. v. U.S. Surgical Corp.*, 435 F.3d 1356, 1359–60 (Fed. Cir. 2006); *Evonik Degussa GmbH v. Materia Inc.*, No. 9-cv-636, 2014 WL 2967653, at \*11 (D. Del. June 30, 2014); *see Hallco v. Foster*, 256 F.3d at 1294. However, the Federal Circuit has not yet explicitly addressed this matter. *Evonik*, 2014 WL 2967653, at \*11. To support its proposition, Apple cites a recent case from the District of Delaware stating that “each theory of invalidity is a separate issue.” *TASER Int’l, Inc. v. Karbon Arms, LLC*, No. 11-cv-426, 2013 WL 6705149, at \*7 (D. Del. Dec. 19, 2013). However, “the overwhelming weight of authority suggests that the

‘issue’ that is to be given issue-preclusive effect to a judgment in the patent context is the ultimate determination on patent validity itself, not the sub-issues or the individual pieces of evidence and arguments that may have been necessary to support the validity determination.” *Crossroads Sys. (Texas), Inc. v. Dot Hill Sys. Corp.*, No. A-03-CA-754-SS, 2006 WL 1544621, at \*5 (W.D. Tex. May 31, 2006); *accord Evonik*, 2014 WL 2967653, at \*12 (holding that “validity is a single issue” for preclusion purposes). In this case, Apple is again contesting the validity of the ’135, ’504, ’151, and ’211 Patents. Since this validity dispute is identical to the issue raised in *Apple I*, the first issue preclusion element is satisfied.

The second element of issue preclusion is not in dispute. Apple concedes that its anticipation defense based on Kiuchi was fully litigated in *Apple I*. Docket No. 155 at 3. The third element of issue preclusion is also satisfied. A finding that the claims presented during the *Apple I* trial were not invalid was necessary to the Court’s judgment entered against those claims. *See Apple I*, Docket No. 732.

The parties dispute the final element of issue preclusion, whether a special circumstance would make it unfair to apply the doctrine. Apple argues that precluding its untried invalidity defenses would be unfair because it had to narrow its case for trial in *Apple I*. Docket No. 155 at 14.<sup>6</sup> However, Apple was not the only party encouraged to narrow its case. Both VirnetX and Apple were encouraged to narrow their cases for the *Apple I* trial and both parties voluntarily did so. Narrowing a case for trial involves strategic risk that the parties will choose unwisely from among their multiple claims and defenses. VirnetX originally asserted ninety claims but reduced that number to sixteen for trial. Claim preclusion now bars VirnetX from asserting, against the

---

<sup>6</sup> Apple also notes that VirnetX has proposed claim constructions that would broaden some of the claims asserted in *Apple I* and this case. Thus, it argues, if the Court adopted these broader constructions, then issue preclusion would not apply to the broadened claims. Docket No. 155 at 15. However, Apple’s concern is moot because the Court’s claim construction order does not adopt VirnetX’s broader proposed constructions for the claims previously construed in *Apple I*.

same allegedly infringing conduct, the seventy-four claims that it dropped before trial. *See Brain Life, LLC v. Elekta Inc.*, 746 F.3d 1045, 1053 (Fed. Cir. 2014) (holding that the unopposed dismissal of patent claims without prejudice and entry of final judgment precluded subsequent assertion of those patent claims against the same allegedly infringing conduct). Likewise, Apple originally asserted several theories of invalidity but only presented an anticipation defense to the jury. However, Apple now asserts that it would be unfair to preclude its foregone *Apple I* defenses against the same patent claims in this subsequent litigation. On the contrary, it would be unfair if these defenses were not precluded. According to Apple's theory, plaintiffs would bear all of the risk inherent in narrowing a complex patent case in order to make trial practicable. The Court will not make an exception and require only one party to bear this burden. Accordingly, there is no special circumstance which would make it unfair to apply issue preclusion to Apple's asserted invalidity theories.

Alternatively, Apple argues that a stipulation between the parties during *Apple I* preserves its invalidity contentions. The stipulation recounts that during the pendency of that case, Apple released a new feature called iMessage. *Apple I*, Docket No. 551 at 1. Given the timing in the *Apple I* case, VirnetX and Apple agreed that iMessage would not be an accused feature in that case, but that VirnetX could assert claims against iMessage in a future litigation.

*Id.* The stipulation goes on to state:

Nothing in this agreement affects any other rights that Apple has to assert its affirmative defenses and counterclaims with respect to iMessage or any VirnetX patent that may be asserted against Apple.

*Id.*

This section of the stipulation does not grant Apple rights it would not otherwise have—namely, a second attempt at asserting invalidity. If the parties had wished to do so, they could

have stipulated that the agreement preserved Apple's affirmative defenses and counterclaims to be reasserted in a later case. Instead, the stipulation states that "[n]othing in th[e] agreement affects" those affirmative defenses and counterclaims. *Id.* Because the stipulated agreement does not affect Apple's invalidity contentions, it does not preserve them.

Accordingly, Apple is **PRECLUDED** from asserting invalidity in this litigation against the patent claims that were tried before a jury in *Apple I*.

#### **b. Derivation and Non-Joinder Invalidity Defenses Against Untried Claims**

VirnetX argues that issue preclusion also bars VirnetX's derivation and non-joinder invalidity defenses to the seventy-four claims not presented at the *Apple I* trial. However, the Court refused to enter judgment on the claims and defenses not presented to the jury in that case. *Apple I*, Docket No. 732. Thus, the validity of the seventy-four untried claims was not necessary to the *Apple I* judgment. *See* 35 U.S.C. § 282(a) ("Each claim of a patent . . . shall be presumed valid independently of the validity of other claims . . . ."). Accordingly, Apple is **NOT PRECLUDED** from raising derivation and non-joinder defenses against the patent claims that were not tried before a jury in *Apple I*.

### **CONCLUSION**

VirnetX's Motion for Partial Summary Judgment on Apple's Invalidity Counterclaims Asserted in the Prior Litigation (Docket No. 149) is **GRANTED IN PART** and **DENIED IN PART**. Although not barred under claim preclusion, the doctrine of issue preclusion **PRECLUDES** Apple from asserting invalidity as a defense against infringement of the claims that were tried before a jury in *Apple I*.<sup>7</sup> Apple is **NOT PRECLUDED** from asserting

---

<sup>7</sup> At the *Apple I* trial, VirnetX presented claims 1, 3, 7, 8 of the '135 Patent; claims 1 and 13 of the '151 Patent; claims 1, 2, 5, 16, 21, and 27 of the '504 Patent; and claims 36, 37, 47 and 51 of the '211 Patent.

invalidity, under any theory, as a defense against infringement of the claims that were not tried before a jury in *Apple I*.

**So ORDERED and SIGNED this 8th day of August, 2014.**

A handwritten signature in black ink, appearing to read 'Leonard Davis', written over a horizontal line.

**LEONARD DAVIS  
UNITED STATES DISTRICT JUDGE**



**IN THE UNITED STATES DISTRICT COURT  
FOR THE EASTERN DISTRICT OF TEXAS  
TYLER DIVISION**

**VIRNETX INC. AND SCIENCE  
APPLICATIONS INTERNATIONAL  
CORPORATION,**

**Plaintiffs,**

**vs.**

**APPLE INC.,**

**Defendant.**

§  
§  
§  
§  
§  
§  
§  
§  
§  
§  
§  
§

**CASE NO. 6:12-CV-855**

**ORDER**

Before the Court are the following motions:

- Plaintiff VirnetX, Inc.’s (“VirnetX”) Motion to Compel Responses to VirnetX’s Interrogatory No. 7 (Docket No. 184);
- VirnetX’s Motion to Compel Document Production (Docket No. 194);
- Defendant Apple Inc.’s (“Apple”) Motion for Partial Summary Judgment of Noninfringement by FaceTime (Docket No. 315);
- VirnetX’s Motion to Strike Portions of the Opinion and Testimony of Mr. Christopher Bakewell (Docket No. 316);
- Apple’s Motion to Exclude the Expert Opinion of Dr. Mark Jones (Docket No. 317);
- VirnetX’s Motion to Exclude Portions of Dr. Matthew Blaze’s Invalidity Report (Docket No. 318);
- VirnetX’s Motion to Strike Portions of the Expert Report of Mr. James T. Carmichael (Docket No. 319);
- VirnetX’s Motion for Partial Summary Judgment of No Invalidity on Dependent Claims of Previously Tried Claims (Docket No. 320);
- VirnetX’s Motion for Partial Summary Judgment of No Invalidity Based on Derivation and Non-Joinder Theories (Docket No. 321);
- VirnetX’s Motion for Summary Judgment of No Inequitable Conduct (Docket No. 322);

- Apple's Motion to Exclude the Expert Damages Opinions of Mr. Roy Weinstein (Docket No. 323); and
- Apple's Motion to Strike VirnetX's Summary Judgment Briefing (Docket No. 326).

On January 7, 2016, the Court heard oral arguments regarding a variety of these motions. Based on the parties' briefing and argument, the Court rules as follows.

The Court **DENIES** VirnetX's Motion to Compel Responses to VirnetX's Interrogatory No. 7 (Docket No. 184). Apple is **ORDERED** to make the 30(b)(6) witness, which was offered during the hearing, available for deposition at VirnetX's convenience. To the extent that this deposition does not allow VirnetX to adequately test the details of a non-infringing alternative, and a witness for Apple discusses that non-infringing alternative at trial, VirnetX may request the Court's permission to ask an Apple witness about previous misrepresentation concerning the non-infringing alternatives in Case No. 6:10-cv-417.

The Court **DENIES** VirnetX's Motion to Compel Document Production (Docket No. 194). However, the Court will carefully consider any objections from VirnetX where Apple criticizes a VirnetX witness for not considering specific usage sought by VirnetX in this motion.

The Court **DENIES** Apple's Motion for Partial Summary Judgment of Noninfringement by FaceTime (Docket No. 315), with opinion to follow. Apple has not shown the absence of a genuine issue of material fact as to whether FaceTime infringes the asserted patents.

The Court **GRANTS** VirnetX's Motion to Strike Portions of the Opinion and Testimony of Mr. Christopher Bakewell (Docket No. 316), with opinion to follow. Mr. Bakewell's new method of calculating damages does not sufficiently relate to the consolidation of Case Nos. 6:10-cv-417 and 6:12-cv-855.

The Court **DENIES** Apple's Motion to Exclude the Expert Opinion of Dr. Mark Jones (Docket No. 317). Although Apple presents valid criticisms of Dr. Jones's opinions, they go to the weight of the evidence rather than admissibility.

The Court **DENIES** VirnetX's Motion to Exclude Portions of Dr. Matthew Blaze's Invalidity Report (Docket No. 318). With respect to Dr. Blaze's invalidity defenses for the asserted claims of Case No. 6:10-cv-417, Apple states it will not present an invalidity defense for these previously tried claims. In addition, Dr. Blaze's opinion on conception of the invention is admissible.

At the hearing, VirnetX withdrew its Motion to Strike Portions of the Expert Report of Mr. James T. Carmichael (Docket No. 319). Therefore, this motion is **DENIED AS MOOT**.

The Court **GRANTS-IN-PART** and **DENIES-IN-PART** VirnetX's Motion for Partial Summary Judgment of No Invalidity on Dependent Claims of Previously Tried Claims (Docket No. 320), with opinion to follow. This Motion is **GRANTED** with respect to Apple's anticipation and obviousness defenses and **DENIED** as to the derivation and non-joinder defenses.

The Court **DENIES** VirnetX's Motion for Partial Summary Judgment of No Invalidity Based on Derivation and Non-Joinder Theories (Docket No. 321). When viewing the facts in the light most favorable to Apple, the Schulzrinne Presentation creates a question of fact as to whether the named inventors of the asserted patents derived their invention from Dr. Henning Schulzrinne.

The Court **GRANTS** VirnetX's Motion for Summary Judgment of No Inequitable Conduct (Docket No. 322), with opinion to follow. Based on the evidence presented, the single

most reasonable inference is not that Mr. Toby Kusmer had a specific intent to deceive the U.S. Patent and Trademark Office.

The Court **DENIES** Apple's Motion to Exclude the Expert Damages Opinions of Mr. Roy Weinstein (Docket No. 323). Although Apple presents valid criticisms of Mr. Weinstein's opinions, they go to the weight of the evidence rather than its admissibility. The Court will be in a better position to evaluate Apple's criticisms, including how VirnetX uses the disputed survey, during Mr. Weinstein's testimony.

The Court **DENIES** Apple's Motion to Strike VirnetX's Summary Judgment Briefing (Docket No. 326). However, the Court is concerned with VirnetX's 29-page motion for summary judgment of no inequitable conduct. In this instance, VirnetX did not file a motion requesting additional pages to brief its motion as it did for its motion *in limine* responses (Docket No. 314). VirnetX states that it did not need to request leave to file additional pages because its motion for summary judgment of no inequitable conduct should be considered case dispositive pursuant to Local Rule CV-7. This argument should have been made prior to, or at least concurrent with, VirnetX filing its motion for summary judgment of no inequitable conduct. By filing the 29-page motion as is, VirnetX prevented the Court from ordering it to simply re-file the motion within the required page limits. Similarly, Apple was essentially forced to concede to VirnetX's interpretation of the Local Rules because of the short briefing schedule (*see* Docket No. 340). Although striking VirnetX's summary judgment briefing is too severe in this situation, the Court **ORDERS** the parties to meet and confer to determine an appropriate remedy. The parties shall file a joint proposal detailing their positions on a remedy by **January 15, 2016 by 5:00 p.m.**

**SIGNED this 11th day of January, 2016.**

A handwritten signature in cursive script, reading "Robert W. Schroeder III", followed by a small circular mark.

ROBERT W. SCHROEDER III  
UNITED STATES DISTRICT JUDGE

**IN THE UNITED STATES DISTRICT COURT  
FOR THE EASTERN DISTRICT OF TEXAS  
TYLER DIVISION**

**VIRNETX INC. AND SCIENCE  
APPLICATIONS INTERNATIONAL  
CORPORATION,**

**Plaintiffs,**

**VS.**

**APPLE INC.,**

**Defendant.**

§ § § § § § § § § § § § § §

**CASE NO. 6:12-CV-855**

## MEMORANDUM OPINION

On January 7, 2016, the Court heard oral argument on various motions. This document provides the written opinion of the Court’s prior rulings on January 11, 2016. *See* Docket No. 362. This opinion addresses: (1) Defendant Apple Inc.’s (“Apple”) denied Motion for Partial Summary Judgment of Noninfringement by FaceTime (Docket No. 315); (2) Plaintiff VirnetX Inc.’s (“VirnetX”) granted-in-part and denied-in-part Motion for Partial Summary Judgment of No Invalidity on Dependent Claims of Previously Tried Claims (Docket No. 320); (3) VirnetX’s granted Motion for Summary Judgment of No Inequitable Conduct (Docket No. 322); and (4) VirnetX’s granted Motion to Strike Portions of the Opinion and Testimony of Mr. Christopher Bakewell (Docket No. 316).

**1. Apple’s Motion for Partial Summary Judgment of Noninfringement by FaceTime (Docket No. 315)**

Summary judgment shall be rendered when there is no genuine issue of material fact and the moving party is entitled to judgment as a matter of law. FED. R. CIV. P. 56(c); *Celotex Corp. v. Catrett*, 477 U.S. 317, 322–25 (1986); *Ragas v. Tenn. Gas Pipeline Co.*, 136 F.3d 455, 458 (5th Cir. 1998). An issue of material fact is genuine if the evidence could lead a reasonable jury

to find for the non-moving party. *Anderson v. Liberty Lobby, Inc.*, 477 U.S. 242, 248 (1986). In determining whether a genuine issue of fact exists, a court views all inferences drawn from the factual record in the light most favorable to the nonmoving party. *Matsushita Elec. Indus. Co., Ltd. v. Zenith Radio Corp.*, 475 U.S. 574, 587 (1986).

In a summary judgment motion, Apple argued that the FaceTime feature does not infringe because it is not anonymous as required by the claim term “secure communication link.” Docket No. 352 at 1; *see VirnetX, Inc. v. Cisco Sys., Inc.*, 767 F.3d 1308, 1319 (Fed. Cir. 2014) (construing “secure communication link” as “a direct communication link that provides data security and anonymity”). Apple first explained that, in the specifications of the patents asserted against the FaceTime feature, the preferred embodiment requires “anonymity” by describing a first layer of obfuscation for content and a second layer of obfuscation for source and destination Internet Protocol (“IP”) addresses. Docket No. 352 at 1. Apple concluded that FaceTime is not anonymous because it does not conceal IP addresses as described in the patent specifications. Docket No. 315 at 1. Apple stated that VirnetX incorrectly interpreted “anonymity” as the inability to “correlate” a person or machine to an IP address, instead of as “concealment of source and designation IP addresses.” Docket No. 352 at 5.

Apple effectively asked the Court to further construe a “secure communication link” as implementing a particular process of providing anonymity. *See* Docket No. 315 at 4–7. The particular examples of providing anonymity to a communication link disclosed in the patent specifications should not limit the claims. *See VirnetX*, 767 F.3d at 1319. Based on how the FaceTime feature operates, a jury determined what degree of anonymity is sufficient to infringe the claims. Therefore, a genuine issue of material fact existed as to whether the FaceTime feature satisfied the “anonymity” requirement of the asserted claims.

Apple further stated that Network Address Translations (“NATs”), which were relied on by VirnetX in one of its two “anonymity” theories, are not part of the FaceTime feature.<sup>1</sup> Docket No. 352 at 1–3. The only specific argument that Apple identified as support for NATs being distinct from the FaceTime feature is third party control. *Id.* at 1–2. Apple described a NAT as a “new device.” *Id.* at 2. However, the asserted claims are not directed to a single device. *E.g.*, U.S. Patent No. 7,921,211 (“the ’211 Patent”) at claim 1 (claiming a system). In addition, Apple did not provide support of its position that the introduction of another component, which is not under Apple’s control, negates infringement of the FaceTime feature. *See* Docket No. 352 at 2.

Apple next argued that NATs do not provide the necessary “anonymity” because private and public IP addresses are the same; however, Apple did not explain in what respects the IP addresses are the same. *Id.* at 3. Further, Apple did not claim that the IP addresses are identical, and a description of an IP address as public or private appears to provide some meaning as to how it operates. *See* Docket No. 336 at 4.

Apple also stated that NATs do not provide anonymity because a communication link contains a participant’s private IP address before it interacts with a NAT. Docket No. 352 at 3–4. During this window before a communication reaches a NAT, the participant’s private IP address is allegedly accessible by eavesdroppers. *Id.* VirnetX retorted that, when eavesdroppers intercept packets of an ongoing FaceTime call between participating devices located behind NATs (*i.e.*, after the packets reach the NATs), eavesdroppers cannot correlate a device to a participant. *See* Docket No. 336 at 4, n.1. A reasonable jury could have found that the IP address conversion performed by a NAT early in the communication’s path is sufficient to establish anonymity.

---

<sup>1</sup> In addition, Apple disagreed with VirnetX’s characterization of anonymous because it would encompass NAT technology that was invented before the asserted patents. Docket No. 352 at 4. This is an invalidity position, which is unrelated to noninfringement.



Apple also shed doubt on VirnetX's second basis for "anonymity" within the FaceTime feature—the call setup process establishing "anonymity" of a communication. Apple stated that any anonymity established during the call setup process is irrelevant because it is the secure communication link that must be anonymous. Docket No. 352 at 4–5. VirnetX responded that the call setup process creates a secure communication link for the remainder of the communication. *Id.* Drawing all inferences in the light most favorable to VirnetX, a reasonable jury could have found that the call setup process establishes anonymity.

Apple finally argued that the construction of "domain name service system" incorporates the Court's construction of "domain name service." Docket No. 365 at 54:24–59:13; *see also* Docket No. 369 (VirnetX filing an Emergency Motion to Clarify Under *O2 Micro*). Apple relied on previous Court proceedings in attempting to infer that the construction of a "domain name service system" was meant to include the construction of a "domain name service." However, the Court previously interpreted "domain name service" and "domain service system" as separate terms with different constructions. Case No. 6:10-cv-417 (*Apple I*), Docket No. 266 at 15, 20. These two separate terms generally appear in different contexts: the claim preamble versus the body of the claim. Docket No. 369 at 8–10; *e.g.*, '211 Patent at claims 1, 36. Accordingly, the original constructions of "domain name service system" and "domain name service" continue to apply.

Apple did not demonstrate the absence of a genuine issue of material fact as to whether the FaceTime feature infringed the asserted patents. Accordingly, the Court denied Apple's Motion for Partial Summary Judgment of Noninfringement by FaceTime (Docket No. 315). Docket No. 362.

**2. VirnetX's Motion for Partial Summary Judgment of No Invalidity on Dependent Claims of Previously Tried Claims (Docket No. 320)**

VirnetX filed a motion for partial summary judgment based upon the *Apple I* jury finding of no invalidity of the asserted claims. Docket No. 320. VirnetX argued that, because the independent claims in U.S. Patent No. 7,418,504 (“the ’504 Patent”) and the ’211 Patent were found not invalid in *Apple I*, the five newly asserted claims that depend from the previously tried claims must also be not invalid. *Id.* at 4–6. VirnetX submitted that, if a claim is not invalid, a claim that depends from it also cannot be invalid because it is narrower in scope. *Id.* at 5–6. More specifically, VirnetX alleged that the five newly asserted dependent claims are not invalid under (1) anticipation; (2) obviousness; (3) derivation; or (4) nonjoinder. Docket No. 359 at 1.

The newly asserted dependent claims are not captured by issue preclusion, because “[e]ach claim of a patent . . . shall be presumed valid independently of the validity of other claims.” *See* 35 U.S.C. § 282(a). Although issue preclusion does not dictate that the newly asserted dependent claims are not invalid as anticipated and obvious, the relationship between the scope of independent claims and that of dependent claims does.

A dependent claim further defines an independent claim. *See* 35 U.S.C. § 112(d); 37 C.F.R. § 1.75(c). In other words, the scope of subject matter captured by an independent claim is broader than a claim that depends from it. *See* 35 U.S.C. § 112(d); 37 C.F.R. § 1.75(c). In the context of anticipation, if a reference does not read on the limitations of an independent claim, it cannot read on the limitations of a dependent claim that includes additional requirements. *See Aspex Eyewear, Inc. v. Zenni Optical, Inc.*, 713 F.3d 1377, 1381 (Fed. Cir. 2013). This is also true of obviousness. *See id.* If an independent claim is nonobvious, then a claim that depends from it is also nonobvious. *See id.*

In contrast to anticipation and obviousness, invalidity based upon derivation from another and nonjoinder of all inventors is not similarly limited by the relationship between independent and dependent claims. If the inventive entity of an independent claim is accurate, a claim that depends from it may not have the same inventive entity. *See* 25 U.S.C. § 116(a). For instance, an inventor may contribute to a patent by conceiving a limitation that is only present in a dependent claim. *See id.* A particular limitation in a dependent claim could be derived from another or cause an inventor to be excluded from a patent, while the inventive entity may accurately reflect the inventors of an independent claim. *See id.*

Accordingly, this motion (Docket No. 320) was granted with respect to Apple's anticipation and obviousness defenses and denied as to its derivation and nonjoinder defenses. Docket No. 362.

### **3. Granting VirnetX's Motion for Summary Judgment of No Inequitable Conduct (Docket No. 322)**

"Inequitable conduct resides in failure to disclose material information, or submission of false material information, with an intent to deceive, and those two elements, materiality and intent, must be proven by clear and convincing evidence." *Kingsdown Med. Consultants, Ltd. v. Hollister Inc.*, 863 F.2d 867, 872 (Fed. Cir. 1988). "Intent and materiality are separate requirements." *See Therasense, Inc. v. Becton, Dickinson & Co.*, 649 F.3d 1276, 1290 (Fed. Cir. 2011) (en banc). But-for materiality is required to establish inequitable conduct. *Id.* at 1291.

"When an applicant fails to disclose prior art to the [U.S. Patent and Trademark Office ("Patent Office")], that prior art is but-for material if the [Patent Office] would not have allowed a claim had it been aware of the undisclosed prior art." *Id.* "Hence, in assessing the materiality of a withheld reference, a court must determine whether the [Patent Office] would have allowed the claim if it had been aware of the undisclosed reference." *Id.*

When examining the intent to deceive requirement, the alleged conduct must be “viewed in light of all the evidence, including evidence indicative of good faith.” *Kingsdown*, 863 F.2d at 876. To demonstrate the intent requirement of inequitable conduct, “the single most reasonable inference able to be drawn from the evidence” must be a specific intent to deceive the Patent Office. *Therasense*, 649 F.3d at 1290. This standard applies at the summary judgment stage. *ROY-G-BIV Corp. v. ABB, Ltd.*, 63 F. Supp. 3d 690, 695 (E.D. Tex. 2014). Intent may be shown from indirect and circumstantial evidence. *Therasense*, 649 F.3d at 1290.

In its motion, VirnetX addressed a number of issues, including an explanation of why the single most reasonable inference is not that VirnetX’s prosecuting attorney, Mr. Toby Kusmer, had an intent to deceive the Patent Office. Docket No. 322 at 7–8. VirnetX explained that there was no intent to deceive the Patent Office when Mr. Kusmer allegedly: (1) disclosed too much information to the Patent Office; (2) failed to disclose evidence from related patents in Case No. 6:07-cv-80 (“the *Microsoft* litigation”) and reexamination proceedings initiated by Apple; and (3) made misleading statements during prosecution regarding the publication date of a prior art reference by Aventail (“the Aventail reference”). *Id.* at 7–21.

Apple responded by stating, among other things, that the intent requirement for inequitable conduct was well-supported by the evidence. Docket No. 339 at 20. Apple identified evidence that it believed supported inferring an intent to deceive the Patent Office. *Id.* at 20–21. Apple’s alleged evidence was the following: (1) Mr. Kusmer incorrectly told the Patent Office that the publication date of the Aventail reference was not discussed during the *Microsoft* litigation; (2) Mr. Kusmer withheld testimony from the *Microsoft* litigation during prosecution of U.S. Patent Nos. 8,051,181 and 8,504,697 (“the ’181 Patent” and “the ’697 Patent,” respectively); (3) Mr. Kusmer misled the Patent Office by inundating it with

approximately 132,500 pages of documents, which were listed on information disclosure statements; (4) Mr. Kusmer admitted he did not review all of the documents listed on the information disclosure statements before submitting them to the Patent Office; (5) Mr. Kusmer lulled the Patent Office into a false sense of complacency by promising to notify it of any evidence of the Aventail reference publication date, but then failed to do so; (6) Mr. Kusmer withheld the three reexamination declarations during prosecution of the '181 Patent; and (7) Mr. Kusmer described the three reexamination declarations related to the publication date of the Aventail reference as "insufficient" during prosecution of the '697 Patent. *Id.* at 20–21.

During the *Microsoft* litigation, the publication date of the Aventail reference was discussed in the context of a confidential deposition transcript. *See* Docket No. 322 at 10–11. Mr. Kusmer had no obligation to disclose documents under the protection of a protective order to the Patent Office. *See id.* With respect to the documents listed on the information disclosure statements, Apple did not indicate how Mr. Kusmer intended to deceive the Patent Office other than by filing information disclosure statements that totaled many pages and were not reviewed by him personally before being filed.<sup>2</sup> *See* Docket No. 339 at 20. Further, the examiner rejected the pending claims using the Aventail reference during prosecution of the '697 Patent, which indicates that the Aventail reference was not hidden from the Patent Office. *See id.* at 11.

Mr. Kusmer promised to update the Patent Office with information relevant to the publication date of the Aventail reference, and did so. *See id.* at 11–12. Apple's main complaint appears to be the speed with which he updated the Patent Office. *See id.* Mr. Kusmer received the three declarations from Apple's reexamination filings a few days before a notice of allowance was mailed for the '181 Patent. *See* Docket No. 322 at 13. However, Mr. Kusmer

---

<sup>2</sup> The Aventail reference was listed on a supplemental information disclosure statement with twenty-seven other references. *See* Docket No. 322 at 3 n.1.

brought the three declarations addressing the publication date of the Aventail reference to the examiner's attention during prosecution of the '697 Patent, which issued after the '181 Patent. *See* Docket No. 339 at 11–12. Additionally, the parties continue to dispute the Aventail reference publication date. *See* Docket No. 322 at 17.

In view of the foregoing, the single most reasonable inference was not that Mr. Kusmer had an intent to deceive the Patent Office by not disclosing the declarations. Therefore, the Court granted VirnetX's Motion for Summary Judgment of No Inequitable Conduct (Docket No. 322). Docket No. 362.

**4. Granting VirnetX's Motion to Strike Portions of the Opinion and Testimony of Mr. Christopher Bakewell (Docket No. 316)**

In the consolidation order, the Court warned that “while there is substantial overlap between the two cases, incorporating the issues remanded in *Apple I* may require limited and focused fact discovery, as well as supplemental expert reports.” Docket No. 220 at 1–2. As a result, Apple served several interrogatories and requests for admission on VirnetX. Docket No. 333 at 4. Apple then supplemented the report of its damages expert, Mr. Christopher Bakewell. *Id.*

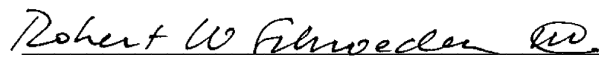
In its motion to strike, VirnetX stated that Mr. Bakewell improperly supplemented his report under the ruse that it was related to the Court's consolidation of *Apple I* and Case No. 6:12-cv-855 (“*Apple II*”). Docket No. 316 at 5. As VirnetX described it, Mr. Bakewell's supplemental expert report introduced a new damages model based on a per-feature-per-product amount, or “a reasonable royalty rate that applies to each of the three accused features [was] \$0.017 per unit (*i.e.*, each worth one-third of \$0.05 per unit).” *Id.* at 3. Apple responded by explaining that the supplemental report accounts for the multiple versions of Virtual Private Network (“VPN”) On Demand and FaceTime that would be at issue in the newly consolidated

case. Docket No. 333 at 5–6. According to Apple, because the cases were consolidated, a distinction needed to be made between the value of various versions of VPN On Demand and FaceTime and this somehow also affected the reasonable royalty if less than three infringing features were on a product. *Id.*

If Mr. Bakewell intended to present information about the relative value of the accused features (*i.e.*, VPN On Demand, FaceTime, or iMessage), that information should have already been included in an earlier expert report. Before the cases were consolidated, Mr. Bakewell's expert report addressed all three accused features. *See* Docket No. 316 at 2. Further, the post-consolidation discovery collected by Apple does not justify a shift to determining a royalty rate based on the number of infringing features on a product. *See* Docket No. 333 at 4.

Mr. Bakewell's new method of calculating damages did not sufficiently relate to the consolidation of *Apple I* and *Apple II*. Accordingly, the Court granted VirnetX's motion (Docket No. 316). Docket No. 362.

**SIGNED this 22nd day of March, 2016.**

  
ROBERT W. SCHROEDER III  
UNITED STATES DISTRICT JUDGE

**IN THE UNITED STATES DISTRICT COURT  
FOR THE EASTERN DISTRICT OF TEXAS  
TYLER DIVISION**

VIRNETX INC.,

Plaintiff,

v.

APPLE INC.,

Defendant.

§  
§  
§  
§  
§  
§  
§  
§  
§

CIVIL ACTION NO. 6:12-CV-00855-  
RWS

**JURY INSTRUCTIONS**

**1. Introduction**

**MEMBERS OF THE JURY:**

You have now heard the evidence in this case, and I will instruct you now on the law that you must apply. It is your duty to follow the law as I give it to you. On the other hand, you, the jury, are the sole judges of the facts. Do not consider any statement that I may have made during the trial or make in these instructions as an indication that I have any opinion about the facts of the case. After I instruct you on the law, as I suggested, the attorneys will have an opportunity to make their closing arguments. Statements and argument of the attorneys are not evidence and are not instructions on the law. They are intended only to assist you in understanding the evidence and what the parties' contentions are.



### **1.1 General Instruction**

A verdict form has been prepared for you, and you will take this form with you to the jury room. And when you have reached a unanimous agreement as to your verdict, you will have your foreperson fill in, date, and sign the form. At the end of the instructions, I will take you through the verdict form before the parties begin their closing arguments. Answer each question on the verdict form from the facts as you find them. Do not decide who you think should win and then answer the questions accordingly. A corporation and all other persons are equal before the law and must be treated as equals in a court of justice. With respect to each question asked, your answers and your verdict must be unanimous.

In determining whether any fact has been proven in this case, you may, unless otherwise instructed, consider the testimony of all witnesses regardless of who may have called them and all exhibits received in evidence regardless of who may have produced them. At times during the trial, it was necessary for the Court to talk with the lawyers here at the bench out of your hearing or by calling a recess. We met at the bench or when you were in the jury room because during trial, sometimes things come up that don't involve the jury. You should not speculate on what was discussed during such times. You are the jurors, and you are the sole judges of the credibility of all the witnesses and the weight and effect of all evidence. By the Court allowing testimony or other evidence to be introduced over the objection of an

attorney, the Court did not indicate any opinion as to the weight or effect of such evidence.

### **1.2 Considering Witness Testimony**

You alone are to determine the questions of credibility or truthfulness of the witnesses. In weighing the testimony of the witnesses, you may consider the witness's manner and demeanor on the witness stand, any feelings or interest in the case, or any prejudice or bias about the case, that he or she may have, and the consistency or inconsistency of his or her testimony considered in the light of the circumstances. Has the witness been contradicted by other credible evidence? Has he or she made statements at other times and places contrary to those made here on the witness stand? You must give the testimony of each witness the credibility that you think it deserves.

Even though a witness may be a party to the action and therefore interested in its outcome, the testimony may be accepted if it is not contradicted by direct evidence or by any inference that may be drawn from the evidence, if you believe the testimony.

You are not to decide this case by counting the number of witnesses who have testified on the opposing sides. Witness testimony is weighed; witnesses are not counted. The test is not the relative number of witnesses, but the relative convincing force of the evidence. The testimony of a single witness is sufficient to prove any

fact, even if a greater number of witnesses testified to the contrary, if after considering all of the other evidence, you believe that witness.

### **1.3 Impeachment by Witness's Inconsistent Statements**

In determining the weight to give to the testimony of a witness, consider whether there was evidence that at some other time the witness said or did something, or failed to say or do something, that was different from the testimony given at the trial.

A simple mistake by a witness does not necessarily mean that the witness did not tell the truth as he or she remembers it. People may forget some things or remember other things inaccurately. If a witness made a misstatement, consider whether that misstatement was an intentional falsehood or simply an innocent mistake. The significance of that may depend on whether it has to do with an important fact or with only an unimportant detail.

### **1.4 How to Examine the Evidence**

Certain testimony in this case has been presented to you through a deposition. A deposition is the sworn, recorded answers to questions asked to a witness in advance of the trial. Under some circumstances, if a witness cannot be present to testify from the witness stand, the witness testimony may be presented under oath in the form of a deposition. Sometime before this trial, attorneys representing the parties in this case questioned this witness under oath. A court reporter was present

and recorded the testimony. This deposition testimony is entitled to the same consideration and is to be judged by you as to the credibility and weight and otherwise considered by you insofar as possible the same as if the witness had been present and had testified from the witness stand in court.

While you should consider only the evidence in this case, you are permitted to draw such reasonable inferences from the testimony and exhibits as you feel are justified in the light of common experience. In other words, you may make deductions and reach conclusions that reason and common sense lead you to draw from the facts that have been established by the testimony and evidence in the case. The testimony of a single witness may be sufficient to prove any fact, even if a greater number of witnesses may have testified to the contrary, if after considering all the other evidence you believe that single witness.

There are two types of evidence that you may consider in properly finding the truth as to the facts in this case. One is direct evidence, such as testimony of an eyewitness. The other is indirect or circumstantial evidence, the proof of a chain of circumstances that indicates the existence or non-existence of certain other facts. As a general rule, the law makes no distinction between direct and circumstantial evidence, but simply requires that you find the facts from all of the evidence, both direct and circumstantial.

The parties have stipulated, or agreed, to some facts in this case. When the lawyers on both sides stipulate to the existence of a fact, you must, unless otherwise instructed, accept the stipulation as evidence and regard the fact as proved. Apple agrees that a prior VPN On Demand “always” mode feature infringed the ’135 and ’151 patents, and a prior version of the FaceTime feature infringed the ’504 and ’211 patents. Infringement and damages for those features are not at issue in this case. Those two features have been redesigned. VirnetX contends that those redesigns infringe. Apple disputes VirnetX’s contentions. As I explained to you at the beginning of the case, a party may attempt to change the design of existing products or methods so they do not fall within the boundaries of those claims. If that change in fact causes the product or method to fall outside of those boundaries, the product or method would no longer infringe the claims.

### **1.5 Objections to Evidence**

Attorneys representing clients in courts such as this one have an obligation in the course of trial to assert objections when they believe testimony or evidence is being offered that is contrary to the rules of evidence. The essence of a fair trial is that it be conducted pursuant to the rules of evidence and that your verdict be based only on legally admissible evidence. So you should not be influenced by the objection or by the Court's ruling on it. If the objection is sustained, then ignore the

question. If the objection is overruled, then you may treat the answer to that question just as you would treat the answer to any other question.

## **1.6 Expert Witnesses**

When knowledge of a technical subject matter may be helpful to the jury, a person who has special training or experience in that technical field, he or she is called an expert witness, is permitted to state his or her opinion on those technical matters. However, you are not required to accept that opinion. As with any other witness, it is up to you to decide whether the witness's testimony is believable or not, whether it is supported by the evidence, and whether to rely upon it. In deciding whether to accept or rely upon the opinion of an expert witness, you may consider any bias of the witness.

## **2. Contentions of the Parties**

I will first give you a summary of each side's contentions in this case. I will then tell you what each side must prove to win on these issues.

VirnetX seeks damages from Apple for allegedly infringing certain claims of four VirnetX patents—specifically the '504, '211, '135, and '151 patents. VirnetX contends that Apple has made, used, sold, or offered for sale in the United States, or imported into the United States, products that practice the asserts claims of these patents. Specifically, VirnetX contends that:

Apple's redesigned VPN on Demand feature infringes claim 1 and 7 of the '135 patent and claim 13 of the '151 patent.

Apple's redesigned FaceTime feature infringes claims 1, 2, 5, and 27 of the '504 and claims 36, 47, and 51 of the '211 patent.

VirnetX seeks damages in the form of a reasonable royalty for this infringement. VirnetX also contends that Apple induces this infringement.

In response to VirnetX's infringement contentions, Apple contends that these features do not infringe any claim of VirnetX's patents. Apple further contends that it has not induced any other party's infringement. Because Apple contends that the VPN On Demand and FaceTime features at issue in this case do not infringe, Apple further contends that VirnetX is not entitled to damages.

### **3. Burdens of Proof**

As I told you at the beginning of this trial, in any legal action, facts must be proved by a required amount of evidence known as the "burden of proof." The burden of proof in this case is on VirnetX. VirnetX has the burden of proving infringement and damages by a preponderance of the evidence. Preponderance of the evidence means the evidence that persuades you that a claim is more likely true than not true. If the proof establishes that all parts of one of VirnetX's infringement claims are more likely true than not true, then you should find for VirnetX as to that claim. But if you find that VirnetX has failed to prove any part of its claim is more

likely than not true, then VirnetX may not recover on its claim. In determining whether any fact has been proved by a preponderance of the evidence, you may, unless otherwise instructed, consider the stipulations, the testimony of all witnesses, regardless of who may have called them, and all exhibits received in evidence, regardless of who may have produced them.

#### **4. The Patent Claims**

Before you can decide many of the issues in this case, you will need to understand the role of patent "claims." The patent claims are the numbered sentences at the end of each patent. The claims are important because it is the words of the claims that define what a patent covers. The figures and text in the rest of the patent provide a description and/or examples of the invention and provide a context for the claims. But it is the claims that define the breadth of the patent's coverage. Each claim is effectively treated as if it were a separate patent. And each claim may cover more or less than another claim. Therefore, what a patent covers depends in turn on what each of its claims covers.

You will first need to understand what the Asserted Claims cover in order to decide whether or not there is infringement. The law says that it is the Court's role to define the terms of the claims and it is your role to apply these definitions to the issues that you are asked to decide in this case. Therefore, as I explained to you at the start of the case, I have determined the meaning of certain claim terms at issue



in this case. And I have provided you those definitions of those terms in your juror notebook. You must accept the definitions of these words in the claims as being correct. It is your job to take these definitions and apply them to the issues that you are deciding, including infringement. The claim language I have not interpreted for you in your notebook is to be given its ordinary and accustomed meaning as understood by one of ordinary skill in the art.

You have also heard discussion about a construction for the word “domain name service.” You are instructed that the construction for “domain name service system,” an element of all of the asserted claims of the ’504 and ’211 patents, does not incorporate or include the Court’s construction for the term “domain name service.”

#### **4.1 How a Patent Claim Defines What It Covers**

I will now explain how a patent claim defines what it covers. A claim sets forth, in words, a set of requirements. Each claim sets forth its requirements in a single sentence. If a device or system satisfies each of these requirements then it is covered by the claim. In patent law, the requirements of a claim are often referred to as "claim elements" or "claim limitations." When a thing, such as a feature, product, process, or system meets all of the requirements of a claim, the claim is said to "cover" that thing, and that thing is said to “fall” within the scope of that claim. In other words, a claim covers a feature, product, process, or system where each of the

claim elements or limitations is present in that feature, product, process, or system. Conversely, if the feature, product, process, or system meets only some, but not all, of the claim elements or limitations, then that feature, product, process, or system is not covered by the claim.

#### **4.2 Independent and Dependent Claims**

This case involves two types of patent claims: Independent claims and dependent claims.

An "independent claim" sets forth all of the requirements that must be met in order to be covered by that claim. Thus, it is not necessary to look at any other claim to determine what an independent claim covers. In this case, for example, Claim 1 of the '504 patent is an independent claim.

Other claims in the case are "dependent claims." In this case, for example, Claim 2 of the '504 patent depends from Claim 1. A dependent claim refers to another claim and includes all the requirements or parts of the claim to which it refers. The dependent claim then adds its own additional requirements. In this way, the claim "depends" on another claim. To determine what a dependent claim covers, it is necessary to look at both the dependent claim and any other claims to which it refers. A product, feature, method or system that meets all of the requirements of both the dependent claim and the claims to which it refers is covered by that dependent claim.

### **4.3 Open-Ended or “Comprising” Claims**

The beginning portion, or preamble, to some of the claims uses the word "comprising." "Comprising" and "comprises" mean "including but not limited to," or "containing but not limited to." Thus, if you decide that an accused feature includes all the requirements in that claim, the claim is infringed. This is true even if the accused instrumentality includes components in addition to those requirements. For example, a claim to a table comprising a tabletop, legs, and glue would be infringed by a table that includes a tabletop, legs, and glue, even if the table also includes wheels on the table's legs.

## **5. Infringement – Generally**

Patent law gives the owner of a valid patent the right to exclude others from importing, making, using, offering to sell, or selling the patented invention. Any person or business entity that has engaged in any of those acts without the patent owner's permission infringes the patent.

You can have more than one patent governing an area of technology, although it may relate to different aspects of that technology. The mere fact that Apple has patents related to part of the technology of the accused features is not a defense to the fact that someone else may have a patent related to another part of those features.

I will now instruct you as to the rules you must follow when deciding whether VirnetX has proven that Apple infringed the Asserted Claims.

### **5.1 Direct Infringement- Literal Infringement**

If a person makes, uses, offers to sell, or sells in the United States or imports into the United States what is covered by the claims of a patent without the patent owner's permission, that person is said to literally infringe the patent. To determine literal infringement, you must independently compare each of the accused features with the Asserted Claims, using my instructions as to the meaning of those patent claims.

A patent claim is literally infringed only if an accused feature, product, system, or method includes each and every element in that patent claim. If the accused feature, product, system, or method does not contain one or more of the elements recited in a claim, then that feature, product, system, or method does not literally infringe that claim. If you find that the accused product or method includes each element of the claim, then that product or method infringes the claim even if such product or method contains additional elements that are not recited in the claims.

If you find that the accused feature, product, system, or method includes each element of the claim, then that feature, product, system, or method infringes the claim, even if such feature, product, system, or method contains additional elements that are not recited in the claims.

A person may literally infringe a patent, even though in good faith the person believes that what it is doing is not an infringement of any patent, and even if it did not know of the patent. Literal infringement does not require proof that the person copied a product or the patent. You must consider each of the Asserted Claims individually. You must be certain to compare each accused feature, product, or system with each claim such feature, product, or system is alleged to infringe. Each accused feature, product, or system should be compared to the limitations recited in the Asserted Claims, not to any preferred or commercial embodiment of the claimed invention.

You must analyze each Asserted Claim and each accused feature separately. If you find that VirnetX has proved by a preponderance of the evidence that each and every limitation of that claim is present in the accused feature, method, or system, then you must find that such feature, method, or system infringes that claim.

### **5.3 Indirect Infringement**

VirnetX alleges that Apple is also liable for indirect infringement by actively inducing others to directly infringe the Asserted Claims. As with direct infringement, you must determine whether there has been indirect infringement by active inducement on a claim-by-claim and feature-by-feature basis.

Although VirnetX need not prove that Apple has directly infringed to prove indirect infringement, VirnetX must prove that someone has directly infringed. If

there is no direct infringement by anyone, Apple cannot have actively induced the infringement of the patent.

To show active inducement of infringement, VirnetX must prove by a preponderance of the evidence that Apple's customers or end-users have directly infringed the asserted claims of the patents-in-suit, and that Apple has actively and knowingly aided and abetted that direct infringement.

Apple is liable for active inducement of a claim only if:

- (1) Apple has taken action during the time the patent is in force which encourages acts by someone else;
- (2) the encouraged acts constitute direct infringement of that claim;
- (3) Apple
  - is aware of the patent and knows that the encouraged acts constitute infringement of the patent, or;
  - is willfully blind to the infringement of the patent. Willful blindness requires that Apple subjectively believed that there was a high probability that the encouraged acts constituted infringement of the patent and Apple took deliberate actions to avoid learning of the infringement.
- (4) Apple has the intent to encourage infringement by someone else; and
- (5) The encouraged acts are actually carried out by someone else.

In order to prove active inducement, VirnetX must prove that each of the above requirements is met by a preponderance of the evidence. That is, that it is more likely true than not that each of the above requirements has been met. If you find that Apple was aware of the patent but believed that the acts it encouraged did not infringe that patent, Apple cannot be liable for active inducement of infringement. In order to establish active inducement of infringement, it is not sufficient that Apple was aware of the acts that allegedly constituted the direct infringement. Rather, you must find that Apple specifically intended to cause the acts that constitute the direct infringement and must have known or was willfully blind that the action would cause direct infringement. If you do not find that Apple meets these specific intent requirements by a preponderance of the evidence, then you must find that Apple has not actively induced the alleged infringement.

## **6. Damages – Generally**

I will now instruct you on damages. If you find that Apple has infringed any claim of VirnetX's patents-in-suit, you must determine the amount of damages to which VirnetX is entitled.

The amount of damages must be adequate to compensate VirnetX for the infringement. At the same time, your damages determination must not include additional sums to punish Apple or to set an example. You may award compensatory

damages only for the loss that VirnetX proves was more likely than not caused by Apple's infringement.

VirnetX seeks damages in the form of a reasonable royalty. Generally, a reasonable royalty is the reasonable amount that someone wanting to use the patented invention should expect to pay to the patent owner and the patent owner should expect to receive.

### **6.1 Damages- Burden of Proof**

Where the parties dispute a matter concerning damages, it is VirnetX's burden to prove the amount of damages by a preponderance of the evidence. VirnetX must prove the amount of damages with reasonable certainty but need not prove the amount of damages with mathematical precision. However, VirnetX is not entitled to damages that are remote or speculative. In other words, you should award only those damages that VirnetX establishes that it more likely than not suffered.

### **6.2 Damages – Reasonable Royalty**

A reasonable royalty is the amount of money a willing patent owner and a willing prospective licensee would have agreed upon at the time the infringement began for a license to make, use, or sell the invention. It is the royalty that would have resulted from an arm's length negotiation between a willing licensor and a willing licensee. This is known as the hypothetical negotiation. Unlike in a real world negotiation, all parties to the hypothetical negotiation are presumed to believe



that the patent is infringed and valid. In considering this hypothetical negotiation, you should focus on what the expectations of patent owner and the infringer would have been had they entered into an agreement at that time and had they acted reasonably in their negotiations.

If infringement is found, the date of the hypothetical negotiation would be September 2013, when the redesigned versions of VPN on Demand and FaceTime were released. The parties agree that if infringement is found, damages would begin on September 18, 2013.

In making your determination of the amount of a reasonable royalty, it is important that you focus on the time period when Apple first infringed that patent and the facts that existed at that time. However, evidence of things that happened after the infringement first began may be considered in evaluating the reasonable royalty only to the extent that the evidence aids in assessing what royalty would have resulted from a hypothetical negotiation.

Your determination does not depend on the actual willingness of the parties to the lawsuit to engage in such negotiations in the real world. Your focus should be on what the parties' expectations would have been had they entered into negotiations for royalties at the time of the hypothetical negotiation.

## **6.2 Reasonable Royalty Factors**

In deciding what is a reasonable royalty that would have resulted from the hypothetical negotiation, you may consider the factors that the patent owner and the alleged infringer would consider in setting the amount the alleged infringer should pay. I will list for you a number of factors you may consider. This is not every possible factor, but it will give you an idea of the kinds of things to consider in setting a reasonable royalty. They are as follows:

- The royalties received by the patentee for licensing of the patents-in-suit, proving or tending to prove an established royalty.
- Royalties paid for other patents comparable to the patents-in-suit.
- The nature and scope of the license, as exclusive or non-exclusive; or as restricted or non-restricted in terms of territory, or with respect to the parties to whom products may be sold.
- Whether or not the licensor had an established policy and marketing program to maintain its patent exclusivity by not licensing others to use the invention or by granting licenses under special conditions designed to preserve that exclusivity.
- The commercial relationship between the licensor and the licensee, such as whether they are competitors in the same territory, in the same line of business, or whether they are inventor and promoter.

- Whether being able to use the patented invention helps in making sales of other products or services.
- The duration of the patent and the term of the license.
- The utility and advantages of the patented invention over the old modes or devices, if any, that had been used for achieving similar results.
- The nature of the patented invention, the character of the commercial embodiment of it as owned and produced by the licensor, and the benefits to those who have used the invention.
- The extent of the licensee's use of the patented invention and any evidence probative of that use.
- The portion of the profits that is due to the patented invention as compared to the portion of the profit due to other factors, such as unpatented elements or unpatented manufacturing processes, or features or improvements developed by the licensee.
- Expert opinions as to what would be a reasonable royalty.
- The amount that a licensor and a licensee would have agreed upon if both sides had been reasonably and voluntarily trying to reach an agreement; that is, the amount which an accused infringer would have been willing to pay as a royalty and yet be able to make a reasonable profit and which amount would

have been acceptable to the patent owner if it had been willing to create a license.

No one factor is dispositive and you can and should consider the evidence that has been presented to you in this case on each of these factors. In determining a reasonable royalty, you may also consider evidence concerning the availability and cost of non-infringing alternatives to the patented invention. A non-infringing alternative must be an acceptable product that is licensed under the patent or that does not infringe the patent. The framework which you should use in determining a reasonable royalty is a hypothetical negotiation between normally prudent business people. In considering the evidence of a reasonable royalty, you're not required to accept one specific figure or another for the reasonable royalty. You are entitled to determine what you consider to be a reasonable royalty based upon your consideration of all of the evidence presented by the parties, whether that evidence is of a specific figure or a range of figures.

When determining a reasonable royalty, you may consider evidence concerning the amounts that other parties have paid for rights to the patents in question or for rights to similar technologies. A license agreement need not be perfectly comparable to a hypothetical license that would have been negotiated between VirnetX and Apple in order for you to consider it. However, if you choose to rely upon evidence from any license agreements, you must account for any

differences between those licenses and the hypothetically negotiated license between VirnetX and Apple when you make your reasonable royalty determination, including the type of technology licensed, whether the license contained a cross-license and/or similar patent protections, whether the license contained any value related to a release of liability, the date when the license was entered, the financial or economic conditions of the parties at the time the parties entered into the license, the extent of use, if any, of any particular licensed patents, the number of patents involved in the license, whether or not the license covered foreign intellectual property rights, the extent to which litigation may have affected the license, and whether contrary to the hypothetical negotiation the licensee in the real world license, at the time of entering the license, believed that the patents were either not infringed or were invalid.

VirnetX has relied on license agreements in which royalties were based on a percentage of the entire price of the licensed end-products. But in determining a reasonable royalty, you must not rely on the overall price of Apple's accused products at issue in this case. Damages for patent infringement must be apportioned to reflect the value the invention contributes to the accused products or features and must not include value from the accused products or features that is not attributable to the patent.

## **7. Instructions for Deliberations**

You must perform your duties as jurors without bias or prejudice as to any party. The law does not permit you to be controlled by sympathy, prejudice, or public opinion. All parties expect that you will carefully and impartially consider all of the evidence, follow the law as it is now being given to you, and reach a just verdict, regardless of the consequences. You should consider and decide this case as a dispute between persons of equal standing in the community, of equal worth, and holding the same or similar stations in life. All persons, including corporations, and other organizations stand equal before the law, regardless of size or who owns them, and are to be treated as equals.

When you retire to the jury room to deliberate on your verdict, you may take this charge with you, as well as the exhibits which the Court has admitted into evidence. You will select your foreperson and conduct your deliberations. If you recess during your deliberations, please follow all of the instructions that the Court has given you about your conduct during the trial.

After you have reached your verdict, your foreperson is to fill in on the form your answers to the questions. Do not reveal your answers until such time as you are discharged, unless otherwise directed by me.

Any notes that you may have taken during the trial, of course as we discuss at the beginning, are only aids to your memory. If your memory should differ from

your notes, then you should rely on your memory and not on the notes. Your notes are not evidence. A juror who has not taken notes should rely on his or her independent recollection of the evidence and should not be unduly influenced by the notes of other jurors. Notes are not entitled to any greater weight than the recollection or impression of each juror about the testimony.

If you want to communicate with me at any time during your deliberations, please give a written message or a question to the Court Security Officer, and we will provide you with sheets on which to do that, and she will bring it to me. I will then respond as promptly as possible, either in writing or by having you brought into the courtroom so that I can address you orally. I will always first disclose to the attorneys your question and my response before I answer your question.

And then finally, after you have reached a verdict, you are not required to talk with anyone about the case unless the Court orders otherwise.

**IN THE UNITED STATES DISTRICT COURT  
FOR THE EASTERN DISTRICT OF TEXAS  
TYLER DIVISION**

VIRNETX INC.,

Plaintiff,

v.

APPLE INC.,

Defendant.

§  
§  
§  
§  
§  
§  
§  
§  
§  
§

CIVIL ACTION NO. 6:12-CV-00855-  
RWS

**VERDICT FORM**

In answering these questions, you are to follow all of the instructions provided by the Court during the Court's jury instructions. Your answers to each question must be unanimous.

As used herein, "'135 patent" means U.S. Patent No. 6,502,135; "'151 patent" means U.S. Patent No. 7,490,151; "'504 patent" means U.S. Patent No. 7,418,504; "'211 patent" means U.S. Patent No. 7,921,211.

1. Did VirnetX prove by a preponderance of the evidence that Apple's redesigned version of its VPN on Demand feature infringes the following claims of VirnetX's '135 & '151 patents?

**Answer "Yes" or "No" for each Claim.**

'135 Patent

'151 Patent

Claim 1

YES

Claim 13

YES

Claim 7

YES

***CONTINUE ON TO NEXT PAGE***



2. Did VirnetX prove by a preponderance of the evidence that Apple's redesigned version of the FaceTime feature infringes the following claims of VirnetX's '504 & '211 patents?

Answer "Yes" or "No" for each Claim.

'504 Patent

Claim 1	<u>YES</u>
Claim 2	<u>YES</u>
Claim 5	<u>YES</u>
Claim 27	<u>YES</u>

'211 Patent

Claim 36	<u>YES</u>
Claim 47	<u>YES</u>
Claim 51	<u>YES</u>

*Answer Question 3 only if you answered "yes" for any of Questions 1 or 2 above. Otherwise, do not answer this question.*

3. What royalty do you find, by a preponderance of the evidence, would fairly and reasonably compensate VirnetX for any infringement that you have found?

\$ 502,567,709.00

You have now reached the end of the verdict form and should review it to ensure it accurately reflects your unanimous determinations. The jury foreperson should then sign and date the verdict form in the spaces below and notify the Court Security Officer that you have reached a verdict. The jury foreperson should retain possession of the verdict form and bring it when the jury is brought back into the courtroom.

Date: 15 APR 2018

By: [REDACTED]

**JURY FOREPERSON**

**IN THE UNITED STATES DISTRICT COURT  
FOR THE EASTERN DISTRICT OF TEXAS  
TYLER DIVISION**

VIRNETX INC., LEIDOS, INC.,

Plaintiffs,

v.

APPLE INC.,

Defendant.

§  
§  
§  
§  
§  
§  
§  
§  
§  
§

CIVIL ACTION NO. 6:12-CV-00855-RWS

**MEMORANDUM OPINION AND ORDER**

Plaintiff VirnetX, Inc. (“VirnetX”) and Defendant Apple Inc.’s (“Apple”) dispute spans over eight years in this Court. An overview of the lengthy history of the dispute provides helpful context for the Court’s opinion below.

On August 11, 2010, VirnetX filed Case No. 6:10-cv-417 against Apple alleging infringement of U.S. Patent Nos. 6,502,135 (“the ’135 Patent”), 7,418,504 (“the ’504 Patent”), 7,490,151 (“the ’151 Patent”) and 7,921,211 (“the ’211 Patent”). Case No. 6:10-cv-417 (“417 action”), Docket No. 1. On November 6, 2012, a jury found that the first versions of Apple’s accused VPN On Demand and FaceTime features infringed the asserted patents and that the asserted patents were not invalid. 417 action, Docket No. 790. On the same day, VirnetX filed the instant case, Case No. 6:12-cv-855, accusing of infringement several redesigned products. Docket No. 1.

In the 417 action, Apple and VirnetX both filed post-trial motions, which the Court resolved in a memorandum opinion. 417 action, Docket No. 851. The matter was appealed, and the Federal Circuit affirmed-in-part, reversed-in-part and remanded for further proceedings. 417

action, Docket No. 853; *see VirnetX, Inc. v. Cisco Sys., Inc.*, 767 F.3d 1308, 1313–14 (Fed. Cir. 2014).

The Federal Circuit affirmed the jury’s finding of infringement by VPN On Demand and affirmed the Court’s denial of Apple’s motion for judgment as a matter of law on invalidity. *Id.* The Federal Circuit vacated the infringement finding for FaceTime based upon a change in claim construction, holding that the term “secure communication link” requires both “security and anonymity,” and vacated damages for VPN On Demand and FaceTime because it found that the jury relied on a flawed damages model. *Id.* at 1314.

Upon receipt of the Federal Circuit’s mandate, the Court solicited the parties’ proposals on how to proceed. 417 action, Docket No. 855. The parties submitted a status report in which VirnetX proposed the Court consolidate the remaining issues in the 417 action with the upcoming trial in the 855 action. Docket No. 864 at 4. Apple opposed the consolidation. *See* 417 action, Docket No. 873 at 45:20–46:6. After a status conference on March 10, 2015, the Court consolidated the 855 and 417 actions, designating the 855 action as the lead case with a revised schedule. Docket No. 220. After extensive motion practice (*see* Docket Nos. 315, 317–323, 326; *see also* Docket Nos. 362, 468), the consolidated action was tried to a jury, and the jury returned a verdict finding infringement of the ’135, ’151, ’504 and ’211 patents.

Again, both Apple and VirnetX filed post-trial motions (Docket Nos. 462, 463), and on July 29, 2016, the Court granted Apple’s Motion for a New Trial Based Upon the Consolidation of Cause Nos. 6:10-cv-417 and 6:12-cv-855. Docket No. 500. The Court reasoned that the consolidation and repeated discussion of the previous jury verdict resulted in an unfair trial. Docket No. 500 at 14. In its Order, the Court explained that “Cause No. 6:10-cv-417 will be retried with jury selection to begin on September 26, 2016, unless the parties agree otherwise on an

alternative date, and immediately followed by a second trial on the issue of willfulness. Cause No. 6:12-cv-855 will be retried after Cause No. 6:10-cv-417.” *Id.* at 15.

After another round of extensive motion practice (*see, e.g.*, 417 action, Docket Nos. 930–931, 937, 944–945), the 417 action was again tried to a jury. The jury found that FaceTime infringed the ’211 and ’504 patents and awarded approximately \$302 million in damages for the collective infringement by the VPN On Demand and FaceTime features in the accused Apple products. 417 action, Docket No. 1025. After the September trial, both parties submitted post-trial motions (*see* 417 action, Docket Nos. 1018–1019, 1047, 1062–1063), which the Court resolved in a memorandum opinion (Docket No. 1079). Apple’s appeal of the 417 action final judgment is pending before the Federal Circuit. *See* 417 case, Docket Nos. 1079, 1089, 1091.

While the post-trial motions were pending, on February 9, 2017, the Court requested that the parties meet and confer about the timing of the 855 trial and propose a schedule. The parties each filed a response (Docket Nos. 519, 520), and Apple simultaneously filed a motion to stay (Docket No. 518). The Court denied Apple’s motion (Docket Nos. 527, 553) and set the case on a schedule (Docket No. 539).

The Court held a jury trial in this matter from April 2, 2018 through April 11, 2018. The trial was bifurcated into (1) a liability and damages phase and (2) a willfulness phase—which were tried in succession to one jury. After the liability and damages phase, the jury returned a verdict finding both VPN on Demand and FaceTime to infringe each asserted patent and awarding \$502,567,709 in damages. Docket No. 723. After the willfulness phase, the jury returned a verdict that Apple’s infringement was willful. Docket No. 729.

Following the verdict, Apple filed its Omnibus Motion for Judgment as a Matter of Law under Rule 50(b) and for a New Trial (Docket No. 775), and VirnetX filed its Motion for Entry of

Judgment and Equitable and Statutory Relief (Docket No. 774). The Court heard argument on the motions on July 18, 2018. The Court now resolves the parties' motions below.

### LEGAL STANDARD

Judgment as a matter of law is only appropriate when “a reasonable jury would not have a legally sufficient evidentiary basis to find for the party on that issue.” FED. R. CIV. P. 50(a). “The grant or denial of a motion for judgment as a matter of law is a procedural issue not unique to patent law, reviewed under the law of the regional circuit in which the appeal from the district court would usually lie.” *Finisar Corp. v. DirecTV Group, Inc.*, 523 F.3d 1323, 1332 (Fed. Cir. 2008).

Under Fifth Circuit law, a court is to be “especially deferential” to a jury’s verdict and must not reverse the jury’s findings unless they are not supported by substantial evidence. *Baisden v. I’m Ready Prods., Inc.*, 693 F.3d 491, 499 (5th Cir. 2012). “Substantial evidence is defined as evidence of such quality and weight that reasonable and fair-minded men in the exercise of impartial judgment might reach different conclusions.” *Threlkeld v. Total Petroleum, Inc.*, 211 F.3d 887, 891 (5th Cir. 2000). The Court will “uphold a jury verdict unless the facts and inferences point so strongly and so overwhelmingly in favor of one party that reasonable men could not arrive at any verdict to the contrary.” *Cousin v. Trans Union Corp.*, 246 F.3d 359, 366 (5th Cir. 2001); *see also Int’l Ins. Co. v. RSR Corp.*, 426 F.3d 281, 296 (5th Cir. 2005). However, “[t]here must be more than a mere scintilla of evidence in the record to prevent judgment as a matter of law in favor of the movant.” *Arismendez v. Nightingale Home Health Care, Inc.*, 493 F.3d 602, 606 (5th Cir. 2007) (citing *Laxton v. Gap, Inc.*, 333 F.3d 572, 577 (5th Cir. 2003)).

In evaluating a motion for judgment as a matter of law, the court must “draw all reasonable inferences in the light most favorable to the verdict and cannot substitute other inferences that [the

court] might regard as more reasonable.” *E.E.O.C. v. Boh Bros. Const. Co., L.L.C.*, 731 F.3d 444, 451 (5th Cir. 2013). Although the court must review the record as a whole, it must disregard all evidence favorable to the moving party that the jury is not required to believe. *Ellis v. Weasler Eng’g Inc.*, 258 F.3d 326, 337 (5th Cir. 2001). However, a court may not make credibility determinations or weigh the evidence, as those are solely functions of the jury. *See id.* (citing *Reeves v. Sanderson Plumbing Prods., Inc.*, 530 U.S. 133, 150–51 (2000)). The Court gives “credence to evidence supporting the moving party that is uncontradicted and unimpeached if that evidence comes from disinterested witnesses.” *Arismendez*, 493 F.3d at 606.

Under Federal Rule of Civil Procedure 59(a), a new trial may be granted on any or all issues “for any reason for which a new trial has heretofore been granted in an action at law in federal court.” Rule 59(a)(1)(A). The Federal Circuit reviews the question of a new trial under the law of the regional circuit. *Z4 Techs., Inc. v. Microsoft Corp.*, 507 F.3d 1340, 1347 (Fed. Cir. 2007). The court can grant a new trial “based on its appraisal of the fairness of the trial and the reliability of the jury’s verdict.” *Smith v. Transworld Drilling Co.*, 773 F.2d 610, 612–13 (5th Cir. 1985). “Courts grant a new trial when it is reasonably clear that prejudicial error has crept into the record or that substantial justice has not been done, and the burden of showing harmful error rests on the party seeking the new trial.” *Sibley v. Lemaire*, 184 F.3d 481, 487 (5th Cir. 1999) (quoting *Del Rio Distributing, Inc. v. Adolph Coors Co.*, 589 F.2d 176, 179 n. 3 (5th Cir. 1979)). “A new trial may be granted, for example, if the district court finds the verdict is against the weight of the evidence, the damages awarded are excessive, the trial was unfair, or prejudicial error was committed in its course.” *Smith*, 773 F.2d at 612–13. The decision to grant or deny a new trial is committed to the sound discretion of the district court. *See Allied Chem. Corp. v. Daiflon, Inc.*, 449 U.S. 33, 36 (1980). “[N]ew trials should not be granted on evidentiary grounds unless, at a

minimum, the verdict is against the great[,] not merely the greater weight of the evidence.”  
*Conway v. Chem. Leaman Tank Lines, Inc.*, 610 F.2d 360, 363 (5th Cir. 1980).

**I. APPLE’S OMNIBUS MOTION FOR JUDGMENT AS A MATTER OF LAW UNDER RULE 50(B) AND FOR A NEW TRIAL (DOCKET NO. 775)**

**A. Apple’s Motion for Judgment as a Matter of Law and New Trial on Infringement**

**(1) *FaceTime***

A short overview of the accused FaceTime products in this case is instructive. There are three versions of FaceTime that the Court discusses below. The “first version” was the product at issue in the 417 action, which the 417 jury found to infringe the asserted claims. A “second version” was implemented in April of 2013 and relayed 100 percent of calls. *See, e.g.*, 4/3 PM Tr. at 165:15–19. The parties agree that the “second version” does not infringe VirnetX’s patents. *Id.* And finally, at issue in this case is a “third version” of FaceTime, which was released in September 2013 via a software update. *Id.* at 165:20–22; Docket No. 775 at 2–3. At trial, the design and operation of this third product was disputed, and the jury determined that the third version infringed each of the asserted claims. Docket No. 723.

**a. *Indication***

VirnetX accuses the third version of FaceTime of infringing claims 1, 2, 5 and 27 of the ’504 patent and claims 36, 47 and 51 of the ’211 patent. Each asserted claim of the ’504 and ’211 patents requires a “domain name service system” that is configured “to comprise an indication,” in the case of the ’504 patent, or “to indicate,” in the case of the ’211 patent, that “the domain name service system supports establishing a secure communication link.” The term “secure communication link” requires “a direct communication link that provides data security and anonymity.” Docket No. 262; *see also VirnetX*, 767 F.3d at 1317–19. The Court’s construction of “indication” is “an indication other than merely returning of requested DNS records, such as an



IP address or key certificate, that the domain name service system supports establishing a secure communication link.” Docket No. 180 at 10.

According to Apple, VirnetX’s expert, Dr. Mark Jones, opined that the claimed “domain name service system” was met by Apple’s FaceTime invitation server, push notification servers and registration database. Docket No. 775 at 1. Specifically, Apple suggests that Dr. Jones identified the “accept push” message as meeting the indication requirement. *Id.* at 1–2. Apple disagrees with Dr. Jones’s assessment because, in the version of FaceTime at issue in this case, the callee’s IP address was removed from the accept push message. *Id.* at 2. Apple maintains that “this change alone had the effect of directing all FaceTime calls through a relay server, an *indirect* connection that the parties agree does not infringe.” *Id.* (emphasis in original). Apple submits that, although FaceTime clients could establish peer-to-peer connections after September 2013, this was the result of a client-side software change, not a server change, so Apple does not infringe. *Id.* at 3.

In response, VirnetX identifies the following trial testimony from Dr. Jones about indication:

Q. And how does that accept push message indicate that the FaceTime domain name service system supports establishing a secured communication link?

A. Well, that accept push message is the culmination of this provisioning process that I described. And that provisioning process includes authenticating the identities of both of the parties and their phones. It includes providing them and -- the certificates for each one of the parties. There are, as I discussed, certificate names, a session token. And then coming back in that accept push message are things like the callee’s certificate, a push token, a certificate name for the callee, and other information.

Docket No. 779 at 2 (citing 4/3 PM Tr. at 121:5–16. According to VirnetX, the accept push message is an indication that the FaceTime servers have successfully authenticated and

provisioned devices to establish a FaceTime call. *Id.* VirnetX also points to Dr. Jones’s testimony that, under the Court’s construction, the FaceTime servers are not merely returning requested DNS records such as IP addresses or certificates and that the FaceTime servers support establishing direct, secure FaceTime calls. *Id.* (citing 4/3 PM Tr. at 122:16–123:17; 123:18–124:4). Aside from expert evidence, VirnetX also identifies the testimony of Mr. Gokul Thirumalai, an Apple corporate representative, who testified at trial that he made a server-side change in May of 2013 to facilitate direct peer-to-peer calls. *Id.* (citing 4/5 PM Tr. at 247:22–248:1).

In its reply, Apple reiterates its belief that the indication necessarily must include the callee’s IP address because the address is necessary for the caller to make a direct call. Docket No. 783 at 1. Apple also suggests that the Court’s construction requires that the “indication do something *more* than ‘merely’ return an IP address—not that it *cannot* return an IP address.” *Id.* at 2 (emphasis in original).

To put the parties’ arguments in context, a review of the claim construction is instructive. The Court construed the “indication” term in the context of the ’504 patent. At Markman, Apple proposed that an “an indication that the domain name service system supports establishing a secure communication link” be construed to mean “an affirmative signal beyond the mere returning of an IP address, public key, digital signature, or certificate that the domain name service system supports establishing a secure communication link.” Docket No. 180 at 8–9.

The Court found that Apple’s proposed construction was more correct in light of statements made by the patentees during reexamination. Specifically, the Court explained that the patentees represented to the examiner that “[t]he ’504 patent specification clearly and unequivocally disclaims merely returning an address or a public key by describing these actions as ‘conventional’ in the prior art, and that “[n]ever does the specification equate the mere return of requested DNS

records, such as an IP address or key certificate, with supporting secure communications.” *Id.* at 10 (citing Docket No. 150-14 at 5–6). These statements were found to clearly distinguish the mere return of requested DNS records, such as an IP address or key certificate, from the claimed “indication” terms. *Id.* The Court ultimately construed “an indication that the domain name service system supports establishing a secure communication link” as “an indication other than merely returning of requested DNS records, such as an IP address or key certificate, that the domain name service system supports establishing a secure communication link.” *Id.*

The Court’s construction requires an indication *other than* the mere return of an IP address, so return of an IP address is not required, and VirnetX was not required to point to the return of an IP address to establish infringement at trial. Apple posits that an IP address is nonetheless necessary for a caller to make a direct call and that the claims require indicating support for a direct link. Docket No. 783 at 1. But VirnetX presented credible evidence at trial that IP addresses from both the caller and callee are *not* necessary to set up a direct call. *See* 4/4 AM Tr. (Jones) at 29:17–30:1 (“Q. And I think that the suggestion was that there were two that -- that setting up a direct connection requires two IP addresses. Is that your impression? A. That’s what I understand the point to be. Q. All right. Dr. Jones, is that true or false? A. That’s false. Q. Will you please explain why? A. Yes. The -- on the internet when a client wants to contact another computer, a server or another device, it just needs the address of that device. It’s going to send a packet from that one -- from, say, the left to the right or the right to the left, and all these packets contain IP addresses of -- so if I’m sending from the callee to the caller, the callee’s IP address is contained in that packet that goes to the caller. That’s how direct communications are set up on the internet. You don’t need IP addresses from both parties to establish that direct communication to begin with.”). That Apple’s expert, Dr. Matthew Blaze, disagreed with Dr. Jones does not entitle Apple to JMOL.

*See* 4/9 AM Tr. at 130:21–131:1. It is not the Court’s job to substitute its judgment for the jury’s and to reweigh the expert testimony to determine a technical fact. *See Ellis v. Weasler Eng’g Inc.*, 258 F.3d 326, 337 (5th Cir. 2001).

Regardless, on this record, the Court concludes that VirnetX introduced substantial evidence at trial to support the jury’s verdict that the “indication” term was met by the third version of FaceTime. *See* 4/3 PM Tr. at 121:2–124:7. Specifically, Dr. Jones testified that the accept push message indicates that the FaceTime domain name service system supports establishing a secured communication link as a “culmination of a provisioning process” and contains the callee’s certificate, a peer-push token, session token, a certificate name for the callee, and other information. *Id.* at 121:8–122:10. Dr. Jones explained that the accept push message indicates that the FaceTime servers have successfully authenticated and provisioned both devices to establish a direct, secure FaceTime call. *See id.* at 121:5–124:4.

Dr. Jones also explained that the FaceTime servers were not conventional DNS servers:

Q. In what way does the FaceTime DNSS do something beyond merely returning DNS records?

A. Well, what it’s returning is in that -- that accept push is, first, an indication that the provisioning process has been completed and it’s the FaceTime servers that facilitating that provision process. It also includes the command number that’s indicating that this is an accept instead of reject. It’s got the certificate, it’s got a push token, and the other information I referenced as well.

*Id.* at 123:8–17. Even Mr. Thirumalai—Apple’s fact witness for FaceTime—did not contest Dr. Jones’s explanation of how FaceTime works. *See* 4/5 PM Tr. (Thirumalai) at 206:17–20 (“Are you aware of anything incorrect about the technical explanation of the operation of FaceTime that was provided by Dr. Jones? A. No, I don’t think so.”).

Apple suggests that Dr. Jones’s testimony “indicates nothing about support for a *direct* call, as [it] applies equally to a *non-infringing* relayed call.” Docket No. 793 at 1 (emphasis in

original). That the accept push message can also be used to establish a relayed FaceTime call does not change the result because “[t]he addition of features does not avoid infringement, if all the elements of the patent claims have been adopted. Nor is infringement avoided if a claimed feature performs not only as shown in the patent, but also performs an additional function.” *N. Telecom, Inc. v. Datapoint Corp.*, 908 F.2d 931, 945 (Fed. Cir. 1990). Although Dr. Blaze disagreed with Dr. Jones and testified that “nothing in [the accept push] indicates support for direct communication,” the jury was not required to credit Dr. Blaze’s testimony, and the Court again refuses to re-weigh evidence and invade the province of the jury. 4/9 AM Tr. at 21:5.

***b. Secure Communication Link***

Apple’s “anonymity defense was tried in the ’417 action. It’s a common feature to the new version of FaceTime as well as the old version.” 4/2 AM Tr. at 17:10–12. In its filings before trial, Apple represented to the Court that it did “not intend to present a defense regarding the absence of anonymity.” Docket No. 628 at 8 n.2. At trial, Apple sought a ruling from the Court precluding the presentation of its anonymity defense. While the Court agreed that Apple could not simply “repeat arguments that are identical to ones from the previous case,” it did not rule that any particular argument was foreclosed. *Id.* at 19:15–21. Regardless, however, based upon Apple’s representation that the secure communication link noninfringement argument in this case and the 417 action are identical, issue preclusion attaches to Apple’s argument in this case, and the Court declines to rule on issues already resolved by the 417 judgment. *See Aspex Eyewear, Inc. v. Zenni Optical Inc.*, 713 F.3d 1377, 1382 (Fed. Cir. 2013).

***c. Domain name service system***

Before the consolidated trial in 2016, the Court held that “domain name service system” did not incorporate the Court’s construction of “domain name service.” Apple does not present any new arguments in support of its construction, and the Court declines to reconsider its previous

rulings. *See* 417 action, Docket No. 266 at 20 (“Defendants seek to improperly import limitations from a preferred embodiment into the claim language. The claim language itself provides a description of the domain name service system. Thus, the Court finds that ‘domain name service system’ does not require construction.”); 417 action, Docket No. 732 at 14, n.3 (“However, the Court did not construe ‘domain name service system’ because the claim language itself provided a description of the term, i.e. that it must ‘comprise an indication that [it] supports establishing a secure communication link.’ ”). Accordingly, Apple is entitled to neither JMOL nor a new trial on this basis.

## **(2) VPN On Demand**

VirnetX accuses Apple’s VPN on Demand feature of infringing Claims 1 and 7 of the ’135 patent and claim 13 of the ’151 patent.

A conventional DNS resolves domain names (e.g., “Yahoo.com”) into Internet Protocol (“IP”) addresses. *See* ’135 patent at 37:22–27. A user’s web browser then utilizes the IP address to request a website. *Id.* at 37:24–29. The ’135 and ’151 patents share a common specification disclosing a system in which, instead of a conventional DNS receiving the request, a DNS proxy intercepts it and determines whether the request is for a secure site. *Id.* at 38:23–25. If the request is for a secure site, the system automatically initiates a virtual private network (“VPN”) between the proxy and the secure site, but if the request is for a non-secure website, then the DNS proxy forwards the request to a conventional DNS for resolution. *Id.* at 38:43–47.

Claim 1 of the ’135 patent has three steps: (1) generating from the client computer a DNS request that requests an IP address corresponding to a domain name associated with the target computer; (2) determining whether the DNS request transmitted in step 1 is requesting access to a secure web site; and (3) in response to determining that the DNS request in step 2 is requesting

access to a secure target web site, automatically initiating the VPN between the client computer and the target computer. Claim 1 and 13 of the '151 patent, respectively, require creating an “encrypted channel” and “secure channel” “between” the client and the secure server. '151 patent at 48:28–29.

The Court’s discussion of VPN on Demand, too, is informed by a review of the VPN On Demand product considered in the 417 action. The VPN On Demand accused of infringement in the 417 action maintained an “Always” and an “If Needed Mode.” The “Always” mode would check to see whether a requested site in a DNS request was on a user-configurable list and, if so, as the name suggests, would always create a VPN without sending a DNS request to a DNS server. 4/3 PM Tr. at 195:4–14. The “Always” mode ignored location: It would create a VPN without regard to whether the user was inside or outside a private network. *Id.* at 195:13–196:3. The “If Needed” mode was location-based: It would create a VPN if a user was outside the private network, but not if a user was inside the private network. 4/6 PM Tr. 151:13–24. The parties agree that the “If Needed” mode did not infringe. *See, e.g.*, Docket No. 779 at 13; 4/3 AM Tr. at 77:25–78:2.

The jury in the 417 action returned a verdict finding the “Always” mode to infringe, and the Court denied Apple’s JMOL motion of noninfringement. *VirnetX Inc. v. Apple Inc.*, 925 F. Supp. 2d 816, 830 (E.D. Tex. 2013). The Federal Circuit affirmed as to literal infringement. *VirnetX*, 767 F.3d at 1322.

VirnetX argued at trial that the redesigned VPN on Demand (“redesigned VOD”) replicates the Always mode functionality with “Evaluate Connection.” 4/3 AM Tr. at 78: 3–6. Apple contends now that it is entitled to JMOL of noninfringement for the redesigned VOD at issue in this case. Its arguments can be grouped into five categories: (1) fact-based noninfringement

arguments about the redesigned VOD functionality; (2) arguments that HTTPS probe is optional and location-based; (3) arguments about an interception requirement; (4) arguments analogizing the redesigned VOD to the “If Needed” mode in the 417 action; and (5) arguments about actual use.

*a. Fact-based noninfringement arguments*

In its motion, Apple first presents the Court with a series of factual representations about redesigned VOD functionality, suggesting that, under Apple’s description of the redesigned product functionality, it is entitled to JMOL of noninfringement. However, in each instance, there is substantial evidence supporting the verdict, and the Court declines to reweigh the evidence on JMOL and substitute its judgment for that of the jury.

For example, Apple argues that “the decision to start a VPN is based on something other than a DNS request.” Docket No. 775 at 8. But at trial, VirnetX’s expert, Dr. Jones, explained that the domain name matching in VPN on Demand is based on the domain name in the DNS request. *See* 4/10 AM at 66:7–25. Dr. Jones also explained that, when VPN on Demand checks the results of the probe, it “is determining whether or not the DNS request that’s being made is being made from the outside to the firewall to the server on the inside or is being made from within the firewall.” *Id.* at 67:10–21. In its reply, Apple disagrees with Dr. Jones and states that, “[b]ecause infringement depends solely on the optional probe’s failure, any ‘determination’ is based on the probe, not the request.” Docket No. 783 at 5. But Apple’s disagreement with Dr. Jones is not a basis for JMOL, and the jury was entitled to credit Dr. Jones’s testimony on this fact.

Apple also argues that the probe failure occurs “only after a DNS request is made and returns a successful result.” Docket No. 775 at 8. But the jury was entitled to credit Dr. Jones’s trial testimony that the results of the HTTPS probe are checked after the DNS request is made, but by that point, “[t]he probe has already failed or succeeded.” *See* 4/10 Sealed Tr. at 92:3–25.



Similarly, Apple argues that the HTTPS probe cannot determine whether a server is a secure server because it accesses “a wholly separate probe server.” Docket No. 775 at 9. Apple provides no basis, however, for the Court to disregard the record evidence that the probe server and the target server are on the same private network and that redesigned VOD determines whether the target server is behind a firewall (and therefore “requires authorization for access”) based on whether the probe is able to reach the probe server. *See* 4/3 AM Tr. at 80:4–11.

Apple also suggests that redesigned VOD creates VPNs based on the DNS response, not the DNS request. Docket No. 775 at 10. But, at trial, Dr. Jones testified that, when the DNS request matched a domain name in the list and the probe failed, VPN on Demand will start a VPN “no matter what comes back” in the DNS response. *See* 4/3 AM Tr. at 81:11–23. He explained that, if the DNS response contains an IP address, VPN on Demand will discard it. *See id.* at 82:4–10.

According to Apple, with respect to claim 13 of the ’151 patent, VPN on Demand always forwards the DNS request to a DNS function that returns an IP address, regardless of whether the DNS request corresponds to a secure server. Docket No. 775 at 10–11. But there is substantial evidence in the record suggesting that, while VPN on Demand forwards all DNS requests to conventional DNS (*see* 4/4 AM Tr. at 16:21–17:15), VPN on Demand only forwards the DNS request to a DNS function that returns an IP address of a nonsecure computer—as is required by claim 13—if the DNS request does not correspond to a secure server. *See* 4/4 AM Tr. at 17:16–22 (“Q. And there is nothing in iOS 7 VPN On Demand which makes a determination whether or not to send a DNS request in DNS function on the basis of whether the request is to secure or unsecure server. Right? A. There is a check to see if it’s going to send to a DNS function that

returns an IP address within the code, yes. If you're talking about sending it to an external server, no.").

*b. The optional HTTPS probe is location based*

According to Apple, the fact that the optional HTTPS is location-based is fatal to VirnetX's claim. Docket No. 775 at 7–10. Apple maintains that, to the extent the optional HTTPS probe performs any "determination," it is a determination as to the location of the requesting device relative to the private network, not a determination as to any requested server's security. *Id.* at 9. Apple argues that Dr. Jones's infringement theory that the requesting device's location can determine whether a server is "secure" or "not secure" is "absurd." *Id.*

The asserted claims of the '135 and '151 patents require determining whether a DNS request is requesting access to a secure web site or secure server. VirnetX introduced evidence at trial that redesigned VOD performs this determination by comparing the domain name of the DNS request against a list of domain names in a configuration file and by consulting the result of the HTTPS probe. *See* 4/3 AM Tr. at 86:14–87:22. Dr. Jones testified that, "when the name matches and the probe has failed, that indicates that to reach that computer it's going to -- it can't reach it without authorization. In other words, the probe failed. It couldn't get there. And by being on the list, the IT administrators indicated that this is a computer that can communicate in a VPN." *Id.* at 87:16-22. This evidence satisfies the Court's claim constructions. *See* Docket No. 180 at 24 and 27 (construing "secure web site" to mean "a web site that requires authorization for access and that can communicate in a VPN" "secure server" to mean "a server that requires authorization for access and that can communicate in an encrypted channel.").

Apple suggests that it is "absurd" for a server to be a "secure server" depending on whether the requesting device is outside the private network. The record evidence demonstrates that whether the requesting device is inside or outside the private network affects whether a server

*requires authorization for access* (which is a requirement of the Court’s construction of “secure server”). *See* 4/4 AM Tr. at 32:4–14.

Even Apple’s fact witness for redesigned VOD, Mr. Simon Patience, explained at trial that the location of a requesting device can bear on whether a target server is a “secure server” that requires authorization for access and can communicate in a VPN. Mr. Patience testified that redesigned VOD creates a VPN based on whether there is a firewall between the device and the target server. *See* 4/6 PM Tr. at 165:18–166:2; 164:23–165:7; 170:10–16. According to Mr. Patience, “the location probe is just another test to determine whether you’re inside or outside the firewall.” *Id.* at 167:22–168:3. Mr. Patience also confirmed that the list of domain names “is supposed to contain a list of things behind the firewall.” *Id.* at 214:8–14. Mr. Patience also testified that servers behind a firewall are secure servers. *Id.* at 198:11–17 (“Q. Do you see this server? A. Yes. Q. Is that a secure server? A. Yes. I – you could consider it to be one, yes. Q. You could consider it to be one or not? A. Well, the fact that it’s behind a firewall would tend to imply that it is a secure server, yes.”). Importantly, Mr. Patience applied the Court’s construction of “secure server,” as he also confirmed that servers behind firewalls require authorization for access that can communicate in a VPN. *Id.* at 204:11–15 (“Q. When the device is outside of the firewall, does it require authorization for access to communicate with a server within the firewall? A. Yes, because that’s what VPN is. VPN is the authorization.”).

On this record, it is clear that the fact that the HTTPS probe is location-based is not fatal to VirnetX’s claims, and Apple is not entitled to JMOL on this basis.

***c. Redesigned VOD does not intercept DNS requests before they are sent to a DNS server***

In its motion, Apple first characterizes the ’135 and ’151 patents as introducing a “DNS proxy” “that intercepts DNS requests *before they are sent to a DNS server* and automatically

creates a VPN if the DNS request corresponds to a secure site.” Docket No. 775 at 6–7 (emphasis in original). In its reply, Apple reprises this argument, stating that “DNS requests must be intercepted before transmission to avoid hampering anonymous communications on the Internet.” Docket No. 783 at 5 (internal citations omitted).

As a preliminary matter, the Court notes that, on a JMOL motion, it applies its claim constructions. The Court has not construed the asserted claims to require a “DNS proxy that intercepts DNS requests *before they are sent* to a DNS server.” Apple did not attempt to seek a ruling from the Court that claims have this temporal limitation, and its contention that the claims are so limited is therefore waived. *See Conoco, Inc. v. Energy & Envtl. Int’l, L.C.*, 460 F.3d 1349, 1359 (Fed. Cir. 2006) (“[L]itigants waive their right to present new claim construction disputes if they are raised for the first time after trial.”). Post-trial motions are not a vehicle to seek new constructions and retroactively apply them to the evidence introduced at trial.

*d. Redesigned VOD replicates the noninfringing “If Needed” mode*

According to Apple, the old “If Needed” mode of VPN on Demand was location-based: “[I]t would create a VPN if a user was outside the private network, but not if a user was inside the private network.” Docket No. 775 at 8. Apple maintains that, “like the prior ‘If Needed’ mode, the redesigned version of VOD is location-based—it creates a VPN only if a user needs one to connect to the requested site.” *Id.* In its reply, Apple also states that redesigned VOD “operates in the same way” as the old “If Needed” mode by first “attempt[ing] to connect insecurely and only creates a VPN if needed.” Docket No. 783 at 4.

The Court questions the value in comparing the accused product to a non-infringing product to determine whether the accused product infringes. Indeed, in its motion, Apple recognizes that an infringement analysis comparing accused products with old products, rather

than the claims, is “undisputedly incorrect.” Docket No. 775 at 37 (citing *Zenith Labs., Inc. v. Bristol-Meyers Squibb Co.*, 19 F.3d 1418, 1423 (Fed. Cir. 1994)).

In any event, VirnetX presented evidence that, while the “If Needed” functionality would attempt to make an unsecure connection to a domain name if it could, the redesigned VOD does not first attempt to connect insecurely. *See* 4/3 AM Tr. at 76:6–10; 80:24–82:10. VirnetX also presented evidence that, unlike redesigned VOD, which creates a VPN based on the domain name matching the results of the HTTPS probe, the “If Needed” mode triggered a VPN only if the domain name could not be resolved by conventional DNS. *Id.* at 76:17–77:14; PX308. Accordingly, Apple is not entitled to JMOL on this basis.

Apple also specifically requests a new trial based on VirnetX’s arguments at trial that Apple “moved” the infringing Always mode functionality into the redesigned VOD. Docket No. 775 at 37. To be clear, *both* parties discussed the previous iterations of VPN on Demand at length at trial. *See, e.g.*, 4/6 PM (Direct examination of Apple’s witness, Mr. Patience) Tr. at 172:9–16 (“Q. Now, how does the operation of If Needed in iOS 7 to 11 compare to the Always mode that was removed from iOS 3 to 6? A. So Always is just driven by the name of the server. It just will always create a VPN if it finds the name of a server on this list. And so it would create a VPN when it's outside of the firewall because the name is on the list. And inside the firewall it will create a VPN also because the name is on the list.”).

However, as discussed above, VirnetX presented sufficient evidence of infringement to withstand Apple’s JMOL motion, without including any reference to the previous Always mode. Indeed, the Court cannot identify any prejudice to Apple from the parties’ discussions of the Always mode. The jury’s verdict is not against the great weight of the evidence, and Apple is not entitled to a new trial on this basis.

*e. Actual use*

In its motion, Apple argues that VirnetX based its infringement theory on a specific and optional operating scenario for redesigned VOD but offered no evidence that anyone configured or used VOD in the infringing manner. Docket No. 775 at 11. According to Apple, VirnetX only offered evidence of redesigned VOD's capability to infringe, which is insufficient because the claim language here is not drawn to mere capability. *Id.* Apple suggests that VirnetX did not identify use of the infringing configuration by Apple or its customers.

VirnetX responds that claim 13 of the '151 patent is an apparatus claim, which can be infringed by a device having the claimed structure capable of functioning as described by the claim. Docket No. 779 at 15. VirnetX contests whether direct infringement of this claim requires proof of use like a method claim and argues that the claim is drawn to structure that has a specified capability. *Id.* at 16. With respect to the method claims of the '135 patent, VirnetX points to Dr. Jones's testimony that Apple directly infringes by testing VPN on Demand and an internal Apple email documenting a VPN On Demand test plan. *Id.* at 17. VirnetX also argues that the failure of the HTTPS probe is not a condition of infringement and that the probe need only be configured to infringe. *Id.*

"[I]n every infringement analysis, the language of the claims, as well as the nature of the accused product, dictates whether an infringement has occurred." *Finjan, Inc. v. Secure Computing Corp.*, 626 F.3d 1197, 1204 (Fed. Cir. 2010). "To infringe a method claim, a person must have practiced all steps of the claimed method." *Lucent Techs., Inc. v. Gateway, Inc.*, 580 F.3d 1301, 1317 (Fed. Cir. 2009). "Direct infringement of a method claim can be based on even one instance of the claimed method being performed." *Mirror Worlds, LLC v. Apple Inc.*, 692 F.3d 1351, 1359 (Fed. Cir. 2012) (citing *Lucent*, 580 F.3d at 1317).

An accused device may be found to infringe if it is reasonably capable of satisfying the claim limitations, even if it is also capable of noninfringing modes of operation. *Hilgraeve Corp. v. Symantec Corp.*, 265 F.3d 1336, 1343 (Fed. Cir. 2001) (collecting cases). A claim that recites *capability* and not actual operation may be infringed when it is capable of operating in the claimed mode. *Finjan*, 626 F.3d at 1204. “[W]hen the asserted claims recite capability, [Federal Circuit] case law supports finding infringement by a ‘reasonably capable’ accused device on a case-by-case basis particularly where . . . there is evidence that the accused device is actually used in an infringing manner and can be so used without significant alterations.” *Ericsson, Inc. v. D-Link Sys., Inc.*, 773 F.3d 1201, 1217 (Fed. Cir. 2014).

#### 1. Method Claims

The asserted claims of the ’135 patent are method claims for which there is record evidence supporting a finding of infringement. For example, Dr. Jones testified that Apple directly infringes by testing VPN on Demand. *See* 4/3 PM Tr. at 131:10–12. Additionally, VirnetX introduced PX1018 into the trial record, which is an internal Apple email documenting a “VPN On Demand test plan” that involves the “RequiredURLStringProbe” (i.e., the HTTPS probe). Although the jury could have found that the testing described in the document never occurred, it could equally and reasonably have concluded that the described test plan was carried out. And the Court must “draw all reasonable inferences in the light most favorable to the verdict and cannot substitute other inferences that [the court] might regard as more reasonable.” *E.E.O.C. v. Boh Bros.*, 731 F.3d at 451.

Apple suggests that actual failure of the HTTPS probe is a condition of infringement. But the asserted claims of the ’135 do not require that a VPN is always established, and the probe’s success or failure only bears on whether a VPN is created. If VPN on Demand determines that the

DNS request is not requesting access to a secure website (e.g., if the HTTPS probe did not fail), then the step need not be carried out in order for the claimed method to be performed. *See Cybersettle, Inc. v. Nat'l Arbitration Forum, Inc.*, 243 Fed. Appx. 603, 607 (Fed. Cir. 2007) (“It is of course true that method steps may be contingent. If the condition for performing a contingent step is not satisfied, the performance recited by the step need not be carried out in order for the claimed method to be performed.”).

Apple also suggests that there is no evidence that its customers directly infringe the asserted claims of the '135 patent. Docket No. 775 at 13. The record suggests that Apple infringed with the Always mode; it announced it was removing that feature; there was customer backlash; and Apple subsequently released the new version of VPN on Demand, which replicates the Always mode. *See* 4/3 PM Tr. at 133:2–134:14; PX1007; PX1012. This is at least circumstantial evidence that some subset of Apple's customers directly infringe the '135 patent with the new version of VPN on Demand. *See Moleculon Research Corp. v. CBS, Inc.*, 793 F.2d 1261, 1272 (Fed. Cir. 1986). Accordingly, Apple is not entitled to JMOL of noninfringement on the method claims.

## 2. Apparatus claims

Claim 13 of the '151 patent is drawn to structure that has a specified capability. The structure it requires is “computer readable medium” that has “computer readable instructions[.]” The instructions are claimed by reference to the steps they perform, but the claim explicitly states that the steps are performed “when executed[.]” As such, Apple directly infringes when it makes, uses, offers to sell, sells, and imports devices containing VPN on Demand because VPN on Demand is capable of operating in an infringing mode. *See* 35 U.S.C. § 271(a); *Intel Corp. v. U.S. Int'l Trade Comm'n*, 946 F.2d 821, 832 (Fed. Cir. 1991) (“Because the language of claim 1 refers to ‘programmable selection means’ and states ‘whereby when said alternate addressing mode is



selected,’ the accused device, to be infringing, need only be capable of operating in the page mode.”).

The functional language in claim 13 does not change the infringement analysis: Functional language in an apparatus claim requires that an accused apparatus be capable of performing the recited functions. *Intel*, 946 F.2d at 832; *see also Revolution Eyewear, Inc. v. Aspex Eyewear, Inc.*, 563 F.3d 1358, 1369–70 (Fed. Cir. 2009) (“[C]laim 22 here only requires a capacity to perform a function . . . .”); *Microprocessor Enhancement Corp. v. Texas Instrum. Inc.*, 520 F.3d 1367, 1375 (Fed. Cir. 2008) (“Claim 7 . . . is clearly limited to a pipelined processor possessing the recited structure and capable of performing the recited functions . . . .”); *UltimatePointer, L.L.C. v. Nintendo Co.*, 816 F.3d 816, 826 (Fed. Cir. 2016) (“[T]he ‘generating data’ limitation reflects the capability of that structure[.]”); *MasterMine Software, Inc. v. Microsoft Corp.*, 874 F.3d 1307, 1315 (Fed. Cir. 2017) (“Though claim 8 includes active verbs—presents, receives, and generates—these verbs represent permissible functional language used to describe capabilities of the ‘reporting module.’”).

Regardless, however, VirnetX presented at least circumstantial evidence at trial of actual infringement. VirnetX introduced PX1018, in which an Apple employee explains that the “VPN On Demand test plan” would “[i]deally . . . be presented to the customer.” PX1018. The jury was entitled to infer from this evidence that Apple carried out its test plan and followed through on its plan to present it to customers. PX1018. Accordingly, Apple is not entitled to JMOL of noninfringement.

**(3) iMessage**

Apple seeks JMOL on its counterclaim that iMessage in iOS 5–8 and OS X 10.8–10.10 does not infringe because VirnetX’s claim of infringement and Apple’s counterclaim for declaratory judgment of noninfringement have not been dismissed. Docket No. 775 at 14.

In the 417 action, VirnetX asked this Court to enter judgment on Apple’s dropped invalidity theories after the 2012 trial. *See* 417 action, Docket No. 732 at 45–46. These theories and references were asserted up to the time of trial but were never presented to the jury. *Id.* at 45. Noting that there must be a “continuing case or controversy with respect to withdrawn or otherwise unasserted claims” for the Court to enter judgment, the Court declined to do so. *Id.* at 46.

“[T]he existence of a case or controversy must be evaluated on a claim-by-claim basis.” *Jervis B. Webb Co. v. S. Sys., Inc.*, 742 F.2d 1388, 1399 (Fed. Cir. 1984). And “jurisdiction must exist at all stages of review, not merely at the time the complaint was filed.” *Streck, Inc. v. Research & Diagnostic Sys., Inc.*, 665 F.3d 1269, 1282 (Fed. Cir. 2012) (internal quotation marks omitted). The Court cannot and will not enter judgment on claims and defenses that were not presented for consideration to the jury because there is no basis to do so. *See Connell v. Sears, Roebuck & Co.*, 722 F.2d 1542, 1553 (Fed. Cir. 1983); *Datascope Corp. v. Smec, Inc.*, 776 F.2d 320, 327 (Fed. Cir. 1985) (noting that references in pleadings do not support a judgment that particular claims are invalid when “the validity of those claims were not litigated by the parties at trial”); *Nordisk Pharm., Inc. v. Bio-Technology Gen. Corp.*, 424 F.3d 1347, 1356 (Fed. Cir. 2005) (vacating the district court’s order invalidating a claim not litigated at trial).

Parties routinely drop asserted claims and defenses in their cases as litigation progresses, and the Court declines to penalize or discourage the parties’ efforts in narrowing the case for trial. Accordingly, Apple is not entitled to JMOL of noninfringement for iMessage.

**(4) Indirect Infringement**

Apple also argues that it is entitled to JMOL and a new trial on indirect infringement because VirnetX failed to prove that Apple had specific intent to encourage infringement. Docket No. 775 at 15. In response, VirnetX suggests that the evidence supporting the jury's willfulness finding equally supports its finding of specific intent. Docket No. 779 at 19.

The trier of fact was provided with sufficient evidence from which to infer Apple's knowledge that use of its products would constitute infringement and Apple's specific intent. For example, for VPN on Demand, VirnetX presented evidence at trial that Apple knew about the patents-in-suit before the redesign, knew that the previous version infringed, and replicated that original design. *See* 4/3 PM Tr. at 132:20–133:1. Further, the test plan document introduced at trial includes an infringing configuration and Apple states therein that the configuration “would be presented to the customer[.]” PX1018. This is sufficient evidence from which the jury could conclude that Apple intended its customers to infringe.

For FaceTime, VirnetX presented evidence that Apple was able to avoid infringement via full relay but chose to resume infringement with direct FaceTime calls. 4/3 AM Tr. at 135:22–136:2. VirnetX also presented evidence that Apple “broke” FaceTime on older phones to force customers to resume direct calls in iOS 7. *See* 4/5 PM Tr. at 272:22–273:2; PX1020; PX1031; PX1106. Similarly, VirnetX presented evidence that Apple has encouraged its users to make FaceTime calls through marketing of the feature (*see, e.g.*, PX1098)—calls that the jury determined to be infringing.

On this record, the Court is persuaded that the jury had a sufficient evidentiary basis from which to conclude that Apple both knowingly induced infringement and possessed specific intent to encourage its customers' infringement. *See ACCO Brands, Inc. v. ABA Locks Mfr. Co.*, 501

F.3d 1307, 1312 (Fed. Cir. 2007). The jury's verdict is supported by substantial evidence and is not against the great weight of the evidence. Accordingly, Apple is not entitled to JMOL or a new trial on indirect infringement.

### **B. Apple's Motion for Judgment as a Matter of Law and a New Trial on Damages**

The Court struggles to identify a single basis Apple provides for its JMOL on damages that it has not already considered at length in the context of *Daubert* motions and the 417 post-trial. The Court questions whether repeated review of the same arguments in detail is a judicious use of the Court's resources. To be sure, the Court concluded at summary judgment that the damages questions between the 417 action and this matter were not identical for the purposes of issue preclusion because of disputes about the hypothetical negotiation date and about the accused products' functionality. Docket No. 553 at 7. At trial, however, Apple presented damages testimony that was nearly identical to that presented in the 417 action.

In fact, a comparison of Apple's post-trial brief in the 417 action and this case reveals that Apple's arguments mirror those in the 417 action. Apple has asserted in both cases' post-trial briefing that (1) no reasonable jury could rely on the VoIP licenses; (2) Mr. Weinstein failed to apportion; (3) Mr. Weinstein's model violated the entire market value rule; (4) Apple is entitled to an offset for units that included Skype; and (5) no reasonable jury could award more than Apple's expert's prescribed royalty rate per unit. The Court considers each argument below.

#### **(1) *Reliance on the VoIP licenses***

Apple argues that Mr. Weinstein failed to account for the differences between the Voice-over-IP licenses he considered and the hypothetical VirnetX-Apple license. Specifically, Apple argues: (1) that each VoIP license is for a longer period of time and for more patents than the hypothetical license; (2) that the VoIP licenses cover products that are less complex and have fewer

features than the accused products; (3) that the VoIP licenses were litigation licenses; (4) that the VoIP licenses were paired with IP PBX servers, which Apple does not sell; and (5) that Mr. Weinstein devalued the Microsoft license. Docket No. 775 at 18–22.

“Prior licenses . . . are almost never perfectly analogous to the infringement action.” *Ericsson*, 773 F.3d at 1227. “For example, allegedly comparable licenses may cover more patents than are at issue in the action, include cross-licensing terms, cover foreign intellectual property rights, or, as here, be calculated as some percentage of the value of a multicomponent product.” *Id.* These differences must be explained to the jury so they can account for “the need to discount reliance on a given license to account only for the value attributed to the licensed technology.” 773 F.3d at 1228; *see also VirnetX*, 767 F.3d at 1330 (“Moreover, all of the other differences that Apple complains of were presented to the jury, allowing the jury to fully evaluate the relevance of the licenses.”).

To some extent, each of Apple’s criticisms is a *Daubert* attack on Mr. Weinstein’s opinion, and the Court has already held that Apple’s concerns are related to the weight of this testimony, not its admissibility. Docket No. 362 at 3 (“Although Apple presents valid criticisms of Dr. Jones’s opinions, they go to the weight of the evidence rather than admissibility.”). Nevertheless, Mr. Weinstein explained the factual circumstances surrounding each license he relied on at length to the jury. *See* 4/5 AM Tr. at 40:10–61:25. The Court can identify no error in the jury relying on these licenses in determining the appropriate royalty rate in this case.

## **(2) *Failure to apportion***

Apple also challenges Mr. Weinstein’s opinion to the extent he failed to apportion the VirnetX licensing policy. According to Apple, the policy applies to the price of the end product, but “[n]othing in the policy explains what, if any, apportionment was done to devise that policy.”

Docket No. 775 at 23. Even if the policy and licenses are apportioned, however, Apple contends that the Apple devices are more complex than the VoIP phones and that this difference was not accounted for. *Id.* Apple also suggests that Mr. Weinstein’s opinion is faulty because he opined that a per-unit royalty rate should remain constant regardless of the number of accused features in the accused device. *Id.* at 23. And Apple maintains that the royalty rate is improper because it exceeds the market price of FaceTime. *Id.* at 24.

To the extent Apple complains that Mr. Weinstein failed to apportion the *licenses*, such apportionment is not necessary because the rates in the license agreements are real-world rates, apportioned to reflect the value of VirnetX’s technology to those companies who entered into them. *See also Ericsson Inc. v. D-Link Corp.*, No. 6:10-cv-473, 2013 WL 2242444, \*2–3 (E.D. Tex. May 21, 2013) (“It goes without saying that the licensees would not have paid value for portions of the 802.11 standard unrelated to Ericsson’s patents. Therefore, Mr. Bone’s report does not implicate the entire market value rule.”); 4/4 PM Tr. at 11:13–12:14 (Larsen).

Apple has not identified any authority suggesting that a company’s general licensing policy—that it applies to all potential licensees—must be apportioned on a per-defendant basis before it can be admissible or relevant to damages. Regardless, Mr. Weinstein understood the licensing policy to be apportioned. *See* 4/5 AM Tr. 121:9–12 (“Well, I think I’ve testified how I believe that VirnetX’s licensing policy contemplates rates that are apportioned to reflect the contribution that VirnetX’s technology makes to the final products of licensees.”). That Mr. Weinstein could not describe how the apportionment “was done” in the initial licensing policy does not render his opinion unreliable, and goes to the credibility the trier of fact could assign to his testimony. *See* Docket No. 775 at 18. Ultimately, Mr. Weinstein’s damages calculations were

based on real-world license agreements entered into pursuant to the policy, which were described in detail for the jury to evaluate.

Apple suggests that Mr. Weinstein's opinion that the per-unit royalty rate should remain constant regardless of the number of accused features present in the given device is improper. Docket No. 775 at 22–23. But the jury heard substantial evidence explaining the basis for this opinion that the royalty rate was not dependent on the number of accused features. *See, e.g.*, 4/5 AM Tr. at 86:19–94:7. The Court is not persuaded that Mr. Weinstein's opinion is unreliable, and the jury was entitled to credit his opinion.

Apple also makes a number of factual arguments for which the Court declines to substitute its judgment for that of the jury. For example, Apple claims that the royalty for FaceTime could not have been more than the alleged price of FaceTime. Docket No. 775 at 23–24, 26. This argument was presented to the jury, and the jury was entitled to reject Apple's argument that its pricing of FaceTime was a "market price" that reflected the value of the product. *Compare* 4/9 PM Tr. at 260:23–261:7, *with* 4/5 PM Tr. at 169:17–172:3.

Additionally, Apple suggests that, because VirnetX accused a single implementation of redesigned VOD of infringement and because VOD has other, noninfringing modes of operation, the damages award is greater than VOD's footprint in the marketplace. Docket No. 775 at 24. In response, VirnetX suggests that Apple discounts the "huge importance of the presence of the infringing functionality in VPN On Demand." Docket No. 779 at 28 (citing 4/5 AM Tr. at 64:18–66:19, 68:7–70:16; PX1007; PX 1121; PX1012.03; PX1010). In its reply, Apple reprises its argument that there is no evidence that anyone used redesigned VOD to infringe, but the Court has already rejected this argument with respect to liability above. Accordingly, Apple is not entitled to JMOL on this basis.

**(3) *Entire market value rule* (“EMVR”)**

Apple argues that use of VirnetX’s licenses to calculate per-unit rates violates the EMVR. Docket No. 775 at 25. As the Federal Circuit explained in *Ericsson, Inc. v. D-Link Sys., Inc.*, 773 F.3d 1201, 1226 (Fed. Cir. 2014), the entire market value rule has two parts: a substantive legal rule and a separate, evidentiary principle. The substantive legal rule mandates that an ultimate reasonable royalty award be based on the incremental value that the patented invention adds to the end product. *Id.* The evidentiary principle, applicable specifically to the choice of a royalty base, is that where a multi-component product is at issue and the patented feature is not the item which imbues the combination of the other features with value, care must be taken to avoid misleading the jury by placing undue emphasis on the value of the entire product. *Id.*

Mr. Weinstein’s reliance on *actual* licenses entered into for the patented technology does not violate the EMVR. Importantly, the jury was never tasked with applying a royalty rate percentage to the entire value of any Apple product. Instead, Mr. Weinstein’s model applied a per-unit rate of \$1.20 based on certain VirnetX licenses (the per-unit royalty rate) to the number of accused units sold (the royalty base). As was true in the 417 action, the Court also instructed the jury not to consider any outside knowledge they may have had about the total revenue or total price of the accused products, which it presumes the jury followed. *See* Docket No. 721 (jury instructions) (“VirnetX has relied on license agreements in which royalties were based on a percentage of the entire price of the licensed end-products. But in determining a reasonable royalty, you must not rely on the overall price of Apple’s accused products at issue in this case.”); *See Francis v. Franklin*, 471 U.S. 307, 324 n.9 (1985). Accordingly, this is not a basis for judgment as a matter of law or a new trial.



**(4) *Skype offset***

Next, Apple argues, as it did in the 417 action, that “it is entitled to a partial offset of any damages for products that are already licensed, including, for example, accused devices that already include Skype.” Docket No. 775 at 25–26; 417 action, Docket No. 1079 at 14. As was true in the 417 action, “Apple never pled a license defense, and Apple presented no evidence that its devices are shipped with Skype included.” 417 action, Docket No. 1079 at 17. “Apple also provides no explanation for why the actions of its end-users to modify its product post-sale should bring Apple under the protection of third-party licenses.” *Id.* Finding no basis to reconsider its prior ruling, the Court denies Apple’s motion for judgment as a matter of law on this basis.

**(5) *Royalty rate***

Apple also asserts in its motion that “no reasonable jury” could disagree with its damages expert. Docket No. 775 at 26–27. The Court declines to substitute its judgment for that of the jury and will not reweigh the evidence and credit Apple’s expert over Mr. Weinstein. The jury was free to disbelieve Apple’s expert and credit Mr. Weinstein’s testimony, which provided substantial evidence for its verdict. *i4i Ltd. P’ship v. Microsoft Corp.*, 598 F.3d 831, 848 (Fed. Cir. 2010). Accordingly, Apple’s motion for JMOL on this basis is denied.

**C. Apple’s Motion for Judgment as Matter of Law and New Trial on Willfulness**

The issue of willfulness was bifurcated; after the jury returned a verdict finding Apple liable for infringement and damages, the jury was tasked with the question of willfulness in a second “phase” of trial. After approximately two hours and thirty minutes of deliberations, the jury ultimately returned a verdict that Apple’s infringement was willful.

Apple now moves for judgment as a matter of law and a new trial on the issue of willfulness. A jury’s finding of willfulness merely “opens the door” to the Court exercising its

discretion to enhance damages. *See Read Corp. v. Portec, Inc.*, 970 F.2d 816, 826 (Fed. Cir. 1992) (“a finding of willful infringement does not mandate that damages be enhanced”); *Halo Elecs., Inc. v. Pulse Elecs., Inc.*, 136 S. Ct. 1923, 1933 (2016).

As is discussed in detail, *infra* Section II.A, the Court has exercised its discretion to decline enhancing damages in this matter. Because the Court declines to award enhanced damages, Apple’s willfulness JMOL and new trial motions are moot. *See Presidio Components, Inc. v. American Technical Ceramics Corp.*, Case No. 14-02061-H-BGS (S.D. Cal. Aug. 17, 2016) at 21 (noting that Defendant’s JMOL motion on willfulness was “essentially moot because the Court, exercising its sound discretion, ultimately declines to award [Plaintiff] enhanced damages despite the jury’s finding of willful infringement.”); *Schwendimann v. Arkwright Advanced Coating, Inc.*, No. CV 11-820 (JRT/HB), 2018 WL 3621206, at \*21 (D. Minn. July 30, 2018) (citing *Laitram Corp. v. NEC Corp.*, 115 F.3d 947, 955 (Fed. Cir. 1997)) (“A court’s denial of enhanced damages renders a motion for judgment as a matter of law on willful infringement moot.”); *Greatbatch Ltd. v. AVX Corp.*, Case No. 13-723 (D. Del. Mar. 30, 2018) at 14 (concluding that, after two trials on the asserted patents and products and having reviewed all of the evidence, the Court determined it could deny enhanced damages “before a finding on willfulness”); *Erfindergemeinschaft Uropep Gbr v. Eli Lilly And Company et al*, Case No. 2:15-cv-1202 (Bryson, J.), Docket No. 346, 4/25/17 Trial Tr. at 1390:25–1391:3 (granting Rule 50 motion on willfulness); 1500:14–22 (later explaining that, “had willfulness gone to the jury and had there been a verdict of willful infringement in this case, I – based on the evidence I heard in the course of the trial, I would have exercised my discretion not to award enhanced damages under 35 USC, Section 284.”).

#### **D. Apple's Remaining Arguments for New Trial**

In its omnibus motion, Apple moved for JMOL and a new trial on several grounds discussed above. Aside from the arguments already considered, Apple made additional arguments which the Court addresses below.

##### **(1) *Domain name service system instruction***

At trial, the Court gave the jury the following instruction:

You have heard discussion about a construction for the word “domain name service.” You were instructed that the construction for “domain name service system,” an element of all of the asserted claims of the ’504 and ’211 patents, does not incorporate or include the Court’s construction for the term “domain name service.”

4/10 AM Tr. at 128:10–15. Apple challenges this instruction as confusing to the jury, prejudicial to Apple and irrelevant to the issues in the case. Docket No. 775 at 35. In its motion, Apple does not explain what prejudice Apple has suffered as a result of the instruction. In fact, Apple explains in its reply that its own expert “*agreed* that the ‘domain name service’ construction was not incorporated into ‘domain name service system.’ ” Docket No. 783 at 14 (citing 4/9 AM Tr. at 113:23–114:1). The Court can identify at least one instance in the record where Dr. Blaze, in response to Apple’s examination, suggested that the construction for “domain name service” applied to “domain name service system.” *See* 4/9 AM Tr. at 104:13–24 (“Q. I have put up on Slide 57 three of the Court’s claim constructions. Okay? A. Right. Q. Secure communication link, domain name service, and the indication term. Okay? A. Yes. Q. Which one of these claim constructions does not apply to the ’504 and ’211 patents, sir? A. Well, the – I’m not sure what – I’m not sure what you’re asking. There’s a domain name service. There’s an indication. There’s a secure communication. All these terms are here.”). This testimony justifies the Court’s instruction.

But even if Apple was correct that the instruction was unnecessary, it is unclear what prejudice Apple suffered from the Court explaining the nuances of its claim construction to the jury. Accordingly, this is not a basis for a new trial.

**(2) *EMVR and hypothetical negotiation instructions***

Apple asks the Court for a new trial based on its exclusion of Apple's proposed EMVR instruction and based upon the formulation of its hypothetical negotiation instruction. As the Court concluded above, Mr. Weinstein's testimony did not implicate the entire market value rule, so any instruction thereto would have been inappropriate. Indeed, as in the 417 case, "[t]hough the prices of Apple's devices were never presented to the jury, the Court instructed the jury, out of an abundance of caution, not to rely on the full price of any Apple product." 417 action, Docket No. 1079; *see* Docket No. 721 (jury instructions) ("VirnetX has relied on license agreements in which royalties were based on a percentage of the entire price of the licensed end-products. But in determining a reasonable royalty, you must not rely on the overall price of Apple's accused products at issue in this case.").

The Court presumes that the jury followed this instruction. *See Francis v. Franklin*, 471 U.S. 307, 324 n.9 (1985). Again, "[f]urther instructions on the precise contours of the entire market value rule may have led the jury to mistakenly believe that it could apply the rule despite the fact that the record did not support the rule's applicability." 417 action, Docket No. 1079 at 28.

Apple also challenges the Court's instructions on the hypothetical negotiation. The Court provided the same instruction in the 417 retrial and again concludes that the instruction as a whole properly conveyed to the jury that the reasonable royalty should reflect the fair market value of the technology. Moreover, Apple has not identified what meaningful difference there is between the Court's instruction and Federal Circuit case law. *Compare* 4/10 AM Tr. at 135:5–8, *with Lucent*,

580 F.3d at 1324 (explaining that, in analyzing the hypothetical negotiation, the Court must ask, “Had the Infringer not infringed, what would the Patent Holder have made?” and that “the hypothetical negotiation or the ‘willing licensor-willing licensee’ approach, attempts to ascertain the royalty upon which the parties would have agreed had they successfully negotiated an agreement just before infringement began.”) (internal citations omitted). Accordingly, Apple is not entitled to a new trial on this basis.

Accordingly, Apple’s motion for a new trial based on the hypothetical negotiation instruction and on the exclusion of its EMVR instruction is **DENIED**.<sup>1</sup>

### **(3) *Great weight of the evidence***

Apple also argues that it is entitled to a new trial because the damages award is excessive and against the weight of the evidence. Docket No. 775 at 42–44. Apple reasons that VirnetX improperly asked the jury to “punish” Apple for its previous infringement and that Mr. Weinstein’s damages methodology is faulty. The Court has discussed Mr. Weinstein’s methodology at length, and declines to reconsider it here.

Apple cites the following statement as “suggesting that Apple should get a worse deal than every other licensee because it previously infringed”: “[Apple] showed up having been trespassing on this property for years. And so in 2013 they say, Yeah, we’re an agreed infringer, but we want a better rate than anyone who showed up and took a license voluntarily. We want a better rate than anybody.” Docket No. 775 at 43; 4/10 Tr. at 161:12–16. Apple fails to articulate, however, how this comment suggests that Apple should be punished for its previous infringement. *Id.*

In support, Apple cites *Gilster v. Primebank*, 747 F.3d 1007, 1011 (8th Cir. 2014). But the remarks in this case are a far cry from those at issue in *Gilster*. In *Gilster*, the Eighth Circuit

---

<sup>1</sup> The Court notes that it provided the same hypothetical negotiation instruction in the 417 action and, likewise, denied Apple’s motion for a new trial on this basis as well. *See* 417 action, Docket No. 1079 at 28.

reversed a district court’s denial of a new trial motion when, during closing argument in a sexual harassment case, Plaintiff’s counsel referred “to an experience in her own life . . . ‘plainly calculated to arouse the jury’s sympathy’ . . . [and] ended the argument by ‘giving’ the jury the ‘power and the responsibility for correcting injustices.’ ” 747 F.3d at 1011. Specifically, the Eighth Circuit noted that “improper vouching permeated counsel’s rebuttal argument.” *Id.* These circumstances are not present here, and the Court declines to grant a new trial based on VirnetX’s closing argument.

**(4) *Mr. Weinstein’s testimony***

Apple also asks the Court to grant a new trial because Mr. Weinstein’s testimony should have been excluded. Docket No. 775 at 39. As discussed at length above and in previous orders, Mr. Weinstein’s methodology withstands Apple’s *Daubert* challenge, and Apple is not entitled to a new trial on this basis.

**(5) *The exclusion of Patent Office (“PTO”) proceedings***

Apple contends that it is entitled to a new trial because the Court excluded evidence from parallel PTO proceedings. Docket No. 775 at 39. Apple suggests that the fact that the PTO has issued final written decisions finding each claim unpatentable over the prior art is relevant to damages. *Id.*

The Court is not persuaded that exclusion of the PTO proceedings warrants a new trial. VirnetX’s appeals of those proceedings are ongoing, and none of the asserted claims has been cancelled. It is particularly unclear what probative value the PTO proceedings have in light of the fact that invalidity is not an issue in this case.

To the extent Apple argues the decisions are relevant to damages, the relevance of the decisions is minimal because a number of the “decisions” cited in Apple’s offer of proof<sup>2</sup> on PTO proceedings were issued *after* the parties’ alleged hypothetical negotiation date of September 2013. *Compare* Docket No. 721 at 18 (“If infringement is found, the date of the hypothetical negotiation would be September 2013, when the redesigned versions of VPN on Demand and FaceTime were released.”), *with* Docket No. 692 (Apple’s offer of proof).

Apple suggests that the evidence would have been relevant to the utility and advantages of the patented property over old modes or devices. *Id.* But, contrary to its assertions, Apple was not *precluded* from introducing evidence that the claimed invention “has no utility or advantages over old modes or devices.” *Id.* The Court only prohibited the use of the PTO proceedings, and Apple was free to present whatever evidence relating to *Georgia-Pacific* factors 9 and 10 it so chose. Apple is not entitled to a new trial on this basis.

\* \* \*

Having considered each of Apple’s arguments in its motion for judgment as a matter of law and a new trial of noninfringement, the Court concludes that Apple’s motion should be **DENIED**.

## **II. VIRNETX’S MOTION FOR ENTRY OF JUDGMENT AND EQUITABLE AND STATUTORY RELIEF (DOCKET NO. 774)**

In its motion for post-trial relief, VirnetX seeks (1) enhanced damages; (2) attorneys’ fees pursuant to 35 U.S.C. § 285; (3) supplemental damages; (4) an injunction or sunset royalty; (5) pre-judgment interest; and (6) post-judgment interest and costs. The Court addresses each request in turn.

---

<sup>2</sup> The Court notes that, in its offer of proof on PTO proceedings, Apple only “submits that the evidence is relevant and admissible because it tends to show . . . VirnetX’s damages demand is excessive and unjustified given that the asserted claims have been held unpatentable, subject only to Federal Circuit appeal.” Docket No. 692 at 21–22.

### A. Enhanced damages

The jury returned a verdict finding Apple's infringement willful. Docket No. 729. VirnetX argues that a 100 percent enhancement of the jury's verdict is warranted under the *Read* factors. The Court disagrees. As detailed below, the Court concludes that enhancement is inappropriate under the totality of the circumstances.

A court may enhance the jury's damages award by up to three times. 35 U.S.C. § 284. "The paramount determination in deciding enhancement and the amount thereof is the egregiousness of the defendant's conduct based on all the facts and circumstances." *Read Corp. v. Portec Inc.*, 970 F.2d 816 (Fed. Cir. 1992). When deciding how much to award in enhanced damages, district courts often apply the non-exclusive factors articulated in *Read*. . . ." *Georgetown Rail Equip. Co. v. Holland L.P.*, 867 F.3d 1229, 1244–45 (Fed. Cir. 2017). The non-exclusive *Read* factors used to evaluate whether to enhance damages—and the amount of any enhancement—include the following: (1) whether the infringer deliberately copied the ideas of another; (2) whether the infringer investigated the scope of the patent and formed a good-faith belief that it was invalid or that it was not infringed; (3) the infringer's behavior as a party to the litigation; (4) the defendant's size and financial condition; (5) the closeness of the case; (6) the duration of the defendant's misconduct; (7) remedial action by the defendant; (8) the defendant's motivation for harm; and (9) whether the defendant attempted to conceal its misconduct. *Read*, 970 F.2d at 827.

An award need not rest on any particular factor, and not all relevant factors need to weigh in favor of an enhanced award. See *SRI Int'l, Inc. v. Advanced Tech. Labs., Inc.*, 127 F.3d 1462, 1469 (Fed. Cir. 1997). While the *Read* factors are helpful to the Court's exercise of its discretion, an analysis focused on "egregious infringement behavior" is the touchstone for determining an



award of enhanced damages rather than a more rigid, mechanical assessment. *See Finjan, Inc. v. Blue Coat Sys., Inc.*, No. 13-cv-3999, 2016 WL 3880774, at \*16 (N.D. Cal. July 18, 2016).

**(1) Copying**

The first *Read* factor is “whether the infringer deliberately copied the ideas or design of another.” As to this factor, VirnetX argues that Apple copied (or was reckless to whether it copied) VirnetX’s inventive methods and systems. Docket No. 774 at 9. Apple contests that it copied any of VirnetX’s ideas or designs and argues that its engineers made a good-faith effort to redesign its features to avoid further infringement. Docket No. 778 at 6. In its reply, VirnetX maintains that the copying factor is met here because Apple went “back to including already-adjudicated infringing components in its products.” Docket No. 782 at 4.

While substantial evidence was presented at trial supporting both sides’ view of how the redesigned products worked, the jury presumably found the facts to be more consistent with VirnetX’s characterization than Apple’s. By replicating the operation and functionality of its products that were already adjudicated to infringe, Apple “copied” the ideas or designs of VirnetX. *See PPC Broadband, Inc. v. Corning Optical Commc’ns RF, LLC*, No. 511CV761GLSDEP, 2016 WL 6537977, at \*6 (N.D.N.Y. Nov. 3, 2016), appeal dismissed, No. 16-4106, 2016 WL 10655596 (2d Cir. Dec. 12, 2016). After the 417 product functionality was adjudicated to infringe, Apple was at least deliberately reckless towards copying VirnetX’s ideas when it reimplemented those features in the redesigns. *Barry v. Medtronic, Inc.*, 250 F. Supp. 3d 107, 114 (E.D. Tex. 2017). Accordingly, the “copying” factor favors enhancement.

**(2) Investigation and good-faith belief**

Next, the Court considers “whether the infringer, when he knew of the other’s patent protection, investigated the scope of the patent and formed a good-faith belief that it was invalid

or that it was not infringed.” As the Court explained in the 417 action, “Apple contends that its good-faith belief in its infringement positions and its post-grant challenges to the patents’ validity mitigate against an award of enhanced damages, but after the jury’s verdict finding the claims infringed and valid, its reliance on these beliefs was no longer reasonable.” 417 action, Docket No. 1079 at 47 (citing *Mondis Tech. Ltd. v. Chimei InnoLux Corp.*, 822 F. Supp. 2d 639, 652 (E.D. Tex. 2011), *aff’d sub nom. Mondis Tech. Ltd. v. InnoLux Corp.*, 530 F. App’x 959 (Fed. Cir. 2013) (enhancing damages for post-verdict infringement despite defendant’s argument that it had a good-faith belief in its appellate positions); *VirnetX*, 767 F.3d at 1324 (noting that actions by the PTO are of limited value when attempting to establish a good faith belief of invalidity)).

With respect to infringement and redesigned VOD, although the jury did not agree, Apple had at least a good-faith belief that the redesign did not infringe based on its location-based reconfiguration. As to FaceTime, the Court notes that Apple raised a noninfringement argument regarding the “indication” term for which there was substantial evidence. The Court cannot conclude on this record that Apple’s noninfringement positions were not in good faith.

Apple also obtained an written opinion of counsel from Mr. Lee Van Pelt, but the Court assigns his opinion little weight because it was obtained after the redesigns were released. Apple claimed to have obtained an earlier, “oral” opinion from Mr. Van Pelt, but the witness alleged to have received the oral opinion stated that he did not “have personal knowledge of a verbal opinion from Mr. Pelt.” 4/11 PM Tr. at 28:24–29:2. The Court likewise assigns this unsubstantiated opinion little weight.

On balance, Apple did not rely on a timely opinion of counsel. To the extent it had a good-faith belief in invalidity based on the PTO proceedings, the weight of such belief is minimized by the fact that invalidity was no longer an issue in this case. But the Court is persuaded that Apple

maintained a good faith belief that its redesigns did not infringe. Indeed, unlike in the 417 retrial,<sup>3</sup> Apple maintained reasonable noninfringement positions for both products at issue in the case. Considering the totality of the circumstances, this factor is neutral.

**(3) *Infringer's behavior as a party to the litigation***

For the next *Read* factor, the Court considers Apple's "behavior as a party to the litigation." The Court is not persuaded that Apple committed litigation misconduct in this trial. The Court agrees with VirnetX that Apple violated a ruling *in limine* by cross-examining VirnetX's CEO, Mr. Kendall Larsen, on his divorce. 4/4 PM Tr. at 168:20–168:1. The Court also notes that Apple's expert, Dr. Blaze, implied to the jury that the construction of "domain name service system" incorporated the construction of "domain name service," which the Court remedied by including a corrective instruction in its jury charge. When the Court considers these actions independently of the 417 action and in the context of this case alone, the Court is not persuaded that Apple's behavior as a party to the litigation supports enhancement, and this factor weighs neutral.

**(4) *Size and financial condition***

*Read* factor 4, the infringer's size and financial condition, weighs in favor of enhancement. This factor can weigh against enhancement when the infringer is in such perilous financial condition that an award of enhanced damages might put it out of business. *Idenix Pharm. LLC v. Gilead Scis., Inc.*, 271 F. Supp. 3d 694, 701 (D. Del. 2017) (citing *Virginia Panel Corp. v. Mac Panel Co.*, 887 F.Supp. 880, 885 (W.D. Va. 1995), *aff'd*, 133 F.3d 860 (Fed. Cir. 1997)). In cases

---

<sup>3</sup> After the first trial in the 417 action, the Federal Circuit affirmed the finding that none of the asserted claims were invalid and that VPN on Demand infringed. *VirnetX, Inc.*, 767 F.3d at 1313–14 ("For the reasons that follow, we affirm the jury's findings that none of the asserted claims are invalid and that many of the asserted claims of the '135 and '151 patents are infringed by Apple's VPN On \*1314 Demand product."). Accordingly, for the retrial, infringement by VPN on Demand was not submitted to the jury.

where enhanced damages would not unduly prejudice the defendant's noninfringing business, it can weigh in favor of enhancement. *See, e.g., Creative Internet Advert. Corp. v. Yahoo! Inc.*, 689 F. Supp. 2d 858, 866 (E.D. Tex. 2010); *Ericsson Inc. v. TCL Commc'n Tech. Holdings, Ltd.*, No. 2:15-CV-00011-RSP, 2018 WL 2149736, at \*11 (E.D. Tex. May 10, 2018).

As the Court noted in the 417 action, "it is undisputed that Apple is one of the largest and most financially successful companies in the world." 417 action, Docket No. 1079 at 45. *See also Arctic Cat Inc. v. Bombardier Recreational Prod., Inc.*, 198 F. Supp. 3d 1343, 1352 (S.D. Fla. 2016), *aff'd*, 876 F.3d 1350 (Fed. Cir. 2017) ("Where, as here, BRP is a multi-billion dollar enterprise and the market leader—due in significant part to sales of products found to willfully infringe Arctic Cat's patents—enhancement of damages is particularly warranted."). And the Court cannot conclude "that a trebled award would "unduly prejudice [Apple's] non-infringing business." *Georgetown Rail Equip. Co. v. Holland L.P.*, No. 6:13-CV-366, 2016 WL 3346084, at \*19 (E.D. Tex. June 16, 2016), *aff'd*, 867 F.3d 1229 (Fed. Cir. 2017).

#### **(5) Closeness of the case**

The next *Read* factor, "closeness of the case," disfavors enhancement. In ruling on Apple's post-trial motions, the Court determined that the verdict was supported by substantial evidence. But the Court is also persuaded that Apple presented reasoned and justified defenses. *Apple Inc. v. Samsung Elecs. Co.*, 258 F. Supp. 3d 1013, 1033 (N.D. Cal. 2017) (citing *Finjan*, 2016 WL 3880774 at \*17; *Power Integrations, Inc. v. Fairchild Semiconductor Int'l, Inc.*, No. 09-CV-05235-MMC, 2017 WL 130236, at \*4 (N.D. Cal. Jan. 13, 2017)). A testament to the closeness of the case is the fact that the jury deliberated for over five hours between the two phases of the case.

**(6) Duration of misconduct and (7) remedial action**

Factors 6 and 7—the duration of the misconduct and the remedial action taken by the infringer—weigh against enhancement. Apple took remedial action quickly after the 417 verdict to redesign its products, collaborating with engineers, management, and its legal team. *See, e.g.*, 4/6/18 AM Tr. at 68:24–69:19; 4/11 PM Tr. at 6:24–8:21, 13:14–15:21. The Court is not persuaded that Apple’s remedial action was not in good faith. Although ultimately the jury found that the redesign was infringing, the Court declines to penalize Apple for shifting course and attempting to make changes in light of the 417 verdict.

**(8) Motion for harm**

Similarly, there is no evidence of a “motivation for harm” that would support enhancement. VirnetX asserts that Apple’s release of a redesign and participation in post-grant review at the Patent Office is evidence of its “motivation to harm.” Docket No. 774 at 24–25. However, VirnetX points to no evidence that Apple’s decision to release redesigns was anything other than profit-driven, let alone motivated to harm VirnetX. In the absence of any supporting authority, the Court also declines to infer motivation to harm a patent owner from participation in post-grant proceedings. Accordingly, this factor is neutral. *See, e.g., Georgetown Rail*, No. 6:13-CV-366, 2016 WL 3346084, at \*20 (finding this factor neutral when there was “nothing to suggest that Holland acted out of spite or ill-will toward Georgetown or for any reason other than a desire to capture a piece of the market”); *Internet Machines*, 2013 WL 4056282, at \*20; *Spectralytics, Inc. v. Cordis Corp.*, 834 F. Supp. 2d 920, 924 (D. Minn. 2011), *aff’d*, 485 F. App’x 437 (Fed. Cir. 2012).

**(9) Concealment**

Finally, the Court considers the ninth *Read* factor, whether Apple concealed its infringement. As the Court described in Section I.A.(1), there are three relevant versions of FaceTime in this case. First, there is the adjudged-infringing version of FaceTime that was at issue in the 417 action. A second version—that the parties agree did not infringe—was implemented in April 2013 and relayed 100 percent of calls. The jury determined the third version of FaceTime, at issue in this case, infringed each of the asserted claims.

The third version of FaceTime was released on September 18, 2013, the day iOS 7 was released. 4/5 PM Tr. at 216:14–17 (Thirumalai); *id.* at 250:13–18. On that date, Apple stopped relaying 100 percent of FaceTime calls. Apple’s corporate representative and fact witness for FaceTime, Mr. Thirumalai, was instrumental in enacting the 100 percent relay non-infringing alternative in April 2013. He was also a member of the team studying ways to reduce relay usage. Mr. Thirumalai made a server change in May of 2013 facilitating peer-to-peer calls. *Id.* at 216:18–218:6; 214:25–215:3; 247:23–248:16. Despite this, however, Apple misrepresented to VirnetX’s counsel during two different depositions—both occurring long after Apple had stopped relaying 100 percent of FaceTime calls—that Apple had never made the change back to supporting direct peer-to-peer FaceTime calls. 4/6 AM Tr. at 25:14–16 (Thirumalai) (“Q: On May 15, 2014, did you tell us that 100 percent of FaceTime calls were being relayed? A: Yes.”); *id.* at 29:4–14; *cf.* 4/11 AM Tr. at 121:4–122:5 (Stauffer) (testifying that, as of 2014, it was *not true* that Apple had relayed 100 percent of FaceTime calls since April 2013).

The evidence clearly supports a finding that Apple attempted to conceal its infringement, and this factor weighs in favor of enhancement.

\* \* \*

Having considered each *Read* factor, the Court concludes that enhancement is inappropriate. In favor of enhancement are the facts that Apple is a large, successful company, that Apple “copied” VirnetX’s ideas in a redesign and that an Apple engineer and corporate witness misled VirnetX in depositions about key infringement facts. The Court should not be interpreted as condoning such conduct. At the same time, however, this case was close, the misconduct brief and the remediation effort significant.

Aside from the *Read* factors, the Court also considers the size of the jury verdict. Enhanced damages inherently deter future, similar conduct. *Affinity Labs of Texas, LLC v. BMW N. Am., LLC*, 783 F. Supp. 2d 891, 899 (E.D. Tex. 2011) (“General deterrence of infringing activity is also a factor to be considered.”). The jury’s damages number is supported by the evidence in this case. But the Court is not persuaded that any enhancement of the verdict would lead to any additional deterrence of future conduct.

“Awards of enhanced damages under the Patent Act over the past 180 years establish that they are not to be meted out in a typical infringement case, but are instead designed as a ‘punitive’ or ‘vindictive’ sanction for egregious infringement behavior. The sort of conduct warranting enhanced damages has been variously described [by the Supreme Court] as willful, wanton, malicious, bad-faith, deliberate, consciously wrongful, flagrant, or—indeed—characteristic of a pirate.” *Halo*, 136 S. Ct. at 1932. In light of its full consideration of the totality of the circumstances, the Court cannot conclude that Apple’s conduct mandates enhancement. *See id.* at 1933 (“Section 284 allows district courts to punish the full range of culpable behavior. Yet none of this is to say that enhanced damages must follow a finding of egregious misconduct.”). Accordingly, the Court exercises its discretion to deny VirnetX’s request for enhanced damages.

**B. Attorneys' fees pursuant to 35 U.S.C. § 285**

VirnetX asks the Court to award it attorneys' fees for "the reasons related to the closeness of the case and litigation conduct under the *Read* factors." Docket No. 774 at 30. As the Court has concluded above, Apple's behavior in this case did not amount to litigation misconduct, and the substantive strength of Apple's positions do not stand out from others. Accordingly, VirnetX's request for attorneys' fees is **DENIED**.

**C. Supplemental damages**

VirnetX requests supplemental damages at the jury's implied royalty rate to account for units not included in the jury verdict. Docket No. 774 at 33–34. Apple requests the Court stay any accounting of these units until the resolution of all appeals in this case, in the 417 action and of the PTO proceedings. Docket No. 778 at 27. According to Apple, the appeals could moot damages awards in this action. *Id.*

Because Apple does not oppose VirnetX's request for supplemental damages at the implied royalty rate, the Court **GRANTS** VirnetX's motion for supplemental damages. A stay of accounting is not warranted in this case, and Apple is directed to provide VirnetX an accounting of post-verdict, pre-judgment infringing units within **thirty (30) days**.

**D. Injunction or sunset royalty**

VirnetX seeks a permanent injunction. Docket No. 774 at 35–42. If the Court declines to enter an injunction, VirnetX alternatively seeks a sunset royalty. *Id.*

**(1) Injunction**

"According to well-established principles of equity, a plaintiff seeking a permanent injunction must satisfy a four-factor test before a court may grant such relief." *eBay Inc. v. MercExchange, L.L.C.*, 547 U.S. 388, 391 (2006). "A plaintiff must demonstrate: (1) that it has



suffered an irreparable injury; (2) that remedies available at law, such as monetary damages, are inadequate to compensate for that injury; (3) that, considering the balance of hardships between the plaintiff and defendant, a remedy in equity is warranted; and (4) that the public interest would not be disserved by a permanent injunction.” *Id.* Because the Court concludes that VirnetX failed to demonstrate irreparable injury, the Court will deny VirnetX’s request for an injunction.

As irreparable injury, VirnetX argues Apple’s infringement has prevented VirnetX from capitalizing on its own product, the Gabriel application. According to VirnetX, the Gabriel product—for which it charges a fee—is forced to compete against Apple’s infringing products that include VirnetX’s technology at no additional cost. *Id.* at 35–36. VirnetX contends that this harm is connected to the patented features and demand for the infringing product because Apple “must use VirnetX’s patented technology in order to offer its Facetime and the infringing mode of VPN on Demand.” *Id.* at 36. VirnetX also argues that the prevalence of Apple’s products in the marketplace have harmed VirnetX’s reputation as an innovator. *Id.* With respect to whether monetary damages can adequately compensate VirnetX, VirnetX claims that “an ongoing royalty fails to account for other contractual terms that VirnetX could otherwise negotiate with Apple, such as terms prohibiting Apple from challenging the validity of the patents and terms constraining Apple’s ability to contest whether royalties are owed for future products,” and that “[n]o monetary remedy can compensate for these terms.” *Id.* at 37–38. For balance of harms, VirnetX suggests that the jury’s willfulness finding and the fact that “VirnetX’s patented technology is critical to its business” tilts the equities towards VirnetX. *Id.* at 39. Finally, VirnetX maintains that an injunction promotes the public interest because “Apple’s infringing products are not essential for public health or welfare.” *Id.* at 39.

In 2013, Judge Davis considered similar arguments from VirnetX regarding an injunction and rejected them. *VirnetX Inc. v. Apple Inc.*, 925 F. Supp. 2d 816, 845 (E.D. Tex. 2013). The only operative fact that has changed since 2013 is that, now, VirnetX has released its Gabriel product.<sup>4</sup> The Court now considers whether the release of Gabriel justifies entry of an injunction.

VirnetX suggests that, now that Gabriel has been released, “VirnetX . . . directly competes with Apple for end-users of secure communication software with the patent-practicing features and continues to suffer competitive harm in the form of lost sales and reputational harm caused by Apple’s distribution of the infringing features.” Docket No. 774 at 34.

The Court is not persuaded that Gabriel competes with the infringing Apple products. Gabriel, unlike FaceTime or redesigned VOD, is a cross-platform application compatible with Apple, Android, Windows, and Linux devices. Docket No. 778-9. These non-Apple devices cover 50 percent of the U.S. smartphone market and nearly 80 percent of the U.S. computer operating system market. For the majority of Gabriel-compatible systems, Apple’s products present no competition at all.

The Court also notes that VirnetX has entered into a license with another competitor: Skype. Skype has been downloaded over 1 billion times on Android (non-Apple) devices, 4/5 PM Tr. 162:1–163:2, whereas Gabriel has only been downloaded between 500 and 1,000 times on the Android platform. Docket Nos. 778-12, 778-25. VirnetX’s decision to license a significant competitor who posed a major threat to its flagship product cautions against any finding of irreparable harm. *Nichia Corp. v. Everlight Ams., Inc.*, 855 F.3d 1328, 1343 (Fed. Cir. 2017).

---

<sup>4</sup> The Court notes, however, that Judge Davis concluded that, even if Gabriel had been on the market, “Apple does not directly compete with VirnetX. Apple sells phones, not security software. Additionally, Apple’s sale of cell phones has not restricted VirnetX’s ability to market and sell its Gabriel technology to other tablet, cellphone or computer manufacturers. VirnetX’s damages are limited to the loss of Apple as a customer.” *VirnetX*, 925 F.Supp.2d at 846.

Even in markets where Apple products and Gabriel do compete, Gabriel functionality differs from FaceTime and redesigned VOD. VirnetX markets Gabriel as an “integrated set of secure applications including Mail, Messaging, File Sharing & Backup, Voice Calls and Video Calls.” Docket No. 778-9. VirnetX markets the security of Gabriel to its small-to-medium business customers, seeking to appeal to companies that place a high value on security. Docket No. 778-14 at 17. FaceTime and VPN on Demand, however, do not offer mail, messaging, or file-sharing and back up, and Apple primarily serves the consumer market. *See, e.g.*, PX-1134–8; PX-1182–8; PX-1186–5.

The record amply demonstrates that VirnetX’s Gabriel product and Apple’s accused products do not compete. The only other irreparable harm VirnetX identifies is reputational harm. But the reputational harm VirnetX complains of—being deemed a “patent troll” in online publications—may occur regardless of whether the Court enters an injunction. Moreover, it is unclear that Apple’s actions have led to these statements.

On this record, the Court concludes that VirnetX will not suffer irreparable harm absent an injunction, and VirnetX’s request for an injunction is **DENIED**.

## **(2) *Sunset royalty***

In the alternative, VirnetX seeks an ongoing royalty. Apple asks this Court to deny an ongoing royalty only on the basis of its JMOL and new trial arguments, which the Court has already rejected. Docket No. 778 at 28.

The Federal Circuit has interpreted 35 U.S.C. § 283 to permit a court to award an ongoing royalty for patent infringement in lieu of an injunction. *Prism Techs. LLC v. Sprint Spectrum L.P.*, 849 F.3d 1360, 1377 (Fed. Cir. 2017). Ongoing royalties may be based on a post-judgment hypothetical negotiation using the Georgia-Pacific factors. *Arctic Cat Inc. v. Bombardier*

*Recreational Prod. Inc.*, 876 F.3d 1350, 1370 (Fed. Cir. 2017). The amount of the ongoing royalty is “committed to the sound discretion of the district court” to be determined in accordance with principles of equity. *Amado v. Microsoft Corp.*, 517 F.3d 1353, 1364 n.2 (Fed. Cir. 2008).

At the outset, the Court determines that imposition of an ongoing royalty is an appropriate exercise of the Court’s discretion in this case. *Whitserve, LLC v. Computer Packages, Inc.*, 694 F.3d 10, 35 (Fed. Cir. 2012). Here, it is clear from the verdict form that the jury awarded damages for past infringement. *See* Docket No. 723 at 4 (“What royalty do you find, by a preponderance of the evidence, would fairly and reasonably compensate VirnetX for any infringement that you have found”); *see also Whitserve*, 694 F.3d at 35 (“The jury was instructed to award ‘damages,’ which by definition covers only past harm.”). Because the jury’s verdict does not compensate VirnetX for future infringement, the Court will award an ongoing royalty. *Telcordia Techs., Inc. v. Cisco Sys., Inc.*, 612 F.3d 1365, 1379 (Fed. Cir. 2010).

The jury’s implied royalty rate is \$1.20, which is supported by substantial evidence. *See supra* Section II.B. From this starting point, the Court conducts a renewed analysis of a reasonable royalty based on a post-verdict hypothetical negotiation. *See Erfindergemeinschaft UroPep GbR v. Eli Lilly & Co.*, No. 2:15-cv-1202-WCB, 2017 WL 3034655, at \*7 (E.D. Tex. July 18, 2017). The burden is on VirnetX to show that it is entitled to a royalty rate in excess of the rate initially determined by the jury. *Creative Internet Advertising Corp. v. Yahoo! Inc.*, 674 F. Supp. 2d 847, 855 (E.D. Tex. 2009).

“There is a fundamental difference . . . between a reasonable royalty for pre-verdict infringement and damages for post-verdict infringement.” *Amado*, 517 F.3d. at 1361. “Prior to judgment, liability for infringement, as well as the validity of the patent, is uncertain, and damages are determined in the context of that uncertainty.” *Id.* at 1362. Once a judgment of validity and

infringement has been entered, however, the calculus is markedly different because different economic factors are involved. *Id.* (citing *Paice LLC v. Toyota Motor Corp.*, 504 F.3d 1293, 1315 (Fed. Cir. 2007)).

VirnetX asks the Court to enhance the ongoing royalty to \$3.00 per unit based on the “totality of the circumstances . . . in combination with the changed post-judgment economic circumstances.” Docket No. 782 at 14–15. The Court declines to enhance the verdict for willfulness for at least the reasons detailed in its analysis of enhanced damages. Importantly, VirnetX bears the burden to show that an enhanced royalty rate is appropriate, and its general statements about “changed circumstances” do not meet that burden. Accordingly, the Court will not enhance the ongoing royalty and **SETS** the ongoing royalty at the jury’s implied rate of \$1.20 per unit. Apple is **ORDERED** to provide an accounting of infringing units on a quarterly basis.

#### **E. Pre-judgment interest**

VirnetX seeks pre-judgment interest at the prime rate compounded annually, to be applied to the jury award beginning at the date of the hypothetical negotiation. Docket No. 774 at 43.

Apple asks the Court to withhold prejudgment interest because “the jury award was generous enough” and will compensate “VirnetX far beyond what the parties would have agreed to in September 2013.” Docket No. 778 at 44–45. If any prejudgment interest is awarded, Apple requests that it be measured from February 4, 2016 because VirnetX “caused” delay by requesting consolidation and “forc[ing] deconsolidation and two retrials.” *Id.* at 45.

Apple’s argument against prejudgment interest was not raised in response to the 417 retrial. *See* Docket No. 1079 at 55–57. Regardless, however, the Court declines to assign responsibility for the lengthy timeline of this case to either party. The Court is not persuaded by Apple’s

argument against prejudgment interest and **AWARDS** prejudgment interest to VirnetX at the prime rate, compounded annually, beginning at the date of the hypothetical negotiation.

#### **F. Post-judgment interest and costs**

VirnetX seeks post-judgment interest, which Apple opposes only insofar as “the Court has not yet ruled on the parties’ post-trial motions, which may affect whether VirnetX is the prevailing party.” Docket No. 778 at 45. Having resolved the parties’ motions above, the Court **AWARDS** VirnetX post-judgment interest pursuant to 28 U.S.C. § 1961. Additionally, VirnetX is the prevailing party and is **AWARDED** costs pursuant to Federal Rule of Civil Procedure 54(d) and 28 U.S.C. §1920.

### **CONCLUSION**

As set forth above, the Court has ruled as follows:

- Apple’s Omnibus Motion for Judgment as a Matter of Law under Rule 50(b) and for a New Trial (Docket No. 775) is **DENIED** in all respects; and
- VirnetX’s Motion for Entry of Judgment and for Equitable and Statutory Relief (Docket No. 774) is **DENIED-IN-PART** and **GRANTED-IN-PART**.

Specifically, with respect to VirnetX’s Motion, the Court has ruled as follows:

- VirnetX’s request for enhanced damages is **DENIED**;
- VirnetX’s request for attorneys’ fees is **DENIED**;
- VirnetX’s request for supplemental damages is **GRANTED**;
- VirnetX’s request for an injunction is **DENIED**;
- VirnetX’s request for a sunset royalty is **GRANTED** and the royalty rate is set at \$1.20;
- VirnetX’s request for pre-judgment and post-judgment interest are **GRANTED**; and
- VirnetX’s request for costs is **GRANTED**.

In light of the above, Apple's Motions for Judgment as a Matter of Law Pursuant to Rule 50(a) (Docket Nos. 713, 714 and 718) are **DENIED AS MOOT**. Final judgment will be entered in accordance with this order.

**So ORDERED and SIGNED this 30th day of August, 2018.**

  
ROBERT W. SCHROEDER III  
UNITED STATES DISTRICT JUDGE

**IN THE UNITED STATES DISTRICT COURT  
FOR THE EASTERN DISTRICT OF TEXAS  
TYLER DIVISION**

VIRNETX INC., LEIDOS, INC.,	§	
	§	
Plaintiffs,	§	CIVIL ACTION NO. 6:12-CV-00855-RWS
	§	
v.	§	LEAD CASE
	§	
APPLE INC.,	§	
	§	
Defendant.	§	

---

VIRNETX INC.,	§	
	§	
Plaintiff,	§	CIVIL ACTION NO. 6:11-CV-00563-RWS
	§	
v.	§	MEMBER CASE
	§	
APPLE INC.,	§	
	§	
Defendant.	§	

**FINAL JUDGMENT**

On this date, the Court entered its Memorandum Opinion and Order denying Defendant Apple Inc.'s Omnibus Motion for Judgment as a Matter of Law under Rule 50(b) and for a New Trial (Docket No. 775) and denying-in-part and granting-in-part VirnetX's Motion for Entry of Judgment and for Equitable and Statutory Relief (Docket No. 774)

A decision having been duly rendered as to all claims and consistent with the Court's Memorandum Opinion and Order, the Court hereby enters **FINAL JUDGMENT**.

The Clerk of the Court is directed to close both the lead and member case.

**So ORDERED and SIGNED this 30th day of August, 2018.**

  
ROBERT W. SCHROEDER III  
UNITED STATES DISTRICT JUDGE



US006502135B1

(12) **United States Patent**  
**Munger et al.**

(10) **Patent No.:** **US 6,502,135 B1**  
(45) **Date of Patent:** **Dec. 31, 2002**

(54) **AGILE NETWORK PROTOCOL FOR  
SECURE COMMUNICATIONS WITH  
ASSURED SYSTEM AVAILABILITY**

(75) Inventors: **Edmund Colby Munger**, Crownsville,  
MD (US); **Douglas Charles Schmidt**,  
Severna Park, MD (US); **Robert  
Dunham Short, III**, Leesburg, VA  
(US); **Victor Larson**, Fairfax, VA (US);  
**Michael Williamson**, South Riding, VA  
(US)

DE	199 24 575	12/1999
EP	2 317 792	4/1998
EP	0 858 189	8/1998
GB	0 814 589	12/1997
WO	WO 98/27783	6/1998
WO	WO 98 59470	12/1998
WO	WO 99 38081	7/1999
WO	WO 99 48303	9/1999
WO	WO 00/70458	11/2000
WO	WO 01 50688	7/2001

**OTHER PUBLICATIONS**

(73) Assignee: **Science Applications International  
Corporation**, San Diego, CA (US)

Fasbender, Kesdogan, and Kubitz: "Variable and Scalable  
Security: Protection of Location Information in Mobile IP",  
IEEE publication, 1996, pp. 963-967.

(\*) Notice: Subject to any disclaimer, the term of this  
patent is extended or adjusted under 35  
U.S.C. 154(b) by 0 days.

(List continued on next page.)

(21) Appl. No.: **09/504,783**

*Primary Examiner*—Krisna Lim

(22) Filed: **Feb. 15, 2000**

(74) *Attorney, Agent, or Firm*—Banner & Witcoff, Ltd.

(57) **ABSTRACT**

**Related U.S. Application Data**

(63) Continuation-in-part of application No. 09/429,643, filed on  
Oct. 29, 1999

(60) Provisional application No. 60/106,261, filed on Oct. 30,  
1998, and provisional application No. 60/137,704, filed on  
Jun. 7, 1999.

(51) **Int. Cl.<sup>7</sup>** ..... **G06F 15/173**

(52) **U.S. Cl.** ..... **709/225; 709/229; 709/245**

(58) **Field of Search** ..... **709/249, 223,  
709/225, 229, 245; 713/201**

(56) **References Cited**

**U.S. PATENT DOCUMENTS**

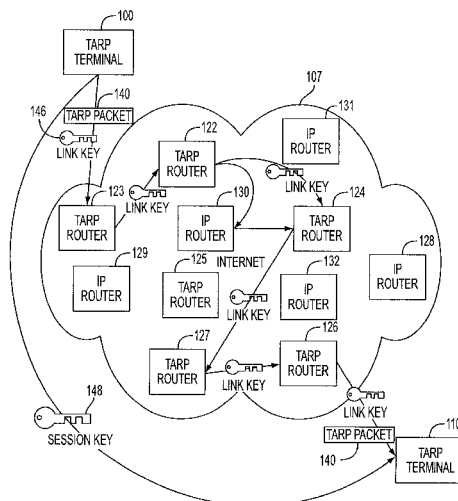
4,933,846 A 6/1990 Humphrey et al.

(List continued on next page.)

**FOREIGN PATENT DOCUMENTS**

DE 0 838 930 12/1999

**17 Claims, 35 Drawing Sheets**



US 6,502,135 B1

Page 2

U.S. PATENT DOCUMENTS

5,588,060	A	12/1996	Aziz	
5,689,566	A	11/1997	Nguyen	
5,796,942	A	8/1998	Esbensen	
5,805,801	A	9/1998	Holloway et al.	
5,842,040	A	11/1998	Hughes et al.	
5,878,231	A *	3/1999	Baehr et al.	709/243
5,892,903	A	4/1999	Klaus	
5,898,830	A *	4/1999	Wesinger et al.	709/225
5,905,859	A	5/1999	Holloway et al.	
6,006,259	A	12/1999	Adelman et al.	
6,016,318	A *	1/2000	Tomoike	370/338
6,052,788	A	4/2000	Wesinger, Jr. et al.	
6,079,020	A *	6/2000	Liu	713/201
6,119,171	A	9/2000	Alkhatib	
6,178,505	B1 *	1/2001	Schneider et al.	713/168
6,226,751	B1 *	5/2001	Arrow et al.	370/351
6,243,749	B1	6/2001	Sitaraman et al.	
6,286,047	B1 *	9/2001	Ramanathan et al.	345/733
6,330,562	B1 *	12/2001	Boden et al.	707/10
6,332,158	B1 *	12/2001	Risley et al.	709/219
6,353,614	B1 *	3/2002	Borella et al.	370/389

OTHER PUBLICATIONS

Linux FreeS/WAN Index File, printed from [http://liberty.freeswan.org/freeswan\\_trees/freeswan-1.3/doc/](http://liberty.freeswan.org/freeswan_trees/freeswan-1.3/doc/) on Feb. 21, 2002, 3 pages.

J. Gilmore, “Swan: Securing the Internet against Wiretapping”, printed from [http://liberty.freeswan.org/freeswan\\_trees/freeswan-1.3/doc/rationale.html](http://liberty.freeswan.org/freeswan_trees/freeswan-1.3/doc/rationale.html) on Feb. 21, 2002, 4 pages.

Glossary for the Linux FreeS/WAN project, printed from [http://liberty.freeswan.org/freeswan\\_trees/freeswan-1.3/doc/glossary.html](http://liberty.freeswan.org/freeswan_trees/freeswan-1.3/doc/glossary.html) on Feb. 21, 2002, 25 pages.

Alan O. Frier et al., “The SSL Protocol Version 3.0”, Nov. 18, 1996, printed from <http://www.netscape.com/eng/ss13/draft302.txt> on Feb. 4, 2002, 56 pages.

Reiter, Michael K. and Rubin, Aviel D. (AT&T Labs—Research), “Crowds: Anonymity for Web Transactions”, pp. 1–23.

Dolev, Shlomi and Ostrovsky, Rafail, “Efficient Anonymous Multicast and Reception” (Extended Abstract), 16 pages.

Rubin, Aviel D., Geer, Daniel, and Ranum, Marcus J. (Wiley Computer Publishing), “Web Security Sourcebook”, pp. 82–94.

Shree Murthy et al., “Congestion–Oriented Shortest Multipath Routing”, Proceedings of IEEE INFOCOM, 1996, pp. 1028–1036.

Jim Jones et al., “Distributed Denial of Service Attacks: Defenses”, Global Integrity Corporation, 2000, pp. 1–14.

Search Report (dated Jun. 18, 2002), International Application No. PCT/US01/13260.

Search Report (dated Jun. 28, 2002), International Application No. PCT/US01/13261.

Donald E. Eastlake, “Domain Name System Security Extensions”, DNS Security Working Group, Apr. 1998, 51 pages.

D. B. Chapman et al., “Building Internet Firewalls”, Nov. 1995, pp. 278–297 and pp. 351–375.

P. Srisuresh et al., “DNS extensions to Network Address Translators”, Jul. 1998, 27 pages.

Laurie Wells, “Security Icon”, Oct. 19, 1998, 1 page.

W. Stallings, “Cryptography And Network Security”, 2<sup>nd</sup> Edition, Chapter 13, IP Security, Jun. 8, 1998, pp. 399–400.

W. Stallings, “New Cryptography and Network Security Book”, Jun. 8, 1998, 3 pages.

\* cited by examiner

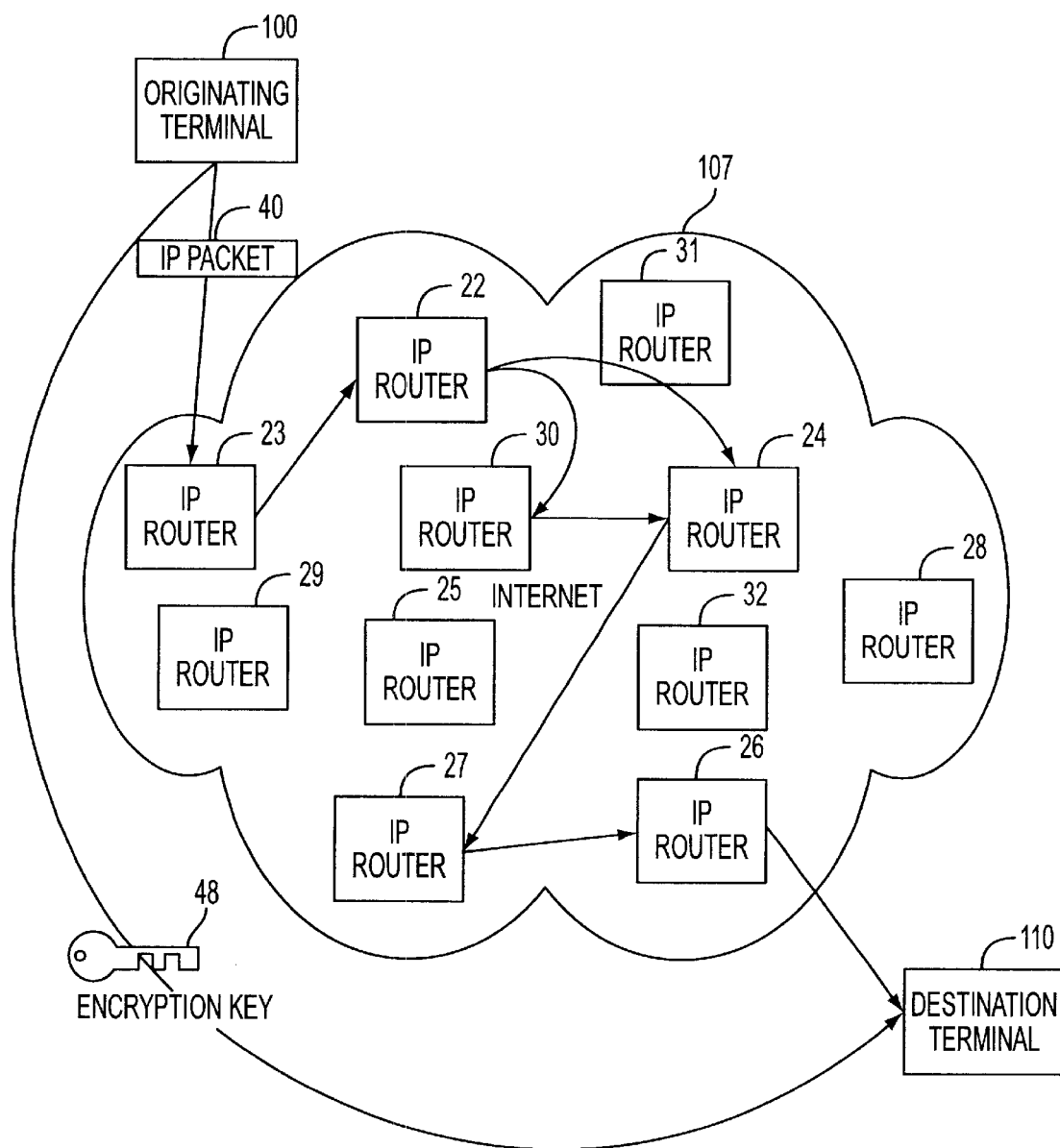


FIG. 1

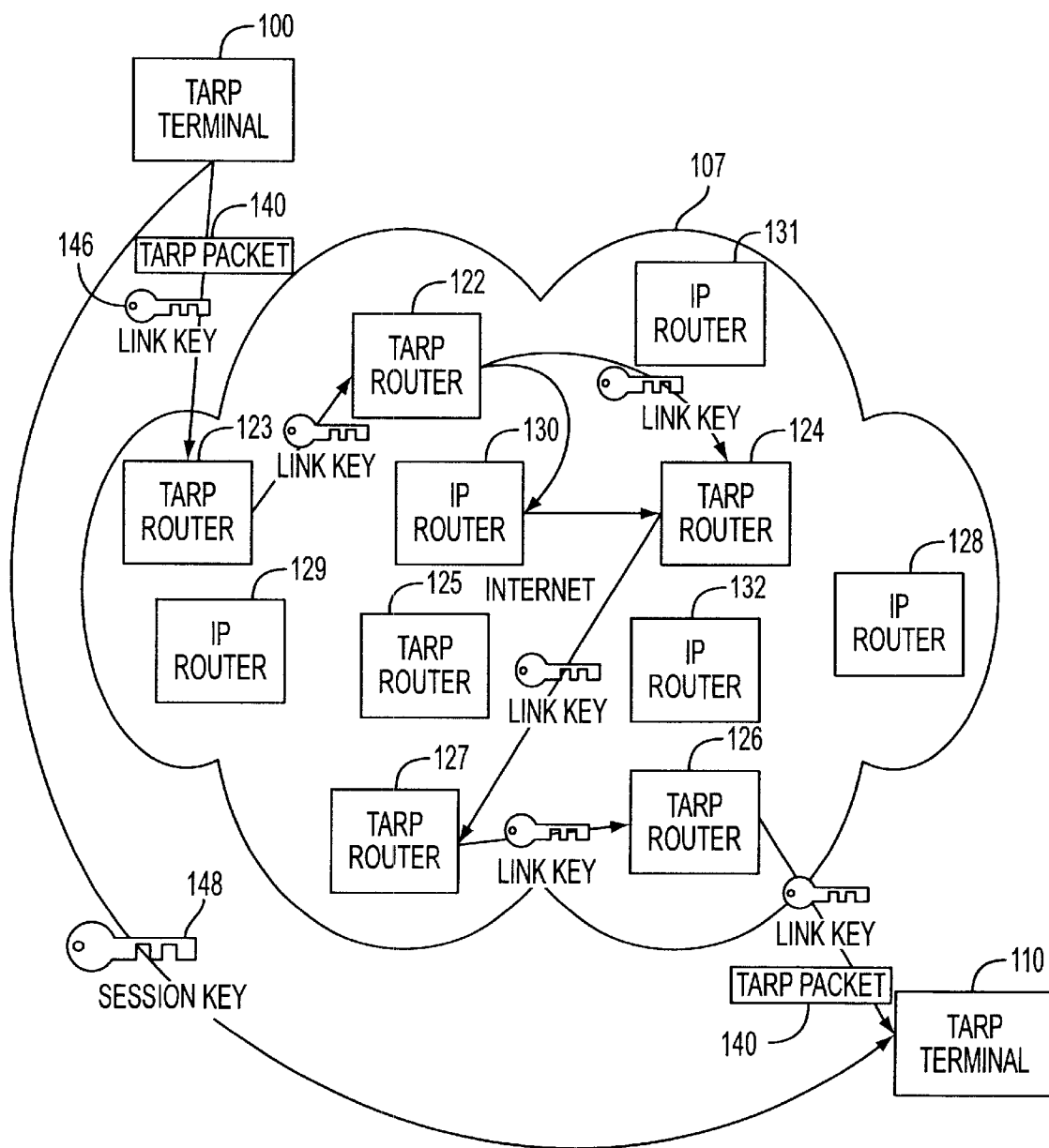


FIG. 2

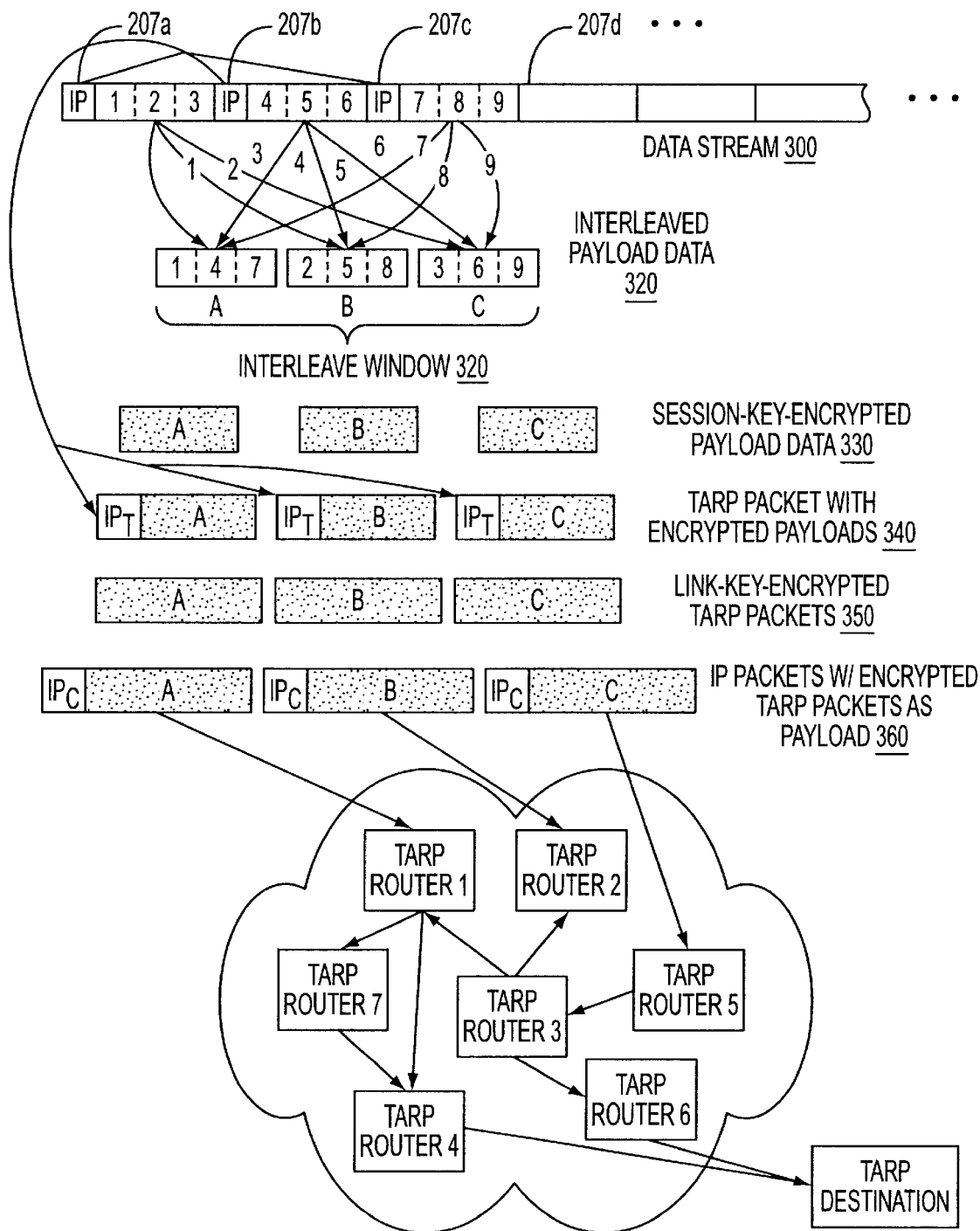


FIG. 3A

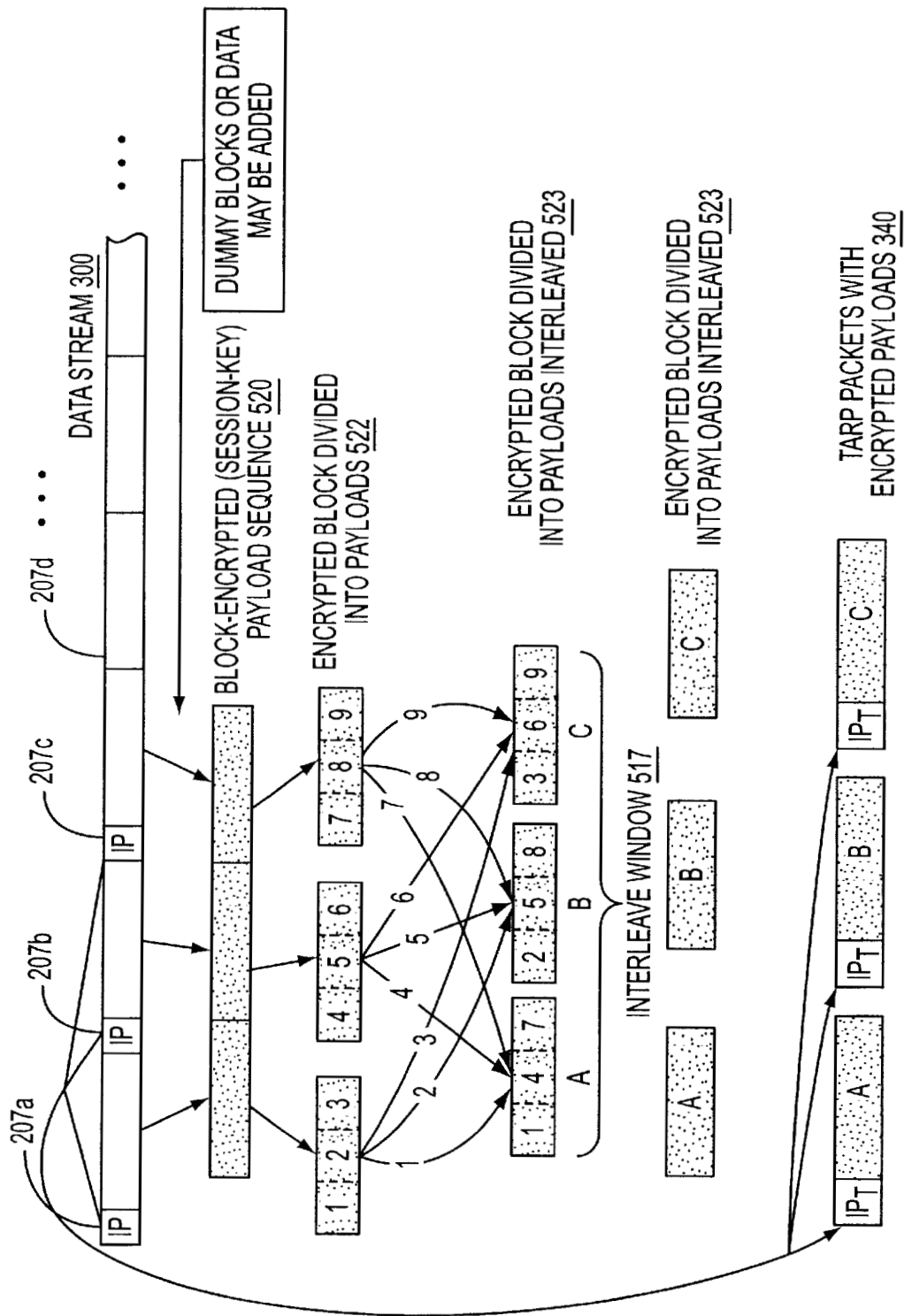


FIG. 3B

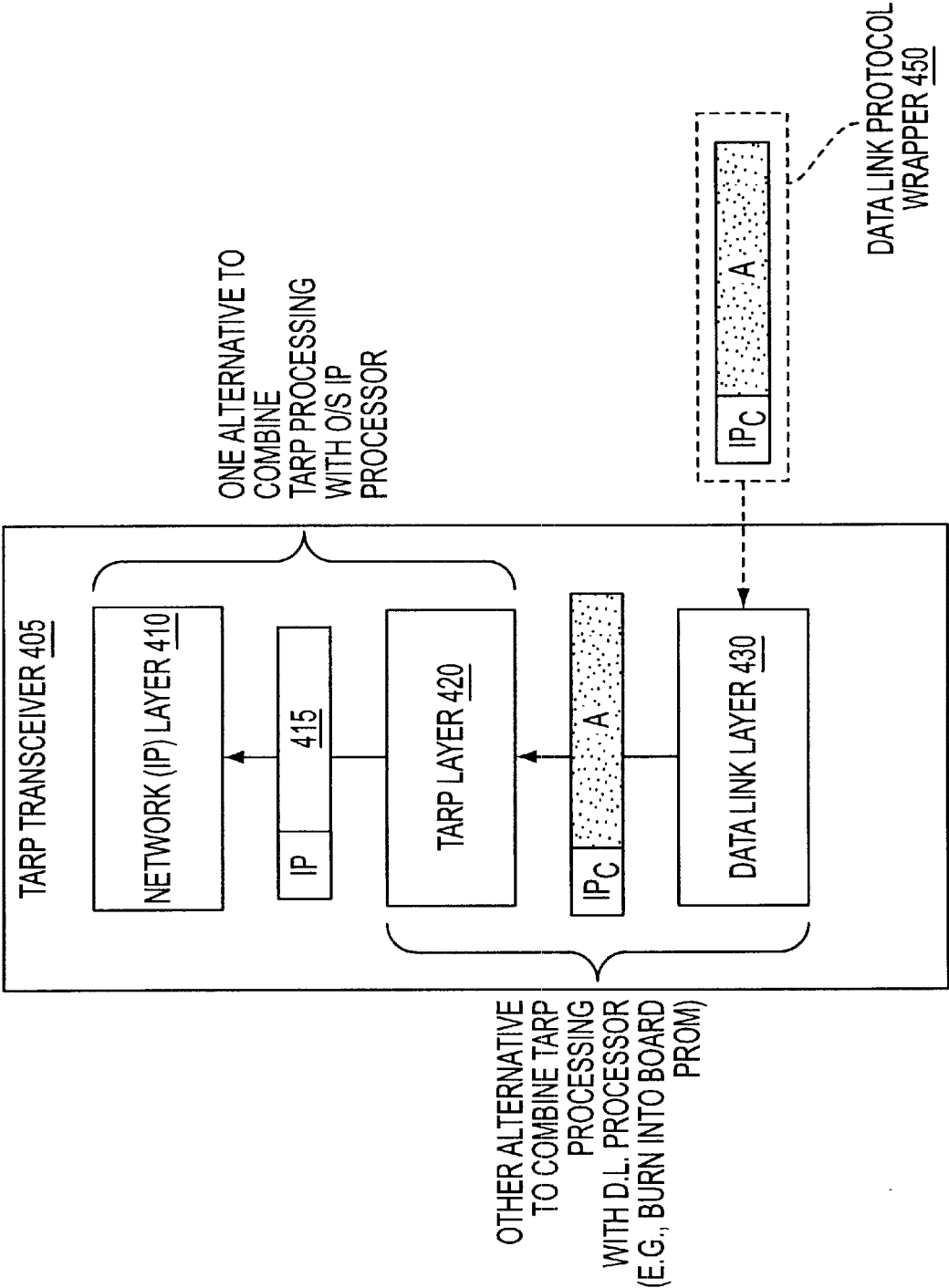


FIG. 4

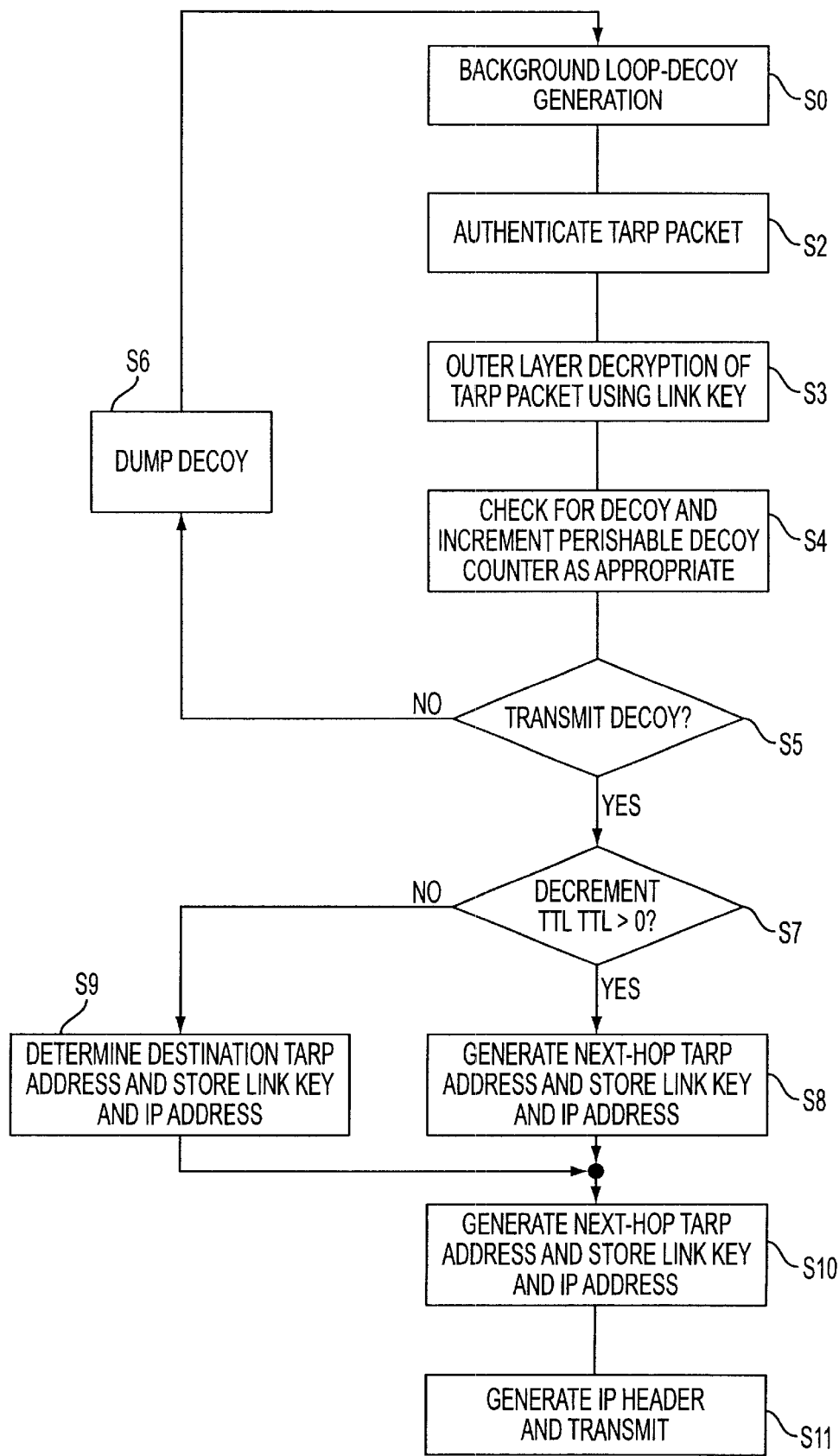


FIG. 5



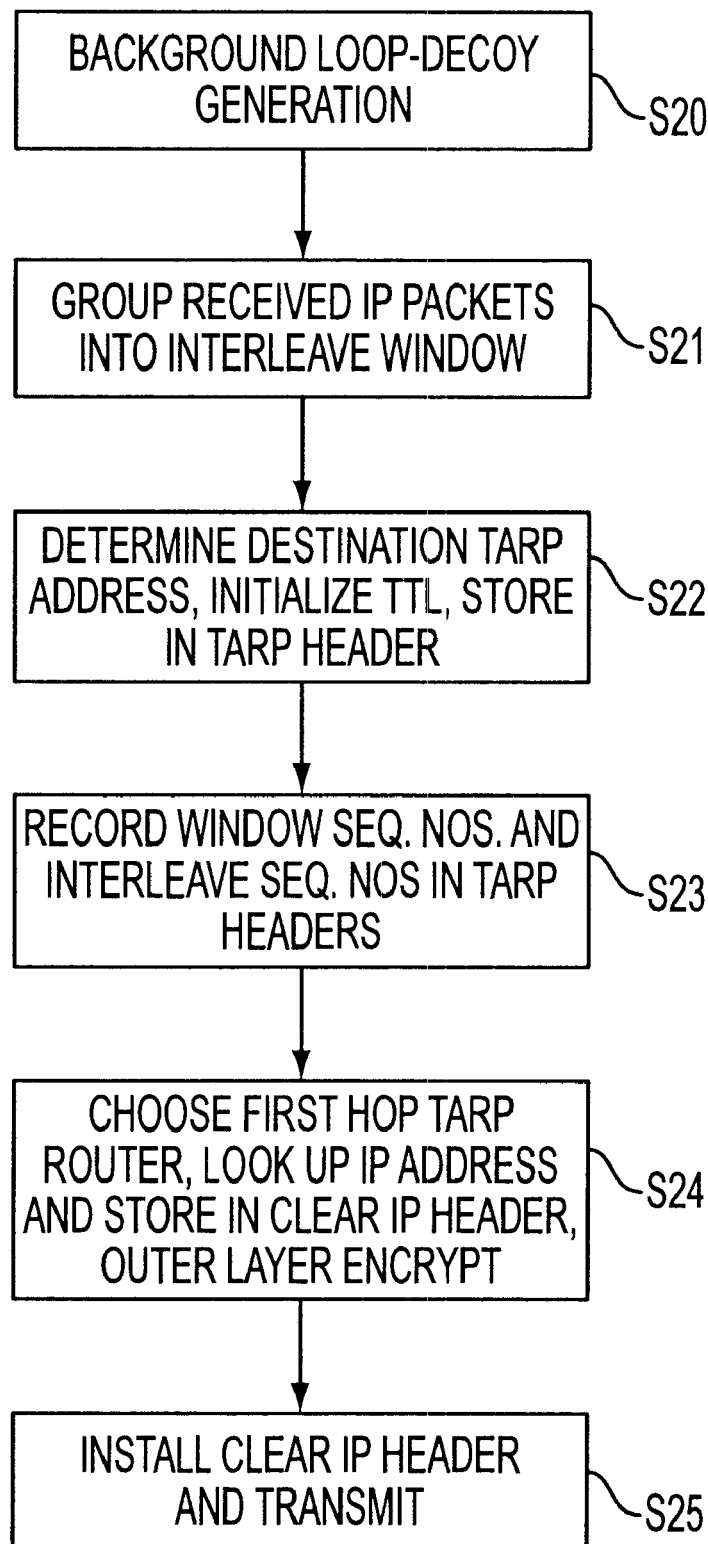


FIG. 6

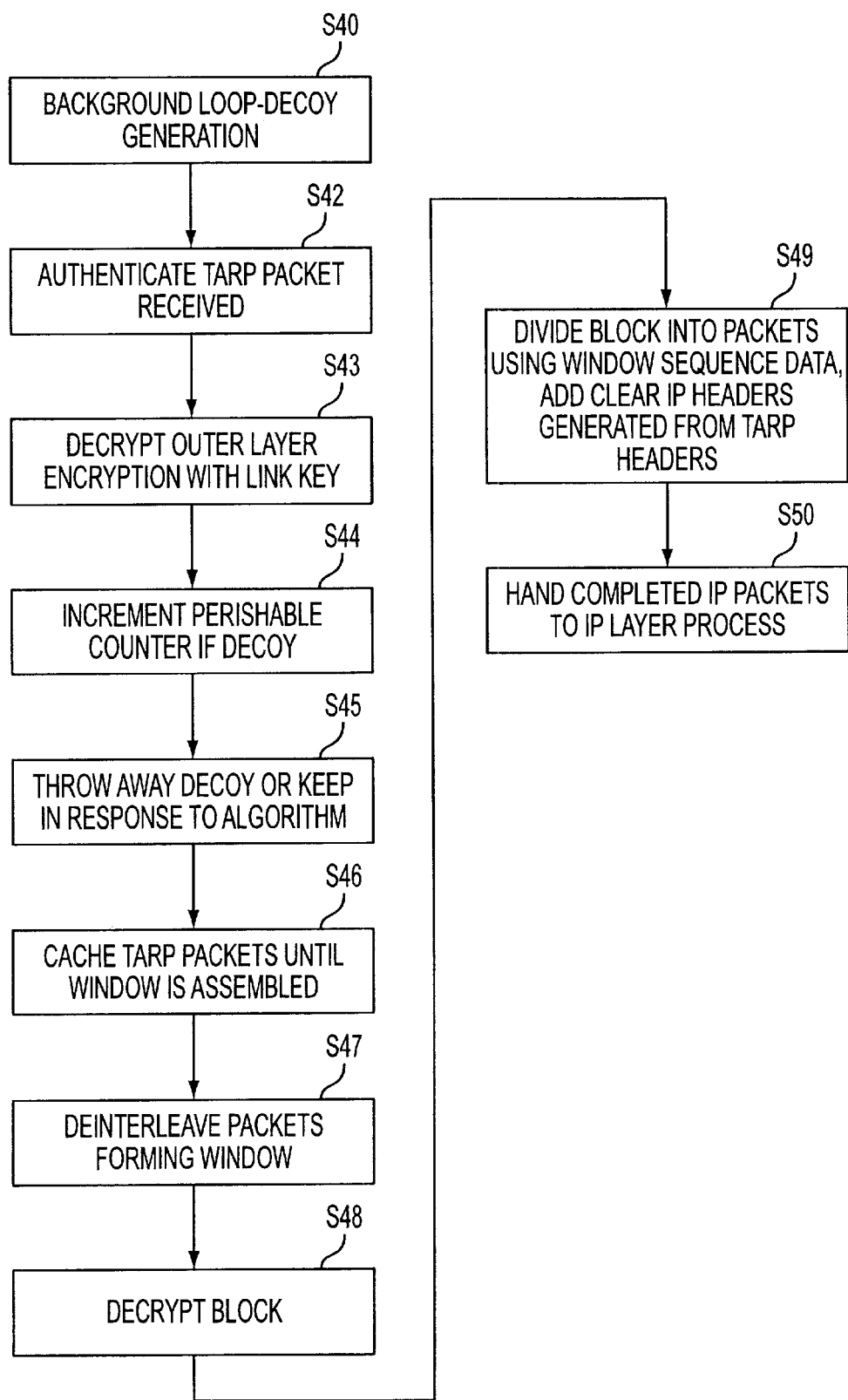


FIG. 7

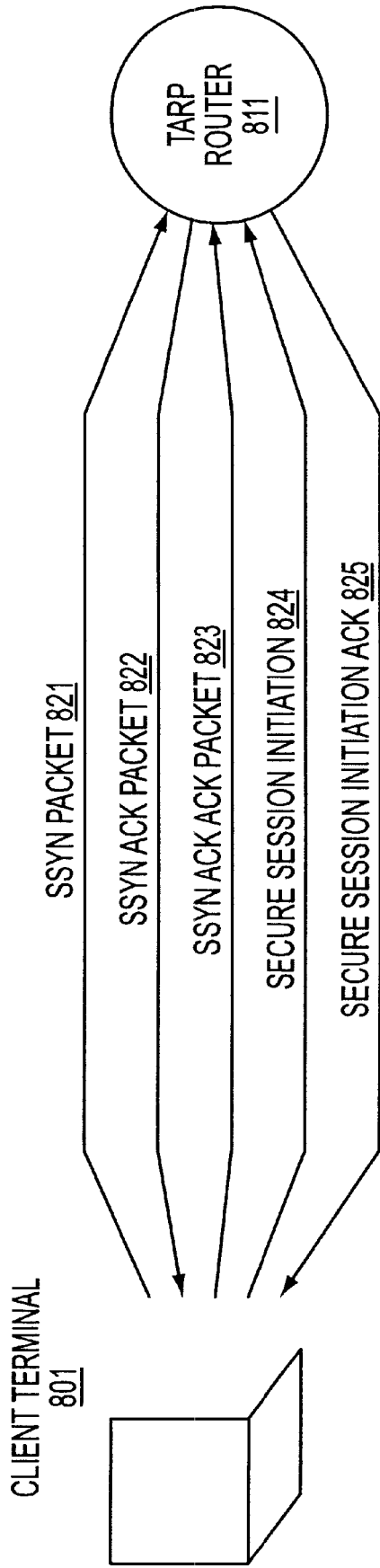


FIG. 8

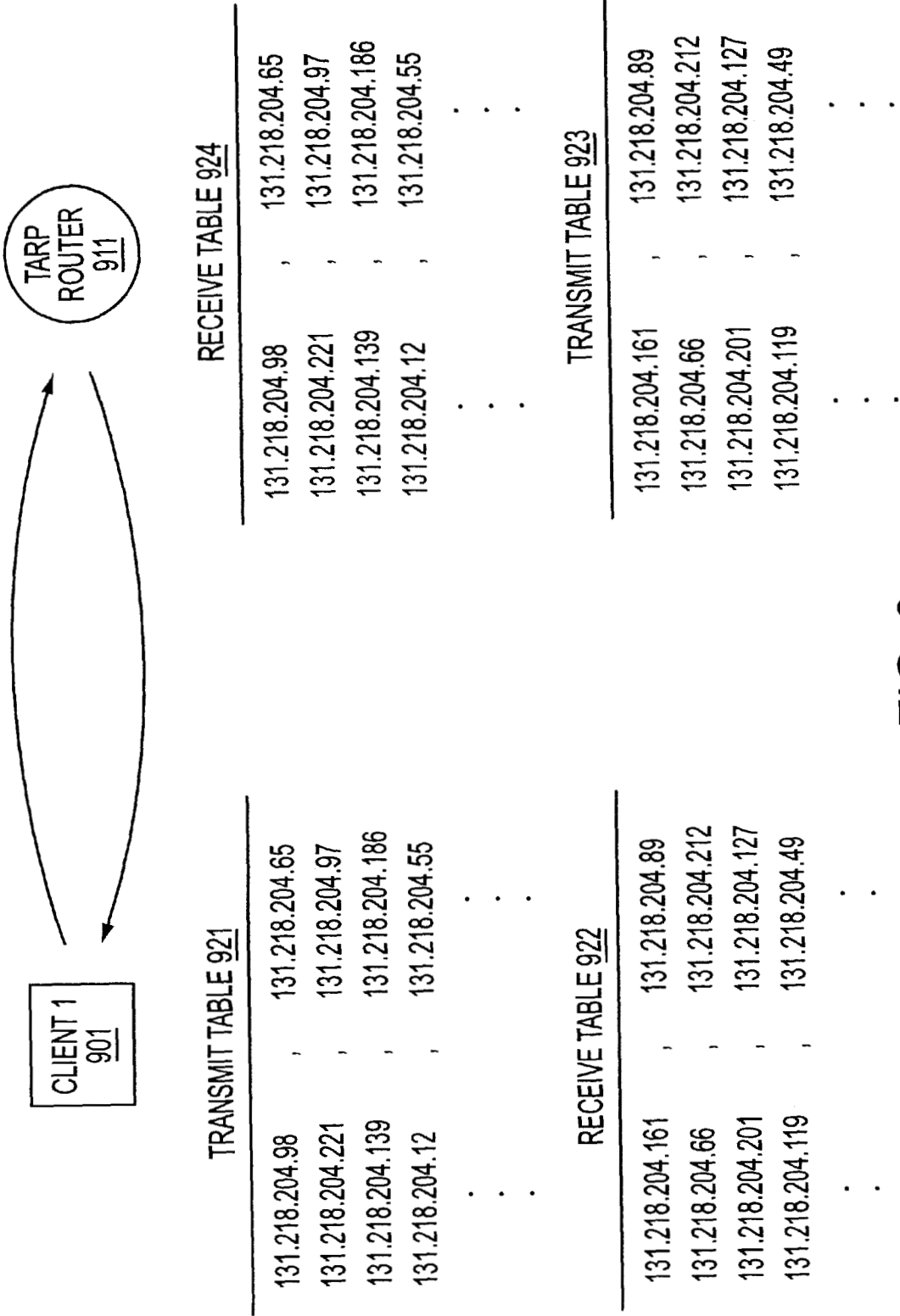


FIG. 9

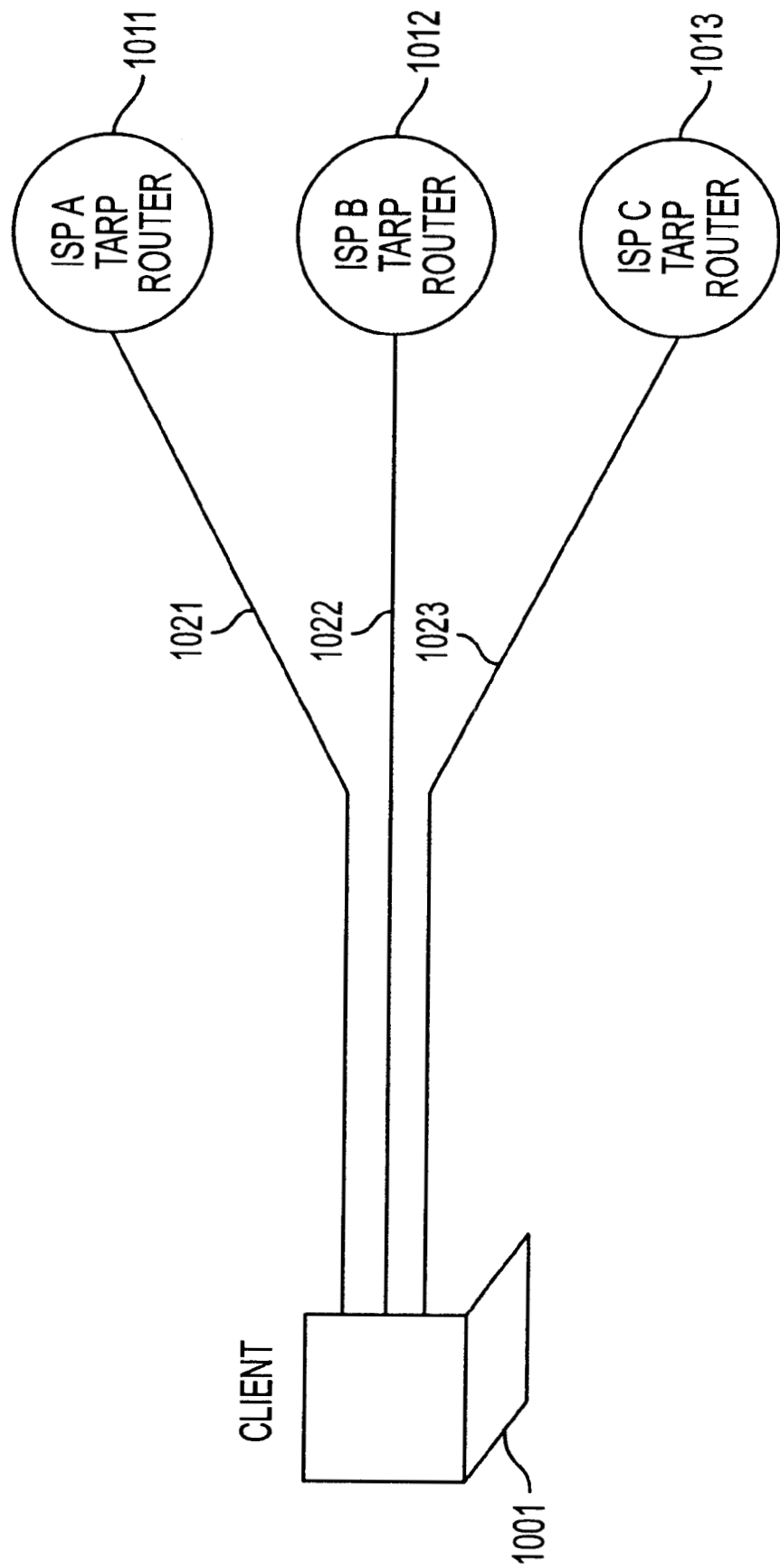


FIG. 10

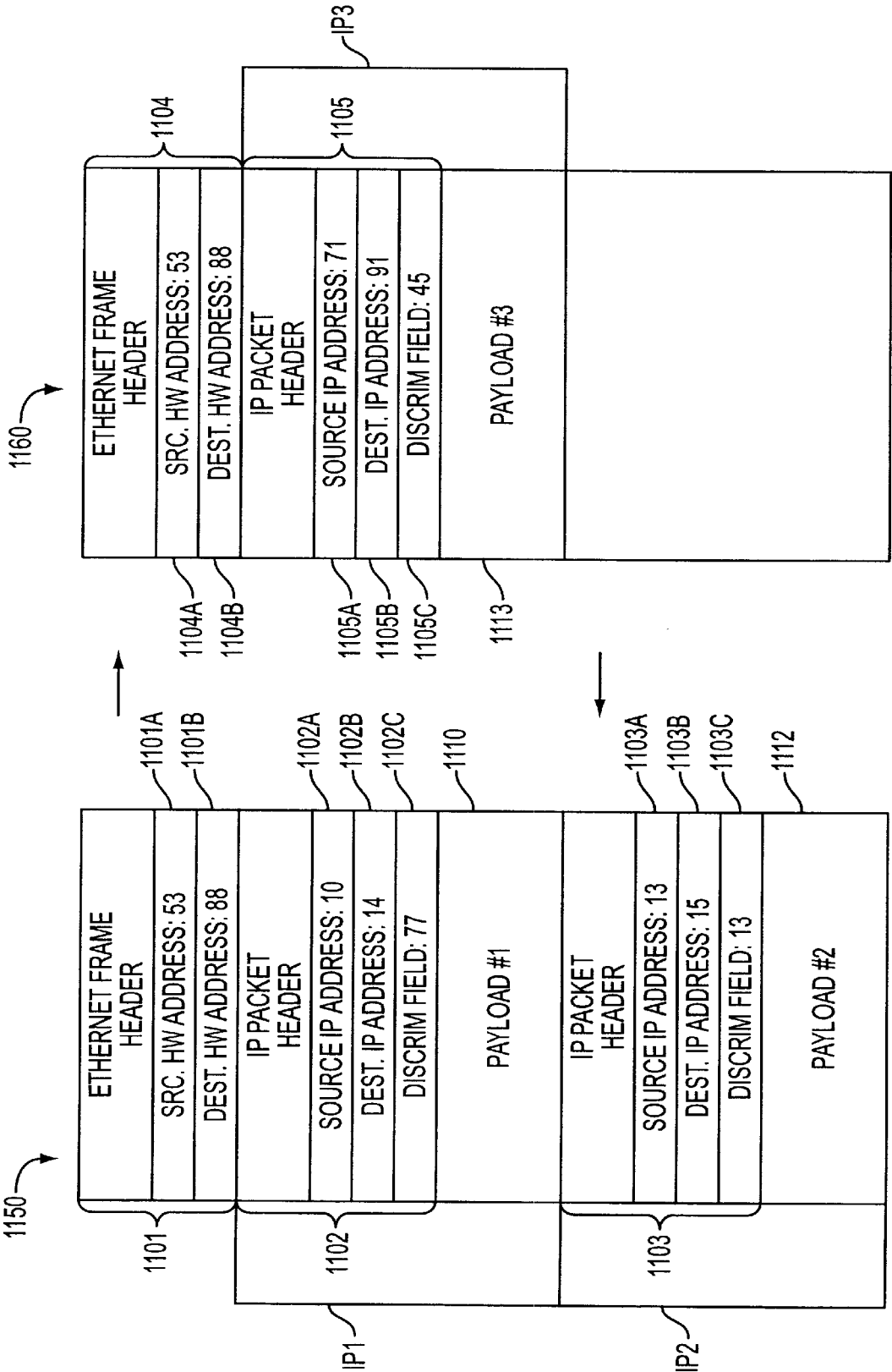


FIG. 11

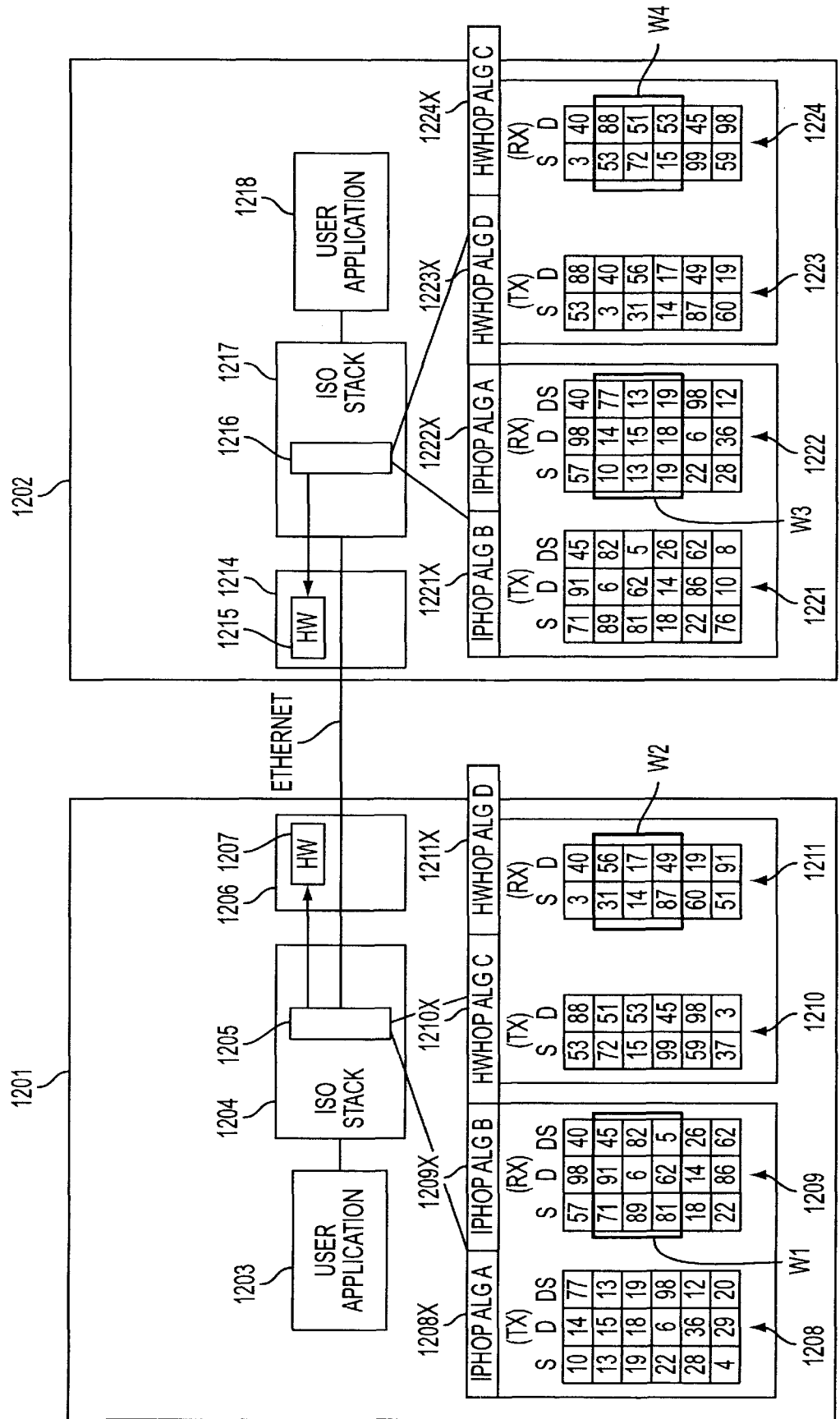


FIG. 12A

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

FIG. 12B



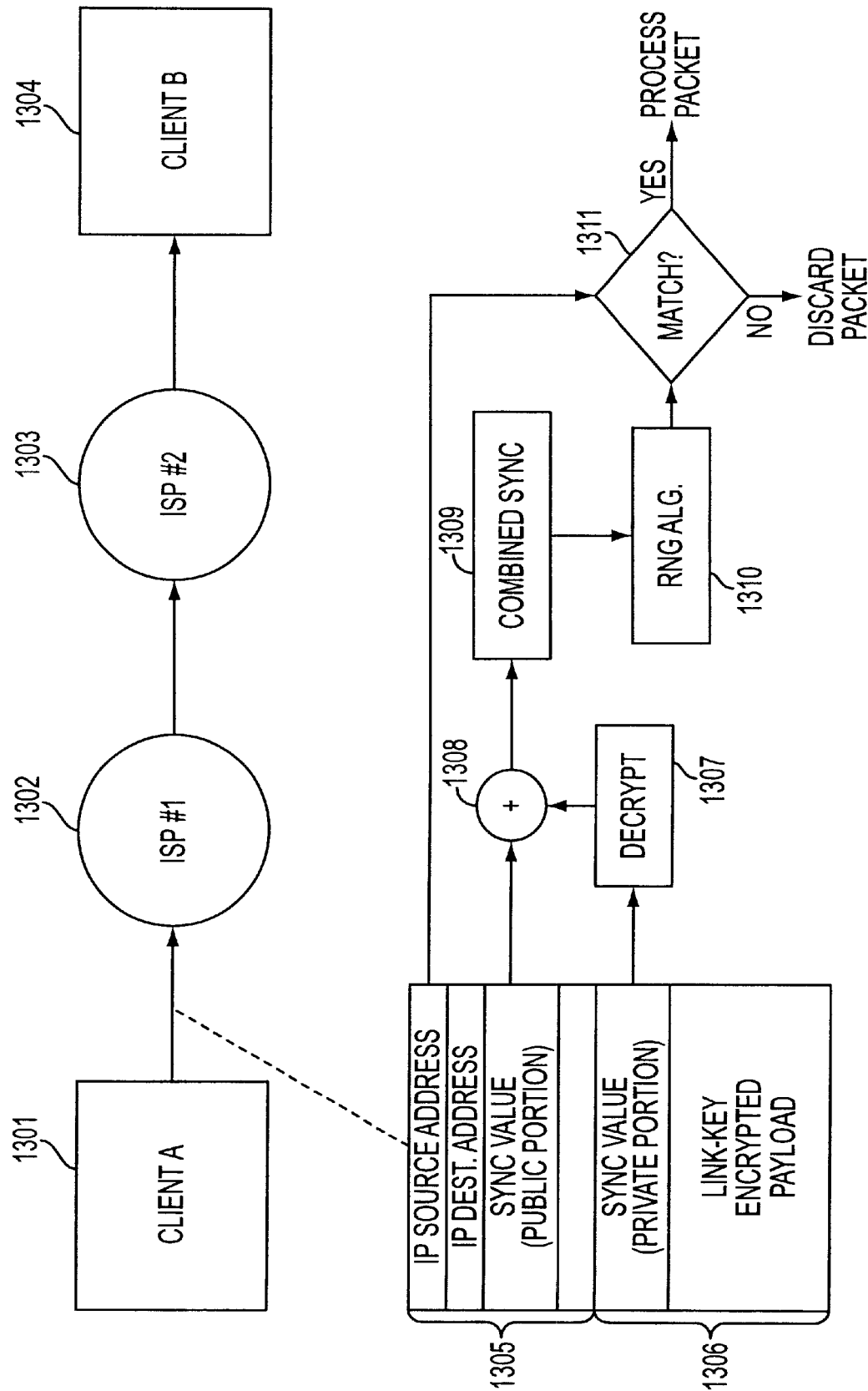


FIG. 13

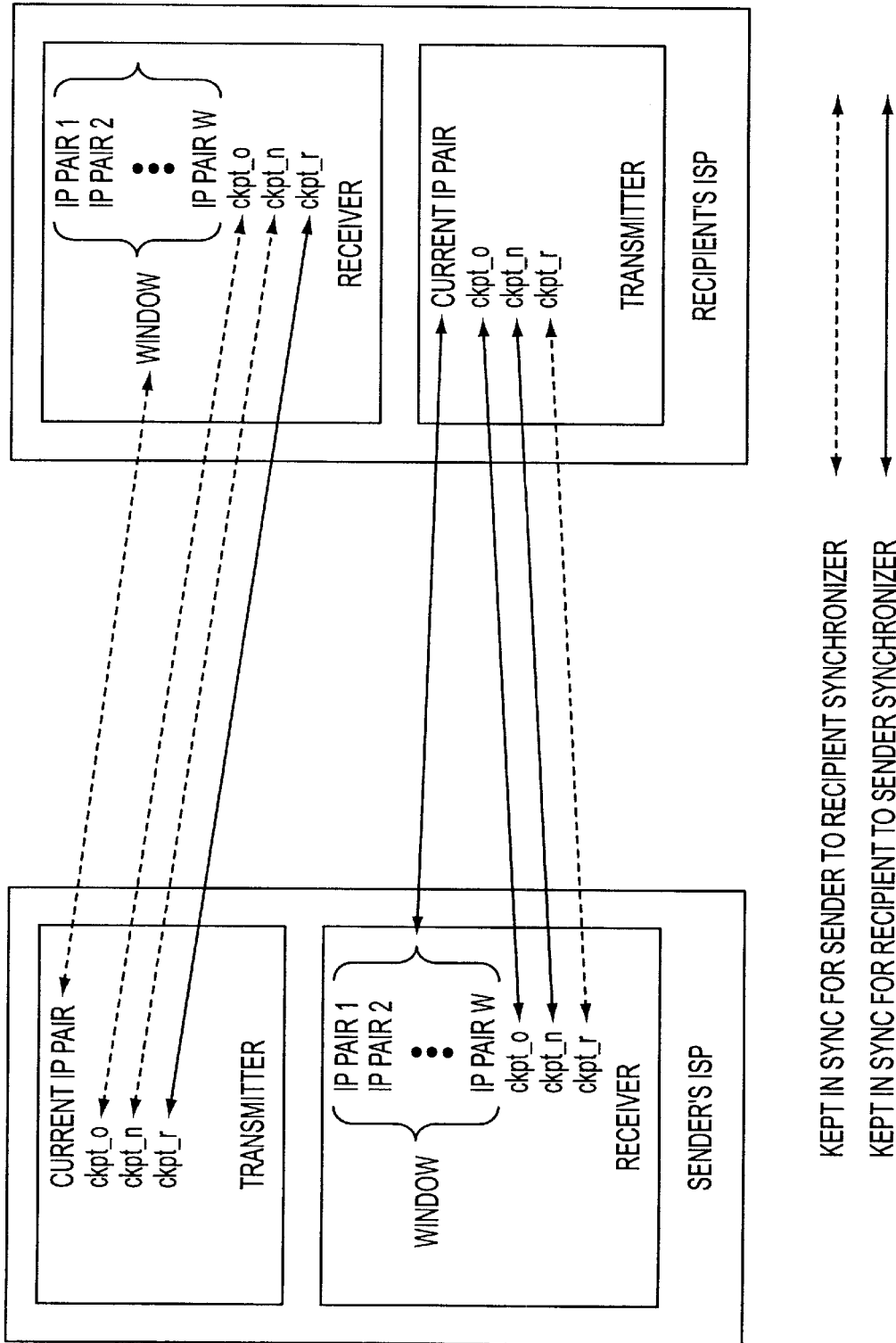


FIG. 14

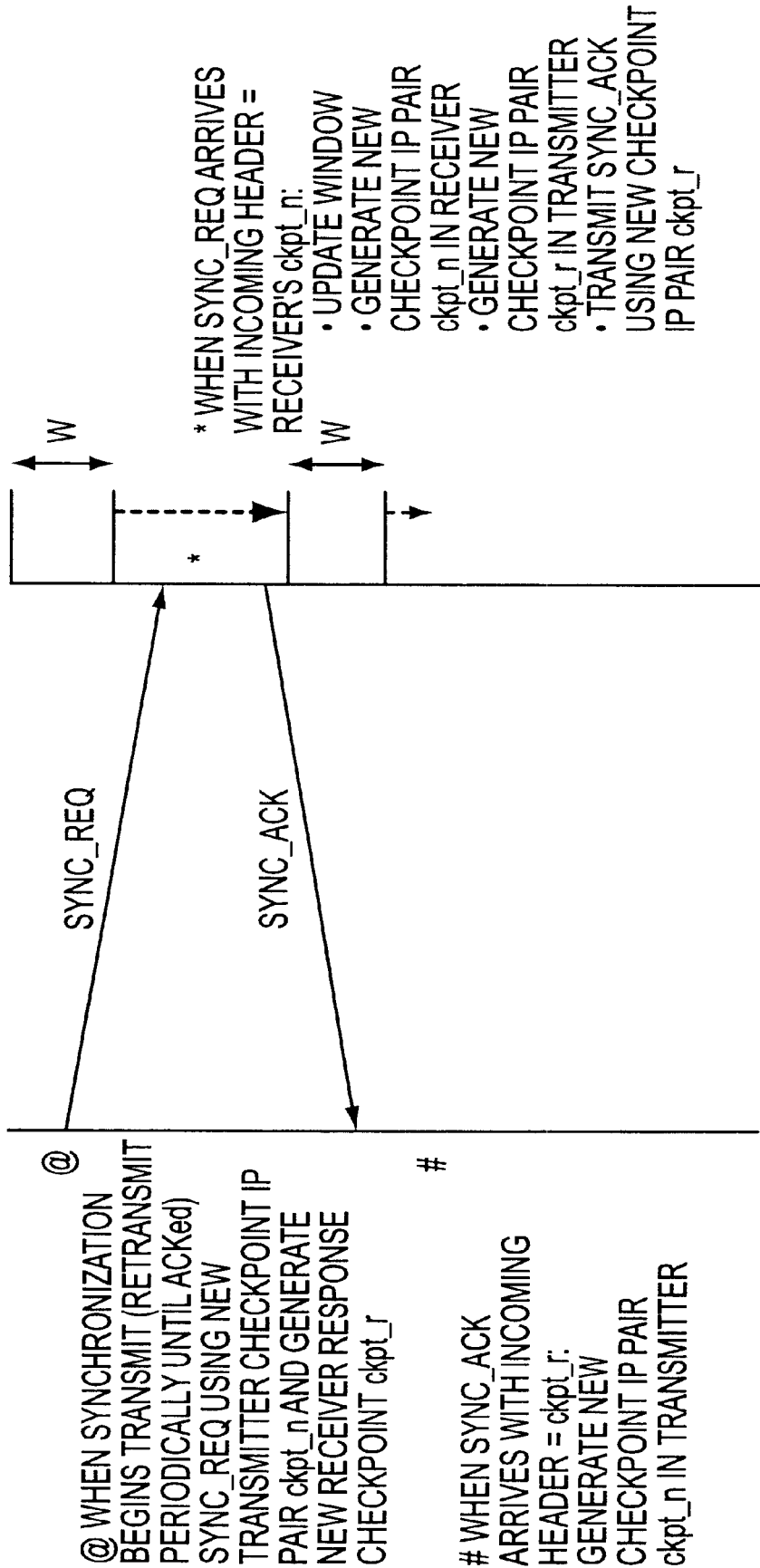


FIG. 15

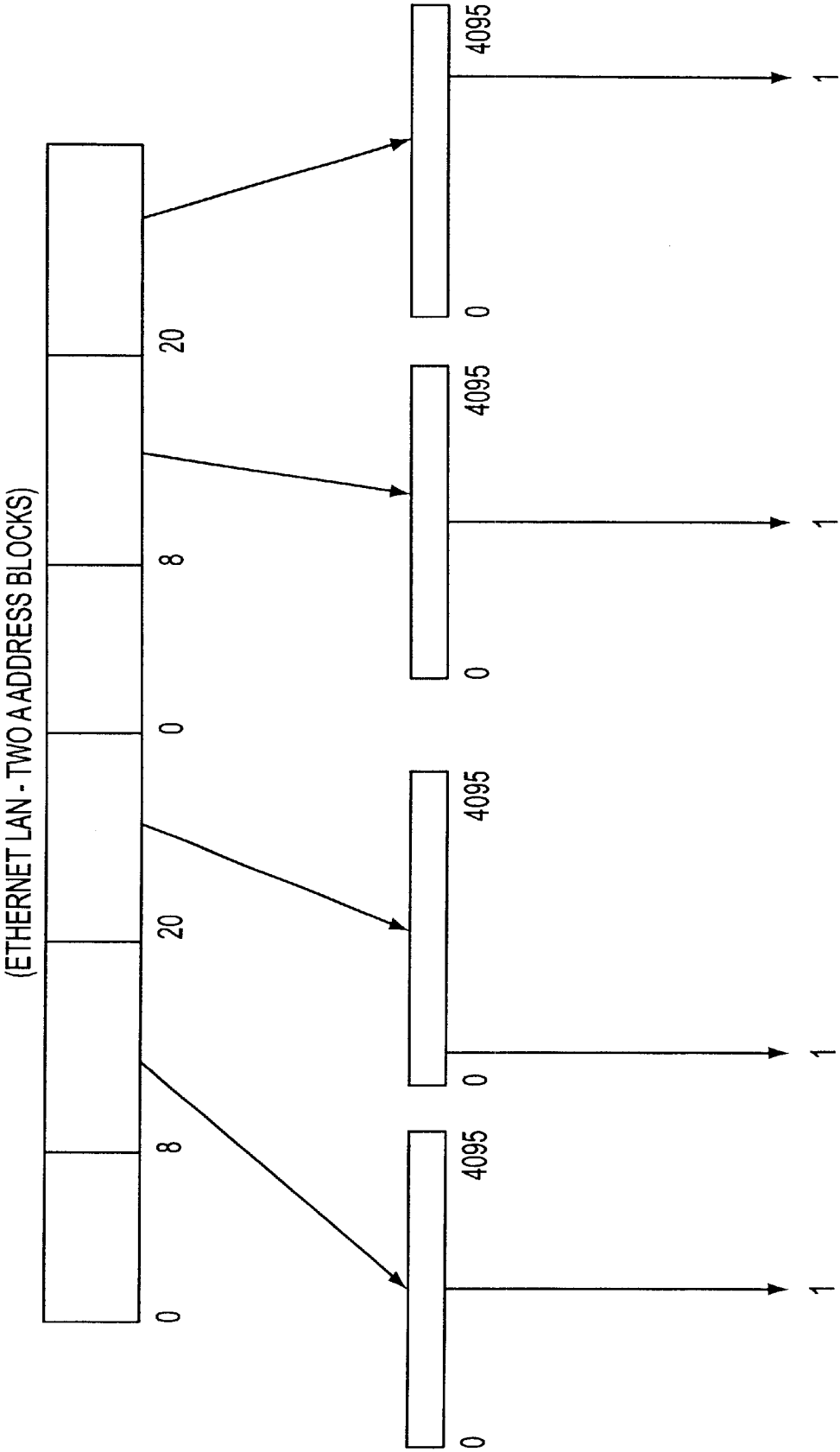


FIG. 16

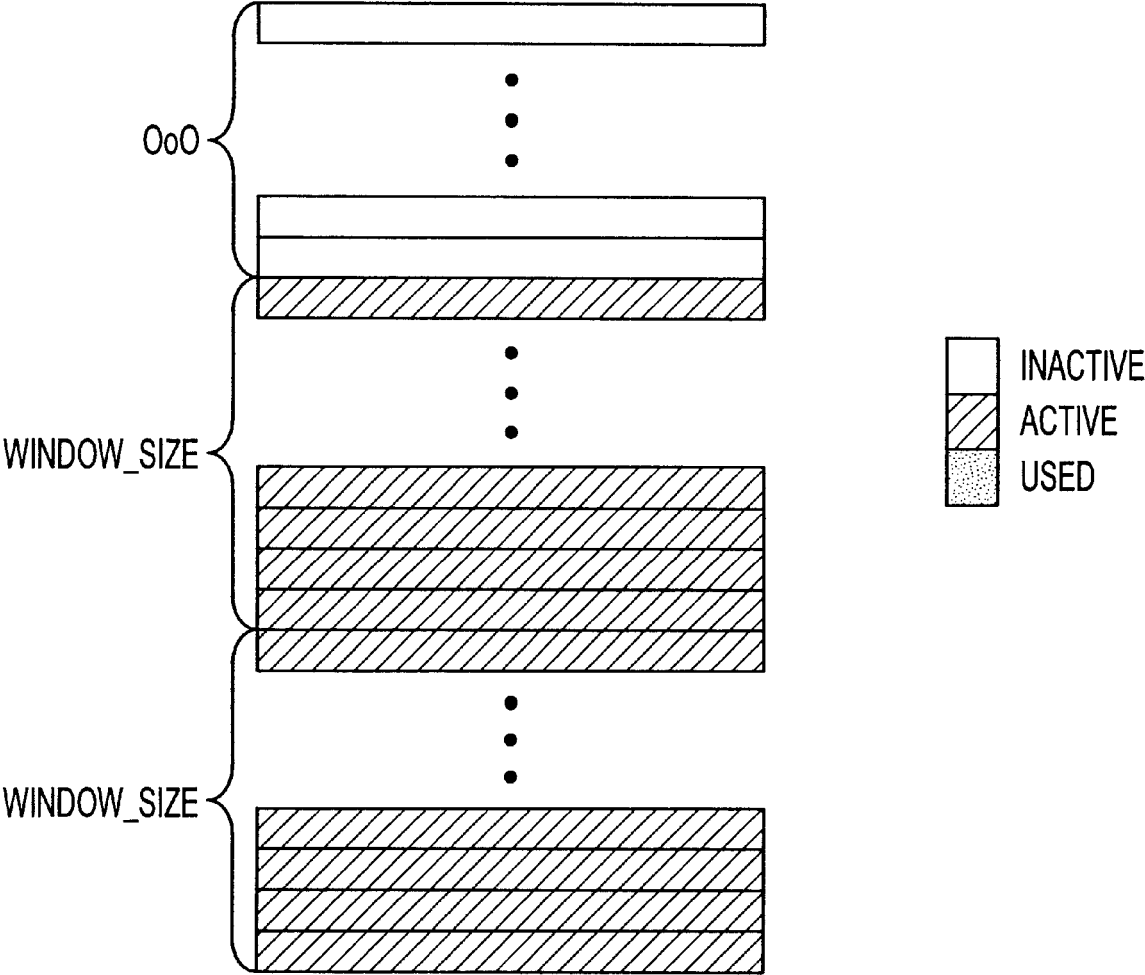


FIG. 17

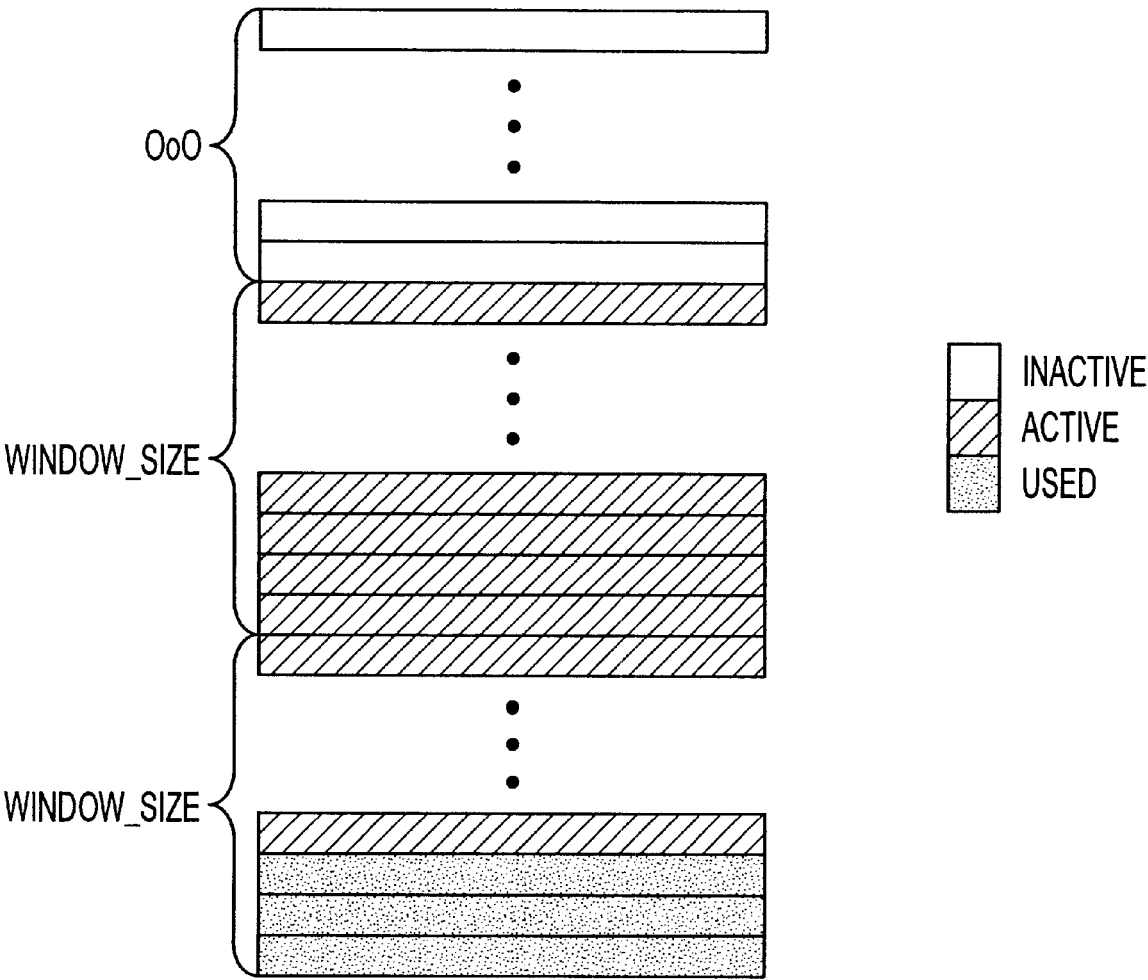


FIG. 18

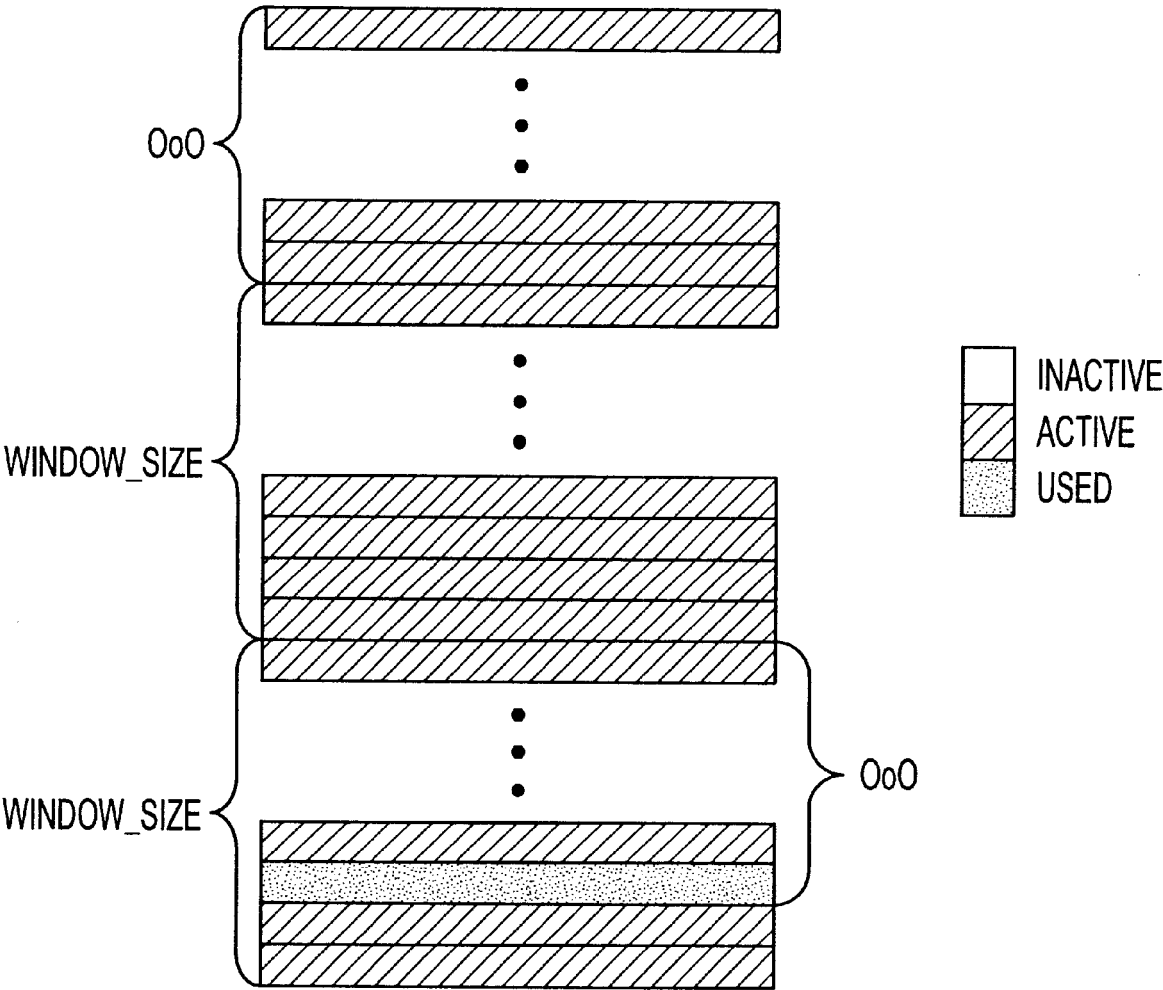


FIG. 19

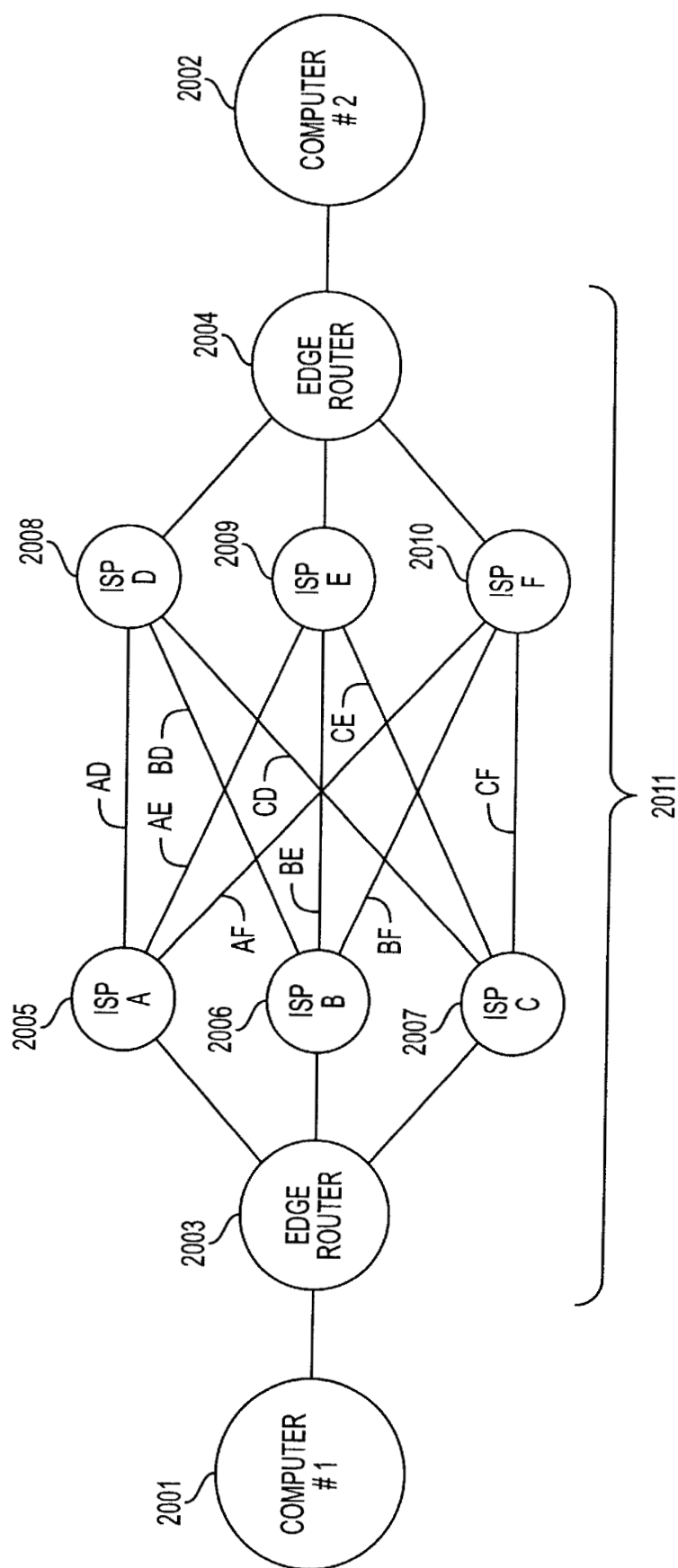


FIG. 20



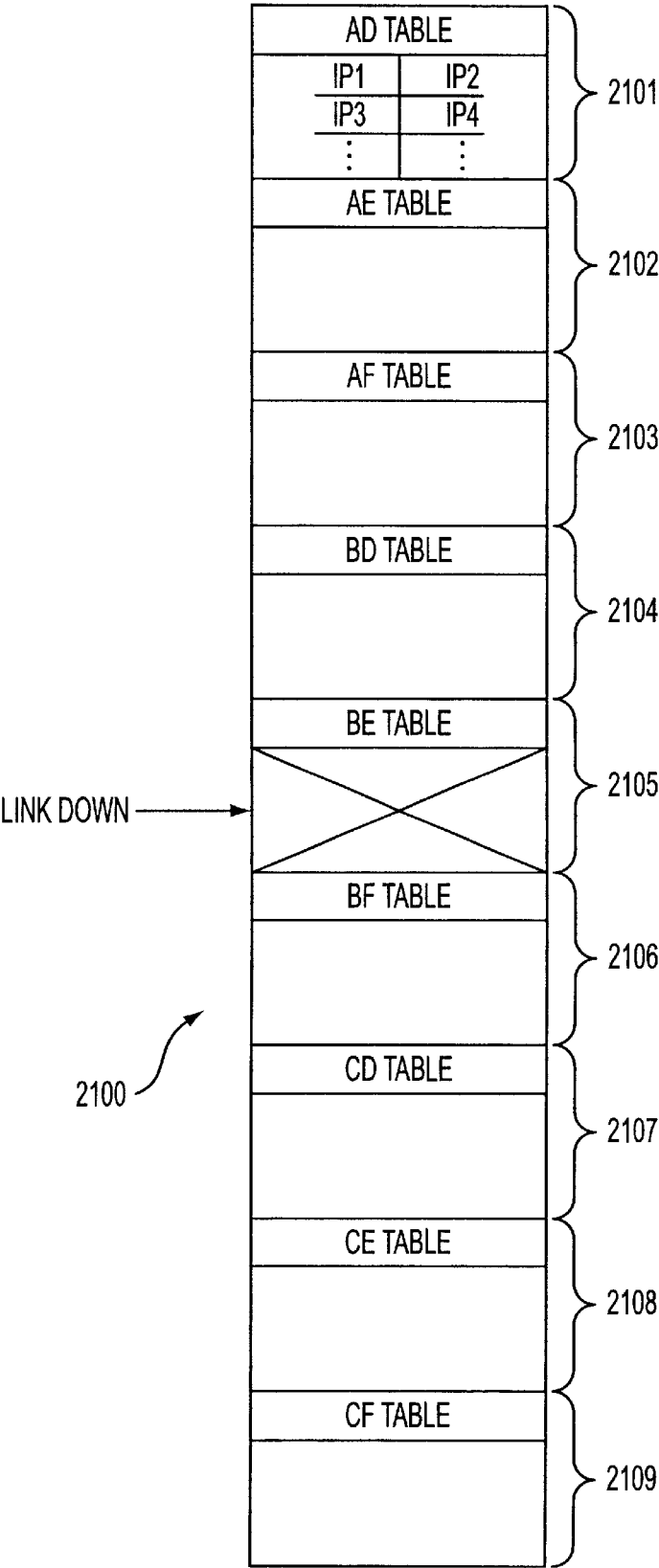


FIG. 21

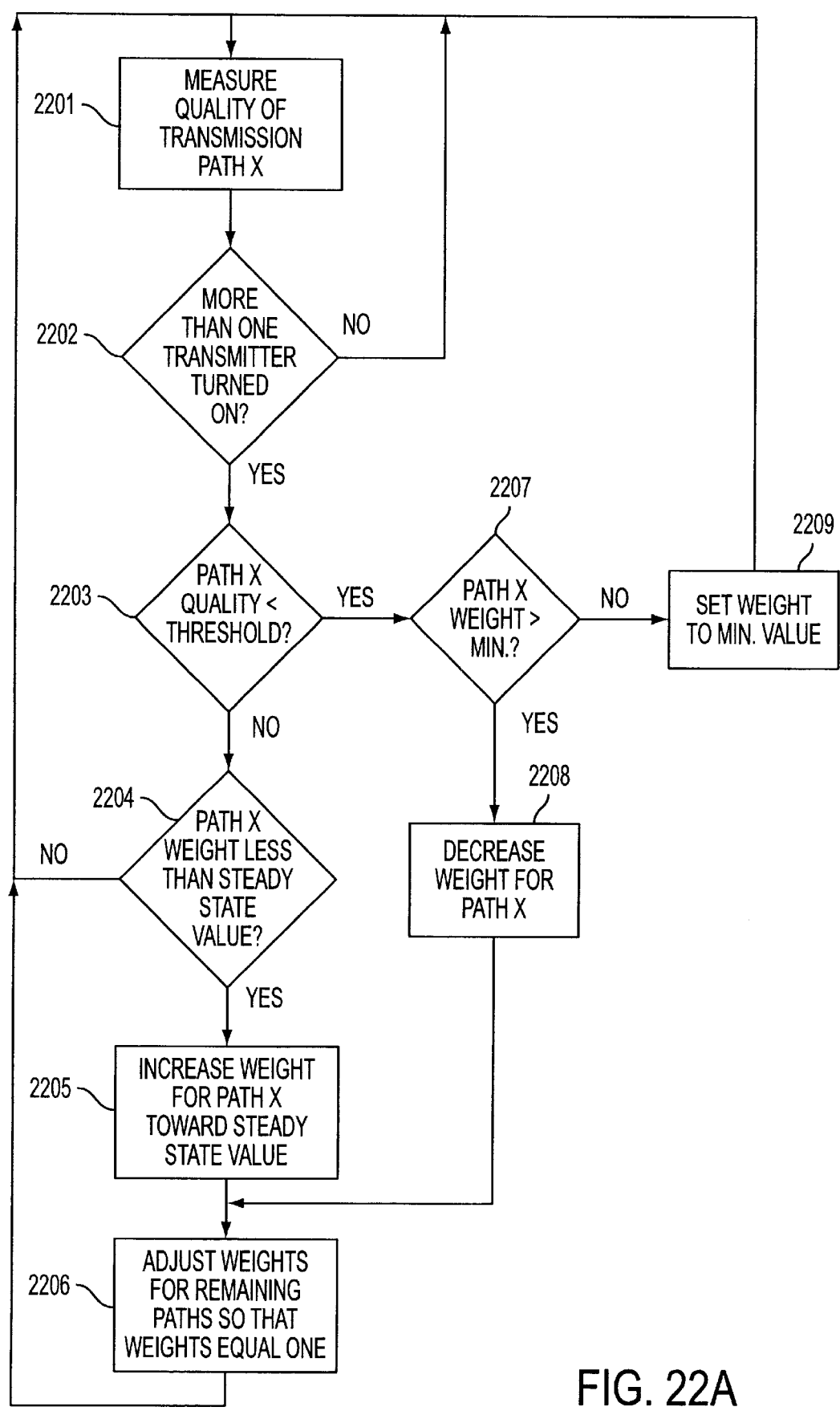


FIG. 22A

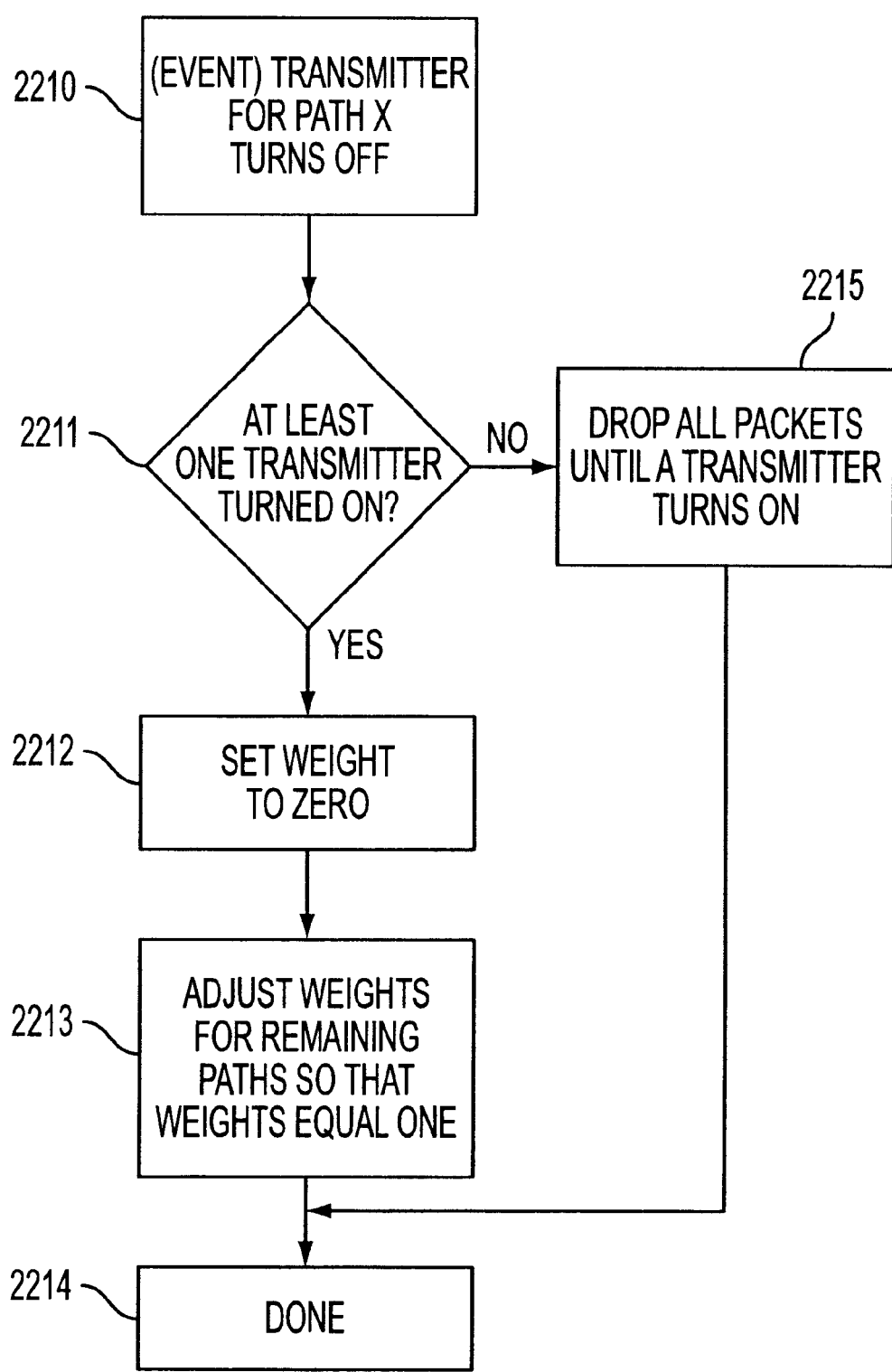


FIG. 22B

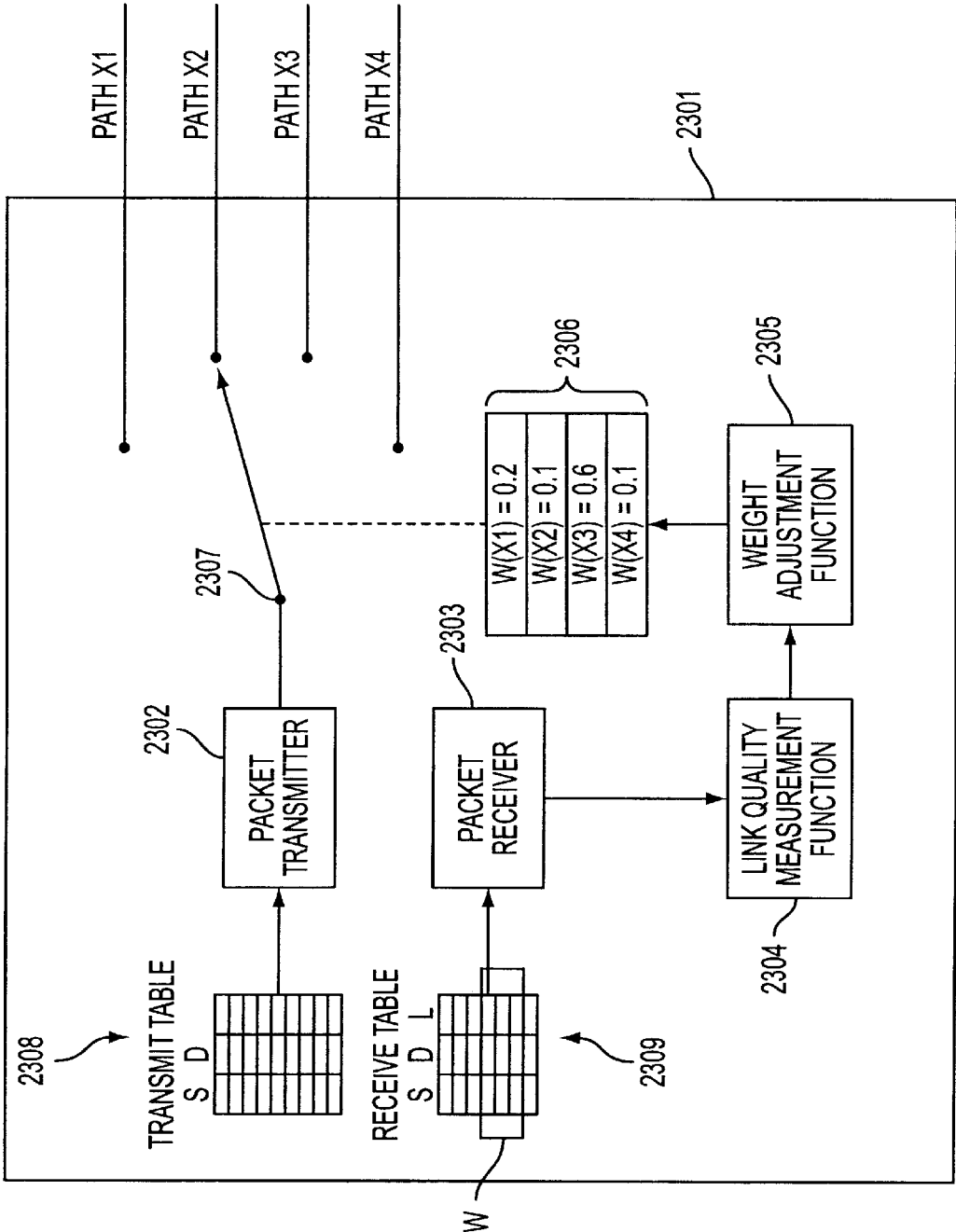


FIG. 23

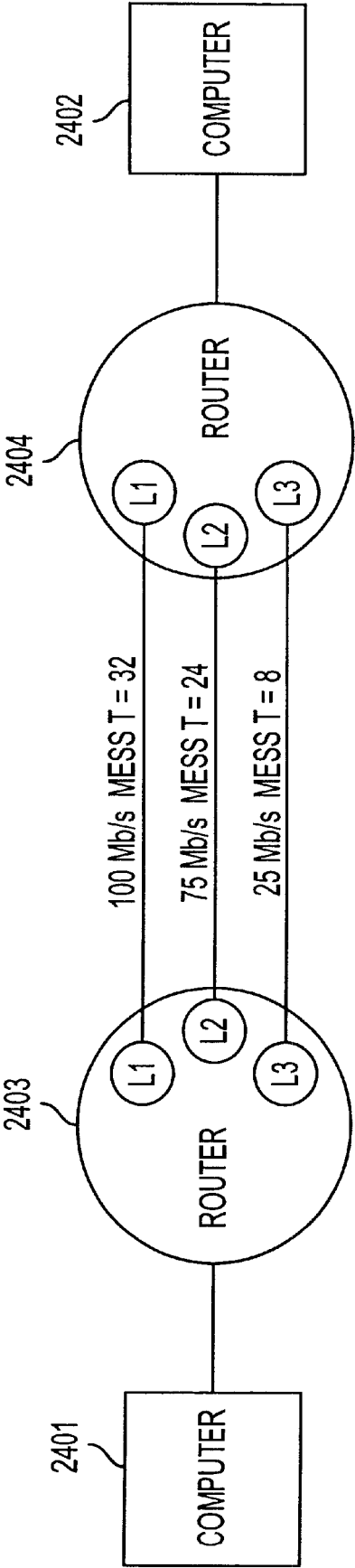


FIG. 24

Case 6:12-cv-00855-RWS Document 1-1 Filed 11/06/12 Page 31 of 73 PageID #: 46

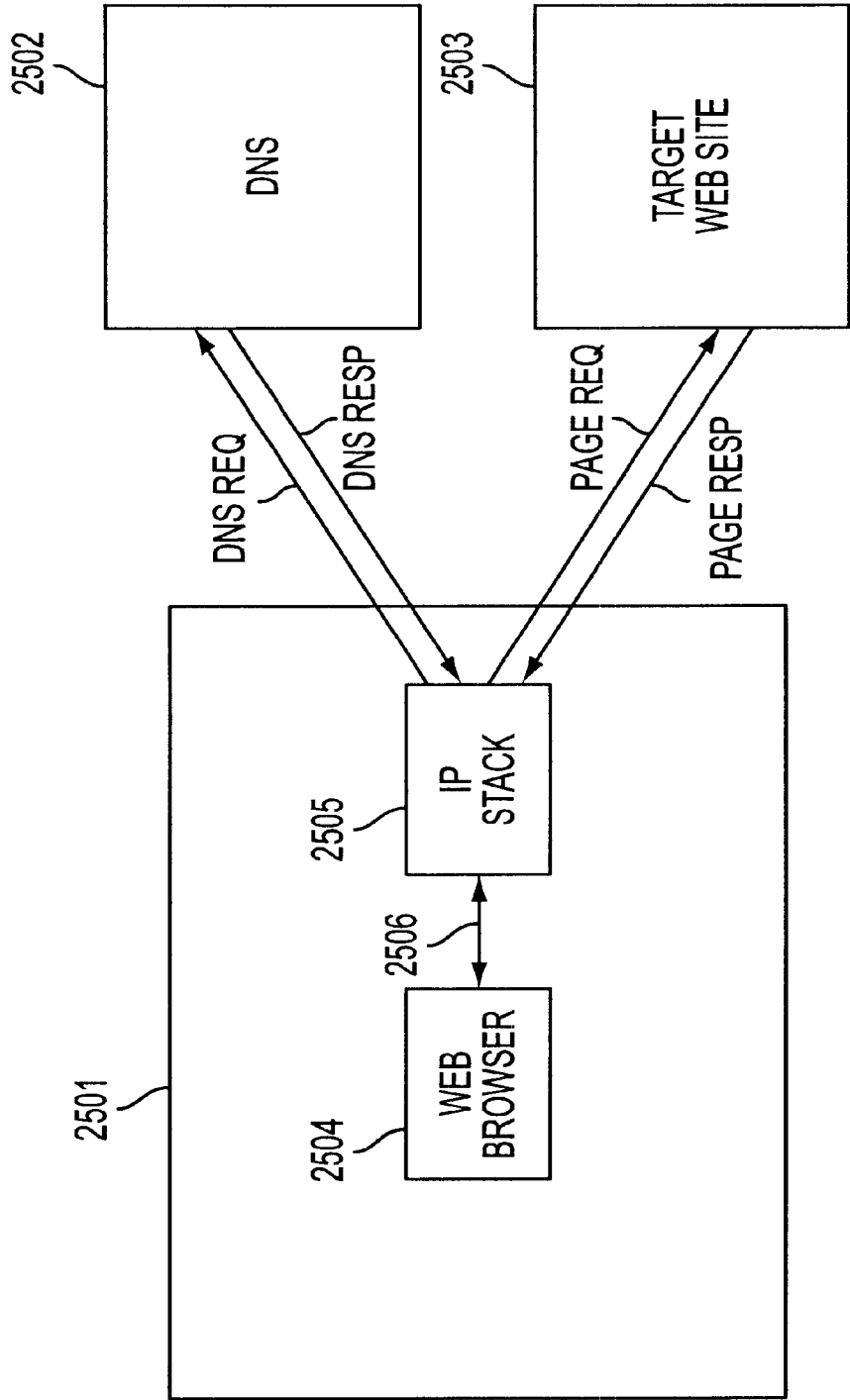


FIG. 25  
(PRIOR ART)

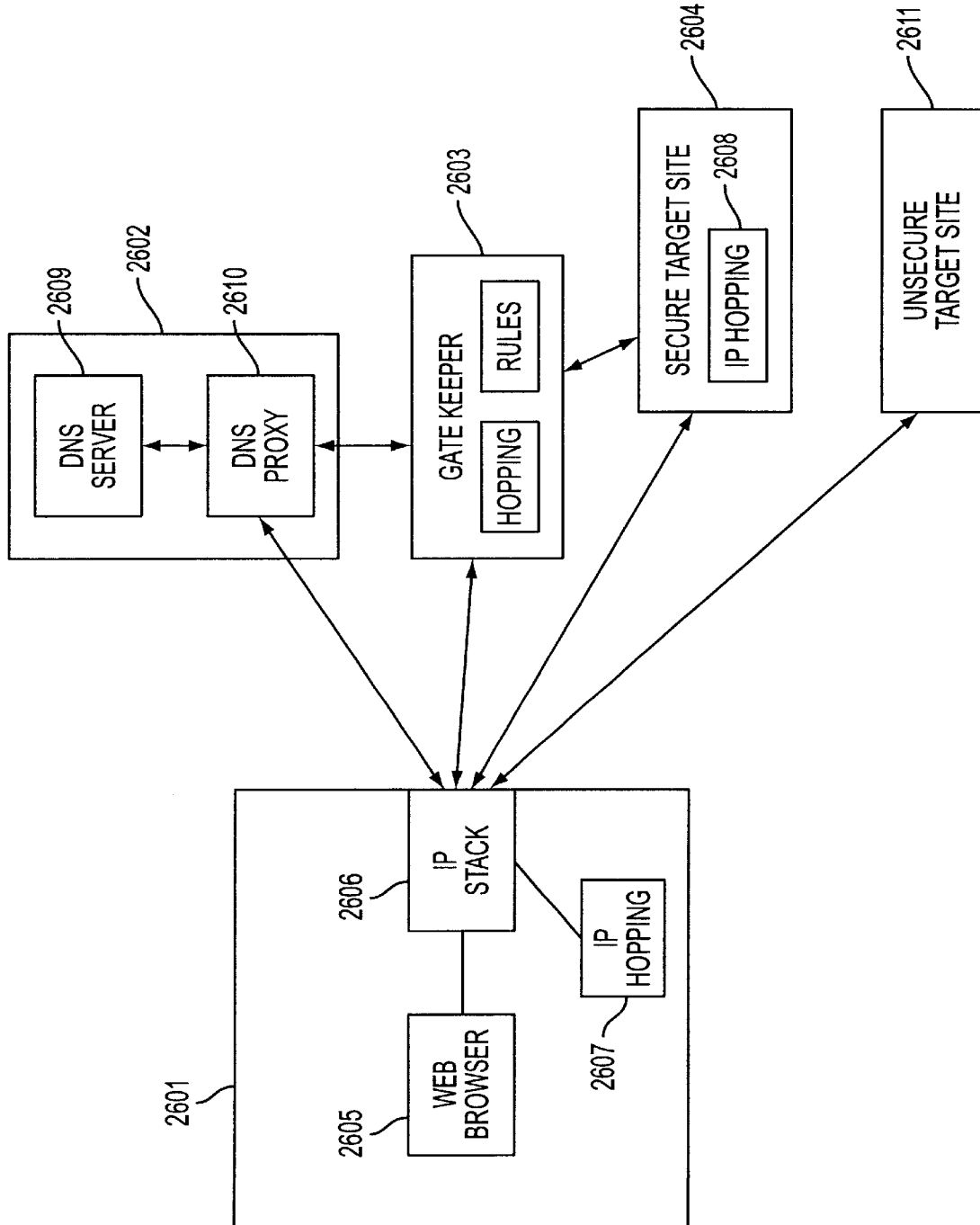


FIG. 26

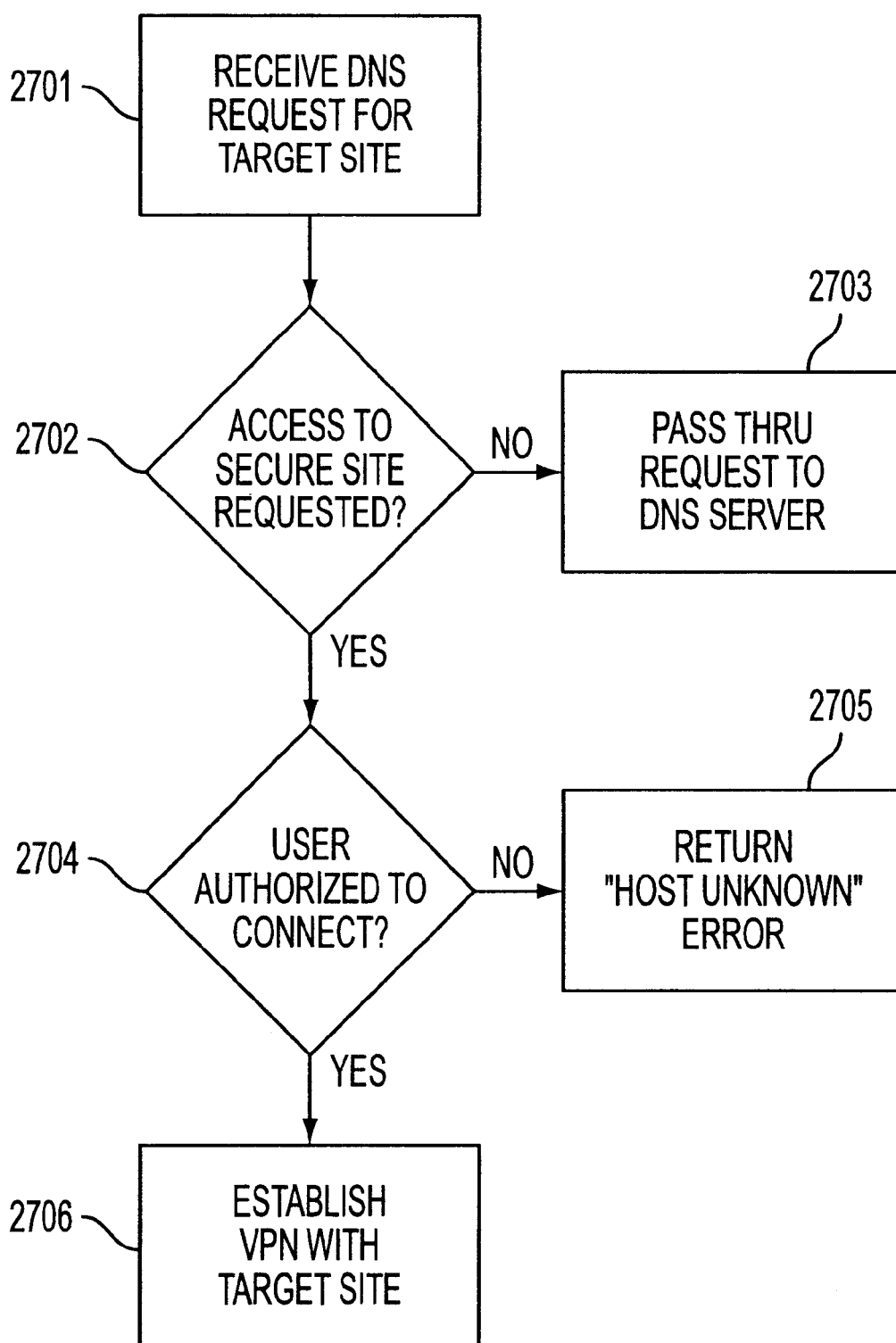


FIG. 27



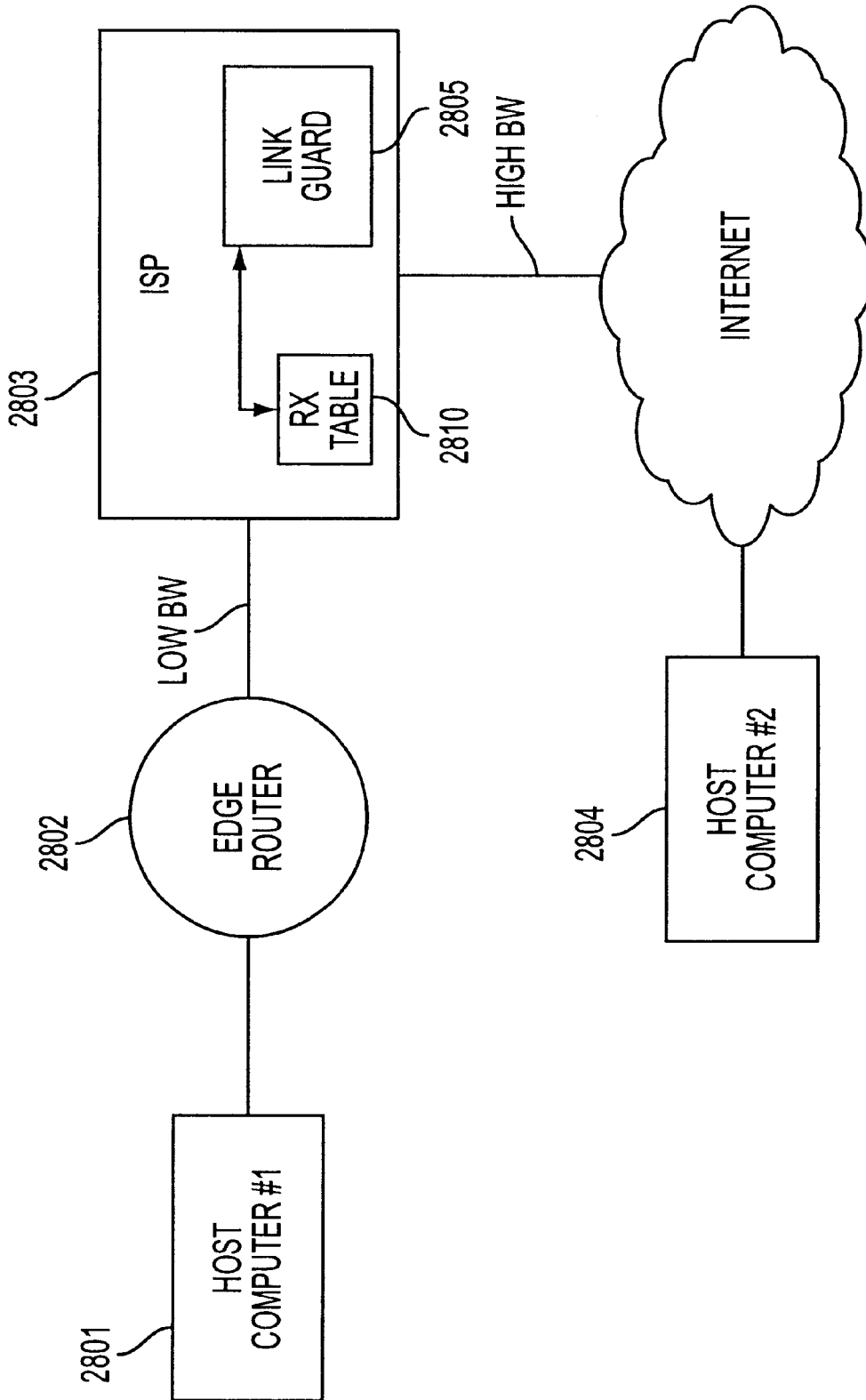


FIG. 28

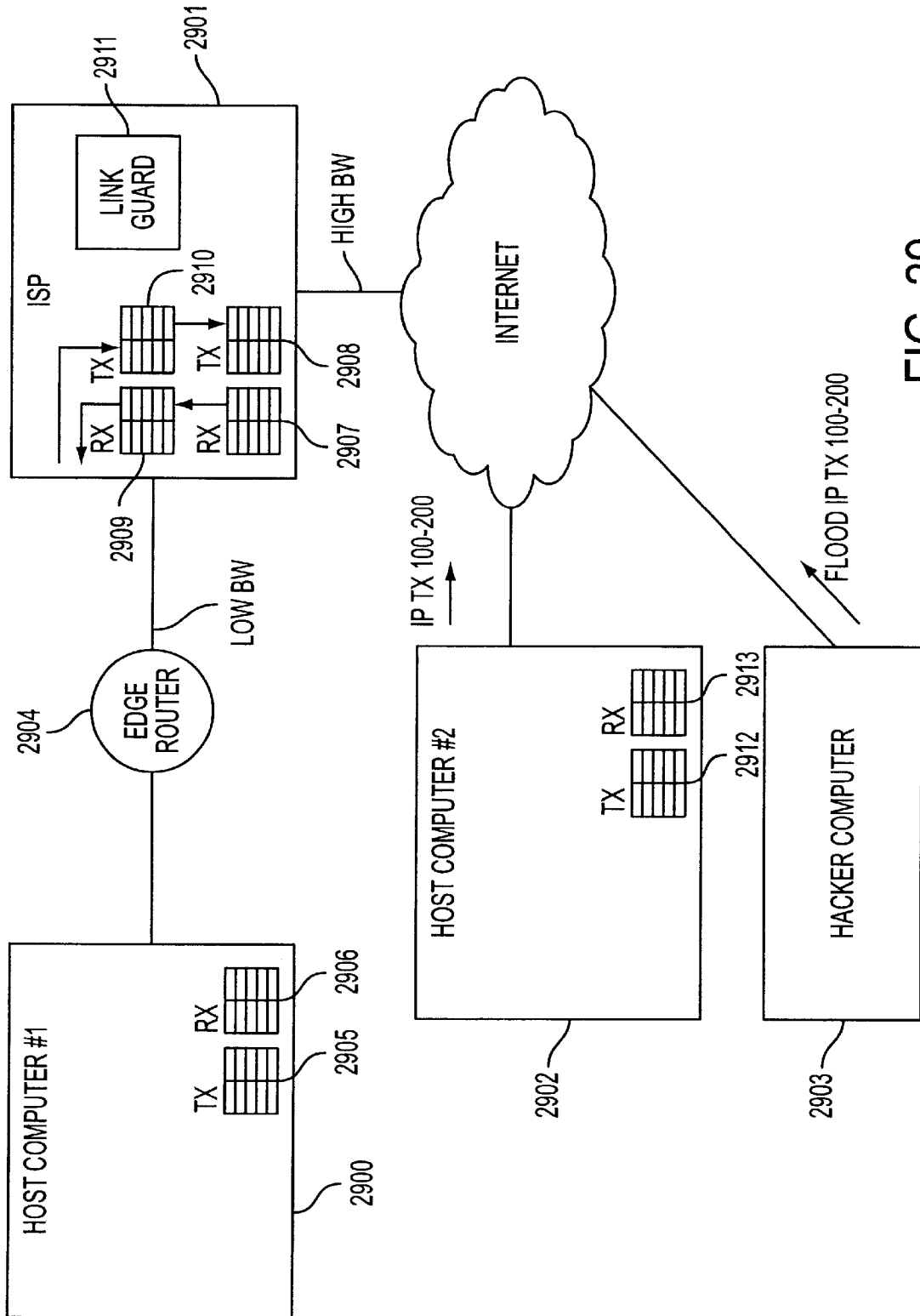
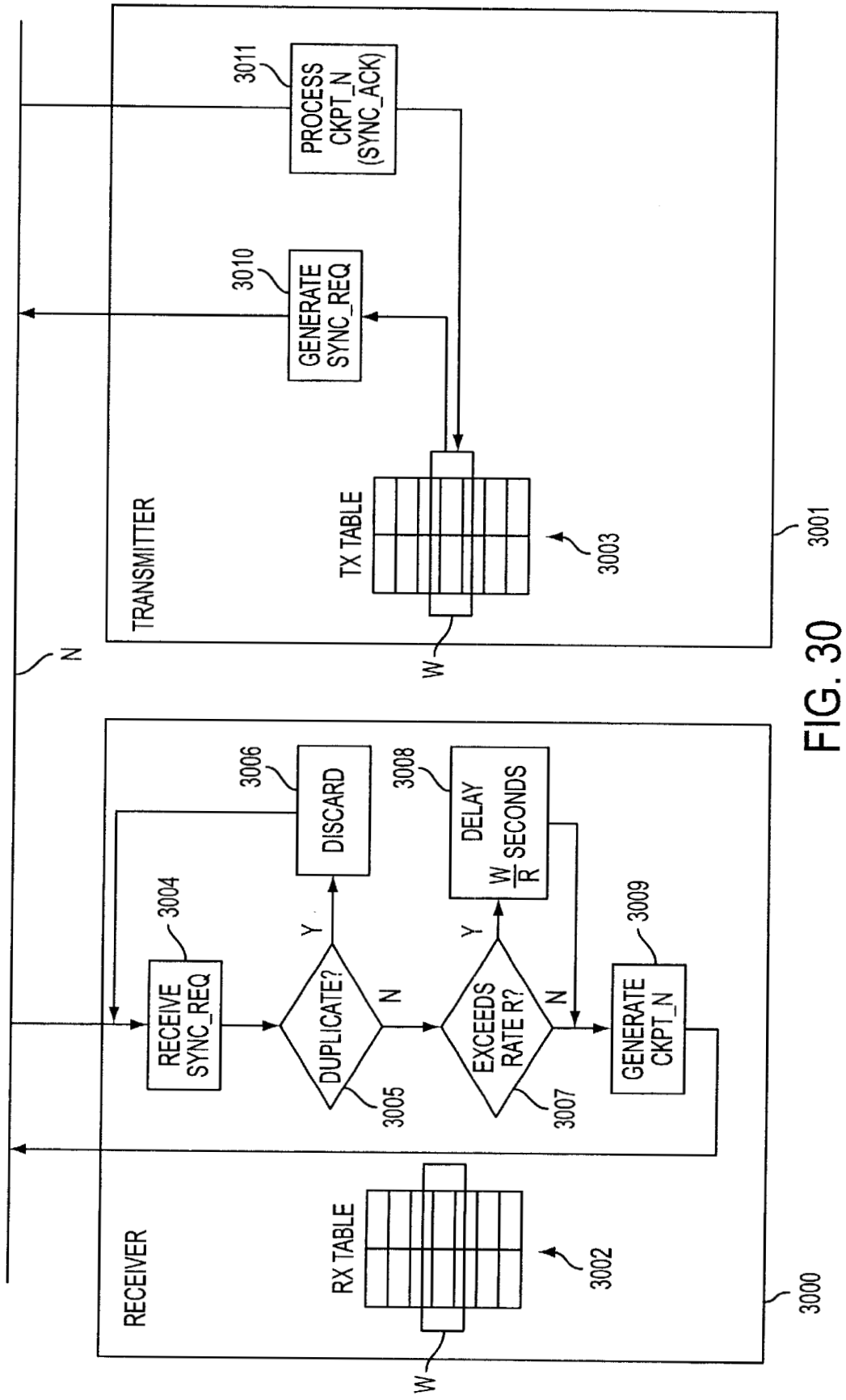


FIG. 29

Case 6:12-cv-00855-RWS Document 1-1 Filed 11/06/12 Page 36 of 73 PageID #: 51



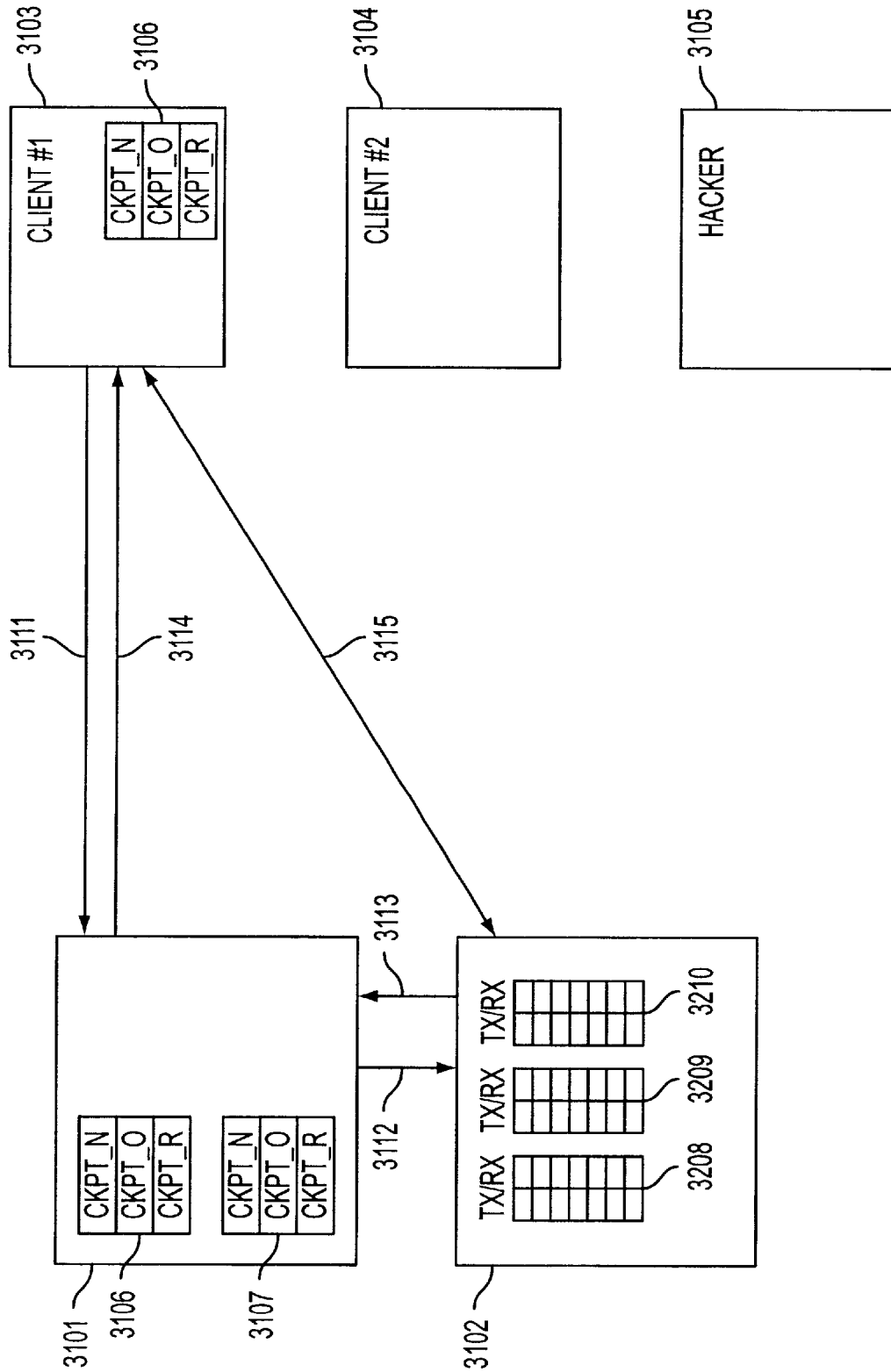


FIG. 31

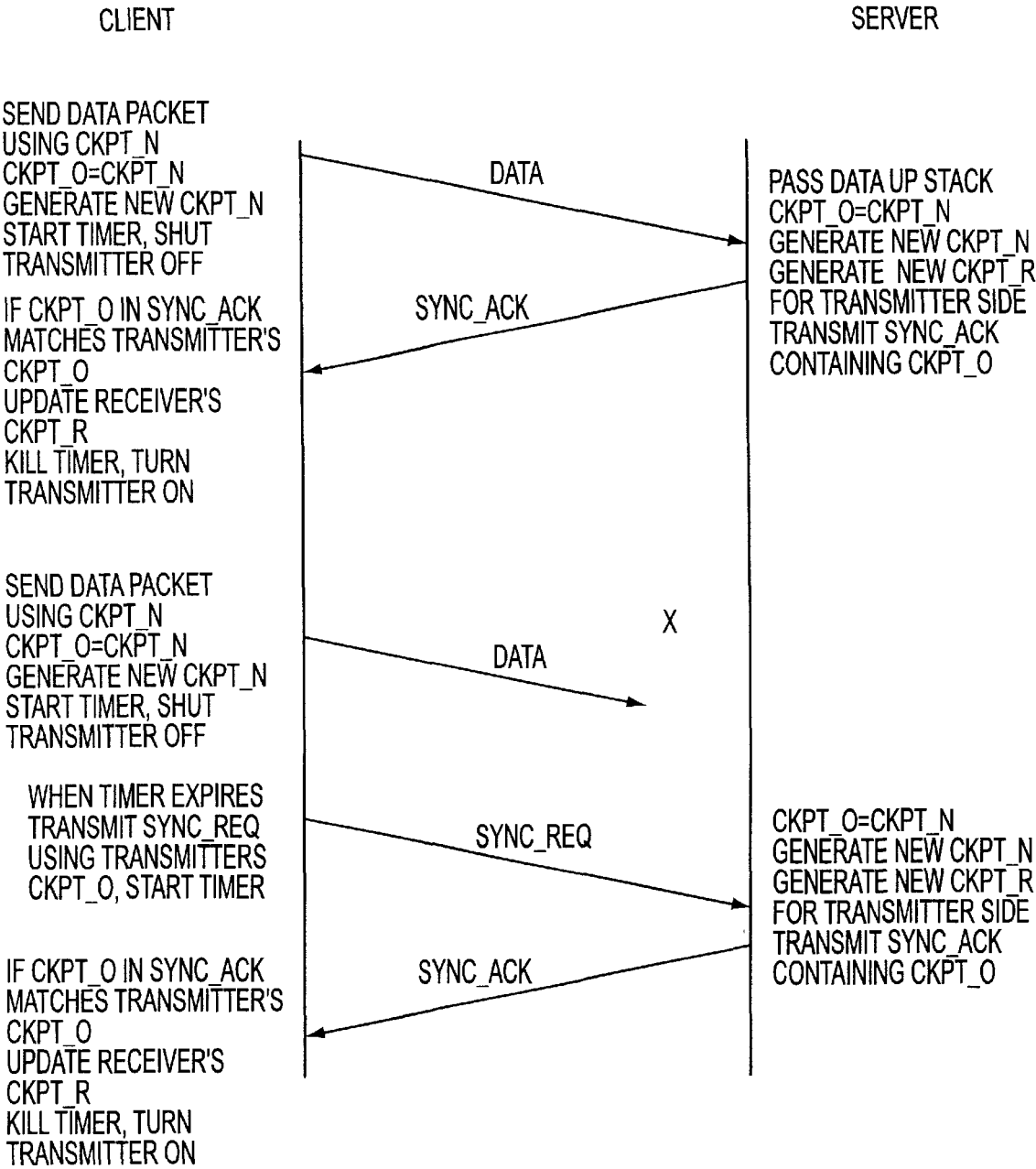


FIG. 32

**AGILE NETWORK PROTOCOL FOR  
SECURE COMMUNICATIONS WITH  
ASSURED SYSTEM AVAILABILITY**

**CROSS-REFERENCE TO RELATED  
APPLICATION**

This application claims priority from and is a continuation-in-part of previously filed U.S. application Ser. No. 09/429,643, filed on Oct. 29, 1999. The subject matter of that application, which is bodily incorporated herein, derives from provisional U.S. application No. 60/106,261 (filed Oct. 30, 1998) and No. 60/137,704 (filed Jun. 7, 1999).

**BACKGROUND OF THE INVENTION**

A tremendous variety of methods have been proposed and implemented to provide security and anonymity for communications over the Internet. The variety stems, in part, from the different needs of different Internet users. A basic heuristic framework to aid in discussing these different security techniques is illustrated in FIG. 1. Two terminals, an originating terminal **100** and a destination terminal **110** are in communication over the Internet. It is desired for the communications to be secure, that is, immune to eavesdropping. For example, terminal **100** may transmit secret information to terminal **110** over the Internet **107**. Also, it may be desired to prevent an eavesdropper from discovering that terminal **100** is in communication with terminal **110**. For example, if terminal **100** is a user and terminal **110** hosts a web site, terminal **100**'s user may not want anyone in the intervening networks to know what web sites he is "visiting." Anonymity would thus be an issue, for example, for companies that want to keep their market research interests private and thus would prefer to prevent outsiders from knowing which web-sites or other Internet resources they are "visiting." These two security issues may be called data security and anonymity, respectively.

Data security is usually tackled using some form of data encryption. An encryption key **48** is known at both the originating and terminating terminals **100** and **110**. The keys may be private and public at the originating and destination terminals **100** and **110**, respectively or they may be symmetrical keys (the same key is used by both parties to encrypt and decrypt). Many encryption methods are known and usable in this context.

To hide traffic from a local administrator or ISP, a user can employ a local proxy server in communicating over an encrypted channel with an outside proxy such that the local administrator or ISP only sees the encrypted traffic. Proxy servers prevent destination servers from determining the identities of the originating clients. This system employs an intermediate server interposed between client and destination server. The destination server sees only the Internet Protocol (IP) address of the proxy server and not the originating client. The target server only sees the address of the outside proxy. This scheme relies on a trusted outside proxy server. Also, proxy schemes are vulnerable to traffic analysis methods of determining identities of transmitters and receivers. Another important limitation of proxy servers is that the server knows the identities of both calling and called parties. In many instances, an originating terminal, such as terminal **A**, would prefer to keep its identity concealed from the proxy, for example, if the proxy server is provided by an Internet service provider (ISP).

To defeat traffic analysis, a scheme called Chaum's mixes employs a proxy server that transmits and receives fixed length messages, including dummy messages. Multiple

originating terminals are connected through a mix (a server) to multiple target servers. It is difficult to tell which of the originating terminals are communicating to which of the connected target servers, and the dummy messages confuse eavesdroppers' efforts to detect communicating pairs by analyzing traffic. A drawback is that there is a risk that the mix server could be compromised. One way to deal with this risk is to spread the trust among multiple mixes. If one mix is compromised, the identities of the originating and target terminals may remain concealed. This strategy requires a number of alternative mixes so that the intermediate servers interposed between the originating and target terminals are not determinable except by compromising more than one mix. The strategy wraps the message with multiple layers of encrypted addresses. The first mix in a sequence can decrypt only the outer layer of the message to reveal the next destination mix in sequence. The second mix can decrypt the message to reveal the next mix and so on. The target server receives the message and, optionally, a multi-layer encrypted payload containing return information to send data back in the same fashion. The only way to defeat such a mix scheme is to collude among mixes. If the packets are all fixed-length and intermixed with dummy packets, there is no way to do any kind of traffic analysis.

Still another anonymity technique, called 'crowds,' protects the identity of the originating terminal from the intermediate proxies by providing that originating terminals belong to groups of proxies called crowds. The crowd proxies are interposed between originating and target terminals. Each proxy through which the message is sent is randomly chosen by an upstream proxy. Each intermediate proxy can send the message either to another randomly chosen proxy in the "crowd" or to the destination. Thus, even crowd members cannot determine if a preceding proxy is the originator of the message or if it was simply passed from another proxy.

ZKS (Zero-Knowledge Systems) Anonymous IP Protocol allows users to select up to any of five different pseudonyms, while desktop software encrypts outgoing traffic and wraps it in User Datagram Protocol (UDP) packets. The first server in a 2+-hop system gets the UDP packets, strips off one layer of encryption to add another, then sends the traffic to the next server, which strips off yet another layer of encryption and adds a new one. The user is permitted to control the number of hops. At the final server, traffic is decrypted with an untraceable IP address. The technique is called onion-routing. This method can be defeated using traffic analysis. For a simple example, bursts of packets from a user during low-duty periods can reveal the identities of sender and receiver.

Firewalls attempt to protect LANs from unauthorized access and hostile exploitation or damage to computers connected to the LAN. Firewalls provide a server through which all access to the LAN must pass. Firewalls are centralized systems that require administrative overhead to maintain. They can be compromised by virtual-machine applications ("applets"). They instill a false sense of security that leads to security breaches for example by users sending sensitive information to servers outside the firewall or encouraging use of modems to sidestep the firewall security. Firewalls are not useful for distributed systems such as business travelers, extranets, small teams, etc.

**SUMMARY OF THE INVENTION**

A secure mechanism for communicating over the internet, including a protocol referred to as the Tunneled Agile

US 6,502,135 B1

3

Routing Protocol (TARP), uses a unique two-layer encryption format and special TARP routers. TARP routers are similar in function to regular IP routers. Each TARP router has one or more IP addresses and uses normal IP protocol to send IP packet messages ("packets" or "datagrams"). The IP packets exchanged between TARP terminals via TARP routers are actually encrypted packets whose true destination address is concealed except to TARP routers and servers. The normal or "clear" or "outside" IP header attached to TARP IP packets contains only the address of a next hop router or destination server. That is, instead of indicating a final destination in the destination field of the IP header, the TARP packet's IP header always points to a next-hop in a series of TARP router hops, or to the final destination. This means there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet since the destination could always be next-hop TARP router as well as the final destination.

Each TARP packet's true destination is concealed behind a layer of encryption generated using a link key. The link key is the encryption key used for encrypted communication between the hops intervening between an originating TARP terminal and a destination TARP terminal. Each TARP router can remove the outer layer of encryption to reveal the destination router for each TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal by the sender/receiver IP numbers in the cleartext IP header.

Once the outer layer of encryption is removed, the TARP router determines the final destination. Each TARP packet undergoes a minimum number of hops to help foil traffic analysis. The hops may be chosen at random or by a fixed value. As a result, each TARP packet may make random trips among a number of geographically disparate routers before reaching its destination. Each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined. This feature is called agile routing. The fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. The associated advantages have to do with the inner layer of encryption discussed below. Agile routing is combined with another feature that furthers this purpose; a feature that ensures that any message is broken into multiple packets.

The IP address of a TARP router can be changed, a feature called IP agility. Each TARP router, independently or under direction from another TARP terminal or router, can change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs.

The message payload is hidden behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the intervening TARP routers. The session key is used to decrypt the payloads of the TARP packets permitting the data stream to be reconstructed.

Communication may be made private using link and session keys, which in turn may be shared and used according to any desired method. For example, public/private keys or symmetric keys may be used.

4

To transmit a data stream, a TARP originating terminal constructs a series of TARP packets from a series of IP packets generated by a network (IP) layer process. (Note that the terms "network layer," "data link layer," "application layer," etc. used in this specification correspond to the Open Systems Interconnection (OSI) network terminology.) The payloads of these packets are assembled into a block and chain-block encrypted using the session key. This assumes, of course, that all the IP packets are destined for the same TARP terminal. The block is then interleaved and the interleaved encrypted block is broken into a series of payloads, one for each TARP packet to be generated. Special TARP headers IPT are then added to each payload using the IP headers from the data stream packets. The TARP headers can be identical to normal IP headers or customized in some way. They should contain a formula or data for deinterleaving the data at the destination TARP terminal, a time-to-live (TTL) parameter to indicate the number of hops still to be executed, a data type identifier which indicates whether the payload contains, for example, TCP or UDP data, the sender's TARP address, the destination TARP address, and an indicator as to whether the packet contains real or decoy data or a formula for filtering out decoy data if decoy data is spread in some way through the TARP payload data.

Note that although chain-block encryption is discussed here with reference to the session key, any encryption method may be used. Preferably, as in chain block encryption, a method should be used that makes unauthorized decryption difficult without an entire result of the encryption process. Thus, by separating the encrypted block among multiple packets and making it difficult for an interloper to obtain access to all of such packets, the contents of the communications are provided an extra layer of security.

Decoy or dummy data can be added to a stream to help foil traffic analysis by reducing the peak-to-average network load. It may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to portion, or entirety, of a message, and that portion or entirety then interleaved into a number of separate packets. Considering the agile IP routing of the packets, and the attendant difficulty of reconstructing an entire sequence of packets to form a single block-encrypted message element, decoy packets can significantly increase the difficulty of reconstructing an entire data stream.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Because the encryption system described above is insertable between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as

5

well. Thus, no operations at or above the Network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicating that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. In addition, it may create a subprocess that maintains the original IP address and continues interacting with the attacker in some manner.

Decoy packets may be generated by each TARP terminal on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis.

In various other embodiments of the invention, a scalable version of the system may be constructed in which a plurality of IP addresses are preassigned to each pair of communicating nodes in the network. Each pair of nodes agrees upon an algorithm for "hopping" between IP addresses (both sending and receiving), such that an eavesdropper sees apparently continuously random IP address pairs (source and destination) for packets transmitted between the pair. Overlapping or "reusable" IP addresses may be allocated to different users on the same subnet, since each node merely verifies that a particular packet includes a valid source/destination pair from the agreed-upon algorithm. Source/destination pairs are preferably not reused between any two nodes during any given end-to-end session, though limited IP block sizes or lengthy sessions might require it.

Further improvements described in this continuation-in-part application include: (1) a load balancer that distributes packets across different transmission paths according to

6

transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is an illustration of secure communications over the Internet according to a prior art embodiment.

FIG. 2 is an illustration of secure communications over the Internet according to an embodiment of the invention.

FIG. 3a is an illustration of a process of forming a tunneled IP packet according to an embodiment of the invention.

FIG. 3b is an illustration of a process of forming a tunneled IP packet according to another embodiment of the invention.

FIG. 4 is an illustration of an OSI layer location of processes that may be used to implement the invention.

FIG. 5 is a flow chart illustrating a process for routing a tunneled packet according to an embodiment of the invention.

FIG. 6 is a flow chart illustrating a process for forming a tunneled packet according to an embodiment of the invention.

FIG. 7 is a flow chart illustrating a process for receiving a tunneled packet according to an embodiment of the invention.

FIG. 8 shows how a secure session is established and synchronized between a client and a TARP router.

FIG. 9 shows an IP address hopping scheme between a client computer and TARP router using transmit and receive tables in each computer.

FIG. 10 shows physical link redundancy among three Internet Service Providers (ISPs) and a client computer.

FIG. 11 shows how multiple IP packets can be embedded into a single "frame" such as an Ethernet frame, and further shows the use of a discriminator field to camouflage true packet recipients.

FIG. 12A shows a system that employs hopped hardware addresses, hopped IP addresses, and hopped discriminator fields.

FIG. 12B shows several different approaches for hopping hardware addresses, IP addresses, and discriminator fields in combination.

FIG. 13 shows a technique for automatically re-establishing synchronization between sender and receiver through the use of a partially public sync value.

FIG. 14 shows a "checkpoint" scheme for regaining synchronization between a sender and recipient.

FIG. 15 shows further details of the checkpoint scheme of FIG. 14.

FIG. 16 shows how two addresses can be decomposed into a plurality of segments for comparison with presence vectors.

FIG. 17 shows a storage array for a receiver's active addresses.

FIG. 18 shows the receiver's storage array after receiving a sync request.



FIG. 19 shows the receiver's storage array after new addresses have been generated.

FIG. 20 shows a system employing distributed transmission paths.

FIG. 21 shows a plurality of link transmission tables that can be used to route packets in the system of FIG. 20.

FIG. 22A shows a flowchart for adjusting weight value distributions associated with a plurality of transmission links.

FIG. 22B shows a flowchart for setting a weight value to zero if a transmitter turns off.

FIG. 23 shows a system employing distributed transmission paths with adjusted weight value distributions for each path.

FIG. 24 shows an example using the system of FIG. 23.

FIG. 25 shows a conventional domain-name look-up service.

FIG. 26 shows a system employing a DNS proxy server with transparent VPN creation.

FIG. 27 shows steps that can be carried out to implement transparent VPN creation based on a DNS look-up function.

FIG. 28 shows a system including a link guard function that prevents packet overloading on a low-bandwidth link LOW BW.

FIG. 29 shows one embodiment of a system employing the principles of FIG. 28.

FIG. 30 shows a system that regulates packet transmission rates by throttling the rate at which synchronizations are performed.

FIG. 31 shows a signaling server 3101 and a transport server 3102 used to establish a VPN with a client computer.

FIG. 32 shows message flows relating to synchronization protocols of FIG. 31.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. 2, a secure mechanism for communicating over the internet employs a number of special routers or servers, called TARP routers 122-127 that are similar to regular IP routers 128-132 in that each has one or more IP addresses and uses normal IP protocol to send normal-looking IP packet messages, called TARP packets 140. TARP packets 140 are identical to normal IP packet messages that are routed by regular IP routers 128-132 because each TARP packet 140 contains a destination address as in a normal IP packet. However, instead of indicating a final destination in the destination field of the IP header, the TARP packet's 140 IP header always points to a next-hop in a series of TARP router hops, or the final destination, TARP terminal 110. Because the header of the TARP packet contains only the next-hop destination, there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet 140 since the destination could always be the next-hop TARP router as well as the final destination, TARP terminal 110.

Each TARP packet's true destination is concealed behind an outer layer of encryption generated using a link key 146. The link key 146 is the encryption key used for encrypted communication between the end points (TARP terminals or TARP routers) of a single link in the chain of hops connecting the originating TARP terminal 100 and the destination TARP terminal 110. Each TARP router 122-127, using the link key 146 it uses to communicate with the previous hop in a chain, can use the link key to reveal the true destination

of a TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal (which may indicate the link key used) by the sender field of the clear IP header. Alternatively, this identity may be hidden behind another layer of encryption in available bits in the clear IP header. Each TARP router, upon receiving a TARP message, determines if the message is a TARP message by using authentication data in the TARP packet. This could be recorded in available bytes in the TARP packet's IP header. Alternatively, TARP packets could be authenticated by attempting to decrypt using the link key 146 and determining if the results are as expected. The former may have computational advantages because it does not involve a decryption process.

Once the outer layer of decryption is completed by a TARP router 122-127, the TARP router determines the final destination. The system is preferably designed to cause each TARP packet 140 to undergo a minimum number of hops to help foil traffic analysis. The time to live counter in the IP header of the TARP message may be used to indicate a number of TARP router hops yet to be completed. Each TARP router then would decrement the counter and determine from that whether it should forward the TARP packet 140 to another TARP router 122-127 or to the destination TARP terminal 110. If the time to live counter is zero or below zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to the destination TARP terminal 110. If the time to live counter is above zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to a TARP router 122-127 that the current TARP terminal chooses at random. As a result, each TARP packet 140 is routed through some minimum number of hops of TARP routers 122-127 which are chosen at random.

Thus, each TARP packet, irrespective of the traditional factors determining traffic in the Internet, makes random trips among a number of geographically disparate routers before reaching its destination and each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined as described above. This feature is called agile routing. For reasons that will become clear shortly, the fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. Agile routing is combined with another feature that furthers this purpose, a feature that ensures that any message is broken into multiple packets.

A TARP router receives a TARP packet when an IP address used by the TARP router coincides with the IP address in the TARP packet's IP header IP<sub>c</sub>. The IP address of a TARP router, however, may not remain constant. To avoid and manage attacks, each TARP router, independently or under direction from another TARP terminal or router, may change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs. In reality, whenever a TARP router looks up the address of a destination in the encrypted header, it must convert a TARP address to a real IP address using its LUT.

While every TARP router receiving a TARP packet has the ability to determine the packet's final destination, the message payload is embedded behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the TARP routers 122-127 intervening between the originating 100 and destination 110 TARP terminals. The session key is used to decrypt the payloads of the TARP packets 140 permitting an entire message to be reconstructed.

In one embodiment, communication may be made private using link and session keys, which in turn may be shared and used according any desired method. For example, a public key or symmetric keys may be communicated between link or session endpoints using a public key method. Any of a variety of other mechanisms for securing data to ensure that only authorized computers can have access to the private information in the TARP packets 140 may be used as desired.

Referring to FIG. 3a, to construct a series of TARP packets, a data stream 300 of IP packets 207a, 207b, 207c, etc., such series of packets being formed by a network (IP) layer process, is broken into a series of small sized segments. In the present example, equal-sized segments 1-9 are defined and used to construct a set of interleaved data packets A, B, and C. Here it is assumed that the number of interleaved packets A, B, and C formed is three and that the number of IP packets 207a-207c used to form the three interleaved packets A, B, and C is exactly three. Of course, the number of IP packets spread over a group of interleaved packets may be any convenient number as may be the number of interleaved packets over which the incoming data stream is spread. The latter, the number of interleaved packets over which the data stream is spread, is called the interleave window.

To create a packet, the transmitting software interleaves the normal IP packets 207a et. seq. to form a new set of interleaved payload data 320. This payload data 320 is then encrypted using a session key to form a set of session-key-encrypted payload data 330, each of which, A, B, and C, will form the payload of a TARP packet. Using the IP header data, from the original packets 207a-207c, new TARP headers IP<sub>T</sub> are formed. The TARP headers IP<sub>T</sub> can be identical to normal IP headers or customized in some way. In a preferred embodiment, the TARP headers IP<sub>T</sub> are IP headers with added data providing the following information required for routing and reconstruction of messages, some of which data is ordinarily, or capable of being, contained in normal IP headers:

1. A window sequence number—an identifier that indicates where the packet belongs in the original message sequence.
2. An interleave sequence number—an identifier that indicates the interleaving sequence used to form the packet so that the packet can be deinterleaved along with other packets in the interleave window.
3. A time-to-live (TTL) datum—indicates the number of TARP-router-hops to be executed before the packet reaches its destination. Note that the TTL parameter may provide a datum to be used in a probabilistic formula for determining whether to route the packet to the destination or to another hop.
4. Data type identifier—indicates whether the payload contains, for example, TCP or UDP data.
5. Sender's address—indicates the sender's address in the TARP network.

6. Destination address—indicates the destination terminal's address in the TARP network.
7. Decoy/Real—an indicator of whether the packet contains real message data or dummy decoy data or a combination.

Obviously, the packets going into a single interleave window must include only packets with a common destination. Thus, it is assumed in the depicted example that the IP headers of IP packets 207a-207c all contain the same destination address or at least will be received by the same terminal so that they can be deinterleaved. Note that dummy or decoy data or packets can be added to form a larger interleave window than would otherwise be required by the size of a given message. Decoy or dummy data can be added to a stream to help foil traffic analysis by leveling the load on the network. Thus, it may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or the entirety, of a message, and that portion or entirety then interleaved into a number of separate packets.

Referring to FIG. 3b, in an alternative mode of TARP packet construction, a series of IP packets are accumulated to make up a predefined interleave window. The payloads of the packets are used to construct a single block 520 for chain block encryption using the session key. The payloads used to form the block are presumed to be destined for the same terminal. The block size may coincide with the interleave window as depicted in the example embodiment of FIG. 3b. After encryption, the encrypted block is broken into separate payloads and segments which are interleaved as in the embodiment of FIG. 3a. The resulting interleaved packets A, B, and C, are then packaged as TARP packets with TARP headers as in the Example of FIG. 3a. The remaining process is as shown in, and discussed with reference to, FIG. 3a.

Once the TARP packets 340 are formed, each entire TARP packet 340, including the TARP header IP<sub>T</sub>, is encrypted using the link key for communication with the first-hop-TARP router. The first hop TARP router is randomly chosen. A final unencrypted IP header IP<sub>C</sub> is added to each encrypted TARP packet 340 to form a normal IP packet 360 that can be transmitted to a TARP router. Note that the process of constructing the TARP packet 360 does not have to be done in stages as described. The above description is just a useful heuristic for describing the final product, namely, the TARP packet.

Note that, TARP header IP<sub>T</sub> could be a completely custom header configuration with no similarity to a normal IP header except that it contain the information identified above. This is so since this header is interpreted by only TARP routers.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Referring to FIG. 4, a TARP transceiver 405 can be an originating terminal 100, a destination terminal

110, or a TARP router 122-127. In each TARP Transceiver 405, a transmitting process is generated to receive normal packets from the Network (IP) layer and generate TARP packets for communication over the network. A receiving process is generated to receive normal IP packets containing TARP packets and generate from these normal IP packets which are "passed up" to the Network (IP) layer. Note that where the TARP Transceiver 405 is a router, the received TARP packets 140 are not processed into a stream of IP packets 415 because they need only be authenticated as proper TARP packets and then passed to another TARP router or a TARP destination terminal 110. The intervening process, a "TARP Layer" 420, could be combined with either the data link layer 430 or the Network layer 410. In either case, it would intervene between the data link layer 430 so that the process would receive regular IP packets containing embedded TARP packets and "hand up" a series of reassembled IP packets to the Network layer 410. As an example of combining the TARP layer 420 with the data link layer 430, a program may augment the normal processes running a communications card, for example, an Ethernet card. Alternatively, the TARP layer processes may form part of a dynamically loadable module that is loaded and executed to support communications between the network and data link layers.

Because the encryption system described above can be inserted between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

Note that IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicates that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. To accomplish this, the TARP process will construct a TARP-formatted message, in the style of Internet Control Message Protocol (ICMP) datagrams as an example; this message will contain the machine's TARP address, its previous IP address, and its new IP address. The TARP layer will transmit this packet to at least one known TARP router; then upon receipt and validation of the message, the TARP router will update its LUT with the new IP address for the stated TARP address. The TARP router will then format a similar message, and broadcast it to the other TARP routers so that they may update their LUTs. Since the total number

of TARP routers on any given subnet is expected to be relatively small, this process of updating the LUTs should be relatively fast. It may not, however, work as well when there is a relatively large number of TARP routers and/or a relatively large number of clients; this has motivated a refinement of this architecture to provide scalability; this refinement has led to a second embodiment, which is discussed below.

Upon detection of an attack, the TARP process may also create a subprocess that maintains the original IP address and continues interacting with the attacker. The latter may provide an opportunity to trace the attacker or study the attacker's methods (called "fishbowling" drawing upon the analogy of a small fish in a fish bowl that "thinks" it is in the ocean but is actually under captive observation). A history of the communication between the attacker and the abandoned (fishbowed) IP address can be recorded or transmitted for human analysis or further synthesized for purposes of responding in some way.

As mentioned above, decoy or dummy data or packets can be added to outgoing data streams by TARP terminals or routers. In addition to making it convenient to spread data over a larger number of separate packets, such decoy packets can also help to level the load on inactive portions of the Internet to help foil traffic analysis efforts.

Decoy packets may be generated by each TARP terminal 100, 110 or each router 122-127 on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. That is, when a series of messages are received, the decoy packet generation rate may be increased. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis. The rate of reception of packets, decoy or otherwise, may be indicated to the decoy packet dropping and generating processes through perishable decoy and regular packet counters. (A perishable counter is one that resets or decrements its value in response to time so that it contains a high value when it is incremented in rapid succession and a small value when incremented either slowly or a small number of times in rapid succession.) Note that destination TARP terminal 110 may generate decoy packets equal in number and size to those TARP packets received to make it appear it is merely routing packets and is therefore not the destination terminal.

Referring to FIG. 5, the following particular steps may be employed in the above-described method for routing TARP packets.

- S0. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.
- S2. The TARP packet may be probed in some way to authenticate the packet before attempting to decrypt it

## US 6,502,135 B1

13

using the link key. That is, the router may determine that the packet is an authentic TARP packet by performing a selected operation on some data included with the clear IP header attached to the encrypted TARP packet contained in the payload. This makes it possible to avoid performing decryption on packets that are not authentic TARP packets.

- S3. The TARP packet is decrypted to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.
- S4. If the packet is a decoy packet, the perishable decoy counter is incremented.
- S5. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the router may choose to throw it away. If the received packet is a decoy packet and it is determined that it should be thrown away (S6), control returns to step S0.
- S7. The TTL parameter of the TARP header is decremented and it is determined if the TTL parameter is greater than zero.
- S8. If the TTL parameter is greater than zero, a TARP address is randomly chosen from a list of TARP addresses maintained by the router and the link key and IP address corresponding to that TARP address memorized for use in creating a new IP packet containing the TARP packet.
- S9. If the TTL parameter is zero or less, the link key and IP address corresponding to the TARP address of the destination are memorized for use in creating the new IP packet containing the TARP packet.
- S10. The TARP packet is encrypted using the memorized link key.
- S11. An IP header is added to the packet that contains the stored IP address, the encrypted TARP packet wrapped with an IP header, and the completed packet transmitted to the next hop or destination.

Referring to FIG. 6, the following particular steps may be employed in the above-described method for generating TARP packets.

- S20. A background loop operation applies an algorithm that determines the generation of decoy IP packets. The loop is interrupted when a data stream containing IP packets is received for transmission.
- S21. The received IP packets are grouped into a set consisting of messages with a constant IP destination address. The set is further broken down to coincide with a maximum size of an interleave window. The set is encrypted, and interleaved into a set of payloads destined to become TARP packets.
- S22. The TARP address corresponding to the IP address is determined from a lookup table and stored to generate the TARP header. An initial TTL count is generated and stored in the header. The TTL count may be random with minimum and maximum values or it may be fixed or determined by some other parameter.
- S23. The window sequence numbers and interleave sequence numbers are recorded in the TARP headers of each packet.
- S24. One TARP router address is randomly chosen for each TARP packet and the IP address corresponding to it stored for use in the clear IP header. The link key corresponding to this router is identified and used to encrypt TARP packets containing interleaved and encrypted data and TARP headers.

14

S25. A clear IP header with the first hop router's real IP address is generated and added to each of the encrypted TARP packets and the resulting packets.

Referring to FIG. 7, the following particular steps may be employed in the above-described method for receiving TARP packets.

- S40. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.
- S42. The TARP packet may be probed to authenticate the packet before attempting to decrypt it using the link key.
- S43. The TARP packet is decrypted with the appropriate link key to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.
- S44. If the packet is a decoy packet, the perishable decoy counter is incremented.
- S45. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the receiver may choose to throw it away.
- S46. The TARP packets are cached until all packets forming an interleave window are received.
- S47. Once all packets of an interleave window are received, the packets are deinterleaved.
- S48. The packets block of combined packets defining the interleave window is then decrypted using the session key.
- S49. The decrypted block is then divided using the window sequence data and the IP<sub>T</sub> headers are converted into normal IP<sub>C</sub> headers. The window sequence numbers are integrated in the IP<sub>C</sub> headers.
- S50. The packets are then handed up to the IP layer processes.

#### 1. SCALABILITY ENHANCEMENTS

The IP agility feature described above relies on the ability to transmit IP address changes to all TARP routers. The embodiments including this feature will be referred to as "boutique" embodiments due to potential limitations in scaling these features up for a large network, such as the Internet. (The "boutique" embodiments would, however, be robust for use in smaller networks, such as small virtual private networks, for example). One problem with the boutique embodiments is that if IP address changes are to occur frequently, the message traffic required to update all routers sufficiently quickly creates a serious burden on the Internet when the TARP router and/or client population gets large. The bandwidth burden added to the networks, for example in ICMP packets, that would be used to update all the TARP routers could overwhelm the Internet for a large scale implementation that approached the scale of the Internet. In other words, the boutique system's scalability is limited.

A system can be constructed which trades some of the features of the above embodiments to provide the benefits of IP agility without the additional messaging burden. This is accomplished by IP address-hopping according to shared algorithms that govern IP addresses used between links participating in communications sessions between nodes such as TARP nodes. (Note that the IP hopping technique is also applicable to the boutique embodiment.) The IP agility feature discussed with respect to the boutique system can be

## US 6,502,135 B1

15

modified so that it becomes decentralized under this scalable regime and governed by the above-described shared algorithm. Other features of the boutique system may be combined with this new type of IP-agility.

The new embodiment has the advantage of providing IP agility governed by a local algorithm and set of IP addresses exchanged by each communicating pair of nodes. This local governance is session-independent in that it may govern communications between a pair of nodes, irrespective of the session or end points being transferred between the directly communicating pair of nodes.

In the scalable embodiments, blocks of IP addresses are allocated to each node in the network. (This scalability will increase in the future, when Internet Protocol addresses are increased to 128-bit fields, vastly increasing the number of distinctly addressable nodes). Each node can thus use any of the IP addresses assigned to that node to communicate with other nodes in the network. Indeed, each pair of communicating nodes can use a plurality of source IP addresses and destination IP addresses for communicating with each other.

Each communicating pair of nodes in a chain participating in any session stores two blocks of IP addresses, called netblocks, and an algorithm and randomization seed for selecting, from each netblock, the next pair of source/destination IP addresses that will be used to transmit the next message. In other words, the algorithm governs the sequential selection of IP-address pairs, one sender and one receiver IP address, from each netblock. The combination of algorithm, seed, and netblock (IP address block) will be called a "hopblock." A router issues separate transmit and receive hopblocks to its clients. The send address and the receive address of the IP header of each outgoing packet sent by the client are filled with the send and receive IP addresses generated by the algorithm. The algorithm is "clocked" (indexed) by a counter so that each time a pair is used, the algorithm turns out a new transmit pair for the next packet to be sent.

The router's receive hopblock is identical to the client's transmit hopblock. The router uses the receive hopblock to predict what the send and receive IP address pair for the next expected packet from that client will be. Since packets can be received out of order, it is not possible for the router to predict with certainty what IP address pair will be on the next sequential packet. To account for this problem, the router generates a range of predictions encompassing the number of possible transmitted packet send/receive addresses, of which the next packet received could leap ahead. Thus, if there is a vanishingly small probability that a given packet will arrive at the router ahead of 5 packets transmitted by the client before the given packet, then the router can generate a series of 6 send/receive IP address pairs (or "hop window") to compare with the next received packet. When a packet is received, it is marked in the hop window as such, so that a second packet with the same IP address pair will be discarded. If an out-of-sequence packet does not arrive within a predetermined timeout period, it can be requested for retransmission or simply discarded from the receive table, depending upon the protocol in use for that communications session, or possibly by convention.

When the router receives the client's packet, it compares the send and receive IP addresses of the packet with the next N predicted send and receive IP address pairs and rejects the packet if it is not a member of this set. Received packets that do not have the predicted source/destination IP addresses falling with the window are rejected, thus thwarting possible hackers. (With the number of possible combinations, even a

16

fairly large window would be hard to fall into at random.) If it is a member of this set, the router accepts the packet and processes it further. This link-based IP-hopping strategy, referred to as "IHOP," is a network element that stands on its own and is not necessarily accompanied by elements of the boutique system described above. If the routing agility feature described in connection with the boutique embodiment is combined with this link-based IP-hopping strategy, the router's next step would be to decrypt the TARP header to determine the destination TARP router for the packet and determine what should be the next hop for the packet. The TARP router would then forward the packet to a random TARP router or the destination TARP router with which the source TARP router has a link-based IP hopping communication established.

FIG. 8 shows how a client computer **801** and a TARP router **811** can establish a secure session. When client **801** seeks to establish an IHOP session with TARP router **811**, the client **801** sends "secure synchronization" request ("SSYN") packet **821** to the TARP router **811**. This SYN packet **821** contains the client's **801** authentication token, and may be sent to the router **811** in an encrypted format. The source and destination IP numbers on the packet **821** are the client's **801** current fixed IP address, and a "known" fixed IP address for the router **811**. (For security purposes, it may be desirable to reject any packets from outside of the local network that are destined for the router's known fixed IP address.) Upon receipt and validation of the client's **801** SSYN packet **821**, the router **811** responds by sending an encrypted "secure synchronization acknowledgment" ("SSYN ACK") **822** to the client **801**. This SSYN ACK **822** will contain the transmit and receive hopblocks that the client **801** will use when communicating with the TARP router **811**. The client **801** will acknowledge the TARP router's **811** response packet **822** by generating an encrypted SSYN ACK ACK packet **823** which will be sent from the client's **801** fixed IP address and to the TARP router's **811** known fixed IP address. The client **801** will simultaneously generate a SSYN ACK ACK packet; this SSYN ACK packet, referred to as the Secure Session Initiation (SSI) packet **824**, will be sent with the first {sender, receiver} IP pair in the client's transmit table **921** (FIG. 9), as specified in the transmit hopblock provided by the TARP router **811** in the SSYN ACK packet **822**. The TARP router **811** will respond to the SSI packet **824** with an SSI ACK packet **825**, which will be sent with the first {sender, receiver} IP pair in the TARP router's transmit table **923**. Once these packets have been successfully exchanged, the secure communications session is established, and all further secure communications between the client **801** and the TARP router **811** will be conducted via this secure session, as long as synchronization is maintained. If synchronization is lost, then the client **801** and TARP router **802** may re-establish the secure session by the procedure outlined in FIG. 8 and described above.

While the secure session is active, both the client **901** and TARP router **911** (FIG. 9) will maintain their respective transmit tables **921**, **923** and receive tables **922**, **924**, as provided by the TARP router during session synchronization **822**. It is important that the sequence of IP pairs in the client's transmit table **921** be identical to those in the TARP router's receive table **924**; similarly, the sequence of IP pairs in the client's receive table **922** must be identical to those in the router's transmit table **923**. This is required for the session synchronization to be maintained. The client **901** need maintain only one transmit table **921** and one receive table **922** during the course of the secure session. Each

sequential packet sent by the client 901 will employ the next {send, receive} IP address pair in the transmit table, regardless of TCP or UDP session. The TARP router 911 will expect each packet arriving from the client 901 to bear the next IP address pair shown in its receive table.

Since packets can arrive out of order, however, the router 911 can maintain a “look ahead” buffer in its receive table, and will mark previously-received IP pairs as invalid for future packets; any future packet containing an IP pair that is in the look-ahead buffer but is marked as previously received will be discarded. Communications from the TARP router 911 to the client 901 are maintained in an identical manner; in particular, the router 911 will select the next IP address pair from its transmit table 923 when constructing a packet to send to the client 901, and the client 901 will maintain a look-ahead buffer of expected IP pairs on packets that it is receiving. Each TARP router will maintain separate pairs of transmit and receive tables for each client that is currently engaged in a secure session with or through that TARP router.

While clients receive their hopblocks from the first server linking them to the Internet, routers exchange hopblocks. When a router establishes a link-based IP-hopping communication regime with another router, each router of the pair exchanges its transmit hopblock. The transmit hopblock of each router becomes the receive hopblock of the other router. The communication between routers is governed as described by the example of a client sending a packet to the first router.

While the above strategy works fine in the IP milieu, many local networks that are connected to the Internet are Ethernet systems. In Ethernet, the IP addresses of the destination devices must be translated into hardware addresses, and vice versa, using known processes (“address resolution protocol,” and “reverse address resolution protocol”). However, if the link-based IP-hopping strategy is employed, the correlation process would become explosive and burdensome. An alternative to the link-based IP hopping strategy may be employed within an Ethernet network. The solution is to provide that the node linking the Internet to the Ethernet (call it the border node) use the link-based IP-hopping communication regime to communicate with nodes outside the Ethernet LAN. Within the Ethernet LAN, each TARP node would have a single IP address which would be addressed in the conventional way. Instead of comparing the {sender, receiver} IP address pairs to authenticate a packet, the intra-LAN TARP node would use one of the IP header extension fields to do so. Thus, the border node uses an algorithm shared by the intra-LAN TARP node to generate a symbol that is stored in the free field in the IP header, and the intra-LAN TARP node generates a range of symbols based on its prediction of the next expected packet to be received from that particular source IP address. The packet is rejected if it does not fall into the set of predicted symbols (for example, numerical values) or is accepted if it does. Communications from the intra-LAN TARP node to the border node are accomplished in the same manner, though the algorithm will necessarily be different for security reasons. Thus, each of the communicating nodes will generate transmit and receive tables in a similar manner to that of FIG. 9; the intra-LAN TARP nodes transmit table will be identical to the border node’s receive table, and the intra-LAN TARP node’s receive table will be identical to the border node’s transmit table.

The algorithm used for IP address-hopping can be any desired algorithm. For example, the algorithm can be a given pseudo-random number generator that generates numbers of

the range covering the allowed IP addresses with a given seed. Alternatively, the session participants can assume a certain type of algorithm and specify simply a parameter for applying the algorithm. For example the assumed algorithm could be a particular pseudo-random number generator and the session participants could simply exchange seed values.

Note that there is no permanent physical distinction between the originating and destination terminal nodes. Either device at either end point can initiate a synchronization of the pair. Note also that the authentication/synchronization-request (and acknowledgment) and hopblock-exchange may all be served by a single message so that separate message exchanges may not be required.

As another extension to the stated architecture, multiple physical paths can be used by a client, in order to provide link redundancy and further thwart attempts at denial of service and traffic monitoring. As shown in FIG. 10, for example, client 1001 can establish three simultaneous sessions with each of three TARP routers provided by different ISPs 1011, 1012, 1013. As an example, the client 1001 can use three different telephone lines 1021, 1022, 1023 to connect to the ISPs, or two telephone lines and a cable modem, etc. In this scheme, transmitted packets will be sent in a random fashion among the different physical paths. This architecture provides a high degree of communications redundancy, with improved immunity from denial-of-service attacks and traffic monitoring.

2. FURTHER EXTENSIONS

The following describes various extensions to the techniques, systems, and methods described above. As described above, the security of communications occurring between computers in a computer network (such as the Internet, an Ethernet, or others) can be enhanced by using seemingly random source and destination Internet Protocol (IP) addresses for data packets transmitted over the network. This feature prevents eavesdroppers from determining which computers in the network are communicating with each other while permitting the two communicating computers to easily recognize whether a given received data packet is legitimate or not. In one embodiment of the above-described systems, an IP header extension field is used to authenticate incoming packets on an Ethernet.

Various extensions to the previously described techniques described herein include: (1) use of hopped hardware or “MAC” addresses in broadcast type network; (2) a self-synchronization technique that permits a computer to automatically regain synchronization with a sender; (3) synchronization algorithms that allow transmitting and receiving computers to quickly re-establish synchronization in the event of lost packets or other events; and (4) a fast-packet rejection mechanism for rejecting invalid packets. Any or all of these extensions can be combined with the features described above in any of various ways.

A. Hardware Address Hopping

Internet protocol-based communications techniques on a LAN—or across any dedicated physical medium—typically embed the IP packets within lower-level packets, often referred to as “frames.” As shown in FIG. 11, for example, a first Ethernet frame 1150 comprises a frame header 1101 and two embedded IP packets IP1 and IP2, while a second Ethernet frame 1160 comprises a different frame header 1104 and a single IP packet IP3. Each frame header generally includes a source hardware address 1101A and a destination hardware address 1101B; other well-known fields in frame headers are omitted from FIG. 11 for clarity. Two

hardware nodes communicating over a physical communication channel insert appropriate source and destination hardware addresses to indicate which nodes on the channel or network should receive the frame.

It may be possible for a nefarious listener to acquire information about the contents of a frame and/or its communicants by examining frames on a local network rather than (or in addition to) the IP packets themselves. This is especially true in broadcast media, such as Ethernet, where it is necessary to insert into the frame header the hardware address of the machine that generated the frame and the hardware address of the machine to which frame is being sent. All nodes on the network can potentially “see” all packets transmitted across the network. This can be a problem for secure communications, especially in cases where the communicants do not want for any third party to be able to identify who is engaging in the information exchange. One way to address this problem is to push the address-hopping scheme down to the hardware layer. In accordance with various embodiments of the invention, hardware addresses are “hopped” in a manner similar to that used to change IP addresses, such that a listener cannot determine which hardware node generated a particular message nor which node is the intended recipient.

FIG. 12A shows a system in which Media Access Control (“MAC”) hardware addresses are “hopped” in order to increase security over a network such as an Ethernet. While the description refers to the exemplary case of an Ethernet environment, the inventive principles are equally applicable to other types of communications media. In the Ethernet case, the MAC address of the sender and receiver are inserted into the Ethernet frame and can be observed by anyone on the LAN who is within the broadcast range for that frame. For secure communications, it becomes desirable to generate frames with MAC addresses that are not attributable to any specific sender or receiver.

As shown in FIG. 12A, two computer nodes 1201 and 1202 communicate over a communication channel such as an Ethernet. Each node executes one or more application programs 1203 and 1218 that communicate by transmitting packets through communication software 1204 and 1217, respectively. Examples of application programs include video conferencing, e-mail, word processing programs, telephony, and the like. Communication software 1204 and 1217 can comprise, for example, an OSI layered architecture or “stack” that standardizes various services provided at different levels of functionality.

The lowest levels of communication software 1204 and 1217 communicate with hardware components 1206 and 1214 respectively, each of which can include one or more registers 1207 and 1215 that allow the hardware to be reconfigured or controlled in accordance with various communication protocols. The hardware components (an Ethernet network interface card, for example) communicate with each other over the communication medium. Each hardware component is typically pre-assigned a fixed hardware address or MAC number that identifies the hardware component to other nodes on the network. One or more interface drivers control the operation of each card and can, for example, be configured to accept or reject packets from certain hardware addresses. As will be described in more detail below, various embodiments of the inventive principles provide for “hopping” different addresses using one or more algorithms and one or more moving windows that track a range of valid addresses to validate received packets. Packets transmitted according to one or more of the inventive principles will be generally referred to as “secure”

packets or “secure communications” to differentiate them from ordinary data packets that are transmitted in the clear using ordinary, machine-correlated addresses.

One straightforward method of generating non-attributable MAC addresses is an extension of the IP hopping scheme. In this scenario, two machines on the same LAN that desire to communicate in a secure fashion exchange random-number generators and seeds, and create sequences of quasi-random MAC addresses for synchronized hopping. The implementation and synchronization issues are then similar to that of IP hopping.

This approach, however, runs the risk of using MAC addresses that are currently active on the LAN—which, in turn, could interrupt communications for those machines. Since an Ethernet MAC address is at present 48 bits in length, the chance of randomly misusing an active MAC address is actually quite small. However, if that figure is multiplied by a large number of nodes (as would be found on an extensive LAN), by a large number of frames (as might be the case with packet voice or streaming video), and by a large number of concurrent Virtual Private Networks (VPNs), then the chance that a non-secure machine’s MAC address could be used in an address-hopped frame can become non-trivial. In short, any scheme that runs even a small risk of interrupting communications for other machines on the LAN is bound to receive resistance from prospective system administrators. Nevertheless, it is technically feasible, and can be implemented without risk on a LAN on which there is a small number of machines, or if all of the machines on the LAN are engaging in MAC-hopped communications.

Synchronized MAC address hopping may incur some overhead in the course of session establishment, especially if there are multiple sessions or multiple nodes involved in the communications. A simpler method of randomizing MAC addresses is to allow each node to receive and process every incident frame on the network. Typically, each network interface driver will check the destination MAC address in the header of every incident frame to see if it matches that machine’s MAC address; if there is no match, then the frame is discarded. In one embodiment, however, these checks can be disabled, and every incident packet is passed to the TARP stack for processing. This will be referred to as “promiscuous” mode, since every incident frame is processed. Promiscuous mode allows the sender to use completely random, unsynchronized MAC addresses, since the destination machine is guaranteed to process the frame. The decision as to whether the packet was truly intended for that machine is handled by the TARP stack, which checks the source and destination IP addresses for a match in its IP synchronization tables. If no match is found, the packet is discarded; if there is a match, the packet is unwrapped, the inner header is evaluated, and if the inner header indicates that the packet is destined for that machine then the packet is forwarded to the IP stack—otherwise it is discarded.

One disadvantage of purely-random MAC address hopping is its impact on processing overhead; that is, since every incident frame must be processed, the machine’s CPU is engaged considerably more often than if the network interface driver is discriminating and rejecting packets unilaterally. A compromise approach is to select either a single fixed MAC address or a small number of MAC addresses (e.g., one for each virtual private network on an Ethernet) to use for MAC-hopped communications, regardless of the actual recipient for which the message is intended. In this mode, the network interface driver can check each incident



frame against one (or a few) pre-established MAC addresses, thereby freeing the CPU from the task of physical-layer packet discrimination. This scheme does not betray any useful information to an interloper on the LAN; in particular, every secure packet can already be identified by a unique packet type in the outer header. However, since all machines engaged in secure communications would either be using the same MAC address, or be selecting from a small pool of predetermined MAC addresses, the association between a specific machine and a specific MAC address is effectively broken.

In this scheme, the CPU will be engaged more often than it would be in non-secure communications (or in synchronized MAC address hopping), since the network interface driver cannot always unilaterally discriminate between secure packets that are destined for that machine, and secure packets from other VPNs. However, the non-secure traffic is easily eliminated at the network interface, thereby reducing the amount of processing required of the CPU. There are boundary conditions where these statements would not hold, of course-e.g., if all of the traffic on the LAN is secure traffic, then the CPU would be engaged to the same degree as it is in the purely-random address hopping case; alternatively, if each VPN on the LAN uses a different MAC address, then the network interface can perfectly discriminate secure frames destined for the local machine from those constituting other VPNs. These are engineering tradeoffs that might be best handled by providing administrative options for the users when installing the software and/or establishing VPNs.

Even in this scenario, however, there still remains a slight risk of selecting MAC addresses that are being used by one or more nodes on the LAN. One solution to this problem is to formally assign one address or a range of addresses for use in MAC-hopped communications. This is typically done via an assigned numbers registration authority; e.g., in the case of Ethernet, MAC address ranges are assigned to vendors by the Institute of Electrical and Electronics Engineers (IEEE). A formally-assigned range of addresses would ensure that secure frames do not conflict with any properly-configured and properly-functioning machines on the LAN.

Reference will now be made to FIGS. 12A and 12B in order to describe the many combinations and features that follow the inventive principles. As explained above, two computer nodes 1201 and 1202 are assumed to be communicating over a network or communication medium such as an Ethernet. A communication protocol in each node (1204 and 1217, respectively) contains a modified element 1205 and 1216 that performs certain functions that deviate from the standard communication protocols. In particular, computer node 1201 implements a first "hop" algorithm 1208X that selects seemingly random source and destination IP addresses (and, in one embodiment, seemingly random IP header discriminator fields) in order to transmit each packet to the other computer node. For example, node 1201 maintains a transmit table 1208 containing triplets of source (S), destination (D), and discriminator fields (DS) that are inserted into outgoing IP packet headers. The table is generated through the use of an appropriate algorithm (e.g., a random number generator that is seeded with an appropriate seed) that is known to the recipient node 1202. As each new IP packet is formed, the next sequential entry out of the sender's transmit table 1208 is used to populate the IP source, IP destination, and IP header extension field (e.g., discriminator field). It will be appreciated that the transmit table need not be created in advance but could instead be created on-the-fly by executing the algorithm when each packet is formed.

At the receiving node 1202, the same IP hop algorithm 1222X is maintained and used to generate a receive table 1222 that lists valid triplets of source IP address, destination IP address, and discriminator field. This is shown by virtue of the first five entries of transmit table 1208 matching the second five entries of receive table 1222. (The tables may be slightly offset at any particular time due to lost packets, misordered packets, or transmission delays). Additionally, node 1202 maintains a receive window W3 that represents a list of valid IP source, IP destination, and discriminator fields that will be accepted when received as part of an incoming IP packet. As packets are received, window W3 slides down the list of valid entries, such that the possible valid entries change over time. Two packets that arrive out of order but are nevertheless matched to entries within window W3 will be accepted; those falling outside of window W3 will be rejected as invalid. The length of window W3 can be adjusted as necessary to reflect network delays or other factors.

Node 1202 maintains a similar transmit table 1221 for creating IP packets and frames destined for node 1201 using a potentially different hopping algorithm 1221X, and node 1201 maintains a matching receive table 1209 using the same algorithm 1209X. As node 1202 transmits packets to node 1201 using seemingly random IP source, IP destination, and/or discriminator fields, node 1201 matches the incoming packet values to those falling within window W1 maintained in its receive table. In effect, transmit table 1208 of node 1201 is synchronized (i.e., entries are selected in the same order) to receive table 1222 of receiving node 1202. Similarly, transmit table 1221 of node 1202 is synchronized to receive table 1209 of node 1201. It will be appreciated that although a common algorithm is shown for the source, destination and discriminator fields in FIG. 12A (using, e.g., a different seed for each of the three fields), an entirely different algorithm could in fact be used to establish values for each of these fields. It will also be appreciated that one or two of the fields can be "hopped" rather than all three as illustrated.

In accordance with another aspect of the invention, hardware or "MAC" addresses are hopped instead of or in addition to IP addresses and/or the discriminator field in order to improve security in a local area or broadcast-type network. To that end, node 1201 further maintains a transmit table 1210 using a transmit algorithm 1210X to generate source and destination hardware addresses that are inserted into frame headers (e.g., fields 1101A and 1101B in FIG. 11) that are synchronized to a corresponding receive table 1224 at node 1202. Similarly, node 1202 maintains a different transmit table 1223 containing source and destination hardware addresses that is synchronized with a corresponding receive table 1211 at node 1201. In this manner, outgoing hardware frames appear to be originating from and going to completely random nodes on the network, even though each recipient can determine whether a given packet is intended for it or not. It will be appreciated that the hardware hopping feature can be implemented at a different level in the communications protocol than the IP hopping feature (e.g., in a card driver or in a hardware card itself to improve performance).

FIG. 12B shows three different embodiments or modes that can be employed using the aforementioned principles. In a first mode referred to as "promiscuous" mode, a common hardware address (e.g., a fixed address for source and another for destination) or else a completely random hardware address is used by all nodes on the network, such that a particular packet cannot be attributed to any one node.



Each node must initially accept all packets containing the common (or random) hardware address and inspect the IP addresses or discriminator field to determine whether the packet is intended for that node. In this regard, either the IP addresses or the discriminator field or both can be varied in accordance with an algorithm as described above. As explained previously, this may increase each node's overhead since additional processing is involved to determine whether a given packet has valid source and destination hardware addresses.

In a second mode referred to as "promiscuous per VPN" mode, a small set of fixed hardware addresses are used, with a fixed source/destination hardware address used for all nodes communicating over a virtual private network. For example, if there are six nodes on an Ethernet, and the network is to be split up into two private virtual networks such that nodes on one VPN can communicate with only the other two nodes on its own VPN, then two sets of hardware addresses could be used: one set for the first VPN and a second set for the second VPN. This would reduce the amount of overhead involved in checking for valid frames since only packets arriving from the designated VPN would need to be checked. IP addresses and one or more discriminator fields could still be hopped as before for secure communication within the VPN. Of course, this solution compromises the anonymity of the VPNs (i.e., an outsider can easily tell what traffic belongs in which VPN, though he cannot correlate it to a specific machine/person). It also requires the use of a discriminator field to mitigate the vulnerability to certain types of DoS attacks. (For example, without the discriminator field, an attacker on the LAN could stream frames containing the MAC addresses being used by the VPN; rejecting those frames could lead to excessive processing overhead. The discriminator field would provide a low-overhead means of rejecting the false packets.)

In a third mode referred to as "hardware hopping" mode, hardware addresses are varied as illustrated in FIG. 12A, such that hardware source and destination addresses are changed constantly in order to provide non-attributable addressing. Variations on these embodiments are of course possible, and the invention is not intended to be limited in any respect by these illustrative examples.

#### B. Extending the Address Space

Address hopping provides security and privacy. However, the level of protection is limited by the number of addresses in the blocks being hopped. A hopblock denotes a field or fields modulated on a packet-wise basis for the purpose of providing a VPN. For instance, if two nodes communicate with IP address hopping using hopblocks of 4 addresses (2 bits) each, there would be 16 possible address-pair combinations. A window of size 16 would result in most address pairs being accepted as valid most of the time. This limitation can be overcome by using a discriminator field in addition to or instead of the hopped address fields. The discriminator field would be hopped in exactly the same fashion as the address fields and it would be used to determine whether a packet should be processed by a receiver.

Suppose that two clients, each using four-bit hopblocks, would like the same level of protection afforded to clients communicating via IP hopping between two A blocks (24 address bits eligible for hopping). A discriminator field of 20 bits, used in conjunction with the 4 address bits eligible for hopping in the IP address field, provides this level of protection. A 24-bit discriminator field would provide a similar level of protection if the address fields were not

hopped or ignored. Using a discriminator field offers the following advantages: (1) an arbitrarily high level of protection can be provided, and (2) address hopping is unnecessary to provide protection. This may be important in environments where address hopping would cause routing problems.

#### C. Synchronization Techniques

It is generally assumed that once a sending node and receiving node have exchanged algorithms and seeds (or similar information sufficient to generate quasi-random source and destination tables), subsequent communication between the two nodes will proceed smoothly. Realistically, however, two nodes may lose synchronization due to network delays or outages, or other problems. Consequently, it is desirable to provide means for re-establishing synchronization between nodes in a network that have lost synchronization.

One possible technique is to require that each node provide an acknowledgment upon successful receipt of each packet and, if no acknowledgment is received within a certain period of time, to re-send the unacknowledged packet. This approach, however, drives up overhead costs and may be prohibitive in high-throughput environments such as streaming video or audio, for example.

A different approach is to employ an automatic synchronizing technique that will be referred to herein as "self-synchronization." In this approach, synchronization information is embedded into each packet, thereby enabling the receiver to re-synchronize itself upon receipt of a single packet if it determines that it has lost synchronization with the sender. (If communications are already in progress, and the receiver determines that it is still in sync with the sender, then there is no need to re-synchronize.) A receiver could detect that it was out of synchronization by, for example, employing a "dead-man" timer that expires after a certain period of time, wherein the timer is reset with each valid packet. A time stamp could be hashed into the public sync field (see below) to preclude packet-retry attacks.

In one embodiment, a "sync field" is added to the header of each packet sent out by the sender. This sync field could appear in the clear or as part of an encrypted portion of the packet. Assuming that a sender and receiver have selected a random-number generator (RNG) and seed value, this combination of RNG and seed can be used to generate a random-number sequence (RNS). The RNS is then used to generate a sequence of source/destination IP pairs (and, if desired, discriminator fields and hardware source and destination addresses), as described above. It is not necessary, however, to generate the entire sequence (or the first N-1 values) in order to generate the Nth random number in the sequence; if the sequence index N is known, the random value corresponding to that index can be directly generated (see below). Different RNGs (and seeds) with different fundamental periods could be used to generate the source and destination IP sequences, but the basic concepts would still apply. For the sake of simplicity, the following discussion will assume that IP source and destination address pairs (only) are hopped using a single RNG sequencing mechanism.

In accordance with a "self-synchronization" feature, a sync field in each packet header provides an index (i.e., a sequence number) into the RNS that is being used to generate IP pairs. Plugging this index into the RNG that is being used to generate the RNS yields a specific random number value, which in turn yields a specific IP pair. That is, an IP pair can be generated directly from knowledge of the RNG, seed, and index number; it is not necessary, in this

scheme, to generate the entire sequence of random numbers that precede the sequence value associated with the index number provided.

Since the communicants have presumably previously exchanged RNGs and seeds, the only new information that must be provided in order to generate an IP pair is the sequence number. If this number is provided by the sender in the packet header, then the receiver need only plug this number into the RNG in order to generate an IP pair—and thus verify that the IP pair appearing in the header of the packet is valid. In this scheme, if the sender and receiver lose synchronization, the receiver can immediately re-synchronize upon receipt of a single packet by simply comparing the IP pair in the packet header to the IP pair generated from the index number. Thus, synchronized communications can be resumed upon receipt of a single packet, making this scheme ideal for multicast communications. Taken to the extreme, it could obviate the need for synchronization tables entirely; that is, the sender and receiver could simply rely on the index number in the sync field to validate the IP pair on each packet, and thereby eliminate the tables entirely.

The aforementioned scheme may have some inherent security issues associated with it—namely, the placement of the sync field. If the field is placed in the outer header, then an interloper could observe the values of the field and their relationship to the IP stream. This could potentially compromise the algorithm that is being used to generate the IP-address sequence, which would compromise the security of the communications. If, however, the value is placed in the inner header, then the sender must decrypt the inner header before it can extract the sync value and validate the IP pair; this opens up the receiver to certain types of denial-of-service (DoS) attacks, such as packet replay. That is, if the receiver must decrypt a packet before it can validate the IP pair, then it could potentially be forced to expend a significant amount of processing on decryption if an attacker simply retransmits previously valid packets. Other attack methodologies are possible in this scenario.

A possible compromise between algorithm security and processing speed is to split up the sync value between an inner (encrypted) and outer (unencrypted) header. That is, if the sync value is sufficiently long, it could potentially be split into a rapidly-changing part that can be viewed in the clear, and a fixed (or very slowly changing) part that must be protected. The part that can be viewed in the clear will be called the “public sync” portion and the part that must be protected will be called the “private sync” portion.

Both the public sync and private sync portions are needed to generate the complete sync value. The private portion, however, can be selected such that it is fixed or will change only occasionally. Thus, the private sync value can be stored by the recipient, thereby obviating the need to decrypt the header in order to retrieve it. If the sender and receiver have previously agreed upon the frequency with which the private part of the sync will change, then the receiver can selectively decrypt a single header in order to extract the new private sync if the communications gap that has led to lost synchronization has exceeded the lifetime of the previous private sync. This should not represent a burdensome amount of decryption, and thus should not open up the receiver to denial-of-service attack simply based on the need to occasionally decrypt a single header.

One implementation of this is to use a hashing function with a one-to-one mapping to generate the private and public sync portions from the sync value. This implementation is shown in FIG. 13, where (for example) a first ISP 1302 is the

sender and a second ISP 1303 is the receiver. (Other alternatives are possible from FIG. 13.) A transmitted packet comprises a public or “outer” header 1305 that is not encrypted, and a private or “inner” header 1306 that is encrypted using for example a link key. Outer header 1305 includes a public sync portion while inner header 1306 contains the private sync portion. A receiving node decrypts the inner header using a decryption function 1307 in order to extract the private sync portion. This step is necessary only if the lifetime of the currently buffered private sync has expired. (If the currently-buffered private sync is still valid, then it is simply extracted from memory and “added” (which could be an inverse hash) to the public sync, as shown in step 1308.) The public and decrypted private sync portions are combined in function 1308 in order to generate the combined sync 1309. The combined sync (1309) is then fed into the RNG (1310) and compared to the IP address pair (1311) to validate or reject the packet.

An important consideration in this architecture is the concept of “future” and “past” where the public sync values are concerned. Though the sync values, themselves, should be random to prevent spoofing attacks, it may be important that the receiver be able to quickly identify a sync value that has already been sent—even if the packet containing that sync value was never actually received by the receiver. One solution is to hash a time stamp or sequence number into the public sync portion, which could be quickly extracted, checked, and discarded, thereby validating the public sync portion itself.

In one embodiment, packets can be checked by comparing the source/destination IP pair generated by the sync field with the pair appearing in the packet header. If (1) they match, (2) the time stamp is valid, and (3) the dead-man timer has expired, then re-synchronization occurs; otherwise, the packet is rejected. If enough processing power is available, the dead-man timer and synchronization tables can be avoided altogether, and the receiver would simply resynchronize (e.g., validate) on every packet.

The foregoing scheme may require large-integer (e.g., 160-bit) math, which may affect its implementation. Without such large-integer registers, processing throughput would be affected, thus potentially affecting security from a denial-of-service standpoint. Nevertheless, as large-integer math processing features become more prevalent, the costs of implementing such a feature will be reduced.

D. Other Synchronization Schemes

As explained above, if W or more consecutive packets are lost between a transmitter and receiver in a VPN (where W is the window size), the receiver’s window will not have been updated and the transmitter will be transmitting packets not in the receiver’s window. The sender and receiver will not recover synchronization until perhaps the random pairs in the window are repeated by chance. Therefore, there is a need to keep a transmitter and receiver in synchronization whenever possible and to re-establish synchronization whenever it is lost.

A “checkpoint” scheme can be used to regain synchronization between a sender and a receiver that have fallen out of synchronization. In this scheme, a checkpoint message comprising a random IP address pair is used for communicating synchronization information. In one embodiment, two messages are used to communicate synchronization information between a sender and a recipient:

- 1. SYNC\_REQ is a message used by the sender to indicate that it wants to synchronize; and
- 2. SYNC\_ACK is a message used by the receiver to inform the transmitter that it has been synchronized.

According to one variation of this approach, both the transmitter and receiver maintain three checkpoints (see FIG. 14):

1. In the transmitter, ckpt\_o (“checkpoint old”) is the IP pair that was used to re-send the last SYNC\_REQ packet to the receiver. In the receiver, ckpt\_o (“checkpoint old”) is the IP pair that receives repeated SYNC\_REQ packets from the transmitter.
2. In the transmitter, ckpt\_n (“checkpoint new”) is the IP pair that will be used to send the next SYNC\_REQ packet to the receiver. In the receiver, ckpt\_n (“checkpoint new”) is the IP pair that receives a new SYNC\_REQ packet from the transmitter and which causes the receiver’s window to be re-aligned, ckpt\_o set to ckpt\_n, a new ckpt\_n to be generated and a new ckpt\_r to be generated.
3. In the transmitter, ckpt\_r is the IP pair that will be used to send the next SYNC\_ACK packet to the receiver. In the receiver, ckpt\_r is the IP pair that receives a new SYNC\_ACK packet from the transmitter and which causes a new ckpt\_n to be generated. Since SYNC\_ACK is transmitted from the receiver ISP to the sender ISP, the transmitter ckpt\_r refers to the ckpt\_r of the receiver and the receiver ckpt\_r refers to the ckpt\_r of the transmitter (see FIG. 14).

When a transmitter initiates synchronization, the IP pair it will use to transmit the next data packet is set to a predetermined value and when a receiver first receives a SYNC\_REQ, the receiver window is updated to be centered on the transmitter’s next IP pair. This is the primary mechanism for checkpoint synchronization.

Synchronization can be initiated by a packet counter (e.g., after every N packets transmitted, initiate a synchronization) or by a timer (every S seconds, initiate a synchronization) or a combination of both. See FIG. 15. From the transmitter’s perspective, this technique operates as follows: (1) Each transmitter periodically transmits a “sync request” message to the receiver to make sure that it is in sync. (2) If the receiver is still in sync, it sends back a “sync ack” message. (If this works, no further action is necessary). (3) If no “sync ack” has been received within a period of time, the transmitter retransmits the sync request again. If the transmitter reaches the next checkpoint without receiving a “sync ack” response, then synchronization is broken, and the transmitter should stop transmitting. The transmitter will continue to send sync\_reqs until it receives a sync\_ack, at which point transmission is reestablished.

From the receiver’s perspective, the scheme operates as follows: (1) when it receives a “sync request” request from the transmitter, it advances its window to the next checkpoint position (even skipping pairs if necessary), and sends a “sync ack” message to the transmitter. If sync was never lost, then the “jump ahead” really just advances to the next available pair of addresses in the table (i.e., normal advancement).

If an interloper intercepts the “sync request” messages and tries to interfere with communication by sending new ones, it will be ignored if the synchronization has been established or it will actually help to re-establish synchronization.

A window is realigned whenever a re-synchronization occurs. This realignment entails updating the receiver’s window to straddle the address pairs used by the packet transmitted immediately after the transmission of the SYNC\_REQ packet. Normally, the transmitter and receiver are in synchronization with one another. However, when network events occur, the receiver’s window may have to be advanced by many steps during resynchronization. In this

case, it is desirable to move the window ahead without having to step through the intervening random numbers sequentially. (This feature is also desirable for the auto-sync approach discussed above).

E. Random Number Generator with a Jump-Ahead Capability

An attractive method for generating randomly hopped addresses is to use identical random number generators in the transmitter and receiver and advance them as packets are transmitted and received. There are many random number generation algorithms that could be used. Each one has strengths and weaknesses for address hopping applications.

Linear congruential random number generators (LCRs) are fast, simple and well characterized random number generators that can be made to jump ahead n steps efficiently. An LCR generates random numbers  $X_1, X_2, X_3 \dots X_k$  starting with seed  $X_0$  using a recurrence

$$X_i = (aX_{i-1} + b) \text{ mod } c, \tag{1}$$

where a, b and c define a particular LCR. Another expression for  $X_i$ ,

$$X_i = ((a^i(X_0 + b) - b) / (a - 1)) \text{ mod } c \tag{2}$$

enables the jump-ahead capability. The factor  $a^i$  can grow very large even for modest i if left unfettered. Therefore some special properties of the modulo operation can be used to control the size and processing time required to compute (2). (2) can be rewritten as:

$$X_i = (a^i(X_0(a-1) + b) - b) / (a-1) \text{ mod } c. \tag{3}$$

It can be shown that:

$$\frac{(a^i(X_0(a-1) + b) - b) / (a-1) \text{ mod } c}{(a-1) \text{ mod } c} = ((a^i \text{ mod } ((a-1)c)(X_0(a-1) + b) - b) / (a-1) \text{ mod } c) \tag{4}$$

$(X_0(a-1) + b)$  can be stored as  $(X_0(a-1) + b) \text{ mod } c$ , b as  $b \text{ mod } c$  and compute  $a^i \text{ mod } ((a-1)c)$  (this requires  $O(\log(i))$  steps).

A practical implementation of this algorithm would jump a fixed distance, n, between synchronizations; this is tantamount to synchronizing every n packets. The window would commence n IP pairs from the start of the previous window. Using  $X_j$ , the random number at the  $j^{\text{th}}$  checkpoint, as  $X_0$  and n as i, a node can store  $a^n \text{ mod } ((a-1)c)$  once per LCR and set

$$X_{j+1} = X_n = ((a^n \text{ mod } ((a-1)c)(X_j^n(a-1) + b) - b) / (a-1) \text{ mod } c, \tag{5}$$

to generate the random number for the  $j+1^{\text{th}}$  synchronization. Using this construction, a node could jump ahead an arbitrary (but fixed) distance between synchronizations in a constant amount of time (independent of n).

Pseudo-random number generators, in general, and LCRs, in particular, will eventually repeat their cycles. This repetition may present vulnerability in the IP hopping scheme.

An adversary would simply have to wait for a repeat to predict future sequences. One way of coping with this vulnerability is to create a random number generator with a known long cycle. A random sequence can be replaced by a new random number generator before it repeats. LCRs can be constructed with known long cycles. This is not currently true of many random number generators.

Random number generators can be cryptographically insecure. An adversary can derive the RNG parameters by examining the output or part of the output. This is true of LCGs. This vulnerability can be mitigated by incorporating an encryptor, designed to scramble the output as part of the random number generator. The random number generator

prevents an adversary from mounting an attack—e.g., a known plaintext attack—against the encryptor.

F. Random Number Generator Example

Consider a RNG where  $a=31$ ,  $b=4$  and  $c=15$ . For this case equation (1) becomes:

$$X_i=(31X_{i-1}+4)\bmod 15. \tag{6}$$

If one sets  $X_0=1$ , equation (6) will produce the sequence 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 0, 4, 8, 12. This sequence will repeat indefinitely. For a jump ahead of 3 numbers in this sequence  $a''=31^3=29791$ ,  $c*(a-1)=15*30=450$  and  $a''\bmod((a-1)c)=31^3\bmod(15*30)=29791\bmod(450)=91$ . Equation (5) becomes:

$$((91(X_i30+4)-4)/30)\bmod 15 \tag{7}$$

Table 1 shows the jump ahead calculations from (7). The calculations start at 5 and jump ahead 3.

TABLE 1

I	$X_i$	$(X_i30 + 4)$	$91(X_i30 + 4) - 4$	$((91(X_i30 + 4) - 4)/30)$	$X_{i+3}$
1	5	154	14010	467	2
4	2	64	5820	194	14
7	14	424	38580	1286	11
10	11	334	30390	1013	8
13	8	244	22200	740	5

G. Fast Packet Filter

Address hopping VPNs must rapidly determine whether a packet has a valid header and thus requires further processing, or has an invalid header (a hostile packet) and should be immediately rejected. Such rapid determinations will be referred to as “fast packet filtering.” This capability protects the VPN from attacks by an adversary who streams hostile packets at the receiver at a high rate of speed in the hope of saturating the receiver’s processor (a so-called “denial of service” attack). Fast packet filtering is an important feature for implementing VPNs on shared media such as Ethernet.

Assuming that all participants in a VPN share an unassigned “A” block of addresses, one possibility is to use an experimental “A” block that will never be assigned to any machine that is not address hopping on the shared medium. “A” blocks have a 24 bits of address that can be hopped as opposed to the 8 bits in “C” blocks. In this case a hopblock will be the “A” block. The use of the experimental “A” block is a likely option on an Ethernet because:

1. The addresses have no validity outside of the Ethernet and will not be routed out to a valid outside destination by a gateway.
2. There are  $2^{24}$  (~16 million) addresses that can be hopped within each “A” block. This yields >280 trillion possible address pairs making it very unlikely that an adversary would guess a valid address. It also provides acceptably low probability of collision between separate VPNs (all VPNs on a shared medium independently generate random address pairs from the same “A” block).
3. The packets will not be received by someone on the Ethernet who is not on a VPN (unless the machine is in promiscuous mode) minimizing impact on non-VPN computers.

The Ethernet example will be used to describe one implementation of fast packet filtering. The ideal algorithm would quickly examine a packet header, determine whether

the packet is hostile, and reject any hostile packets or determine which active IP pair the packet header matches. The problem is a classical associative memory problem. A variety of techniques have been developed to solve this problem (hashing, B-trees etc). Each of these approaches has its strengths and weaknesses. For instance, hash tables can be made to operate quite fast in a statistical sense, but can occasionally degenerate into a much slower algorithm. This slowness can persist for a period of time. Since there is a need to discard hostile packets quickly at all times, hashing would be unacceptable.

H. Presence Vector Algorithm

A presence vector is a bit vector of length  $2^n$  that can be indexed by n-bit numbers (each ranging from 0 to  $2^n-1$ ). One can indicate the presence of k n-bit numbers (not necessarily unique), by setting the bits in the presence vector indexed by each number to 1. Otherwise, the bits in the presence vector are 0. An n-bit number, x, is one of the k numbers if and only if the  $x^{th}$  bit of the presence vector is 1. A fast packet filter can be implemented by indexing the presence vector and looking for a 1, which will be referred to as the “test.”

For example, suppose one wanted to represent the number 135 using a presence vector. The 135<sup>th</sup> bit of the vector would be set. Consequently, one could very quickly determine whether an address of 135 was valid by checking only one bit: the 135<sup>th</sup> bit. The presence vectors could be created in advance corresponding to the table entries for the IP addresses. In effect, the incoming addresses can be used as indices into a long vector, making comparisons very fast. As each RNG generates a new address, the presence vector is updated to reflect the information. As the window moves, the presence vector is updated to zero out addresses that are no longer valid.

There is a trade-off between efficiency of the test and the amount of memory required for storing the presence vector (s). For instance, if one were to use the 48 bits of hopping addresses as an index, the presence vector would have to be 35 terabytes. Clearly, this is too large for practical purposes. Instead, the 48 bits can be divided into several smaller fields. For instance, one could subdivide the 48 bits into four 12-bit fields (see FIG. 16). This reduces the storage requirement to 2048 bytes at the expense of occasionally having to process a hostile packet. In effect, instead of one long presence vector, the decomposed address portions must match all four shorter presence vectors before further processing is allowed. (If the first part of the address portion doesn’t match the first presence vector, there is no need to check the remaining three presence vectors).

A presence vector will have a 1 in the  $y^{th}$  bit if and only if one or more addresses with a corresponding field of y are active. An address is active only if each presence vector indexed by the appropriate sub-field of the address is 1.

Consider a window of 32 active addresses and 3 checkpoints. A hostile packet will be rejected by the indexing of one presence vector more than 99% of the time. A hostile packet will be rejected by the indexing of all 4 presence vectors more than 99.999995% of the time. On average, hostile packets will be rejected in less than 1.02 presence vector index operations.

The small percentage of hostile packets that pass the fast packet filter will be rejected when matching pairs are not found in the active window or are active checkpoints. Hostile packets that serendipitously match a header will be rejected when the VPN software attempts to decrypt the header. However, these cases will be extremely rare. There are many other ways this method can be configured to arbitrate the space/speed tradeoffs.

I. Further Synchronization Enhancements

A slightly modified form of the synchronization techniques described above can be employed. The basic principles of the previously described checkpoint synchronization scheme remain unchanged. The actions resulting from the reception of the checkpoints are, however, slightly different. In this variation, the receiver will maintain between OoO (“Out of Order”) and  $2 \times \text{WINDOW\_SIZE} + \text{OoO}$  active addresses ( $1 \leq \text{OoO} \leq \text{WINDOW\_SIZE}$  and  $\text{WINDOW\_SIZE} \geq 1$ ). OoO and WINDOW\_SIZE are engineerable parameters, where OoO is the minimum number of addresses needed to accommodate lost packets due to events in the network or out of order arrivals and WINDOW\_SIZE is the number of packets transmitted before a SYNC\_REQ is issued. FIG. 17 depicts a storage array for a receiver’s active addresses.

The receiver starts with the first  $2 \times \text{WINDOW\_SIZE}$  addresses loaded and active (ready to receive data). As packets are received, the corresponding entries are marked as “used” and are no longer eligible to receive packets. The transmitter maintains a packet counter, initially set to 0, containing the number of data packets transmitted since the last initial transmission of a SYNC\_REQ for which SYNC\_ACK has been received. When the transmitter packet counter equals WINDOW\_SIZE, the transmitter generates a SYNC\_REQ and does its initial transmission. When the receiver receives a SYNC\_REQ corresponding to its current CKPT\_N, it generates the next WINDOW\_SIZE addresses and starts loading them in order starting at the first location after the last active address wrapping around to the beginning of the array after the end of the array has been reached. The receiver’s array might look like FIG. 18 when a SYNC\_REQ has been received. In this case a couple of packets have been either lost or will be received out of order when the SYNC\_REQ is received.

FIG. 19 shows the receiver’s array after the new addresses have been generated. If the transmitter does not receive a SYNC\_ACK, it will re-issue the SYNC\_REQ at regular intervals. When the transmitter receives a SYNC\_ACK, the packet counter is decremented by WINDOW\_SIZE. If the packet counter reaches  $2 \times \text{WINDOW\_SIZE} - \text{OoO}$  then the transmitter ceases sending data packets until the appropriate SYNC\_ACK is finally received. The transmitter then resumes sending data packets. Future behavior is essentially a repetition of this initial cycle. The advantages of this approach are:

1. There is no need for an efficient jump ahead in the random number generator,
2. No packet is ever transmitted that does not have a corresponding entry in the receiver side
3. No timer based re-synchronization is necessary. This is a consequence of 2.
4. The receiver will always have the ability to accept data messages transmitted within OoO messages of the most recently transmitted message.

J. Distributed Transmission Path Variant

Another embodiment incorporating various inventive principles is shown in FIG. 20. In this embodiment, a message transmission system includes a first computer 2001 in communication with a second computer 2002 through a network 2011 of intermediary computers. In one variant of this embodiment, the network includes two edge routers 2003 and 2004 each of which is linked to a plurality of Internet Service Providers (ISPs) 2005 through 2010. Each ISP is coupled to a plurality of other ISPs in an arrangement as shown in FIG. 20, which is a representative configuration only and is not intended to be limiting. Each connection

between ISPs is labeled in FIG. 20 to indicate a specific physical transmission path (e.g., AD is a physical path that links ISP A (element 2005) to ISP D (element 2008)). Packets arriving at each edge router are selectively transmitted to one of the ISPs to which the router is attached on the basis of a randomly or quasi-randomly selected basis.

As shown in FIG. 21, computer 2001 or edge router 2003 incorporates a plurality of link transmission tables 2100 that identify, for each potential transmission path through the network, valid sets of IP addresses that can be used to transmit the packet. For example, AD table 2101 contains a plurality of IP source/destination pairs that are randomly or quasi-randomly generated. When a packet is to be transmitted from first computer 2001 to second computer 2002, one of the link tables is randomly (or quasi-randomly) selected, and the next valid source/destination address pair from that table is used to transmit the packet through the network. If path AD is randomly selected, for example, the next source/destination IP address pair (which is pre-determined to transmit between ISP A (element 2005) and ISP B (element 2008)) is used to transmit the packet. If one of the transmission paths becomes degraded or inoperative, that link table can be set to a “down” condition as shown in table 2105, thus preventing addresses from being selected from that table. Other transmission paths would be unaffected by this broken link.

3. CONTINUATION-IN-PART IMPROVEMENTS

The following describes various improvements and features that can be applied to the embodiments described above. The improvements include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities. Each is discussed separately below.

A. Load Balancer

Various embodiments described above include a system in which a transmitting node and a receiving node are coupled through a plurality of transmission paths, and wherein successive packets are distributed quasi-randomly over the plurality of paths. See, for example, FIGS. 20 and 21 and accompanying description. The improvement extends this basic concept to encompass distributing packets across different paths in such a manner that the loads on the paths are generally balanced according to transmission link quality.

In one embodiment, a system includes a transmitting node and a receiving node that are linked via a plurality of transmission paths having potentially varying transmission quality. Successive packets are transmitted over the paths based on a weight value distribution function for each path. The rate that packets will be transmitted over a given path can be different for each path. The relative “health” of each transmission path is monitored in order to identify paths that have become degraded. In one embodiment, the health of each path is monitored in the transmitter by comparing the number of packets transmitted to the number of packet acknowledgements received. Each transmission path may comprise a physically separate path (e.g., via dial-up phone

line, computer network, router, bridge, or the like), or may comprise logically separate paths contained within a broad-band communication medium (e.g., separate channels in an FDM, TDM, CDMA, or other type of modulated or unmodulated transmission link).

When the transmission quality of a path falls below a predetermined threshold and there are other paths that can transmit packets, the transmitter changes the weight value used for that path, making it less likely that a given packet will be transmitted over that path. The weight will preferably be set no lower than a minimum value that keeps nominal traffic on the path. The weights of the other available paths are altered to compensate for the change in the affected path. When the quality of a path degrades to where the transmitter is turned off by the synchronization function (i.e., no packets are arriving at the destination), the weight is set to zero. If all transmitters are turned off, no packets are sent.

Conventional TCP/IP protocols include a “throttling” feature that reduces the transmission rate of packets when it is determined that delays or errors are occurring in transmission. In this respect, timers are sometimes used to determine whether packets have been received. These conventional techniques for limiting transmission of packets, however, do not involve multiple transmission paths between two nodes wherein transmission across a particular path relative to the others is changed based on link quality.

According to certain embodiments, in order to damp oscillations that might otherwise occur if weight distributions are changed drastically (e.g., according to a step function), a linear or an exponential decay formula can be applied to gradually decrease the weight value over time that a degrading path will be used. Similarly, if the health of a degraded path improves, the weight value for that path is gradually increased.

Transmission link health can be evaluated by comparing the number of packets that are acknowledged within the transmission window (see embodiments discussed above) to the number of packets transmitted within that window and by the state of the transmitter (i.e., on or off). In other words, rather than accumulating general transmission statistics over time for a path, one specific implementation uses the “windowing” concepts described above to evaluate transmission path health.

The same scheme can be used to shift virtual circuit paths from an “unhealthy” path to a “healthy” one, and to select a path for a new virtual circuit.

FIG. 22A shows a flowchart for adjusting weight values associated with a plurality of transmission links. It is assumed that software executing in one or more computer nodes executes the steps shown in FIG. 22A. It is also assumed that the software can be stored on a computer-readable medium such as a magnetic or optical disk for execution by a computer.

Beginning in step 2201, the transmission quality of a given transmission path is measured. As described above, this measurement can be based on a comparison between the number of packets transmitted over a particular link to the number of packet acknowledgements received over the link (e.g., per unit time, or in absolute terms). Alternatively, the quality can be evaluated by comparing the number of packets that are acknowledged within the transmission window to the number of packets that were transmitted within that window. In yet another variation, the number of missed synchronization messages can be used to indicate link quality. Many other variations are of course possible.

In step 2202, a check is made to determine whether more than one transmitter (e.g., transmission path) is turned on. If not, the process is terminated and resumes at step 2201.

In step 2203, the link quality is compared to a given threshold (e.g., 50%, or any arbitrary number). If the quality falls below the threshold, then in step 2207 a check is made to determine whether the weight is above a minimum level (e.g., 1%). If not, then in step 2209 the weight is set to the minimum level and processing resumes at step 2201. If the weight is above the minimum level, then in step 2208 the weight is gradually decreased for the path, then in step 2206 the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are increased).

If in step 2203 the quality of the path was greater than or equal to the threshold, then in step 2204 a check is made to determine whether the weight is less than a steady-state value for that path. If so, then in step 2205 the weight is increased toward the steady-state value, and in step 2206 the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are decreased). If in step 2204 the weight is not less than the steady-state value, then processing resumes at step 2201 without adjusting the weights.

The weights can be adjusted incrementally according to various functions, preferably by changing the value gradually. In one embodiment, a linearly decreasing function is used to adjust the weights; according to another embodiment, an exponential decay function is used. Gradually changing the weights helps to damp oscillators that might otherwise occur if the probabilities were abruptly.

Although not explicitly shown in FIG. 22A the process can be performed only periodically (e.g., according to a time schedule), or it can be continuously run, such as in a background mode of operation. In one embodiment, the combined weights of all potential paths should add up to unity (e.g., when the weighting for one path is decreased, the corresponding weights that the other paths will be selected will increase).

Adjustments to weight values for other paths can be prorated. For example, a decrease of 10% in weight value for one path could result in an evenly distributed increase in the weights for the remaining paths. Alternatively, weightings could be adjusted according to a weighted formula as desired (e.g., favoring healthy paths over less healthy paths). In yet another variation, the difference in weight value can be amortized over the remaining links in a manner that is proportional to their traffic weighting.

FIG. 22B shows steps that can be executed to shut down transmission links where a transmitter turns off. In step 2210, a transmitter shut-down event occurs. In step 2211, a test is made to determine whether at least one transmitter is still turned on. If not, then in step 2215 all packets are dropped until a transmitter turns on. If in step 2211 at least one transmitter is turned on, then in step 2212 the weight for the path is set to zero, and the weights for the remaining paths are adjusted accordingly.

FIG. 23 shows a computer node 2301 employing various principles of the above-described embodiments. It is assumed that two computer nodes of the type shown in FIG. 23 communicate over a plurality of separate physical transmission paths. As shown in FIG. 23, four transmission paths X1 through X4 are defined for communicating between the two nodes. Each node includes a packet transmitter 2302 that operates in accordance with a transmit table 2308 as described above. (The packet transmitter could also operate without using the IP-hopping features described above, but the following description assumes that some form of hopping is employed in conjunction with the path selection mechanism.). The computer node also includes a packet receiver 2303 that operates in accordance with a receive table 2309, including a moving window W that moves as

valid packets are received. Invalid packets having source and destination addresses that do not fall within window W are rejected.

As each packet is readied for transmission, source and destination IP addresses (or other discriminator values) are selected from transmit table 2308 according to any of the various algorithms described above, and packets containing these source/destination address pairs, which correspond to the node to which the four transmission paths are linked, are generated to a transmission path switch 2307. Switch 2307, which can comprise a software function, selects from one of the available transmission paths according to a weight distribution table 2306. For example, if the weight for path X1 is 0.2, then every fifth packet will be transmitted on path X1. A similar regime holds true for the other paths as shown. Initially, each link's weight value can be set such that it is proportional to its bandwidth, which will be referred to as its "steady-state" value.

Packet receiver 2303 generates an output to a link quality measurement function 2304 that operates as described above to determine the quality of each transmission path. (The input to packet receiver 2303 for receiving incoming packets is omitted for clarity). Link quality measurement function 2304 compares the link quality to a threshold for each transmission link and, if necessary, generates an output to weight adjustment function 2305. If a weight adjustment is required, then the weights in table 2306 are adjusted accordingly, preferably according to a gradual (e.g., linearly or exponentially declining) function. In one embodiment, the weight values for all available paths are initially set to the same value, and only when paths degrade in quality are the weights changed to reflect differences.

Link quality measurement function 2304 can be made to operate as part of a synchronizer function as described above. That is, if resynchronization occurs and the receiver detects that synchronization has been lost (e.g., resulting in the synchronization window W being advanced out of sequence), that fact can be used to drive link quality measurement function 2304. According to one embodiment, load balancing is performed using information garnered during the normal synchronization, augmented slightly to communicate link health from the receiver to the transmitter. The receiver maintains a count, MESS\_R(W), of the messages received in synchronization window W. When it receives a synchronization request (SYNC\_REQ) corresponding to the end of window W, the receiver includes counter MESS\_R in the resulting synchronization acknowledgement (SYNC\_ACK) sent back to the transmitter. This allows the transmitter to compare messages sent to messages received in order to assess the health of the link.

If synchronization is completely lost, weight adjustment function 2305 decreases the weight value on the affected path to zero. When synchronization is regained, the weight value for the affected path is gradually increased to its original value. Alternatively, link quality can be measured by evaluating the length of time required for the receiver to acknowledge a synchronization request. In one embodiment, separate transmit and receive tables are used for each transmission path.

When the transmitter receives a SYNC\_ACK, the MESS\_R is compared with the number of messages transmitted in a window (MESS\_T). When the transmitter receives a SYNC\_ACK, the traffic probabilities will be examined and adjusted if necessary. MESS\_R is compared with the number of messages transmitted in a window (MESS\_T). There are two possibilities:

1. If MESS\_R is less than a threshold value, THRESH, then the link will be deemed to be unhealthy. If the

transmitter was turned off, the transmitter is turned on and the weight P for that link will be set to a minimum value MIN. This will keep a trickle of traffic on the link for monitoring purposes until it recovers. If the transmitter was turned on, the weight P for that link will be set to:

$$P' = \alpha \times \text{MIN} + (1 - \alpha) \times P \tag{1}$$

Equation 1 will exponentially damp the traffic weight value to MIN during sustained periods of degraded service.

2. If MESS\_R for a link is greater than or equal to THRESH, the link will be deemed healthy. If the weight P for that link is greater than or equal to the steady state value S for that link, then P is left unaltered. If the weight P for that link is less than THRESH then P will be set to:

$$P' = \beta \times S + (1 - \beta) \times P \tag{2}$$

where  $\beta$  is a parameter such that  $0 \leq \beta \leq 1$  that determines the damping rate of P.

Equation 2 will increase the traffic weight to S during sustained periods of acceptable service in a damped exponential fashion.

A detailed example will now be provided with reference to FIG. 24. As shown in FIG. 24, a first computer 2401 communicates with a second computer 2402 through two routers 2403 and 2404. Each router is coupled to the other router through three transmission links. As described above, these may be physically diverse links or logical links (including virtual private networks).

Suppose that a first link L1 can sustain a transmission bandwidth of 100 Mb/s and has a window size of 32; link L2 can sustain 75 Mb/s and has a window size of 24; and link L3 can sustain 25 Mb/s and has a window size of 8. The combined links can thus sustain 200 Mb/s. The steady state traffic weights are 0.5 for link L1; 0.375 for link L2, and 0.125 for link L3. MIN=1 Mb/s, THRESH=0.8 MESS\_T for each link,  $\alpha=0.75$  and  $\beta=0.5$ . These traffic weights will remain stable until a link stops for synchronization or reports a number of packets received less than its THRESH. Consider the following sequence of events:

1. Link L1 receives a SYNC\_ACK containing a MESS\_R of 24, indicating that only 75% of the MESS\_T (32) messages transmitted in the last window were successfully received. Link L1 would be below THRESH (0.8). Consequently, link L1's traffic weight value would be reduced to 0.12825, while link L2's traffic weight value would be increased to 0.65812 and link L3's traffic weight value would be increased to 0.217938.
2. Link L2 and L3 remained healthy and link L1 stopped to synchronize. Then link L1's traffic weight value would be set to 0, link L2's traffic weight value would be set to 0.75, and link L3's traffic weight value would be set to 0.25.
3. Link L1 finally received a SYNC\_ACK containing a MESS\_R of 0 indicating that none of the MESS\_T (32) messages transmitted in the last window were successfully received. Link L1 would be below THRESH. Link L1's traffic weight value would be increased to 0.005, link L2's traffic weight value would be decreased to 0.74625, and link L3's traffic weight value would be decreased to 0.24875.
4. Link L1 received a SYNC\_ACK containing a MESS\_R of 32 indicating that 100% of the MESS\_T



- (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.2525, while link L2's traffic weight value would be decreased to 0.560625 and link L3's traffic weight value would be decreased to 0.186875.
- 5
5. Link L1 received a SYNC\_ACK containing a MESS\_R of 32 indicating that 100% of the MESS\_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.37625; link L2's traffic weight value would be decreased to 0.4678125, and link L3's traffic weight value would be decreased to 0.1559375.
- 10
6. Link L1 remains healthy and the traffic probabilities approach their steady state traffic probabilities.
- 15

B. Use of a DNS Proxy to Transparently Create Virtual Private Networks

A second improvement concerns the automatic creation of a virtual private network (VPN) in response to a domain-name server look-up function.

20

Conventional Domain Name Servers (DNSs) provide a look-up function that returns the IP address of a requested computer or host. For example, when a computer user types in the web name "Yahoo.com," the user's web browser transmits a request to a DNS, which converts the name into a four-part IP address that is returned to the user's browser and then used by the browser to contact the destination web site.

25

This conventional scheme is shown in FIG. 25. A user's computer 2501 includes a client application 2504 (for example, a web browser) and an IP protocol stack 2505. When the user enters the name of a destination host, a request DNS REQ is made (through IP protocol stack 2505) to a DNS 2502 to look up the IP address associated with the name. The DNS returns the IP address DNS RESP to client application 2504, which is then able to use the IP address to communicate with the host 2503 through separate transactions such as PAGE REQ and PAGE RESP.

30

In the conventional architecture shown in FIG. 25, nefarious listeners on the Internet could intercept the DNS REQ and DNS RESP packets and thus learn what IP addresses the user was contacting. For example, if a user wanted to set up a secure communication path with a web site having the name "Target.com," when the user's browser contacted a DNS to find the IP address for that web site, the true IP address of that web site would be revealed over the Internet as part of the DNS inquiry. This would hamper anonymous communications on the Internet.

35

One conventional scheme that provides secure virtual private networks over the Internet provides the DNS server with the public keys of the machines that the DNS server has the addresses for. This allows hosts to retrieve automatically the public keys of a host that the host is to communicate with so that the host can set up a VPN without having the user enter the public key of the destination host. One implementation of this standard is presently being developed as part of the FreeS/WAN project(RFC 2535).

40

The conventional scheme suffers from certain drawbacks. For example, any user can perform a DNS request. Moreover, DNS requests resolve to the same value for all users.

45

According to certain aspects of the invention, a specialized DNS server traps DNS requests and, if the request is from a special type of user (e.g., one for which secure communication services are defined), the server does not return the true IP address of the target node, but instead

50

automatically sets up a virtual private network between the target node and the user. The VPN is preferably implemented using the IP address "hopping" features of the basic invention described above, such that the true identity of the two nodes cannot be determined even if packets during the communication are intercepted. For DNS requests that are determined to not require secure services (e.g., an unregistered user), the DNS server transparently "passes through" the request to provide a normal look-up function and return the IP address of the target web server, provided that the requesting host has permissions to resolve unsecured sites. Different users who make an identical DNS request could be provided with different results.

FIG. 26 shows a system employing various principles summarized above. A user's computer 2601 includes a conventional client (e.g., a web browser) 2605 and an IP protocol stack 2606 that preferably operates in accordance with an IP hopping function 2607 as outlined above. A modified DNS server 2602 includes a conventional DNS server function 2609 and a DNS proxy 2610. A gatekeeper server 2603 is interposed between the modified DNS server and a secure target site 2704. An "unsecure" target site 2611 is also accessible via conventional IP protocols.

According to one embodiment, DNS proxy 2610 intercepts all DNS lookup functions from client 2605 and determines whether access to a secure site has been requested. If access to a secure site has been requested (as determined, for example, by a domain name extension, or by reference to an internal table of such sites), DNS proxy 2610 determines whether the user has sufficient security privileges to access the site. If so, DNS proxy 2610 transmits a message to gatekeeper 2603 requesting that a virtual private network be created between user computer 2601 and secure target site 2604. In one embodiment, gatekeeper 2603 creates "hopping-blocks" to be used by computer 2601 and secure target site 2604 for secure communication. Then, gatekeeper 2603 communicates these to user computer 2601. Thereafter, DNS proxy 2610 returns to user computer 2601 the resolved address passed to it by the gatekeeper (this address could be different from the actual target computer) 2604, preferably using a secure administrative VPN. The address that is returned need not be the actual address of the destination computer.

Had the user requested lookup of a non-secure web site such as site 2611, DNS proxy would merely pass through to conventional DNS server 2609 the look-up request, which would be handled in a conventional manner, returning the IP address of non-secure web site 2611. If the user had requested lookup of a secure web site but lacked credentials to create such a connection, DNS proxy 2610 would return a "host unknown" error to the user. In this manner, different users requesting access to the same DNS name could be provided with different look-up results.

Gatekeeper 2603 can be implemented on a separate computer (as shown in FIG. 26) or as a function within modified DNS server 2602. In general, it is anticipated that gatekeeper 2703 facilitates the allocation and exchange of information needed to communicate securely, such as using "hopped" IP addresses. Secure hosts such as site 2604 are assumed to be equipped with a secure communication function such as an IP hopping function 2608.

It will be appreciated that the functions of DNS proxy 2610 and DNS server 2609 can be combined into a single server for convenience. Moreover, although element 2602 is shown as combining the functions of two servers, the two servers can be made to operate independently.

FIG. 27 shows steps that can be executed by DNS proxy server 2610 to handle requests for DNS look-up for secure



hosts. In step 2701, a DNS look-up request is received for a target host. In step 2702, a check is made to determine whether access to a secure host was requested. If not, then in step 2703 the DNS request is passed to conventional DNS server 2609, which looks up the IP address of the target site and returns it to the user's application for further processing.

In step 2702, if access to a secure host was requested, then in step 2704 a further check is made to determine whether the user is authorized to connect to the secure host. Such a check can be made with reference to an internally stored list of authorized IP addresses, or can be made by communicating with gatekeeper 2603 (e.g., over an "administrative" VPN that is secure). It will be appreciated that different levels of security can also be provided for different categories of hosts. For example, some sites may be designated as having a certain security level, and the security level of the user requesting access must match that security level. The user's security level can also be determined by transmitting a request message back to the user's computer requiring that it prove that it has sufficient privileges.

If the user is not authorized to access the secure site, then a "host unknown" message is returned (step 2705). If the user has sufficient security privileges, then in step 2706 a secure VPN is established between the user's computer and the secure target site. As described above, this is preferably done by allocating a hopping regime that will be carried out between the user's computer and the secure target site, and is preferably performed transparently to the user (i.e., the user need not be involved in creating the secure link). As described in various embodiments of this application, any of various fields can be "hopped" (e.g., IP source/destination addresses; a field in the header; etc.) in order to communicate securely.

Some or all of the security functions can be embedded in gatekeeper 2603, such that it handles all requests to connect to secure sites. In this embodiment, DNS proxy 2610 communicates with gatekeeper 2603 to determine (preferably over a secure administrative VPN) whether the user has access to a particular web site. Various scenarios for implementing these features are described by way of example below:

Scenario #1: Client has permission to access target computer, and gatekeeper has a rule to make a VPN for the client. In this scenario, the client's DNS request would be received by the DNS proxy server 2610, which would forward the request to gatekeeper 2603. The gatekeeper would establish a VPN between the client and the requested target. The gatekeeper would provide the address of the destination to the DNS proxy, which would then return the resolved name as a result. The resolved address can be transmitted back to the client in a secure administrative VPN.

Scenario #2: Client does not have permission to access target computer. In this scenario, the client's DNS request would be received by the DNS proxy server 2610, which would forward the request to gatekeeper 2603. The gatekeeper would reject the request, informing DNS proxy server 2610 that it was unable to find the target computer. The DNS proxy 2610 would then return a "host unknown" error message to the client.

Scenario #3: Client has permission to connect using a normal non-VPN link, and the gatekeeper does not have a rule to set up a VPN for the client to the target site. In this scenario, the client's DNS request is received by DNS proxy server 2610, which would check its rules and determine that no VPN is needed. Gatekeeper 2603 would then inform the DNS proxy server to forward the request to conventional

DNS server 2609, which would resolve the request and return the result to the DNS proxy server and then back to the client.

Scenario #4: Client does not have permission to establish a normal/non-VPN link, and the gatekeeper does not have a rule to make a VPN for the client to the target site. In this scenario, the DNS proxy server would receive the client's DNS request and forward it to gatekeeper 2603. Gatekeeper 2603 would determine that no special VPN was needed, but that the client is not authorized to communicate with non-VPN members. The gatekeeper would reject the request, causing DNS proxy server 2610 to return an error message to the client.

C. Large Link to Small Link Bandwidth Management

One feature of the basic architecture is the ability to prevent so-called "denial of service" attacks that can occur if a computer hacker floods a known Internet node with packets, thus preventing the node from communicating with other nodes. Because IP addresses or other fields are "hopped" and packets arriving with invalid addresses are quickly discarded, Internet nodes are protected against flooding targeted at a single IP address.

In a system in which a computer is coupled through a link having a limited bandwidth (e.g., an edge router) to a node that can support a much higher-bandwidth link (e.g., an Internet Service Provider), a potential weakness could be exploited by a determined hacker. Referring to FIG. 28, suppose that a first host computer 2801 is communicating with a second host computer 2804 using the IP address hopping principles described above. The first host computer is coupled through an edge router 2802 to an Internet Service Provider (ISP) 2803 through a low bandwidth link (LOW BW), and is in turn coupled to second host computer 2804 through parts of the Internet through a high bandwidth link (HIGH BW). In this architecture, the ISP is able to support a high bandwidth to the internet, but a much lower bandwidth to the edge router 2802.

Suppose that a computer hacker is able to transmit a large quantity of dummy packets addressed to first host computer 2801 across high bandwidth link HIGH BW. Normally, host computer 2801 would be able to quickly reject the packets since they would not fall within the acceptance window permitted by the IP address hopping scheme. However, because the packets must travel across low bandwidth link LOW BW, the packets overwhelm the lower bandwidth link before they are received by host computer 2801. Consequently, the link to host computer 2801 is effectively flooded before the packets can be discarded.

According to one inventive improvement, a "link guard" function 2805 is inserted into the high-bandwidth node (e.g., ISP 2803) that quickly discards packets destined for a low-bandwidth target node if they are not valid packets. Each packet destined for a low-bandwidth node is cryptographically authenticated to determine whether it belongs to a VPN. If it is not a valid VPN packet, the packet is discarded at the high-bandwidth node. If the packet is authenticated as belonging to a VPN, the packet is passed with high preference. If the packet is a valid non-VPN packet, it is passed with a lower quality of service (e.g., lower priority).

In one embodiment, the ISP distinguishes between VPN and non-VPN packets using the protocol of the packet. In the case of IPSEC [rfc 2401], the packets have IP protocols 420 and 421. In the case of the TARP VPN, the packets will have an IP protocol that is not yet defined. The ISP's link guard, 2805, maintains a table of valid VPNs which it uses to validate whether VPN packets are cryptographically valid.

According to one embodiment, packets that do not fall within any hop windows used by nodes on the low-bandwidth link are rejected, or are sent with a lower quality of service. One approach for doing this is to provide a copy of the IP hopping tables used by the low-bandwidth nodes to the high-bandwidth node, such that both the high-bandwidth and low-bandwidth nodes track hopped packets (e.g., the high-bandwidth node moves its hopping window as valid packets are received). In such a scenario, the high-bandwidth node discards packets that do not fall within the hopping window before they are transmitted over the low-bandwidth link. Thus, for example, ISP **2903** maintains a copy **2910** of the receive table used by host computer **2901**. Incoming packets that do not fall within this receive table are discarded. According to a different embodiment, link guard **2805** validates each VPN packet using a keyed hashed message authentication code (HMAC) [rfc 2104].

According to another embodiment, separate VPNs (using, for example, hopblocks) can be established for communicating between the low-bandwidth node and the high-bandwidth node (i.e., packets arriving at the high-bandwidth node are converted into different packets before being transmitted to the low-bandwidth node).

As shown in FIG. **29**, for example, suppose that a first host computer **2900** is communicating with a second host computer **2902** over the Internet, and the path includes a high bandwidth link HIGH BW to an ISP **2901** and a low bandwidth link LOW BW through an edge router **2904**. In accordance with the basic architecture described above, first host computer **2900** and second host computer **2902** would exchange hopblocks (or a hopblock algorithm) and would be able to create matching transmit and receive tables **2905**, **2906**, **2912** and **2913**. Then in accordance with the basic architecture, the two computers would transmit packets having seemingly random IP source and destination addresses, and each would move a corresponding hopping window in its receive table as valid packets were received.

Suppose that a nefarious computer hacker **2903** was able to deduce that packets having a certain range of IP addresses (e.g., addresses 100 to 200 for the sake of simplicity) are being transmitted to ISP **2901**, and that these packets are being forwarded over a low-bandwidth link. Hacker computer **2903** could thus "flood" packets having addresses falling into the range 100 to 200, expecting that they would be forwarded along low bandwidth link LOW BW, thus causing the low bandwidth link to become overwhelmed. The fast packet reject mechanism in first host computer **3000** would be of little use in rejecting these packets, since the low bandwidth link was effectively jammed before the packets could be rejected. In accordance with one aspect of the improvement, however, VPN link guard **2911** would prevent the attack from impacting the performance of VPN traffic because the packets would either be rejected as invalid VPN packets or given a lower quality of service than VPN traffic over the lower bandwidth link. A denial-of-service flood attack could, however, still disrupt non-VPN traffic.

According to one embodiment of the improvement, ISP **2901** maintains a separate VPN with first host computer **2900**, and thus translates packets arriving at the ISP into packets having a different IP header before they are transmitted to host computer **2900**. The cryptographic keys used to authenticate VPN packets at the link guard **2911** and the cryptographic keys used to encrypt and decrypt the VPN packets at host **2902** and host **2901** can be different, so that link guard **2911** does not have access to the private host data; it only has the capability to authenticate those packets.

According to yet a third embodiment, the low-bandwidth node can transmit a special message to the high-bandwidth

node instructing it to shut down all transmissions on a particular IP address, such that only hopped packets will pass through to the low-bandwidth node. This embodiment would prevent a hacker from flooding packets using a single IP address. According to yet a fourth embodiment, the high-bandwidth node can be configured to discard packets transmitted to the low-bandwidth node if the transmission rate exceeds a certain predetermined threshold for any given IP address; this would allow hopped packets to go through. In this respect, link guard **2911** can be used to detect that the rate of packets on a given IP address are exceeding a threshold rate; further packets addressed to that same IP address would be dropped or transmitted at a lower priority (e.g., delayed).

#### D. Traffic Limiter

In a system in which multiple nodes are communicating using "hopping" technology, a treasonous insider could internally flood the system with packets. In order to prevent this possibility, one inventive improvement involves setting up "contracts" between nodes in the system, such that a receiver can impose a bandwidth limitation on each packet sender. One technique for doing this is to delay acceptance of a checkpoint synchronization request from a sender until a certain time period (e.g., one minute) has elapsed. Each receiver can effectively control the rate at which its hopping window moves by delaying "SYNC ACK" responses to "SYNC\_REQ" messages.

A simple modification to the checkpoint synchronizer will serve to protect a receiver from accidental or deliberate overload from an internally treasonous client. This modification is based on the observation that a receiver will not update its tables until a SYNC\_REQ is received on hopped address CKPT\_N. It is a simple matter of deferring the generation of a new CKPT\_N until an appropriate interval after previous checkpoints.

Suppose a receiver wished to restrict reception from a transmitter to 100 packets a second, and that checkpoint synchronization messages were triggered every 50 packets. A compliant transmitter would not issue new SYNC\_REQ messages more often than every 0.5 seconds. The receiver could delay a non-compliant transmitter from synchronizing by delaying the issuance of CKPT\_N for 0.5 second after the last SYNC\_REQ was accepted.

In general, if M receivers need to restrict N transmitters issuing new SYNC\_REQ messages after every W messages to sending R messages a second in aggregate, each receiver could defer issuing a new CKPT\_N until  $M \times N \times W / R$  seconds have elapsed since the last SYNC\_REQ has been received and accepted. If the transmitter exceeds this rate between a pair of checkpoints, it will issue the new checkpoint before the receiver is ready to receive it, and the SYNC\_REQ will be discarded by the receiver. After this, the transmitter will re-issue the SYNC\_REQ every  $T_i$  seconds until it receives a SYNC\_ACK. The receiver will eventually update CKPT\_N and the SYNC\_REQ will be acknowledged. If the transmission rate greatly exceeds the allowed rate, the transmitter will stop until it is compliant. If the transmitter exceeds the allowed rate by a little, it will eventually stop after several rounds of delayed synchronization until it is in compliance. Hacking the transmitter's code to not shut off only permits the transmitter to lose the acceptance window. In this case it can recover the window and proceed only after it is compliant again.

Two practical issues should be considered when implementing the above scheme:

1. The receiver rate should be slightly higher than the permitted rate in order to allow for statistical fluctuations in traffic arrival times and non-uniform load balancing.

2. Since a transmitter will rightfully continue to transmit for a period after a SYNC\_REQ is transmitted, the algorithm above can artificially reduce the transmitter's bandwidth. If events prevent a compliant transmitter from synchronizing for a period (e.g. the network dropping a SYNC\_REQ or a SYNC\_ACK) a SYNC\_REQ will be accepted later than expected. After this, the transmitter will transmit fewer than expected messages before encountering the next checkpoint. The new checkpoint will not have been activated and the transmitter will have to retransmit the SYNC\_REQ. This will appear to the receiver as if the transmitter is not compliant. Therefore, the next checkpoint will be accepted late from the transmitter's perspective. This has the effect of reducing the transmitter's allowed packet rate until the transmitter transmits at a packet rate below the agreed upon rate for a period of time.

To guard against this, the receiver should keep track of the times that the last C SYNC\_REQs were received and accepted and use the minimum of  $M \times N \times W/R$  seconds after the last SYNC\_REQ has been received and accepted,  $2 \times M \times N \times W/R$  seconds after next to the last SYNC\_REQ has been received and accepted,  $C \times M \times N \times W/R$  seconds after  $(C-1)^{th}$  to the last SYNC\_REQ has been received, as the time to activate CKPT\_N. This prevents the receiver from inappropriately limiting the transmitter's packet rate if at least one out of the last C SYNC\_REQs was processed on the first attempt.

FIG. 30 shows a system employing the above-described principles. In FIG. 30, two computers 3000 and 3001 are assumed to be communicating over a network N in accordance with the "hopping" principles described above (e.g., hopped IP addresses, discriminator values, etc.). For the sake of simplicity, computer 3000 will be referred to as the receiving computer and computer 3001 will be referred to as the transmitting computer, although full duplex operation is of course contemplated. Moreover, although only a single transmitter is shown, multiple transmitters can transmit to receiver 3000.

As described above, receiving computer 3000 maintains a receive table 3002 including a window W that defines valid IP address pairs that will be accepted when appearing in incoming data packets. Transmitting computer 3001 maintains a transmit table 3003 from which the next IP address pairs will be selected when transmitting a packet to receiving computer 3000. (For the sake of illustration, window W is also illustrated with reference to transmit table 3003). As transmitting computer moves through its table, it will eventually generate a SYNC\_REQ message as illustrated in function 3010. This is a request to receiver 3000 to synchronize the receive table 3002, from which transmitter 3001 expects a response in the form of a CKPT\_N (included as part of a SYNC\_ACK message). If transmitting computer 3001 transmits more messages than its allotment, it will prematurely generate the SYNC\_REQ message. (If it has been altered to remove the SYNC\_REQ message generation altogether, it will fall out of synchronization since receiver 3000 will quickly reject packets that fall outside of window W, and the extra packets generated by transmitter 3001 will be discarded).

In accordance with the improvements described above, receiving computer 3000 performs certain steps when a SYNC\_REQ message is received, as illustrated in FIG. 30. In step 3004, receiving computer 3000 receives the SYNC\_REQ message. In step 3005, a check is made to determine whether the request is a duplicate. If so, it is discarded in step 3006. In step 3007, a check is made to determine whether the

SYNC\_REQ received from transmitter 3001 was received at a rate that exceeds the allowable rate R (i.e., the period between the time of the last SYNC\_REQ message). The value R can be a constant, or it can be made to fluctuate as desired. If the rate exceeds R, then in step 3008 the next activation of the next CKPT\_N hopping table entry is delayed by  $W/R$  seconds after the last SYNC\_REQ has been accepted.

Otherwise, if the rate has not been exceeded, then in step 3109 the next CKPT\_N value is calculated and inserted into the receiver's hopping table prior to the next SYNC\_REQ from the transmitter 3101. Transmitter 3101 then processes the SYNC\_REQ in the normal manner.

#### E. Signaling Synchronizer

In a system in which a large number of users communicate with a central node using secure hopping technology, a large amount of memory must be set aside for hopping tables and their supporting data structures. For example, if one million subscribers to a web site occasionally communicate with the web site, the site must maintain one million hopping tables, thus using up valuable computer resources, even though only a small percentage of the users may actually be using the system at any one time. A desirable solution would be a system that permits a certain maximum number of simultaneous links to be maintained, but which would "recognize" millions of registered users at any one time. In other words, out of a population of a million registered users, a few thousand at a time could simultaneously communicate with a central server, without requiring that the server maintain one million hopping tables of appreciable size.

One solution is to partition the central node into two nodes: a signaling server that performs session initiation for user log-on and log-off (and requires only minimally sized tables), and a transport server that contains larger hopping tables for the users. The signaling server listens for the millions of known users and performs a fast-packet reject of other (bogus) packets. When a packet is received from a known user, the signaling server activates a virtual private link (VPL) between the user and the transport server, where hopping tables are allocated and maintained. When the user logs onto the signaling server, the user's computer is provided with hop tables for communicating with the transport server, thus activating the VPL. The VPLs can be torn down when they become inactive for a time period, or they can be torn down upon user log-out. Communication with the signaling server to allow user log-on and log-off can be accomplished using a specialized version of the checkpoint scheme described above.

FIG. 31 shows a system employing certain of the above-described principles. In FIG. 31, a signaling server 3101 and a transport server 3102 communicate over a link. Signaling server 3101 contains a large number of small tables 3106 and 3107 that contain enough information to authenticate a communication request with one or more clients 3103 and 3104. As described in more detail below, these small tables may advantageously be constructed as a special case of the synchronizing checkpoint tables described previously. Transport server 3102, which is preferably a separate computer in communication with signaling server 3101, contains a smaller number of larger hopping tables 3108, 3109, and 3110 that can be allocated to create a VPN with one of the client computers.

According to one embodiment, a client that has previously registered with the system (e.g., via a system administration function, a user registration procedure, or some other method) transmits a request for information from a computer (e.g., a web site). In one variation, the request is

made using a “hopped” packet, such that signaling server 3101 will quickly reject invalid packets from unauthorized computers such as hacker computer 3105. An “administrative” VPN can be established between all of the clients and the signaling server in order to ensure that a hacker cannot flood signaling server 3101 with bogus packets. Details of this scheme are provided below.

Signaling server 3101 receives the request 3111 and uses it to determine that client 3103 is a validly registered user. Next, signaling server 3101 issues a request to transport server 3102 to allocate a hopping table (or hopping algorithm or other regime) for the purpose of creating a VPN with client 3103. The allocated hopping parameters are returned to signaling server 3101 (path 3113), which then supplies the hopping parameters to client 3103 via path 3114, preferably in encrypted form.

Thereafter, client 3103 communicates with transport server 3102 using the normal hopping techniques described above. It will be appreciated that although signaling server 3101 and transport server 3102 are illustrated as being two separate computers, they could of course be combined into a single computer and their functions performed on the single computer. Alternatively, it is possible to partition the functions shown in FIG. 31 differently from as shown without departing from the inventive principles.

One advantage of the above-described architecture is that signaling server 3101 need only maintain a small amount of information on a large number of potential users, yet it retains the capability of quickly rejecting packets from unauthorized users such as hacker computer 3105. Larger data tables needed to perform the hopping and synchronization functions are instead maintained in a transport server 3102, and a smaller number of these tables are needed since they are only allocated for “active” links. After a VPN has become inactive for a certain time period (e.g., one hour), the VPN can be automatically torn down by transport server 3102 or signaling server 3101.

A more detailed description will now be provided regarding how a special case of the checkpoint synchronization feature can be used to implement the signaling scheme described above.

The signaling synchronizer may be required to support many (millions) of standing, low bandwidth connections. It therefore should minimize per-VPL memory usage while providing the security offered by hopping technology. In order to reduce memory usage in the signaling server, the data hopping tables can be completely eliminated and data can be carried as part of the SYNC\_REQ message. The table used by the server side (receiver) and client side (transmitter) is shown schematically as element 3106 in FIG. 31.

The meaning and behaviors of CKPT\_N, CKPT\_O and CKPT\_R remain the same from the previous description, except that CKPT\_N can receive a combined data and SYNC\_REQ message or a SYNC\_REQ message without the data.

The protocol is a straightforward extension of the earlier synchronizer. Assume that a client transmitter is on and the tables are synchronized. The initial tables can be generated “out of band.” For example, a client can log into a web server to establish an account over the Internet. The client will receive keys etc encrypted over the Internet. Meanwhile, the server will set up the signaling VPN on the signaling server.

Assuming that a client application wishes to send a packet to the server on the client’s standing signaling VPL:

1. The client sends the message marked as a data message on the inner header using the transmitter’s CKPT\_N address. It turns the transmitter off and starts a timer T1 noting CKPT\_O. Messages can be one of three types:

DATA, SYNC\_REQ and SYNC\_ACK. In the normal algorithm, some potential problems can be prevented by identifying each message type as part of the encrypted inner header field. In this algorithm, it is important to distinguish a data packet and a SYNC\_REQ in the signaling synchronizer since the data and the SYNC\_REQ come in on the same address.

2. When the server receives a data message on its CKPT\_N, it verifies the message and passes it up the stack. The message can be verified by checking message type and other information (i.e user credentials) contained in the inner header. It replaces its CKPT\_O with CKPT\_N and generates the next CKPT\_N. It updates its transmitter side CKPT\_R to correspond to the client’s receiver side CKPT\_R and transmits a SYNC\_ACK containing CKPT\_O in its payload.
3. When the client side receiver receives a SYNC\_ACK on its CKPT\_R with a payload matching its transmitter side CKPT\_O and the transmitter is off, the transmitter is turned on and the receiver side CKPT\_R is updated. If the SYNC\_ACK’s payload does not match the transmitter side CKPT\_O or the transmitter is on, the SYNC\_ACK is simply discarded.
4. T1 expires: If the transmitter is off and the client’s transmitter side CKPT\_O matches the CKPT\_O associated with the timer, it starts timer T1 noting CKPT\_O again, and a SYNC\_REQ is sent using the transmitter’s CKPT\_O address. Otherwise, no action is taken.
5. When the server receives a SYNC\_REQ on its CKPT\_N it replaces its CKPT\_O with CKPT\_N and generates the next CKPT\_N. It updates its transmitter side CKPT\_R to correspond to the client’s receiver side CKPT\_R and transmits a SYNC\_ACK containing CKPT\_O in its payload.
6. When the server receives a SYNC\_REQ on its CKPT\_O, it updates its transmitter side CKPT\_R to correspond to the client’s receiver side CKPT\_R and transmits a SYNC\_ACK containing CKPT\_O in its payload.

FIG. 32 shows message flows to highlight the protocol. Reading from top to bottom, the client sends data to the server using its transmitter side CKPT\_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT\_N into CKPT\_O and updates CKPT\_N. This message is successfully received and a passed up the stack. It also synchronizes the receiver i.e, the server loads CKPT\_N into CKPT\_O and generates a new CKPT\_N, it generates a new CKPT\_R in the server side transmitter and transmits a SYNC\_ACK containing the server side receiver’s CKPT\_O the server. The SYNC\_ACK is successfully received at the client. The client side receiver’s CKPT\_R is updated, the transmitter is turned on and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

Next, the client sends data to the server using its transmitter side CKPT\_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT\_N into CKPT\_O and updates CKPT\_N. This message is lost. The client side timer expires and as a result a SYNC\_REQ is transmitted on the client side transmitter’s CKPT\_O (this will keep happening until the SYNC\_ACK has been received at the client). The SYNC\_REQ is successfully received at the server. It synchronizes the receiver i.e, the server loads CKPT\_N into CKPT\_O and generates a new

US 6,502,135 B1

47

CKPT\_N, it generates a new CKPT\_R in the server side transmitter and transmits a SYNC\_ACK containing the server side receiver's CKPT\_O to the server. The SYNC\_ACK is successfully received at the client. The client side receiver's CKPT\_R is updated, the transmitter is turned off and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

There are numerous other scenarios that follow this flow. For example, the SYNC\_ACK could be lost. The transmitter would continue to re-send the SYNC\_REQ until the receiver synchronizes and responds.

The above-described procedures allow a client to be authenticated at signaling server 3201 while maintaining the ability of signaling server 3201 to quickly reject invalid packets, such as might be generated by hacker computer 3205. In various embodiments, the signaling synchronizer is really a derivative of the synchronizer. It provides the same protection as the hopping protocol, and it does so for a large number of low bandwidth connections.

What is claimed is:

1. A method of transparently creating a virtual private network (VPN) between a client computer and a target computer, comprising the steps of:

- (1) generating from the client computer a Domain Name Service (DNS) request that requests an IP address corresponding to a domain name associated with the target computer;
- (2) determining whether the DNS request transmitted in step (1) is requesting access to a secure web site; and
- (3) in response to determining that the DNS request in step (2) is requesting access to a secure target web site, automatically initiating the VPN between the client computer and the target computer.

2. The method of claim 1, wherein steps (2) and (3) are performed at a DNS server separate from the client computer.

3. The method of claim 1, further comprising the step of:

- (4) in response to determining that the DNS request in step (2) is not requesting access to a secure target web site, resolving the IP address for the domain name and returning the IP address to the client computer.

4. The method of claim 1, wherein step (3) comprises the step of, prior to automatically initiating the VPN between the client computer and the target computer, determining whether the client computer is authorized to establish a VPN with the target computer and, if not so authorized, returning an error from the DNS request.

5. The method of claim 1, wherein step (3) comprises the step of, prior to automatically initiating the VPN between the client computer and the target computer, determining whether the client computer is authorized to resolve addresses of non secure target computers and, if not so authorized, returning an error from the DNS request.

6. The method of claim 1, wherein step (3) comprises the step of establishing the VPN by creating an IP address hopping scheme between the client computer and the target computer.

7. The method of claim 1, wherein step (3) comprises the step of using a gatekeeper computer that allocates VPN resources for communicating between the client computer and the target computer.

8. The method of claim 1, wherein step (2) is performed in a DNS proxy server that passes through the request to a DNS server if it is determined in step (3) that access is not being requested to a secure target web site.

9. The method of claim 5, wherein step (3) comprises the step of transmitting a message to the client computer to

48

determine whether the client computer is authorized to establish the VPN target computer.

10. A system that transparently creates a virtual private network (VPN) between a client computer and a secure target computer, comprising:

a DNS proxy server that receives a request from the client computer to look up an IP address for a domain name, wherein the DNS proxy server returns the IP address for the requested domain name if it is determined that access to a non-secure web site has been requested, and wherein the DNS proxy server generates a request to create the VPN between the client computer and the secure target computer if it is determined that access to a secure web site has been requested; and

a gatekeeper computer that allocates resources for the VPN between the client computer and the secure web computer in response to the request by the DNS proxy server.

11. The system of claim 10, wherein the gatekeeper computer creates the VPN by establishing an IP address hopping regime that is used to pseudorandomly change IP addresses in packets transmitted between the client computer and the secure target computer.

12. The system of claim 10, wherein the gatekeeper computer determines whether the client computer has sufficient security privileges to create the VPN and, if the client computer lacks sufficient security privileges, rejecting the request to create the VPN.

13. A method of establishing communication between one of a plurality of client computers and a central computer that maintains a plurality of authentication tables each corresponding to one of the client computers, the method comprising the steps of:

(1) in the central computer, receiving from one of the plurality of client computers a request to establish a connection;

(2) authenticating, with reference to one of the plurality of authentication tables, that the request received in step (1) is from an authorized client;

(3) responsive to a determination that the request is from an authorized client, allocating resources to establish a virtual private link between the client and a second computer; and

(4) communicating between the authorized client and the second computer using the virtual private link.

14. The method of claim 13, wherein step (4) comprises the step of communicating according to a scheme by which at least one field in a series of data packets is periodically changed according to a known sequence.

15. The method of claim 14, wherein step (4) comprises the step of comparing an Internet Protocol (IP) address in a header of each data packet to a table of valid IP addresses maintained in a table in the second computer.

16. The method of claim 15, wherein step (4) comprises the step of comparing the IP address in the header of each data packet to a moving window of valid IP addresses, and rejecting data packets having IP addresses that do not fall within the moving window.

17. The method of claim 13, wherein step (2) comprises the step of using a checkpoint data structure that maintains synchronization of a periodically changing parameter known by the central computer and the client computer to authenticate the client.

\* \* \* \* \*

UNITED STATES PATENT AND TRADEMARK OFFICE  
**CERTIFICATE OF CORRECTION**

PATENT NO. : 6,502,135 B1  
DATED : December 31, 2002  
INVENTOR(S) : Edmund Colby Munger et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page,

Item [56], **References Cited**, OTHER PUBLICATIONS, insert the following:

-- Search Report (dated 8/20/02), International Application No. PCT/US01/04340  
Search Report (dated 8/23/02), International Application No. PCT/US01/13260  
James E. Bellaire, "New Statement of Rules – Naming Internet Domains", Internet Newsgroup, July 30, 1995, 1 page.  
D. Clark, "US Calls for Private Domain-Name System", Computer, IEEE Computer Society, August 1, 1998, pages 22-25.  
August Bequai, "Balancing Legal Concerns Over Crime and Security in Cyberspace", Computer & Security, Vol. 17, No. 4, 1998, pages 293-298.  
Rich Winkel, "CAQ: Networking With Spooks: The NET & The Control Of Information", Internet Newsgroup, June 21, 1997, 4 pages. --

Column 48,

Line 2, "VPN target computer" has been replaced with -- VPN with the target computer --.

Signed and Sealed this

Ninth Day of September, 2003

A handwritten signature in black ink, appearing to read "James E. Rogan", with a horizontal line drawn underneath it.

JAMES E. ROGAN  
*Director of the United States Patent and Trademark Office*

US006502135C1

(12) **INTER PARTES REEXAMINATION CERTIFICATE** (0271st)**United States Patent****Munger et al.**(10) **Number:** **US 6,502,135 C1**(45) **Certificate Issued:** **Jun. 7, 2011**(54) **AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS WITH ASSURED SYSTEM AVAILABILITY**

4,933,846 A	6/1990	Humphrey et al.
4,988,990 A	1/1991	Warrior
5,276,735 A	1/1994	Boebert et al.
5,303,302 A	4/1994	Burrows

(75) **Inventors:** **Edmund Colby Munger**, Crownsville, MD (US); **Douglas Charles Schmidt**, Severna Park, MD (US); **Robert Dunham Short, III**, Leesburg, VA (US); **Victor Larson**, Fairfax, VA (US); **Michael Williamson**, South Riding, VA (US)

(Continued)

**FOREIGN PATENT DOCUMENTS**

DE	199 24 575	12/1999
EP	0 814 589	12/1997
EP	836306 A1	4/1998
EP	0 838 930	4/1998
EP	0 858 189	8/1998

(Continued)

(73) **Assignee:** **Virnetx, Inc.**, Scotts Valley Drive, CA (US)**Reexamination Request:**

No. 95/001,269, Dec. 8, 2009

**Reexamination Certificate for:**

Patent No.: **6,502,135**  
 Issued: **Dec. 31, 2002**  
 Appl. No.: **09/504,783**  
 Filed: **Feb. 15, 2000**

**OTHER PUBLICATIONS**

Alan O. Frier et al., "The SSL Protocol Version 3.0", Nov. 18, 1996, printed from <http://www.netscape.com/eng/ss13/draft302.txt> on Feb. 4, 2002, 56 pages.

(Continued)

*Primary Examiner*—Andrew L Nalven

Certificate of Correction issued Sep. 9, 2003.

**Related U.S. Application Data**

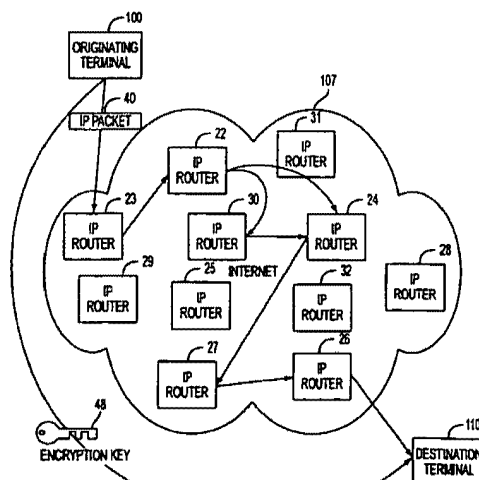
- (63) Continuation of application No. 09/429,643, filed on Oct. 29, 1999, now Pat. No. 7,010,604.  
 (60) Provisional application No. 60/106,261, filed on Oct. 30, 1998, and provisional application No. 60/137,704, filed on Jun. 7, 1999.

(51) **Int. Cl.**  
**G06F 15/173** (2006.01)(52) **U.S. Cl.** ..... **709/225; 709/229; 709/245**(58) **Field of Classification Search** ..... **709/225**  
See application file for complete search history.(56) **References Cited****U.S. PATENT DOCUMENTS**

2,895,502 A 7/1959 Roper et al.

(57) **ABSTRACT**

A plurality of computer nodes communicate using seemingly random Internet Protocol source and destination addresses. Data packets matching criteria defined by a moving window of valid addresses are accepted for further processing, while those that do not meet the criteria are quickly rejected. Improvements to the basic design include (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities.

**Appx182**

## US 6,502,135 C1

Page 2

## U.S. PATENT DOCUMENTS

5,311,593 A	5/1994	Carmi	6,256,671 B1	7/2001	Strentzsch et al.
5,329,521 A	7/1994	Walsh et al.	6,262,987 B1	7/2001	Mogul
5,341,426 A	8/1994	Barney et al.	6,263,445 B1	7/2001	Blumenau
5,367,643 A	11/1994	Chang et al.	6,286,047 B1	9/2001	Ramanathan et al.
5,384,848 A	1/1995	Kikuchi	6,298,341 B1	10/2001	Mann et al.
5,511,122 A	4/1996	Atkinson	6,301,223 B1	10/2001	Hrastar et al.
5,559,883 A	9/1996	Williams	6,308,274 B1	10/2001	Swift
5,561,669 A	10/1996	Lenney et al.	6,311,207 B1	10/2001	Mighdoll et al.
5,588,060 A	12/1996	Aziz	6,314,463 B1	11/2001	Abbott et al.
5,625,626 A	4/1997	Umekita	6,324,161 B1	11/2001	Kirch
5,629,984 A	5/1997	McManis	6,330,562 B1	12/2001	Boden et al.
5,654,695 A	8/1997	Olnowich et al.	6,332,158 B1	12/2001	Risley et al.
5,682,480 A	10/1997	Nakagawa	6,333,272 B1	12/2001	McMillin et al.
5,689,566 A	11/1997	Nguyen	6,338,082 B1	1/2002	Schneider
5,740,375 A	4/1998	Dunne et al.	6,353,614 B1	3/2002	Borella et al.
5,764,906 A	6/1998	Edelstein et al.	6,430,155 B1	8/2002	Davie et al.
5,771,239 A	6/1998	Moroney et al.	6,430,610 B1	8/2002	Carter
5,774,660 A	6/1998	Brendel et al.	6,487,598 B1	11/2002	Valencia
5,787,172 A	7/1998	Arnold	6,502,135 B1	12/2002	Munger et al.
5,796,942 A	8/1998	Esbensen	6,505,232 B1	1/2003	Mighdoll et al.
5,805,801 A	9/1998	Holloway et al.	6,510,154 B1	1/2003	Mayes et al.
5,805,803 A	9/1998	Birrell et al.	6,549,516 B1	4/2003	Albert et al.
5,822,434 A	10/1998	Caronni et al.	6,557,037 B1	4/2003	Provino
5,842,040 A	11/1998	Hughes et al.	6,571,296 B1	5/2003	Dillon
5,845,091 A	12/1998	Dunne et al.	6,571,338 B1	5/2003	Shaio et al.
5,864,666 A	1/1999	Shrader	6,581,166 B1	6/2003	Hirst et al.
5,867,650 A	2/1999	Osterman	6,618,761 B2	9/2003	Munger et al.
5,870,610 A	2/1999	Beyda et al.	6,671,702 B2	12/2003	Kruglikov et al.
5,878,231 A	3/1999	Baehr et al.	6,687,551 B2	2/2004	Steindl
5,892,903 A	4/1999	Klaus	6,687,746 B1	2/2004	Shuster et al.
5,898,830 A	4/1999	Wesinger et al.	6,701,437 B1	3/2004	Hoke et al.
5,905,859 A	5/1999	Holloway et al.	6,714,970 B1	3/2004	Fiveash et al.
5,918,019 A	6/1999	Valencia	6,717,949 B1	4/2004	Boden et al.
5,950,195 A	9/1999	Stockwell et al.	6,752,166 B2	6/2004	Lull et al.
5,996,016 A	11/1999	Thalheimer et al.	6,757,740 B1	6/2004	Parekh et al.
6,006,259 A	12/1999	Adelman et al.	6,760,766 B1	7/2004	Sahlqvist
6,006,272 A	12/1999	Aravamudan et al.	6,826,616 B2	11/2004	Larson et al.
6,016,318 A	1/2000	Tomoike	6,839,759 B2	1/2005	Larson et al.
6,016,512 A	1/2000	Huitema	6,937,597 B1	8/2005	Rosenberg et al.
6,041,342 A	3/2000	Yamaguchi	7,010,604 B1	3/2006	Munger et al.
6,052,788 A	4/2000	Wesinger et al.	7,039,713 B1	5/2006	Van Gunter et al.
6,055,574 A	4/2000	Smorodinsky et al.	7,072,964 B1	7/2006	Whittle et al.
6,061,346 A	5/2000	Nordman	7,133,930 B2	11/2006	Munger et al.
6,061,736 A	5/2000	Rochberger et al.	7,167,904 B1	1/2007	Devarajan et al.
6,079,020 A	6/2000	Liu	7,188,175 B1	3/2007	McKeeth
6,081,900 A	6/2000	Subramaniam et al.	7,188,180 B2	3/2007	Larson et al.
6,092,200 A	7/2000	Muniyappa et al.	7,197,563 B2	3/2007	Sheymov et al.
6,101,182 A	8/2000	Sistanizadeh et al.	7,353,841 B2	4/2008	Kono et al.
6,119,171 A	9/2000	Alkhatib	7,461,334 B1	12/2008	Lu et al.
6,119,234 A	9/2000	Aziz et al.	7,490,151 B2	2/2009	Munger et al.
6,147,976 A	11/2000	Shand et al.	7,493,403 B2	2/2009	Shull et al.
6,157,957 A	12/2000	Berthaud	2001/0049741 A1	12/2001	Skene et al.
6,158,011 A	12/2000	Chen et al.	2002/0004898 A1	1/2002	Droge
6,168,409 B1	1/2001	Fare	2004/0199493 A1	10/2004	Ruiz et al.
6,173,399 B1	1/2001	Gilbrech	2004/0199520 A1	10/2004	Ruiz et al.
6,175,867 B1	1/2001	Taghadoss	2004/0199608 A1	10/2004	Rechtermann et al.
6,178,409 B1	1/2001	Weber et al.	2004/0199620 A1	10/2004	Ruiz et al.
6,178,505 B1	1/2001	Schneider et al.	2005/0055306 A1	3/2005	Miller et al.
6,179,102 B1	1/2001	Weber et al.	2007/0208869 A1	9/2007	Adelman et al.
6,199,112 B1	3/2001	Wilson	2007/0214284 A1	9/2007	King et al.
6,202,081 B1	3/2001	Naudus	2007/0266141 A1	11/2007	Norton
6,222,842 B1	4/2001	Sasyan et al.	2008/0235507 A1	9/2008	Ishikawa et al.
6,223,287 B1	4/2001	Douglas et al.			
6,226,748 B1	5/2001	Bots et al.			
6,226,751 B1	5/2001	Arrow et al.			
6,233,618 B1	5/2001	Shannon			
6,243,360 B1	6/2001	Basilico			
6,243,749 B1	6/2001	Sitaraman et al.			
6,243,754 B1	6/2001	Guerin et al.			
6,246,670 B1	6/2001	Karlsson et al.			

## FOREIGN PATENT DOCUMENTS

GB	2 317 792	4/1998
GB	2 334 181 A	8/1999
JP	62-214744	9/1987
JP	04-363941	12/1992
JP	09-018492	1/1997
JP	10-070531	3/1998
JP	WO 9827783 A	6/1998



## US 6,502,135 C1

Page 3

WO	WO 98/27783	6/1998
WO	WO 98 55930	12/1998
WO	WO 98 59470	12/1998
WO	WO 99 38081	7/1999
WO	WO 99 48303	9/1999
WO	WO 00/17775	3/2000
WO	WO 001/17775	3/2000
WO	WO 00/70458	11/2000
WO	WO 01/016766	3/2001
WO	WO 01 50688	7/2001

## OTHER PUBLICATIONS

August Bequai, "Balancing Legal Concerns Over Crime and Security in Cyberspace", Computer & Security, vol. 17, No. 4, 1998, pp. 293-298.

D. B. Chapman et al., "Building Internet Firewalls", Nov. 1995, pp. 278-375.

D. Clark, "US Calls for Private Domain-Name System", Computer, IEEE Computer Society, Aug. 1, 1998, pp. 22-25.

Davila J et al, "Implementation of Virtual Private Networks at the Transport Layer", Information Security, Second International Work-shop, ISW'99. Proceedings (Lecture Springer-Verlag Berlin, Germany, [Online] 1999, pp. 85-102, XP002399276, ISBN 3-540-666.

Dolev, Shlomi and Ostrovsky, Rafil, "Efficient Anonymous Multicast and Reception" (Extended Abstract), 16 pages.

Donald E. Eastlake, 3rd, "Domain Name System Security Extensions", Internet Draft, Apr. 1998, pp. 1-51.

F. Halsall, "Data Communications, Computer Networks and Open Systems", Chapter 4, Protocol Basics, 1996, pp. 198-203.

Fasbender, Kesdogan, and Kubitz: "Variable and Scalable Security" Protection of Location Information in Mobile IP, IEEE publication, 1996, pp. 963-967.

Glossary for the Linux FreeS/WAN project, printed from [http://liberty.freesswan.org/freeswan\\_trees/freeswan-1.3/doc/glossary.html](http://liberty.freesswan.org/freeswan_trees/freeswan-1.3/doc/glossary.html) on Feb. 21, 2002, 25 pages.

J. Gilmore, "Swan: Securing the Internet against Wiretapping", printed from [http://liberty.freesswan.org/freeswan\\_trees/freeswan-1.3/doc/rationale.html](http://liberty.freesswan.org/freeswan_trees/freeswan-1.3/doc/rationale.html) on Feb. 21, 2002, 4 pages.

James E. Bellaire, "New Statement of Rules-Naming Internet Domains", Internet Newsgroup, Jul. 30, 1995, 1 page.

Jim Jones et al., "Distributed Denial of Service Attacks: Defenses", Global Integrity Corporation. 2000, pp. 1-14.

Laurie Wells (LANCASTERBIBELMAIL MSN COM); "Subject: Security Icon" USENET Newsgroup, Oct. 19, 1998, XP002200606, 1 page.

Linux FreeS/WAN Index File, printed from [http://liberty.freesswan.org/freeswan\\_trees/freeswan-1.3/doc/](http://liberty.freesswan.org/freeswan_trees/freeswan-1.3/doc/) on Feb. 21, 2002, 3 Pages.

P. Srisuresh et al., "DNS extensions to Network address Translators (DNS\_ALG)", Internet Draft, Jul. 1998, pp. 1-27.

RFC 2401 (dated Nov. 1998) Security Architecture for the Internet Protocol (RTP).

RFC 2543-SIP (dated Mar. 1999): Session Initiation Protocol (SIP or SIPS).

Rich Winkel, "CAQ: Networking With Spooks: The NET & The Control Of information", Internet Newsgroup, Jun. 21, 1997, 4 pages.

Rubin, Aviel D., Geer, Daniel, and Ranum, Marcus J. (Wiley Computer Publishing), "Web Security Sourcebook", pp. 82-94.

Search Report (dated Aug. 20, 2002), International Application No. PCT/US01/04340.

Search Report (dated Aug. 23, 2002), International Application No. PCT/US01/13260.

Search Report (dated Oct. 7, 2002), International Application No. PCT/US01/13261.

Search Report, IPER (dated Nov. 13, 2002), International Application No. PCT/US01/04340.

Search Report, IPER (dated Feb. 6, 2002), International Application No. PCT/US01/13261.

Search Report, IPER (dated Jan. 14, 2003), International Application No. PCT/US01/13260.

Sankar, A.U. "A verified sliding window protocol with variable flow control". Proceedings of ACM SIGCOMM conference on Communications architectures & protocols. pp. 84-91, ACM Press, NY, NY 1986.

Shree Murthy et al., "Congestion-Oriented Shortest Multipath Routing", Proceedings of IEEE INFOCOM, 1996, pp. 1028-1036.

W. Stallings, "Cryptography And Network Security", 2nd, Edition, Chapter 13, IP Security, Jun. 8, 1998, pp. 399-440.

Fasbender, A. et al., Variable and Scalable Security: Protection of Location Information in Mobile IP, IEEE VTS, 46th, 1996, 5 pp.

156. Finding Your Way Through the VPN Maze (1999) ("PGP").

WatchGuard Technologies, Inc., WatchGuard LiveSecurity for MSS Powerpoint (Feb. 14 2000) (resubmitted).

WatchGuard Technologies, Inc., MSS Version 2.5, Add-On for WatchGuard SOHO Release Notes (Jul. 21, 2000).

Yuan Dong Feng, "A novel scheme combining interleaving technique with cipher in Rayleigh fading channels," Proceedings of the International Conference on Communication technology, 2:S47-02-1-S47-02-4 (1998).

D.W. Davies and W.L. Price, edited by Tadahiro Uezona, "Network Security", Japan, Nikkei McGraw-Hill, Dec. 5, 1958, First Edition, first copy, p. 102-108.

U.S. Appl. No. 60/134,547 filed May 17, 1999, Victor Sheymov.

U.S. Appl. No. 60/151,563 filed Aug. 31, 1999, Bryan Whittles.

U.S. Appl. No. 09/399,753 filed Sep. 22, 1998, Graig Miller et al.

Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009, *VirnetX Inc. and Science Applications International Corp. v. Microsoft Corporation*.

Appendix A of the Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009.

Concordance Table For the References Cited in Tables on pp. 6-15, 71-80 and 116-124 of the Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009.

I. P. Mockapetris, "DNS Encoding of Network Names and Other Types," Network Working Group, RFC 1101 (Apr. 1989) (RFC1101, DNS SRV).

DNS-related corresponding dated Sep. 7, 1993 to Sep. 20, 1993. (Pre KX, KX Records).

R. Atkinson, "An Internetwork Authentication Architecture," Naval Research Laboratory, Center for High Assurance Computing Systems (Aug. 5, 1993). (Atkinson NRL, KX Records).

## US 6,502,135 C1

Page 4

Henning Schulzrinne, *Personal Mobility For Multimedia Services In The Internet*, Proceedings of the Interactive Distributed Multimedia Systems and Services European Workshop at 143 (1996) (Schulzrinne 96).

Microsoft Corp., *Microsoft Virtual Private Networking: Using Point-to-Point Tunneling Protocol for Low-Cost, Secure, Remote Access Across the Internet* (1996) (printed from 1998 PDC DVD-ROM) (Point to Point, Microsoft Prior Art VPN Technology).

"Safe Surfing: How to Build a Secure World Wide Web Connection," IBM Technical Support Organization, (Mar. 1996). (Safe Surfing, Website Art).

Goldschlag, et al., "Hiding Routing Information," Workshop on Information Hiding, Cambridge, UK (May 1996). (Goldschlag II, Onion Routing).

"IPSec Minutes From Montreal", IPSEC Working Group Meeting Notes, <http://www.sandleman.ca/ipsec/1996/08/msg00018.html> (Jun. 1996). (IPSec Minutes, FreeS/WAN).

J.M. Galvin, "Public Key Distribution with Secure DNS," Proceedings of the Sixth USENIX UNIX Security Symposium, San Jose, California, Jul. 1996. (Galvin, DNSSEC).

J. Gilmore, et al. "Re: Key Management, anyone? (DNS Keying)," IPsec Working Group Mailing List Archives (Aug. 1996). (Gilmore DNS, FreeS/WAN).

H. Orman, et al. "Re: Re: DNS? was Re: Key Management, anyone?" IETF IPsec Working Group Mailing List Archive (Aug. 1996/Sep. 1996). (Orman DNS, FreeS/WAN).

Arnt Gulbrandsen & Paul Vixie, *A DNS RR for specifying the location of services (DNS SRV)*, IETF RFC 2052 (Oct. 1996). (RFC 2052, DNS SRV).

Freier, et al. "The SSL Protocol Version 3.0," Transport Layer Security Working Group (Nov. 18, 1996). (SSL, Underlying Security Technology).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Dec. 2, 1996). (RFC 2543 Internet Draft 1).

M.G. Reed, et al. "Proxies for Anonymous Routing," 12th Annual Computer Security Applications Conference, San Diego, CA, Dec. 9-13, 1996. (Reed, Onion Routing).

Kenneth F. Alden & Edward P. Wobber, *The AltaVista Tunnel: Using the Internet to Extend Corporate Networks*, Digital Technical Journal (1997) (Alden, AltaVista).

Automotive Industry Action Group, "ANX Release 1 Document Publication," AIAG (1997). (AIAG, ANX).

Automotive Industry Action Group, "ANX Release 1 Draft Document Publication," AIAG Publications (1997). (AIAG Release, ANX).

Aventail Corp., "AutoSOCKS v. 2.1 Datasheet," available at <http://www.archive.org/web/19970212013409/www.aventail.com/prod/autosk2ds.html> (1997). (AutoSOCKS, Aventail).

Aventail Corp. "Aventail VPN Data Sheet," available at <http://www.archive.org/web/19970212013043/www.aventail.com/prod/vpndata.html> (1997). (Data Sheet, Aventail).

Aventail Corp., "Directed VPN Vs. Tunnel," available at <http://web.archive.org/web/19970620030312/www.aventail.com/educate/directvpn.html> (1997). (Directed VPN, Aventail).

Aventail Corp., "Managing Corporate Access to the Internet," Aventail AutoSOCKS White Paper available at <http://web.archive.org/web/19970620030312/www.aventail.com/educate/whitepaper/ipmwp.html> (1997). (Corporate Access, Aventail).

Aventail Corp., "Socks Version 5," Aventail Whitepaper, available at <http://web.archive.org/web/19970620030312/www.aventail.com/educate/whitepaper/sockswp.html>

(1997). (Socks, Aventail).

Aventail Corp., "VPN Server V2.0 Administration Guide," (1997). (VPN, Aventail).

Goldschlag, et al. "*Privacy on the Internet*," Naval Research Laboratory, Center for High Assurance Computer Systems (1997). (Goldschlag I, Onion Routing).

Microsoft Corp., *Installing Configuring and Using PPTP with Microsoft Clients and Servers* (1997). (Using PPTP, Microsoft Prior Art VPN Technology).

Microsoft Corp., *IP Security for Microsoft Windows NT Server 5.0* (1997) (printed from 1998 PDC DVD-ROM). (IP Security, Microsoft Prior Art VPN Technology).

Microsoft Corp., *Microsoft Windows NT Active Directory: An Introduction to the Next Generation Directory Services* (1997) (printed from 1998 PDC DVD-ROM). (Directory, Microsoft Prior Art VPN Technology).

Microsoft Corp. *Routing and Remote Access Service for Windows NT Server New Opportunities Today and Looking Ahead* (1997) (printed from 1998 PDC DVD-ROM). (Routing, Microsoft Prior Art VPN Technology).

Microsoft Corp., *Understanding Point-to-Point Tunneling Protocol PPTP* (1997) (printed from 1998 PDC DVD-ROM). (Understanding PPTP, Microsoft Prior Art VPN Technology).

J. Mark Smith et al., *Protecting a Private Network: The AltaVista Firewall*, Digital Technical Journal (1997). (Smith, AltaVista).

Naganand Doraswamy *Implementation of Virtual Private Networks (VPNs) with IPSecurity*, <draft-ietf-ipsec-vpn-00.txt> (Mar. 12, 1997). (Doraswamy).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Mar. 27, 1997). (RFC 2543 Internet Draft 2).

Aventail Corp., "Aventail and Cybersafe to Provide Secure Authentication For Internet and Intranet Communication," Press Release, Apr. 3, 1997. (Secure Authentication, Aventail).

D. Wagner, et al. "Analysis of the SSL 3.0 Protocol," (Apr. 15, 1997). (Analysis, Underlying Security Technologies).

Automotive Industry Action Group, "ANXO Certification Authority Service and Directory Service Definition for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (May 9, 1997). (AIAG Definition, ANX).

Automotive Industry Action Group, "ANXO Certification Process and ANX Registration Process Definition for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (May 9, 1997). (AIAG Certification, ANX).

Aventail Corp. "Aventail Announces the First VPN Solution to Assure Interoperability Across Emerging Security Protocols," Jun. 2, 1997. (First VPN, Aventail).

Syverson, et al. "Private Web Browsing," Naval Research Laboratory, Center for High Assurance Computer Systems (Jun. 2, 1997). (Syverson, Onion Routing).

Bellcore, "Metrics, Criteria, and Measurement Technique Requirements for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (Jun. 16, 1997). (AIAG Requirements, ANX).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jul. 31, 1997). (RFC 2543 Internet Draft 3).

## US 6,502,135 C1

Page 5

- R. Atkinson, "Key Exchange Delegation Record for the DNS," Network Working Group, RFC 2230 (Nov. 1997). (RFC 2230, KX Records).
- M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Nov. 11, 1997). (RFC 2543 Internet Draft 4).
- 1998 Microsoft Professional Developers Conference DVD ("1998 PDC DVD-ROM") (including screenshots captured therefrom and produced as MSFTVX 00018827-00018832). (Conference, Microsoft Prior Art VPN Technology).
- Microsoft Corp., *Virtual Private Networking An Overview* (1998) (printed from 1998 PDC DVD-ROM) (Overview, Microsoft Prior Art VPN Technology).
- Microsoft Corp., *Windows NT 5.0 Beta Has Public Premiere at Seattle Mini-Camp Seminar attendees get first look at the performance and capabilities of Windows NT 5.0* (1998) (available at <http://www.microsoft.com/presspass/features/1998/10-19nt5.mspxpftrue>). (NT Beta, Microsoft Prior Art VPN Technology).
- "What ports does SSL use" available at [stason.org/TULARC/security/ssl-talk/3-4-What-ports-does-ssl-use.html](http://stason.org/TULARC/security/ssl-talk/3-4-What-ports-does-ssl-use.html) (1998). (Ports, DNS SRV).
- Aventail Corp., "Aventail VPN V2.6 Includes Support for More Than Ten Authentication Methods Making Extranet VPN Development Secure and Simple," Press Release, Jan. 19, 1998. (VPN V2.6, Aventail).
- R. G. Moskowitz, "Network Address Translation Issues with IPsec," Internet Draft, Internet Engineering Task Force, Feb. 6, 1998. (Moskowitz).
- H. Schulzrinne, et al., "Internet Telephony Gateway Location," Proceedings of IEEE INFOCOM '98, The Conference on Computer Communications, vol. 2 (Mar. 29-Apr. 2, 1998). (Gateway, Schulzrinne).
- C. Huitema, 45 al. "Simple Gateway Control Protocol," Version 1.0 (May 5, 1998). (SGCP).
- DISA "Secret Internet Protocol Router Network," SIPRNET Program Management Office (D3113) DISN Networks, DISN Transmission Services (May 8, 1998). (DISA, SIPRNET).
- M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (May 14, 1998). (RFC 2543 Internet Draft 5).
- M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jun. 17, 1998). (RFC 2543 Internet Draft 6).
- D. McDonald, et al. "PF\_KEY Management API, Version 2," Network Working Group, RFC 2367 (Jul. 1998). (RFC 2367).
- M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jul. 16, 1998). (RFC 2543 Internet Draft 7).
- M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Aug. 7, 1998). (RFC 2543 Internet Draft 8).
- Microsoft Corp., *Company Focuses on Quality and Customer Feedback* (Aug. 18, 1998). (Focus, Microsoft Prior Art VPN Technology).
- M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Sep 18, 1998). (RFC 2543 Internet Draft 9).
- Atkinson, et al. "Security Architecture for the Internet Protocol," Network Working Group, RFC 2401 (Nov. 1998). (RFC 2401, Underlying Security Technologies).
- M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Nov. 12, 1998). (RFC 2543 Internet Draft 10) 9.
- Donald Eastlake, *Domain Name System Security Extensions*, IETF-DNS Security Working Group (Dec. 1998). (DNS-SEC-7).
- M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Dec. 15, 1998). (RFC 2543 Internet Draft 11).
- Aventail Corp., "Aventail Connect 3.1/2.6 Administrator's Guide," (1999). (Aventail Administrator 3.1, Aventail).
- Aventail Corp., "Aventail Connect 3.1/2.6 User's Guide," (1999). (Aventail Administrator 3.1, Aventail).
- Aventail Corp., "Aventail ExtraWeb Server v3.2 Administrator's Guide," (1999). (Aventail ExtraWeb 3.2, Aventail).
- Kaufman et al, "Implementing IPsec," (Copyright 1999). (Implementing IPSEC, VPN References).
- Network Solutions, Inc. "Enabling SSL," NSI Registry (1999). (Enabling SSL, Underlying Security Technologies).
- Check Point Software Technologies Ltd. (1999) (Check Point, Checkpoint FW).
- Arnt Gulbrandsen & Paul Vixie, *A DNS RR for specifying the location of services (DNS SRV)*, <draft-ietf-dnsind-frc2052bis-02.txt> (Jan. 1999). (Gulbrandsen 99, DNS SRV).
- C. Scott, et al. *Virtual Private Networks*, O'Reilly and Associates, Inc., 2nd ed. (Jan. 1999). (Scott VPNs).
- M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jan. 15, 1999). (RFC 2543 Internet Draft 12).
- Goldschlag, et al., "Onion Routing for Anonymous and Private Internet Connections," Naval Research Laboratory, Center for High Assurance Computer Systems (Jan. 28, 1999). (Goldschlag III, Onion Routing).
- H. Schulzrinne, "Internet Telephony: architecture and protocols—an IETF perspective," Computer Networks, vol. 31, No. 3 (Feb. 1999). (Telephony, Schulzrinne).
- M. Handley, et al, "SIP: Session Initiation Protocol," Network Working Group, RFC 2543 and Internet Drafts (Dec. 1996-Mar. 1999). (Handley, RFC 2543).
- FreeS/WAN Project, *Linux FreeS/WAN Compatibility Guide* (Mar. 4, 1999). (FreeS/WAN Compatibility Guide, FreeS/WAN).
- Telcordia Technologies, "ANX Release 1 Document Corrections," AIAG (May 11, 1999). (Telcordia, ANX).
- Ken Hornstein & Jeffrey Altman, *Distributing Kerberos KDC and Realm Information with DNS* <draft-ietf-cat-krb-dns-locate-00.txt> (Jun. 21, 1999). (Hornstein, DNS SRV).
- Bhattacharya et al. "An LDAP Schema for Configuration and Administration of IPsec Based Virtual Private Networks (VPNs)," IETF Internet Draft (Oct. 1999). (Bhattacharya LDAP VPN).
- B. Patel, et al, "DHCP Configuration of IPSEC Tunnel Mode," IPSEC Working Group, Internet Draft 02 (Oct. 15, 1999). (Patel).
- Goncalves, et al. Check Point FireWall—1 Administration Guide, McGraw-Hill Companies (2000). (Goncalves, Checkpoint FW).
- "Building a Microsoft VPN: A Comprehensive Collection of Microsoft Resources," FirstVPN, (Jan. 2000). (FirstVPN Microsoft).

## US 6,502,135 C1

Page 6

Gulbrandsen, Vixie & Esibov, *A DNS RR for specifying the location of services (DNS SRV)*, IETF RFC 2782 (Feb. 2000). (RFC 2782, DNS SRV).

Mitre Organization, "Technical Description," Collaborative Operations in Joint Expeditionary Force Experiment (JEFX) 99 (Feb. 2000). (Mitre, SIPRNET).

H. Schulzrinne, et al. "Application-Layer Mobility Using SIP," *Mobile Computing and Communications Review*, vol. 4, No. 3, pp. 47-57 (Jul. 2000). (Application, SIP).

Kindred et al., "Dynamic VPN Communities: Implementation and Experience," DARPA Information Survivability Conference and Exposition II (Jun. 2001). (DARPA, VPN Systems).

ANX 101: Basic ANX Service Outline. (Outline, ANX).

ANX 201: Advanced ANX Service. (Advanced, ANX).

Appendix A: Certificate Profile for ANX IPsec Certificates. (Appendix, ANX).

Assured Digital Products. (Assured Digital).

Aventail Corp., "Aventail AutoSOCKS the Client Key to Network Security," Aventail Corporation White Paper. (Network Security, Aventail).

Cindy Moran, "DISN Data Networks: Secret Internet Protocol Router Network (SIPRNet)." (Moran, SIPRNET).

Data Fellows F-Secure VPN+ (F-Secure VPN+).

Interim Operational Systems Doctrine for the Remote Access Security Program (RASP) Secret Dial-In Solution. (RASP, SIPRNET).

Onion Routing, "Investigation of Route Selection Algorithms," available at <http://www.onion-router.net/Archives/Route/Index.html>. (Route Selection, Onion Routing).

Secure Computing, "Buttlet-Proofing an Army Net," Washington Technology. (Secure, SIPRNET).

Sparta "Dynamic Virtual Private Network," (Sparta, VPN Systems).

Standard Operation Procedure for Using the 1910 Secure Modems. (Standard, SIPRNET).

Publically available emails relating to FreeS/WAN (MSFTVX00018833-MSFTVX00019206). (FreeS/WAN emails, FreeS/WAN).

Kaufman et al., "implementing IPsec," (Copyright 1999) (Implementing IPsec).

Network Associates *Gauntlet Firewall For Unix User's Guide Version 5.0* (1999). (Gauntlet User's Guide—Unix, Firewall Products).

Network Associates *Gauntlet Firewall For Windows NT Getting Started Guide Version 5.0* (1999) (Gauntlet Getting Started Guide—NT, Firewall Products).

Network Associates *Gauntlet Firewall For Unix Getting Started Guide Version 5.0* (1999) (Gauntlet Unix Getting Started Guide, Firewall Products).

Network Associates *Release Notes Gauntlet Firewall for Unix 5.0* (Mar. 19, 1999) (Gauntlet Unix Release Notes, Firewall Products).

Network Associates *Gauntlet Firewall For Windows NT Administrator's Guide Version 5.0* (1999) (Gauntlet NT Administrator's Guide, Firewall Products).

Trusted Information Systems, Inc. *Gauntlet Internet Firewall Firewall-to-Firewall Encryption Guide Version 3.1* (1996) (Gauntlet Firewall-to-Firewall, Firewall Products).

Network Associates *Gauntlet Firewall Global Virtual Private Network User's Guide for Windows NT Version 5.0* (1999) (Gauntlet NT GVPN, GVPN).

Network Associates *Gauntlet Firewall For Unix Global Virtual Private Network User's Guide Version 5.0* (1999) (Gauntlet Unix GVPN, GVPN).

Dan Sterne *Dynamic Virtual Private Networks* (May 23, 2000) (Sterne DVPN, DVPN).

Darrell Kindred *Dynamic Virtual Private Networks (DVPN)* (Dec. 21, 1999) (Kindred DVPN, DVPN).

Dan Sterne et al. *TIS Dynamic Security Perimeter Research Project Demonstration* (Mar. 9, 1998) (Dynamic Security Perimeter, DVPN).

Darrell Kindred *Dynamic Virtual Private Networks Capability Description* (Jan. 5, 2000) (Kindred DVPN Capability, DVPN) 11.

Oct. 7, and 28 1997 email from Domenic J. Turchi Jr. (SPARTA00001712-1714, 1808-1811) (Turchi DVPN email, DVPN).

James Just & Dan Sterne *Security Quickstart Task Update* (Feb. 5, 1997) (Security Quickstart, DVPN).

Virtual Private Network Demonstration dated Mar. 21, 1998 (SPARTA00001844-54) (DVPN Demonstration, DVPN).

GTE Internetworking & BBN Technologies *DARPA Information Assurance Program Integrated Feasibility Demonstration* (IFD) 1.1 Plan (Mar. 10, 1998) (IFD 1.1, DVPN).

Microsoft Corp. Windows NT Server Product Documentation: Administration Guide—Connection Point Services, available at <http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/cpsops.mspx> (Connection Point Services) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit).

Microsoft Corp. Windows NT Server Product Documentation: Administration Kit Guide—Connection Manager, available at <http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/cmak.mspx> (Connection Manager) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

Microsoft Corp. Autodial Heuristics, available at <http://support.microsoft.com/kb/164249> (Autodial Heuristics) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

Microsoft Corp., Cariplo: Distributed Component Object Model, (1996) available at [http://msdn2.microsoft.com/en-us/library/ms809332\(printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms809332(printer).aspx) (Cariplo I).

Marc Levy, COM Internet Services (Apr. 23, 1999), available at [http://msdn2.microsoft.com/en-us/library/ms809302\(printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms809302(printer).aspx) (Levy).

Markus Horstmann and Mary Kirtland, DCOM Architecture (Jul. 23, 1997), available at [http://msdn2.microsoft.com/en-us/library/ms809311\(printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms809311(printer).aspx) (Horstmann).

Microsoft Corp., DCOM: A Business Overview (Apr. 1997), available at [http://msdn2.microsoft.com/en-us/library/ms809320\(printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms809320(printer).aspx) (DCOM Business Overview I).

Microsoft Corp., DCOM Technical Overview (Nov. 1996), available at [http://msdn2.microsoft.com/en-us/library/ms809340\(printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms809340(printer).aspx) (DCOM Technical Overview I).

Microsoft Corp., DCOM Architecture White Paper (1998) available in PDC DVD-ROM (DCOM Architecture).

## US 6,502,135 C1

Page 7

Microsoft Corp., DCOM—The Distributed Component Object Model, A Business Overview White Paper (Microsoft 1997) available in PDC DVD-ROM (DCOM Business Overview II).

Microsoft Corp., DCOM—Cariplo Home Banking Over The Internet White Paper (Microsoft 1996) available in PDC DVD-ROM (Cariplo II).

Microsoft Corp., DCOM Solutions in Action White Paper (Microsoft 1996) available in PDC DVD-ROM (DCOM Solutions in Action).

Microsoft Corp., DCOM Technical Overview White Paper (Microsoft 1996) available in PDC DVD-ROM (DCOM Technical Overview II).

125. Scott Suhay & Glenn Wood, DNS and Microsoft Windows NT 4.0 (1996) available at [http://msdn2.microsoft.com/en-us/library/ms810277\(printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms810277(printer).aspx) (Suhay).

126. Aaron Skonnard, *Essential Winlnet* 313–423 (Addison Wesley Longman 1998) (Essential Winlnet).

Microsoft Corp. Installing, Configuring, and Using PPTP with Microsoft Clients and Servers, (1998) available at [http://msdn2.microsoft.com/enus/library/ms811078\(printer\).aspx](http://msdn2.microsoft.com/enus/library/ms811078(printer).aspx) (Using PPTP).

Microsoft Corp. Internet Connection Services for MS RAS, Standard Edition, <http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/bcgstart.mspix> (Internet Connection Services I).

Microsoft Corp. Internet Connection Services for RAS, Commercial Edition, available at <http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/bcgstrtc.mspix> (Internet Connection Services II).

Microsoft Corp., Internet Explorer 5 Corporate Deployment Guide—Appendix B: Enabling Connections with the Connection Manager Administration Kit, available at <http://www.microsoft.com/technet/prodtechnol/ie/deploy/deploy5/appendb.mspix> (IE5 Corporate Development).

Mark Minasi, *Mastering Windows NT Server 4* 1359–1442 (6th ed., Jan. 15, 1999) (Mastering Windows NT Server).

*Hands On, Self-Paced Training for Supporting Verion 4.0* 371–473 (Microsoft Press 1998) (Hands On).

Microsoft Corp., MS Point-to-Point Tunneling Protocol (Windows NT 4.0), available at <http://www.microsoft.com/technet/archive/winntas/maintain/featusability/pptpwp3.mspix> (MS PPTP).

Kenneth Gregg, et al., *Microsoft Windows NT Server Administrator's Bible* 173–206, 883–911, 974–1076 (IDG Books Worldwide 1999) (Gregg).

Microsoft Corp., Remote Access (Windows), available at [http://msdn2.microsoft.com/en-us/library/bb545687\(VS.85,printer\).aspx](http://msdn2.microsoft.com/en-us/library/bb545687(VS.85,printer).aspx) (Remote Access).

Microsoft Corp., Understanding PPTP (Windows NT 4.0), available at <http://www.microsoft.com/technet/archive/winntas/plan/pptpudst.mspix> (Understanding PPTP NT 4) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

Microsoft Corp., Windows NT 4.0: Virtual Private Networking, available at <http://www.microsoft.com/technet/archive/winntas/deploy/confeat/vpntwk.mspix> (NT4 VPN) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

Anthony Northrup, *NT Network Plumbing: Routers, Proxies, and Web Services* 299–399 (IDG Books Worldwide 1998) (Network Plumbing).

Microsoft Corp., Chapter 1—Introduction to Windows NT Routing with Routing and Remote Access Service, Available at <http://www.microsoft.com/technet/archive/winntas/proddocs/ras40/rasch01.mspix> (Intro to RRAS) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.) 13.

Microsoft Corp., Windows NT Server Product Documentation: Chapter 5—Planning for Large-Scale Configurations, available at <http://www.microsoft.com/technet/archive/winntas/proddocs/ras40/rasch05.mspix> (Large-Scale Configurations) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

F-Secure, F-Secure Evaluation Kit (May 1999) (FSECURE 00000003) (Evaluation Kit 3).

F-Secure, F-Secure NameSurfer (May 1999) (FSECURE 00000003) (NameSurfer 3).

F-Secure, F-Secure VPN Administrator's Guide (May 1999) (from FSECURE 00000003) (F-Secure VPN 3).

F-Secure, F-Secure SSH User's & Administrator's Guide (May 1999) (from FSECURE 00000003) (SSH Guide 3).

F-Secure, *F-Secure SSH 2.0 for Windows NT and 95* (May 1999) (from FSECURE 00000003) (SSH 2.0 Guide 3).

F-Secure, *F-Secure VPN+ Administrator's Guide* (May 1999) (from FSECURE 00000003) (VPN+ Guide 3).

F-Secure, *F-Secure VPN+ 4.1* (1999) (from FSECURE 00000006) (VPN+ 4.1 Guide 6).

F-Secure, *F-Secure SSH* (1996) (from FSECURE 00000006) (F-Secure SSH 6).

F-Secure, *F-Secure SSH 2.0 for Windows NT and 95* (1998) (from FSECURE 00000006) (F-Secure SSH 2.0 Guide 6).

F-Secure, *F-Secure Evaluation Kit* (Sep. 1998) (FSECURE 00000009) (Evaluation Kit 9).

F-Secure, *F-Secure SSH User's & Administrator's Guide* (Sep. 1998) (from FSECURE 00000009) (SSH Guide 9).

F-Secure, *F-Secure SSH 2.0 for Windows NT and 95* (Sep. 1998) (from FSECURE 00000009) (F-Secure SSH 2.0 Guide 9).

F-Secure, *F-Secure VPN+* (Sep. 1998) (from FSECURE 00000009) (VPN+ Guide 9).

F-Secure, *F-Secure Management Tools Administrator's Guide* (1999) (from FSECURE 00000003) (F-Secure Management Tools).

F-Secure, *F-Secure Desktop, User's Guide* (1997) (from FSECURE 00000009) (F-Secure Desktop User's Guide).

SafeNet, Inc., *VPN Policy Manager* (Jan. 2000) (VPN Policy Manager).

F-Secure, *F-Secure VPN+ for Windows NT 4.0* (1998) (from FSECURE 00000009) (F-Secure VPN+).

IRE, Inc., *SafeNet/Soft-PK Version 4* (Mar. 28, 2000) (Soft-PK Version 4).

IRE/SafeNet Inc., *VPN Technologies Overview* (Mar. 28, 2000) (Safenet VPN Overview).

IRE, Inc., *SafeNet/Security Center Technical Reference Addendum* (Jun. 22, 1999) (Safenet Addendum).

IRE, Inc., *System Description for VPN Policy Manager and SafeNet/SoftPK* (Mar. 30, 2000) (VPN Policy Manager System Description).

## US 6,502,135 C1

Page 8

- IRE, Inc., *About SafeNet/VPN Policy Manager* (1999) (About Safenet VPN Policy Manager).
- IRE, Inc., *SafeNet/VPN Policy Manager Quick Start Guide Version 1* (1999) (SafeNet VPN Policy Manager).
- Trusted Information Systems, Inc., *Gauntlet Internet Firewall, Firewall Product Functional Summary* (Jul. 22, 1996) (Gauntlet Functional Summary).
- Trusted Information Systems, Inc., *Running the Gauntlet Internet Firewall, An Administrator's Guide to Gauntlet Version 3.0* (May 31, 1995) (Running the Gauntlet Internet Firewall).
- Ted Harwood, *Windows NT Terminal Server and Citrix Metaframe* (New Riders 1999) (Windows NT Harwood) 79.
- Todd W. Mathers and Shawn P. Genoway, *Windows NT Thing Client Solutions: Implementing Terminal Server and Citrix MetaFrame* (Macmillan Technical Publishing 1999) (Windows NT Mathers).
- Bernard Aboba et al., *Securing L2TP using IPSEC* (Feb. 2, 1999).
156. *Finding Your Way Through the VPN Maze* (1999) ("PGP").
- Linux FreeS/WAN Overview (1999) (Linux FreeS/WAN Overview).
- TimeStep, *The Business Case for Secure VPNs* (1998) ("TimeStep").
- WatchGuard Technologies, Inc., *WatchGuard Firebox System Powerpoint* (2000).
- WatchGuard Technologies, Inc., *MSS Firewall Specifications* (1999).
- WatchGuard Technologies, Inc., *Request for Information, Security Services* (2000).
- WatchGuard Technologies, Inc., *Protecting the Internet Distributed Enterprise, White Paper* (Feb. 2000).
- WatchGuard Technologies, Inc., *WatchGuard LiveSecurity for MSS Powerpoint* (Feb. 14, 2000).
- WatchGuard Technologies, Inc., *MSS Version 2.5, Add-On for WatchGuard SOHO Release Notes* (Jul. 21, 2000).
- Air Force Research Laboratory, *Statement of Work for Information Assurance System Architecture and Integration, PR No. N-8-6106* (Contract No. F30602-98-C-0012) (Jan. 29, 1998).
- GTE Internetworking & BBN Technologies *DARPA Information Assurance Program Integrated Feasibility Demonstration (IFD) 1.2 Report, Rev. 1.0* (Sep. 21, 1998).
- BBN Information Assurance Contract, *TIS Labs Monthly Status Report* (Mar. 16-Apr. 30, 1998).
- DARPA, *Dynamic Virtual Private Network (VPN) Powerpoint*.
- GTE Internetworking, *Contractor's Program Progress Report* (Mar. 16-Apr. 30, 1998).
- Darrell Kindred, *Dynamic Virtual Private Networks (DVPN) Countermeasure Characterization* (Jan. 30, 2001).
- Virtual Private Networking Countermeasure Characterization* (Mar. 30, 2000).
- Virtual Private Network Demonstration* (Mar. 21, 1998).
- Information Assurance/NAI Labs, *Dynamic Virtual Private Networks (VPNs) and Integrated Security Management* (2000).
- Information Assurance/NAI Labs, *Create/Add DVPN Enclave* (2000).
- NAI Labs, *IFE 3.1 Integration Demo* (2000).
- Information Assurance, *Science Fair Agenda* (2000).
- Darrell Kindred et al., *Proposed Threads for IFE 3.1* (Jan. 13, 2000).
- IFE 3.1 Technology Dependencies* (2000).
- IFE 3.1 Topology* (Feb. 9, 2000).
- Information Assurance, *Information Assurance Integration: IFE 3.1, Hypothesis & Thread Development* (Jan. 10-11, 2000).
- Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation* (2000).
- Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation v.2* (2000).
- Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation v.3* (2000).
- T. Braun et al., *Virtual Private Network Architecture*, Charging and Accounting Technology for the Internet (Aug. 1, 1999) (VPNA).
- Network Associates Products—*PGP Total Network Security Suite, Dynamic Virtual Private Networks* (1999).
- Microsoft Corporation, *Microsoft Proxy Server 2.0* (1997) (Proxy Server 2.0, Microsoft Prior Art VPN Technology).
- David Johnson et al., *A Guide To Microsoft Proxy Server 2.0* (1999) (Johnson, Microsoft Prior Art VPN Technology).
- Microsoft Corporation, *Setting Server Parameters* (1997) (Proxy Server 2.0 CD labeled MSFTVX00157288) (Setting Server Parameters, Microsoft Prior Art VPN Technology).
- Kevin Schuler, *Microsoft Proxy Server 2* (1998) (Schuler, Microsoft Prior Art VPN Technology).
- Erik Rozell et al., *MCSE Proxy Server 2 Study Guide* (1998) (Rozell, Microsoft Prior Art VPN Technology).
- M. Shane Stigler & Mark A. Linsenhardt, *IIS 4 and Proxy Server 2* (1999) (Stigler, Microsoft Prior Art VPN Technology).
- David G. Schaer, *MCSE Test Success: Proxy Server 2* (1998) (Schaer, Microsoft Prior Art VPN Technology).
- John Savill, *The Windows NT and Windows 2000 Answer Book* (1999) (Savill, Microsoft Prior Art VPN Technology).
- Network Associates *Gauntlet Firewall Global Virtual Private Network User's Guide for Windows NT Version 5.0* (1999) (Gauntlet NT GVPN, GVPN).
- Network Associates *Gauntlet Firewall For UNIX Global Virtual Private Network User's Guide Version 5.0* (1999) (Gauntlet Unix GVPN, GVPN).
- File History for U.S. Appl. No. 09/653,201, Applicant(s): Whittle Bryan, et al., filed Aug. 31, 2000.
- AutoSOCKS v2.1*, Datasheet, <http://web.archive.org/web/19970212013409/www.aventail.com/prod/autoskds.html>.
- Ran Atkinson, *Use of DNS to Distribute Keys*, Sep. 7, 1993, <http://ops.ietf.org/lists/namedroppers/namedroppers.199x/msg00945.html>.
- FirstVPN Enterprise Networks, Overview.
- Chapter 1: Introduction to Firewall Technology, Administration Guide: Dec. 19, 2007, [http://www.books24x7.com/book/id\\_762/viewer\\_r.asp?bookid=762&chunked=41065062](http://www.books24x7.com/book/id_762/viewer_r.asp?bookid=762&chunked=41065062).
- The TLS Protocol Version 1.0; Jan. 1999; p. 65 of 71.
- Elizabeth D. Zwicky, et al., *Building Internet Firewalls*, 2nd Ed.
- Virtual Private Networks—Assured Digital Incorporated—ADI 4500; <http://web.archive.org/web/1990224050035/www.assured-digital.com/products/prodvpn/adia4500.htm>.
- Accessware—The Third Wave in Network Security, Conclave from Internet Dynamics; <http://web.archive.org/web/11980210013830/interdyn.com/Accessware.html>.
- Extended System Press Release, Sep. 2, 1997; *Extended VPN Uses The Internet to Create Virtual Private Networks*, [www.extendedsystems.com](http://www.extendedsystems.com).

## US 6,502,135 C1

Page 9

Socks Version 5; Executive Summary; <http://web.archive.org/web/199970620031945/www.aventail.com/educate/whitepaper/sockswp.html>.

Internet Dynamics First to Ship Integrated Security Solutions for Enterprise Intranets and Extranets; Sep. 15, 1997; <http://web.archive.org/web/19980210014150/interdyn.com>. E-mails from various individuals to Linux IPsec re:DNS-LDAP Splicing.

Microsoft Corporation's Fifth Amended Invalidity Contentions dated Sep. 18, 2009, *VirnetX Inc. and Science Applications International Corp. v. Microsoft Corporation* and invalidity claim charts for U.S. Patent Nos. 7,188,180 and 6,839,759.

The IPSEC Protocol as described in Atkinson, et al., "Security Architecture for the Internet Protocol," Networking Working Group, RFC 2401 (Nov. 1998) ("RFC 2401"); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).

S. Kent and R. Atkinson, "IP Authentication Header," RFC 2402 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).

C. Madson and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH," RFC 2403 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).

C. Madson and R. Glenn, "The Use HMAC-SHA-1-96 with ESP and AH," RFC 2404 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).

C. Madson and N. Doraswamy, "The ESP DES-CBC Cipher Algorithm With Explicit IV," RFC 2405 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).

S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," RFC 2406 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).

Derrell Piper, "The Internet IP Security Domain of Interpretation for ISAKMP," RFC 2407 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).

Douglas Maughan, et al., "Internet Security Association and Key Management Protocol (ISAKMP)," RFC 2408 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).

D. Harkins and D. Carrell, "The Internet Key Exchange (IKE)," RFC 2409 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).

R. Glenn and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec," RFC 2410 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).

R. Thayer, et al., "IP Security Document Roadmap," RFC 2411 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).

Hilarie K. Orman, "The Oakley Key Determination Protocol," RFC 2412 (Nov. 1998) in combination with J.M. Galvin, "Public Key Distribution with Secure DNS," Proceedings of the Sixth USENIX UNIX Security Symposium, San Jose California (Jul. 1996) ("Galvin").

David Kosiur, "Building and Managing Virtual Private Networks" (1998).

P. Mockapetris, "Domain Names—Implementation and Specification," Network Working Group, RFC 1035 (Nov. 1987).

Request for Inter Partes Reexamination of Patent No. 7,188,180, dated Nov. 25, 2009.

Exhibit 2 "Aventail Connect v3.1/v2.6 Administrator's Guide", 120 pages, 1996–1999.

Exhibit 3A, "Gauntlet Firewall for Windows", pp. 1–137, 1998–1999.

Exhibit 3B, "Gauntlet Firewall for Windows", pp. 138–275, 1998–1999.

Exhibit 4, "Kosiur", Building and Managing VPNs, pp. 1–396, 1998.

Exhibit 5, Building a Microsoft VPN; A comprehensive Collection of Microsoft Resources, pp. 1–216.

Exhibit 6, Windows NT Server, Virtual Private Network; An Overview, pp. 1–26, 1998.

Exhibit 7, "Networking Working Group Request for Comments: 1035" pp. 1–56, 1987.

US 6,502,135 C1

**1**  
**INTER PARTES**  
**REEXAMINATION CERTIFICATE**  
**ISSUED UNDER 35 U.S.C. 316**

THE PATENT IS HEREBY AMENDED AS  
INDICATED BELOW.

Matter enclosed in heavy brackets [ ] appeared in the patent, but has been deleted and is no longer a part of the patent; matter printed in italics indicates additions made to the patent.

AS A RESULT OF REEXAMINATION, IT HAS BEEN DETERMINED THAT:

The patentability of claims 1-10 and 12 is confirmed.

New claim 18 is added and determined to be patentable.

Claims 11 and 13-17 were not reexamined.

18. A method of transparently creating a virtual private network (VPN) between a client computer and a target computer, comprising the steps of:

**2**

(1) generating from the client computer a Domain Name Service (DNS) request that requests an IP address corresponding to a domain name associated with the target computer;

(2) determining whether the DNS request transmitted in step (1) is requesting access to a secure web site; and

(3) in response to determining that the DNS request in step (2) is requesting access to a secure target web site, automatically initiating the VPN between the client computer and the target computer, wherein:

steps (2) and (3) are performed at a DNS server separate from the client computer, and step (3) comprises the step of, prior to automatically initiating the VPN between the client computer and the target computer, determining whether the client computer is authorized to resolve addresses of non secure target computers and, if not so authorized, returning an error from the DNS request.

\* \* \* \* \*



(12) **United States Patent**  
**Larson et al.**

(10) **Patent No.:** **US 7,418,504 B2**  
(45) **Date of Patent:** **Aug. 26, 2008**

(54) **AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES**

(58) **Field of Classification Search** ..... 709/226, 709/221; 713/201  
See application file for complete search history.

(56) **References Cited**

(75) Inventors: **Victor Larson**, Fairfax, VA (US);  
**Robert Dunham Short, III**, Leesburg, VA (US); **Edmund Colby Munger**,  
Crownsville, MD (US); **Michael Williamson**, South Riding, VA (US)

U.S. PATENT DOCUMENTS

4,933,846	A	6/1990	Humphrey et al.
4,988,990	A	1/1991	Warrior
5,164,988	A	11/1992	Matyas et al.
5,276,735	A	1/1994	Boebert et al.
5,311,593	A	5/1994	Carmi

(73) Assignee: **VirnetX, Inc.**, Scotts Valley, CA (US)

(Continued)

FOREIGN PATENT DOCUMENTS

(\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 646 days.

DE 199 24 575 12/1999

(Continued)

OTHER PUBLICATIONS

(21) Appl. No.: **10/714,849**

Laurie Wells (Lancasterbibemail MSN Com); "Subject: Security Icon" Usenet Newsgroup, Oct. 19, 1998, XP002200606.

(22) Filed: **Nov. 18, 2003**

(Continued)

(65) **Prior Publication Data**

US 2004/0098485 A1 May 20, 2004

Primary Examiner—Krisna Lim

(74) Attorney, Agent, or Firm—McDermott Will & Emery, LLP

**Related U.S. Application Data**

(63) Continuation of application No. 09/558,210, filed on Apr. 26, 2000, now abandoned, which is a continuation-in-part of application No. 09/504,783, filed on Feb. 15, 2000, now Pat. No. 6,502,135, which is a continuation-in-part of application No. 09/429,643, filed on Oct. 29, 1999, now Pat. No. 7,010,604.

(57) **ABSTRACT**

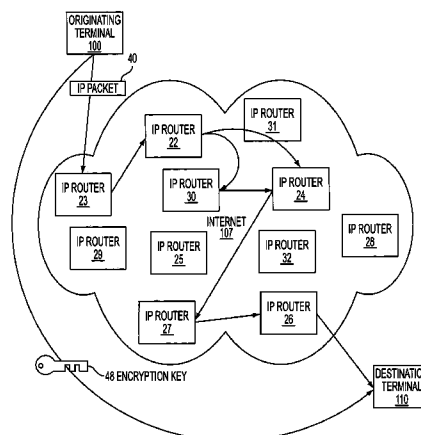
A secure domain name service for a computer network is disclosed that includes a portal connected to a computer network, such as the Internet, and a domain name database connected to the computer network through the portal. The portal authenticates a query for a secure computer network address, and the domain name database stores secure computer network addresses for the computer network. Each secure computer network address is based on a non-standard top-level domain name, such as .scom, .sorg, .snet, .snet, .sedu, .smil and .sint.

(60) Provisional application No. 60/137,704, filed on Jun. 7, 1999, provisional application No. 60/106,261, filed on Oct. 30, 1998.

(51) **Int. Cl.**  
**G06F 15/173** (2006.01)

(52) **U.S. Cl.** ..... 709/226

**60 Claims, 40 Drawing Sheets**



## US 7,418,504 B2

Page 2

## U.S. PATENT DOCUMENTS

5,329,521 A	7/1994	Walsh et al.	6,557,037 B1	4/2003	Provino ..... 709/227
5,341,426 A	8/1994	Barney et al.	6,571,296 B1	5/2003	Dillon
5,367,643 A	11/1994	Chang et al.	6,571,338 B1	5/2003	Shaio et al.
5,559,883 A	9/1996	Williams	6,581,166 B1	6/2003	Hirst et al.
5,561,669 A	10/1996	Lenney et al.	6,606,708 B1	8/2003	Devine et al.
5,588,060 A	12/1996	Aziz	6,618,761 B2	9/2003	Munger et al.
5,625,626 A	4/1997	Umekita	6,671,702 B2	12/2003	Kruglikov et al.
5,654,695 A	8/1997	Olnowich et al.	6,687,551 B2	2/2004	Steindl
5,682,480 A	10/1997	Nakagawa	6,714,970 B1	3/2004	Fiveash et al.
5,689,566 A	11/1997	Nguyen	6,717,949 B1	4/2004	Boden et al.
5,740,375 A	4/1998	Dunne et al.	6,751,738 B2	6/2004	Wesinger, Jr. et al.
5,774,660 A	6/1998	Brendel et al.	6,760,766 B1	7/2004	Sahlqvist
5,787,172 A	7/1998	Arnold	6,826,616 B2	11/2004	Larson et al.
5,790,548 A	8/1998	Sistanizadeh et al.	6,839,759 B2	1/2005	Larson et al.
5,796,942 A	8/1998	Esbensen	7,010,604 B1	3/2006	Munger et al.
5,805,801 A	9/1998	Holloway et al.	7,133,930 B2	11/2006	Munger et al.
5,842,040 A	11/1998	Hughes et al.	7,188,180 B2	3/2007	Larson et al.
5,845,091 A	12/1998	Dunne et al.	7,197,563 B2	3/2007	Sheymov et al.
5,867,650 A	2/1999	Osterman	2002/0004898 A1	1/2002	Droge
5,870,610 A	2/1999	Beyda et al.	2003/0196122 A1	10/2003	Wesinger, Jr. et al.
5,878,231 A	3/1999	Baehr et al.	2005/0055306 A1	3/2005	Miller et al.
5,892,903 A	4/1999	Klaus	2006/0059337 A1	3/2006	Polyhonen et al.
5,898,830 A	4/1999	Wesinger, Jr. et al.	FOREIGN PATENT DOCUMENTS		
5,905,859 A	5/1999	Holloway et al.	DE	199 24 575 A1	12/1999
5,918,019 A	6/1999	Valencia	EP	0 814 589	12/1997
5,996,016 A	11/1999	Thalheimer et al.	EP	0 814 589 A	12/1997
6,006,259 A	12/1999	Adelman et al.	EP	0 838 930	4/1998
6,006,272 A	12/1999	Aravamudan et al.	EP	0 838 930 A	4/1998
6,016,318 A	1/2000	Tomoike	EP	0 838 930 A2	4/1998
6,016,512 A	1/2000	Huitema	EP	836306 A1	4/1998
6,041,342 A	3/2000	Yamaguchi	EP	0 858 189	8/1998
6,052,788 A	4/2000	Wesinger, Jr. et al.	GB	2 317 792	4/1998
6,055,574 A	4/2000	Smorodinsky et al.	GB	2 317 792 A	4/1998
6,061,736 A	5/2000	Rochberger et al.	GB	2 334 181 A	8/1999
6,079,020 A	6/2000	Liu	WO	9827783 A	6/1998
6,092,200 A	7/2000	Muniyappa et al.	WO	WO 98/27783	6/1998
6,101,182 A	8/2000	Sistanizadeh et al.	WO	WO 98 55930	12/1998
6,119,171 A	9/2000	Alkhatib	WO	WO 98 59470	12/1998
6,119,234 A	9/2000	Aziz et al.	WO	WO 99 38081	7/1999
6,147,976 A	11/2000	Shand et al.	WO	WO 99 48303	9/1999
6,157,957 A	12/2000	Berthaud	WO	WO 00/17775	3/2000
6,158,011 A	12/2000	Chen et al.	WO	WO 00/70458	11/2000
6,168,409 B1	1/2001	Fare	WO	WO 01 50688	7/2001
6,175,867 B1	1/2001	Taghadoss	OTHER PUBLICATIONS		
6,178,409 B1	1/2001	Weber et al.	Davila J et al., "Implementatin of Virtual Private Networks at the Transport Layer", Information Security, Second International Workshop, ISW'99. Proceedings (Lecture Springer-Verlag Berlin, Germany, [Online] 1999, pp. 85-102, XP002399276, ISBN 3-540-66695-B, retrieved from the Internet: URL: <a href="http://www.springerlink.com/content/4uac0tb0heccma89/fulltext.pdf">http://www.springerlink.com/content/4uac0tb0heccma89/fulltext.pdf</a> -(Abstract).		
6,178,505 B1	1/2001	Schneider et al.	Donald E. Eastlake, III, "Domain Name System Security Extensions", Internet Draft, Apr. 1998.		
6,179,102 B1	1/2001	Weber et al.	P. Srisuresh, et al., "DNS Extensions to Network Address Translators", Internet Draft, Jul. 1998.		
6,222,842 B1	4/2001	Sasyan et al.	D.B. Chapman, et al., "Building Internet Firewalls, chapters 8 and 10 (parts)", pp. 278-296 and pp. 351-375.		
6,226,751 B1	5/2001	Arrow et al.	Search Report (dated Jun. 18, 2002), International Application No. PCT/US01/13260.		
6,233,618 B1	5/2001	Shannon	Search Report (dated Jun. 28, 2002), International Application No. PCT/US01/13261.		
6,243,360 B1	6/2001	Basilico	Donald E. Eastlake, "Domain Name System Security Extensions", DNS Security Working Group. Apr. 1998, 51 pages.		
6,243,749 B1	6/2001	Sitaraman et al.	D. B. Chapman et al., "Building Internet Firewalls", Nov. 1995, pp. 278-297 and pp. 351-375.		
6,243,754 B1	6/2001	Guerin et al.	P. Srisuresh et al., "DNS extensions to Network Address Translators", Jul. 1998, 27 pages.		
6,256,671 B1	7/2001	Strentzsch et al.	Laurie Wells, "Security Icon", Oct. 19, 1998, 1 page.		
6,263,445 B1	7/2001	Blumenau	W. Stallings, "Cryptography And Network Security", 2 <sup>nd</sup> Edition, Chapter 13, IP Security, Jun. 8, 1998, pp. 399-440.		
6,286,047 B1	9/2001	Ramanathan et al.			
6,301,223 B1	10/2001	Hrastar et al.			
6,308,274 B1	10/2001	Swift			
6,311,207 B1	10/2001	Mighdoll et al.			
6,324,161 B1	11/2001	Kirch			
6,330,562 B1	12/2001	Boden et al.			
6,332,158 B1	12/2001	Risley et al.			
6,353,614 B1	3/2002	Borella et al.			
6,425,003 B1	7/2002	Herzog et al.			
6,430,155 B1	8/2002	Davie et al.			
6,430,610 B1	8/2002	Carter			
6,487,598 B1	11/2002	Valencia			
6,502,135 B1	12/2002	Munger et al.			
6,505,232 B1	1/2003	Mighdoll et al.			
6,510,154 B1	1/2003	Mayes et al.			
6,549,516 B1	4/2003	Albert et al.			

**US 7,418,504 B2**

Page 3

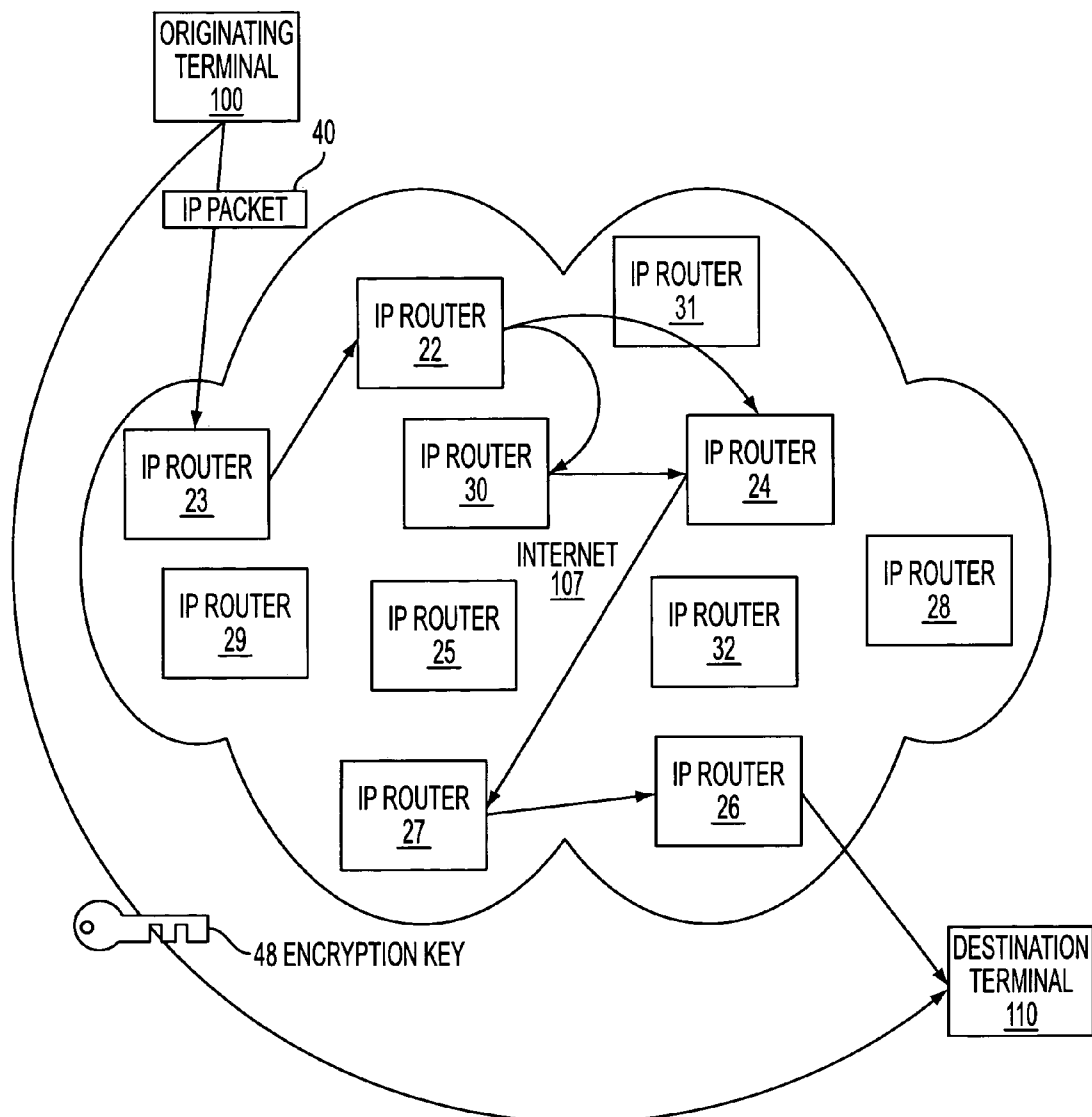
- W. Stallings, "New Cryptography and Network Security Book", Jun. 8, 1998, 3 pages.
- Fasbender, Kesdogan, and Kubitz: "Variable and Scalable Security: Protection of Location Information in Mobile IP", IEEE publication, 1996, pp. 963-967.
- Linux FreeS/WAN Index File, printed from [http://liberty.freeswan.org/freeswan\\_trees/freeswan-1.3/doc/](http://liberty.freeswan.org/freeswan_trees/freeswan-1.3/doc/) on Feb. 21, 2002, 3 Pages.
- J. Gilmore, "Swan: Securing the Internet against Wiretapping", printed from [http://liberty.freeswan.org/freeswan\\_trees/freeswan-1.3/doc/rationale.html](http://liberty.freeswan.org/freeswan_trees/freeswan-1.3/doc/rationale.html) on Feb. 21, 2002, 4 pages.
- Glossary for the Linux FreeS/WAN project, printed from [http://liberty.freeswan.org/freeswan\\_trees/freeswan-1.3/doc/glossary.html](http://liberty.freeswan.org/freeswan_trees/freeswan-1.3/doc/glossary.html) on Feb. 21, 2002, 25 pages.
- Alan O. Frier et al., "The SSL Protocol Version 3.0", Nov. 18, 1996, printed from <http://www.netscape.com/eng/ss13/draft302.txt> on Feb. 4, 2002, 56 pages.
- Search Report (dated Aug. 20, 2002), International Application No. PCT/US01/04340.
- Search Report (dated Aug. 23, 2002), International Application No. PCT/US01/13260.
- Shree Murthy et al., "Congestion-Oriented Shortest Multipath Routing", Proceedings of IEEE INFOCOM, 1996, pp. 1028-1036.
- Jim Jones et al., "Distributed Denial of Service Attacks: Defenses", Global Integrity Corporation, 2000, pp. 1-14.
- James E. Bellaire, "New Statement of Rules—Naming Internet Domains", Internet Newsgroup, Jul. 30, 1995, 1 page.
- D. Clark, "US Calls for Private Domain-Name System", Computer, IEEE Computer Society, Aug. 1, 1998, pp. 22-25.
- August Bequai, "Balancing Legal Concerns Over Crime and Security in Cyberspace", Computer & Security, vol. 17, No. 4, 1998, pp. 293-298.
- Rich Winkel, "CAQ: Networkinig With Spooks: The NET & The Control Of Information", Internet Newsgroup, Jun. 21, 1997, 4 pages.
- Search Report (dated Oct. 7, 2002), International Application No. PCT/US01/13261.
- F. Halsall, "Data Communications, Computer Networks And Open Systems", Chapter 4, Protocol Basics, 1996, pp. 198-203.
- Reiter, Michael K. and Rubin, Aviel D. (AT&T Labs—Research), "Crowds: Anonymity for Web Transmissoins", pp. 1-23.
- Dolev, Shlomi and Ostrovsky, Rafil, "Efficient Anonymous Multicast and Reception"(Extended Abstract), 16 pages.
- Rubin, Aviel D., Greer, Daniel, and Ranum, Marcus J. (Wiley Computer Publishing), "Web Security Sourcebook", pp. 82-94.
- Fasbender, Kesdogan, and Kubitz: "Variable and Scalable Security" Protection of Location Information in Mobile IP, IEEE publication, 1996, pp. 963-967.
- Eastlake, D. E., "Domain Name System Security Extensions", Internet Draft, Apr. 1998, XP002199931, Sections 1, 2.3 and 2.4.
- RFC 2401 (dated Nov. 1998) Security Architecture for the Internet Protocol (RTP).
- RFC 2543-SIP (dated Mar. 1999): Session Initiation Protocol (SIP or SIPS).
- Search Report, IPER (dated Nov. 13, 2002), International Application No. PCT/US01/04340.
- Search Report, IPER (dated Feb. 6, 2002), International Application No. PCT/US01/13261.
- Search Report, IPER (dated Jan. 14, 2003), International Application No. PCT/US01/13260.
- Shankur, A.U. "A verified sliding window protocol with variable flow control". Proceedings of ACM SIGCOMM conference on Communications architectures & protocols. pp. 84-91, ACM Press, NY, NY 1986.
- W. Stallings, "Cryptography and Network Security", 2nd, Edition, Chapter 13, IP Security, Jun. 8, 1998, pp. 399-440.

**U.S. Patent**

Aug. 26, 2008

Sheet 1 of 40

**US 7,418,504 B2**



**FIG. 1**

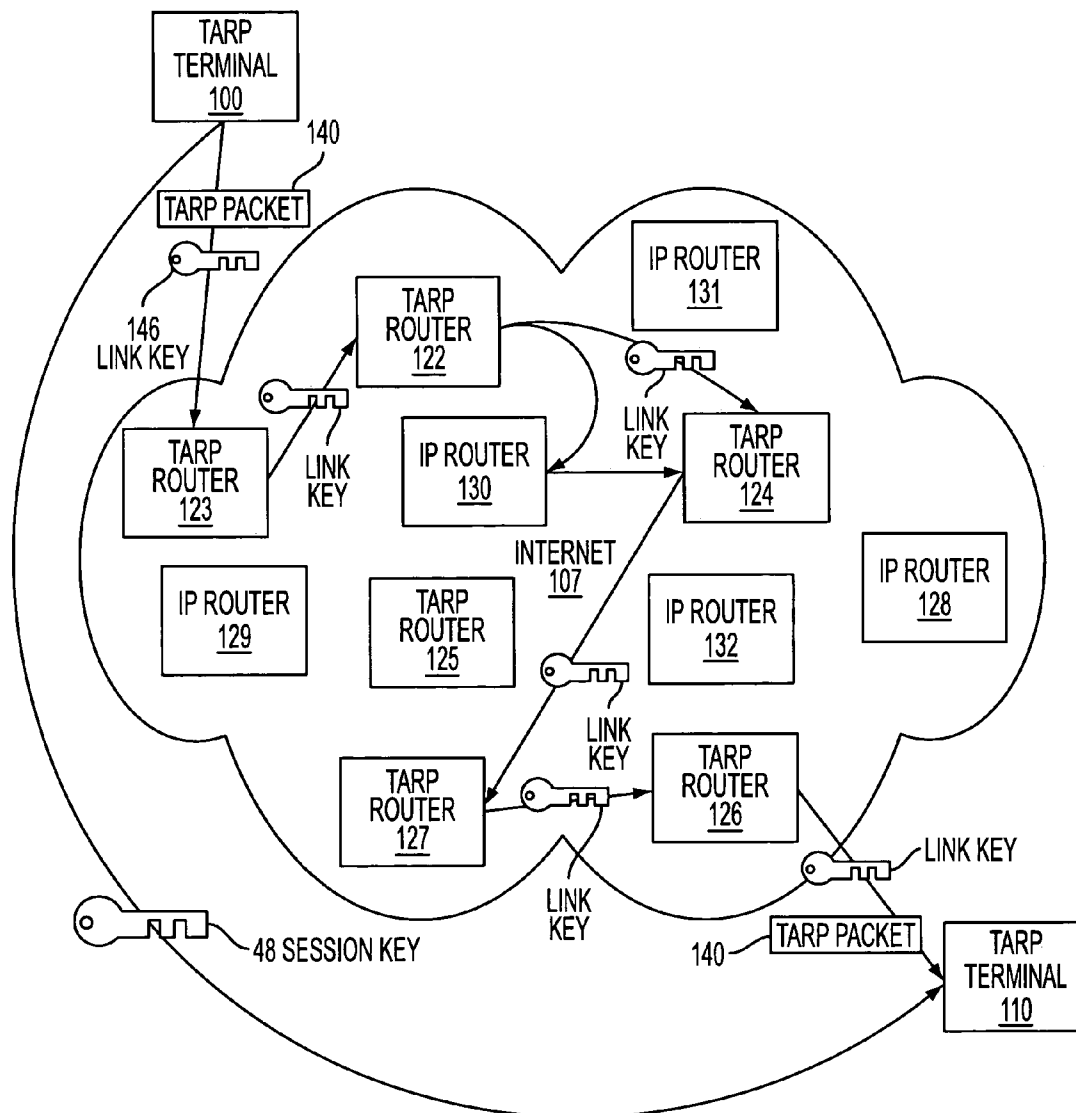


FIG. 2

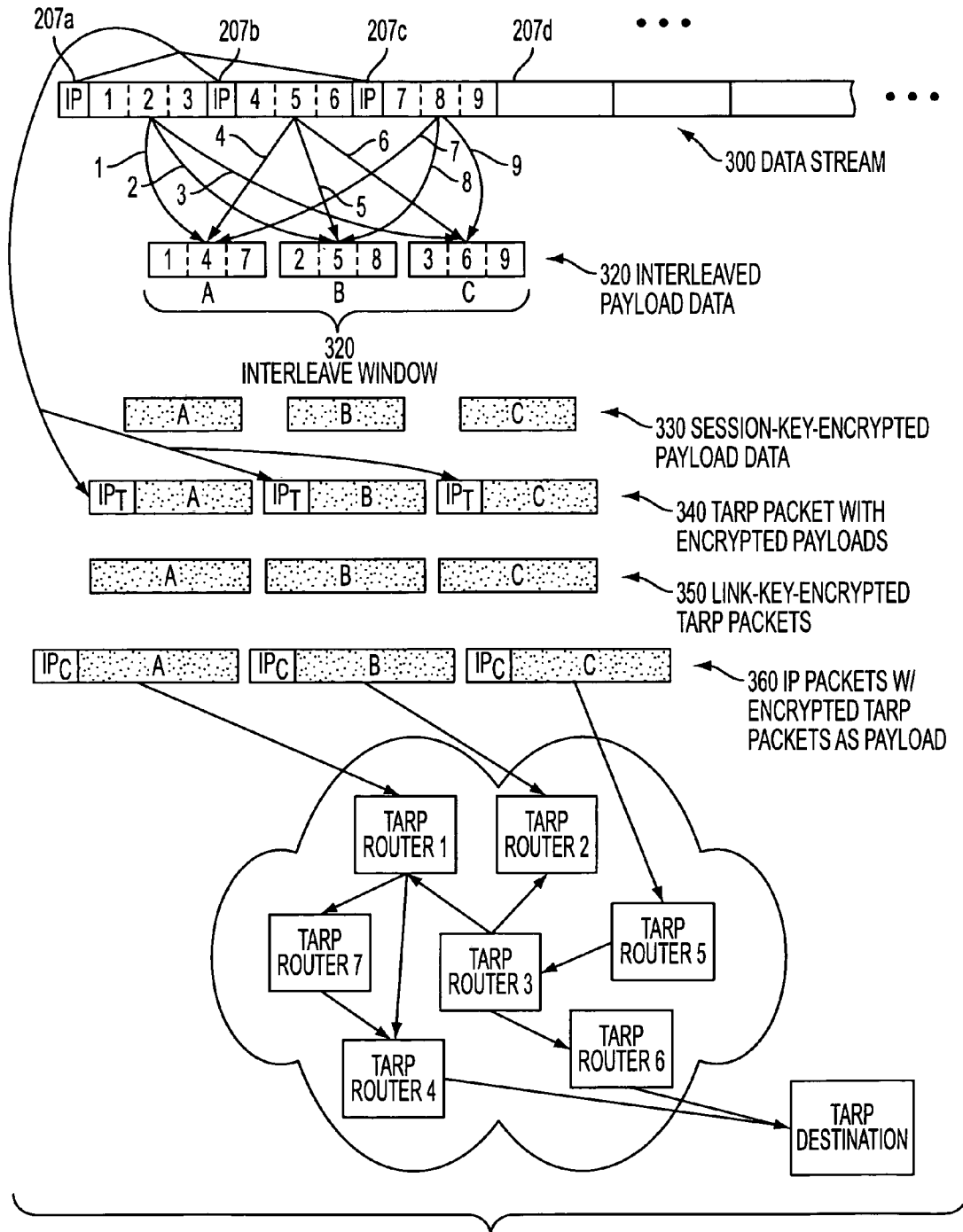


FIG. 3A

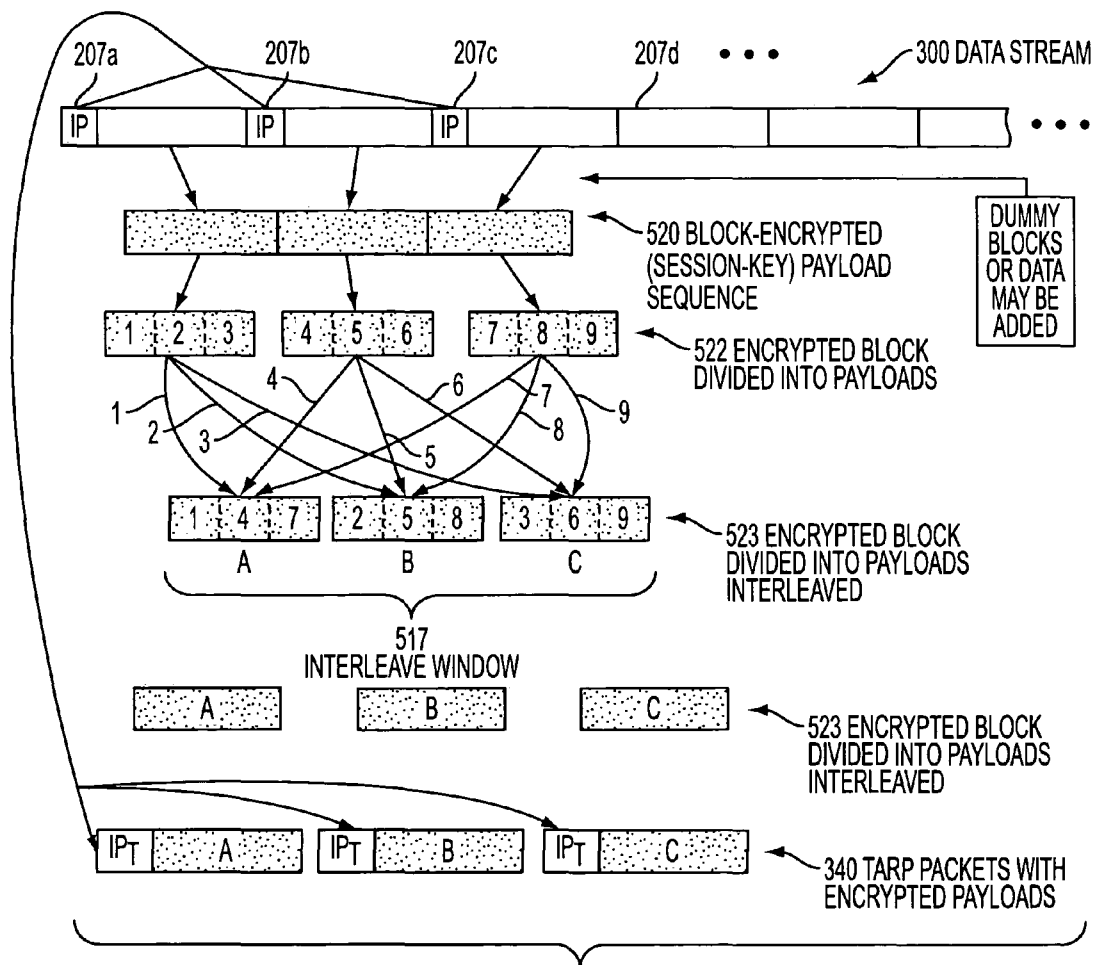


FIG. 3B

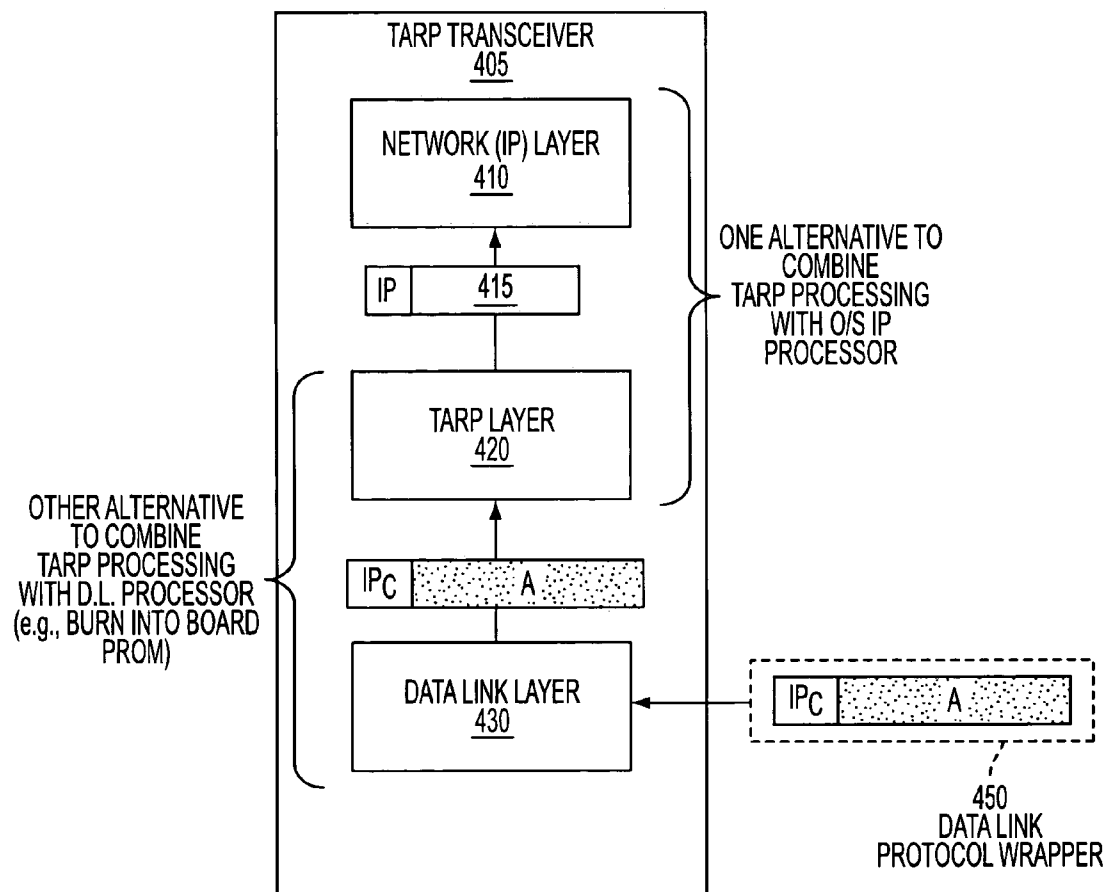


FIG. 4



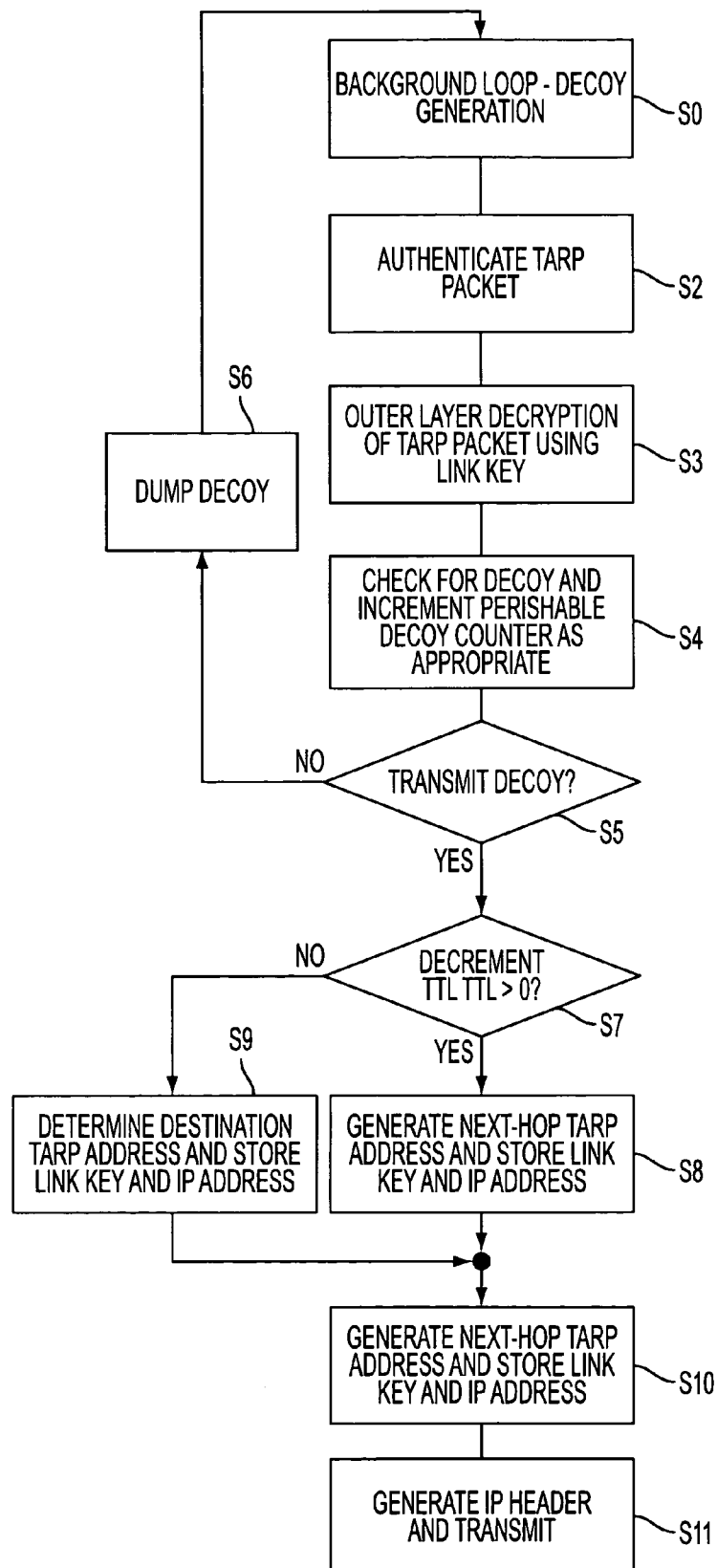
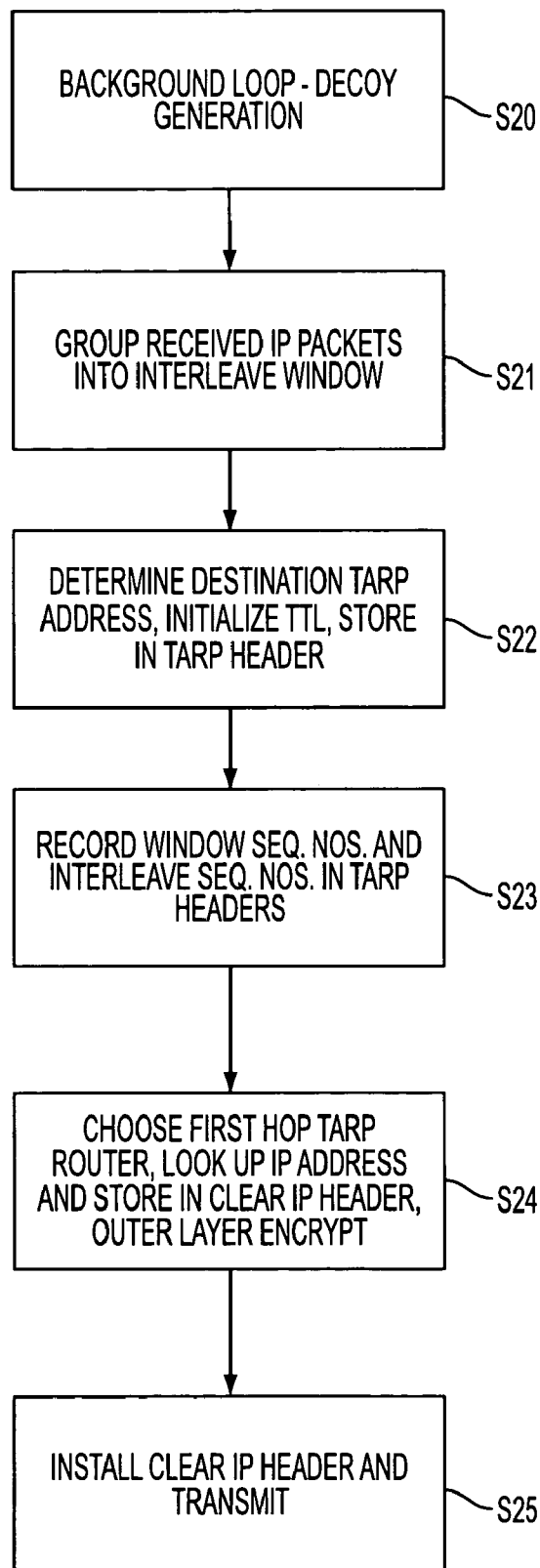


FIG. 5

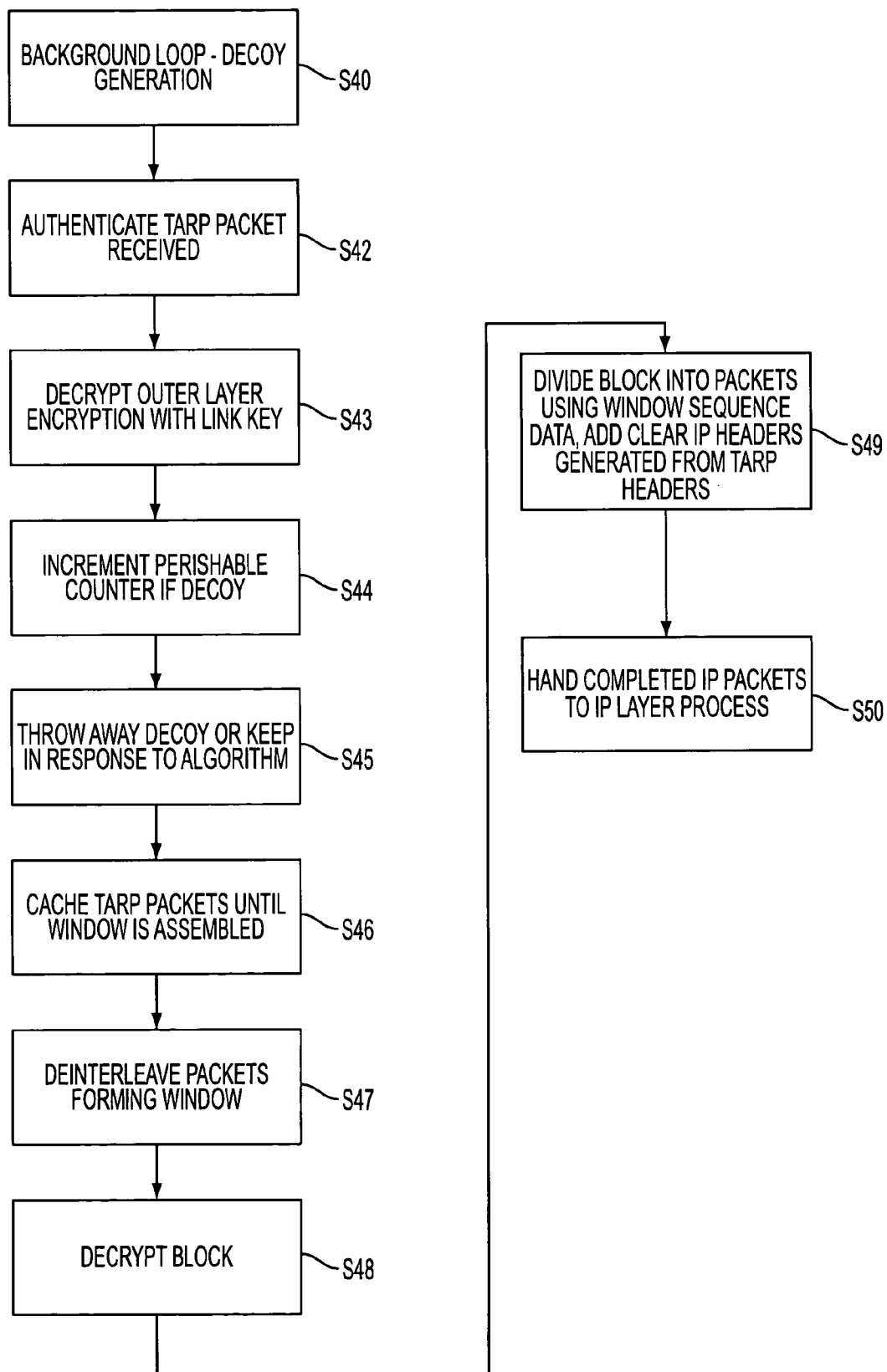


**FIG. 6**

**U.S. Patent**

Aug. 26, 2008

Sheet 8 of 40

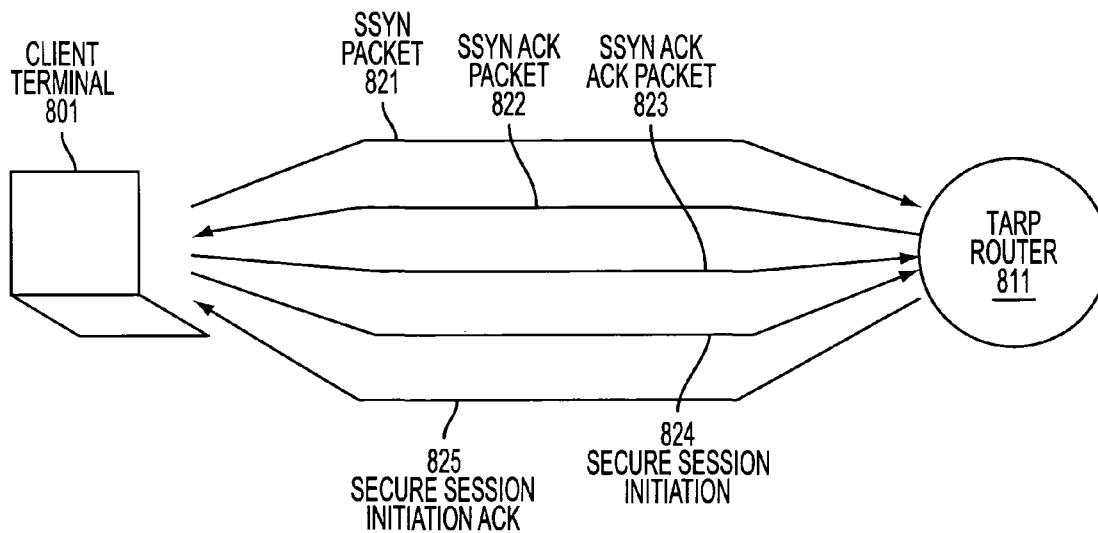
**US 7,418,504 B2****FIG. 7****Appx202**

**U.S. Patent**

**Aug. 26, 2008**

**Sheet 9 of 40**

**US 7,418,504 B2**

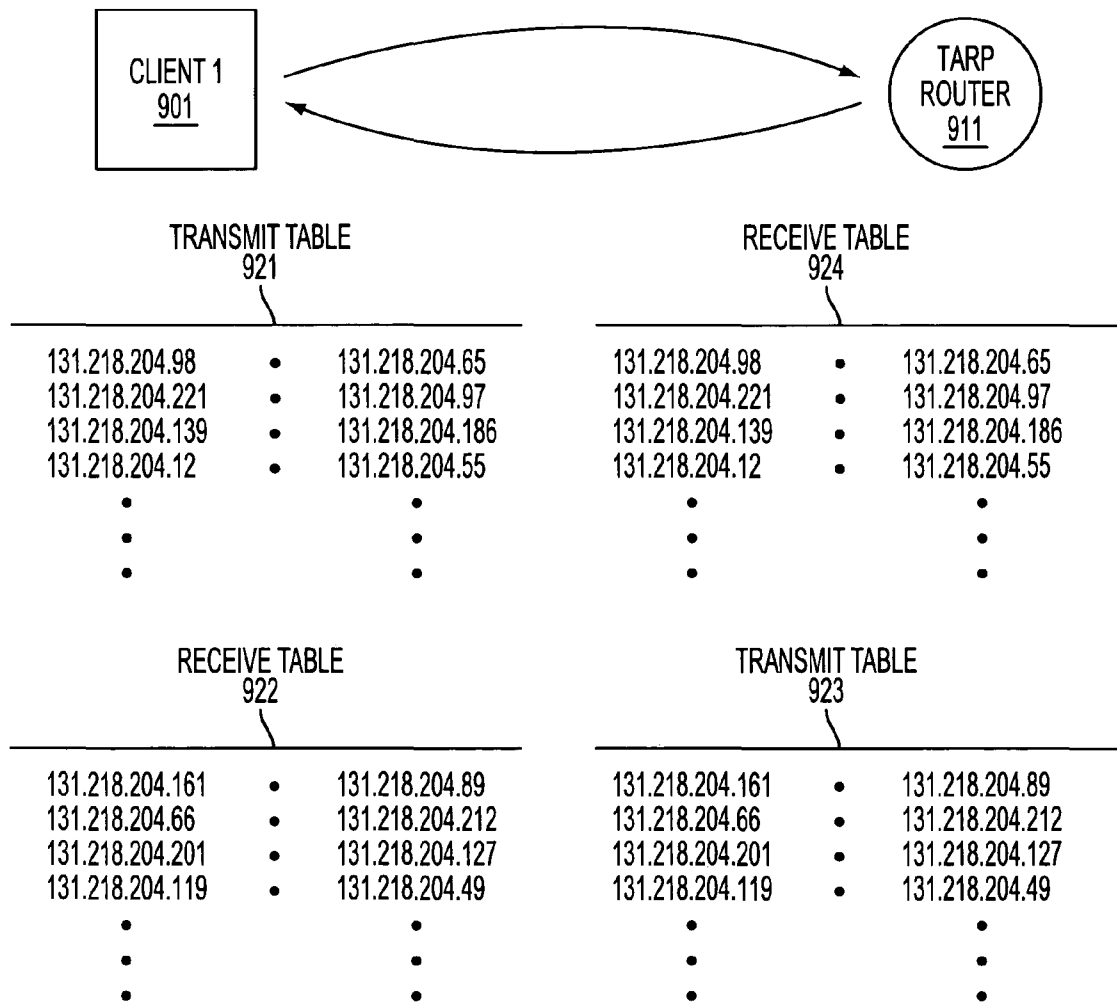


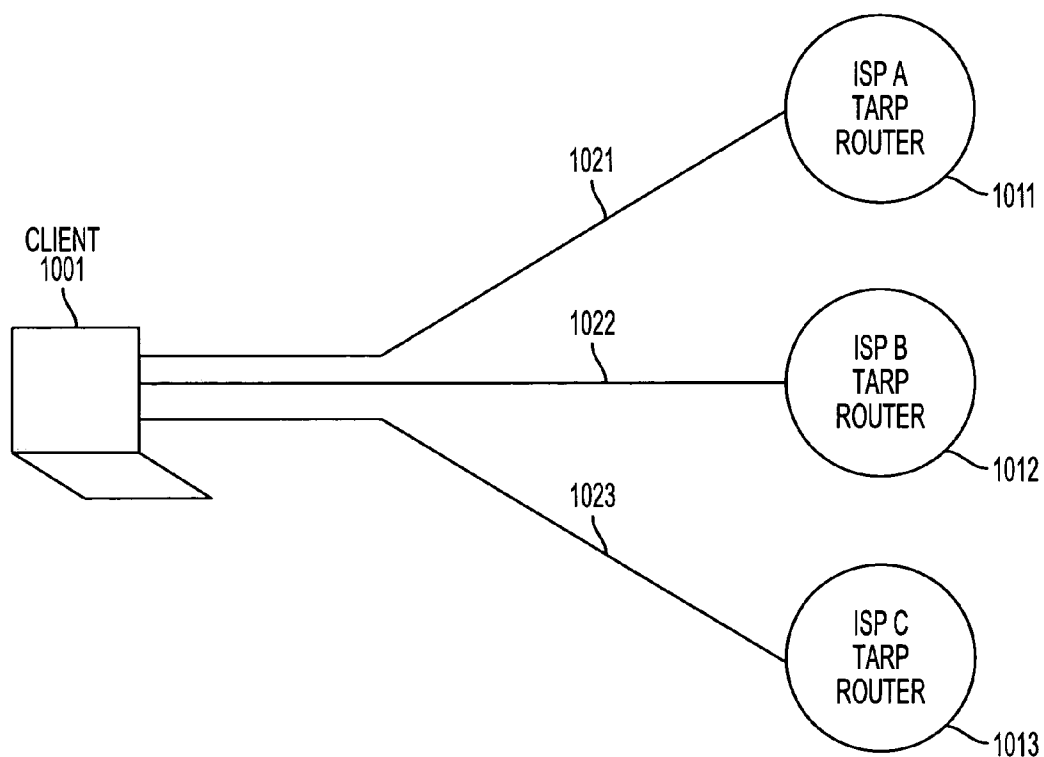
**FIG. 8**

**U.S. Patent**

Aug. 26, 2008

Sheet 10 of 40

**US 7,418,504 B2****FIG. 9**



**FIG. 10**

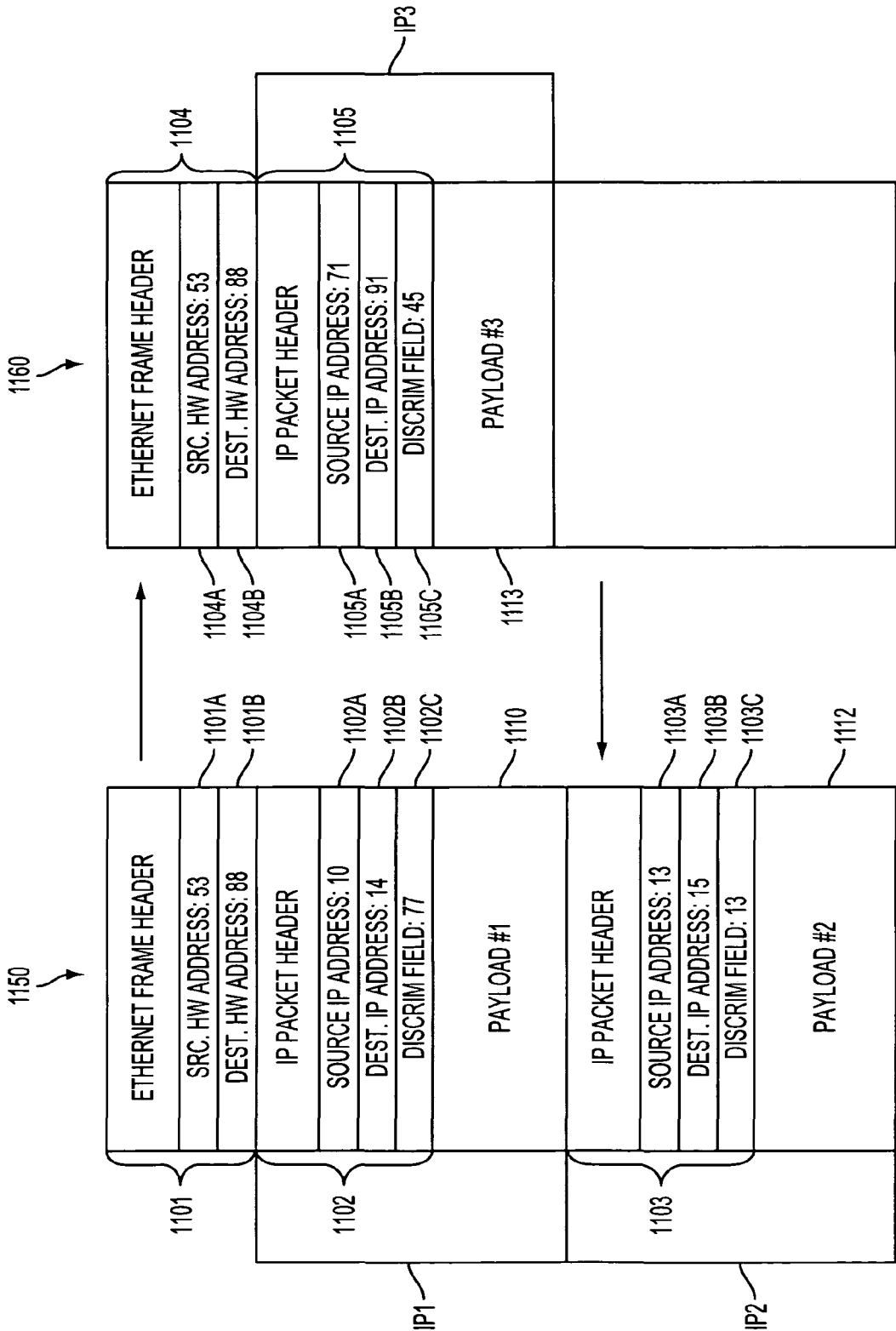


FIG. 11

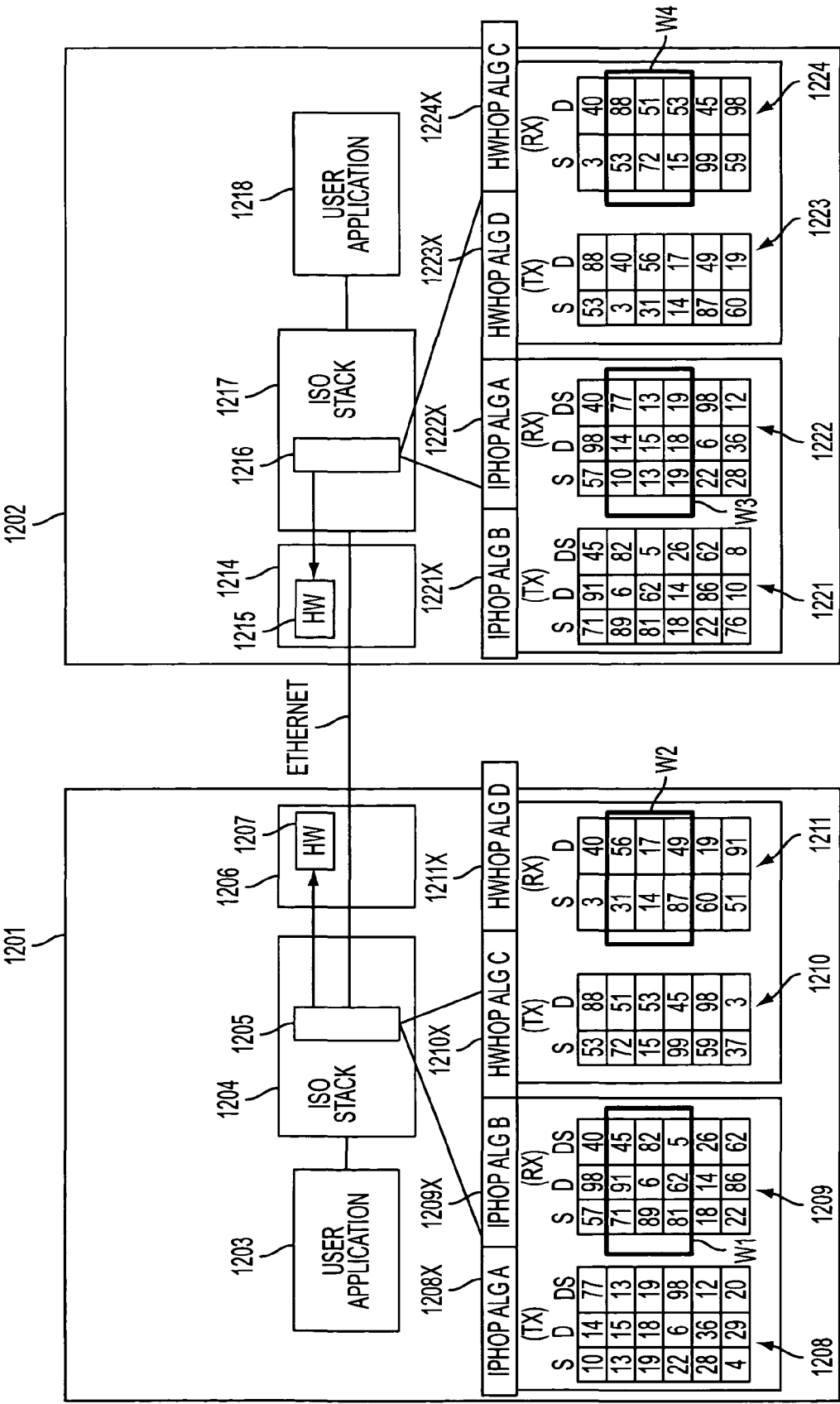


FIG. 12A



**U.S. Patent****Aug. 26, 2008****Sheet 14 of 40****US 7,418,504 B2**

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

**FIG. 12B**

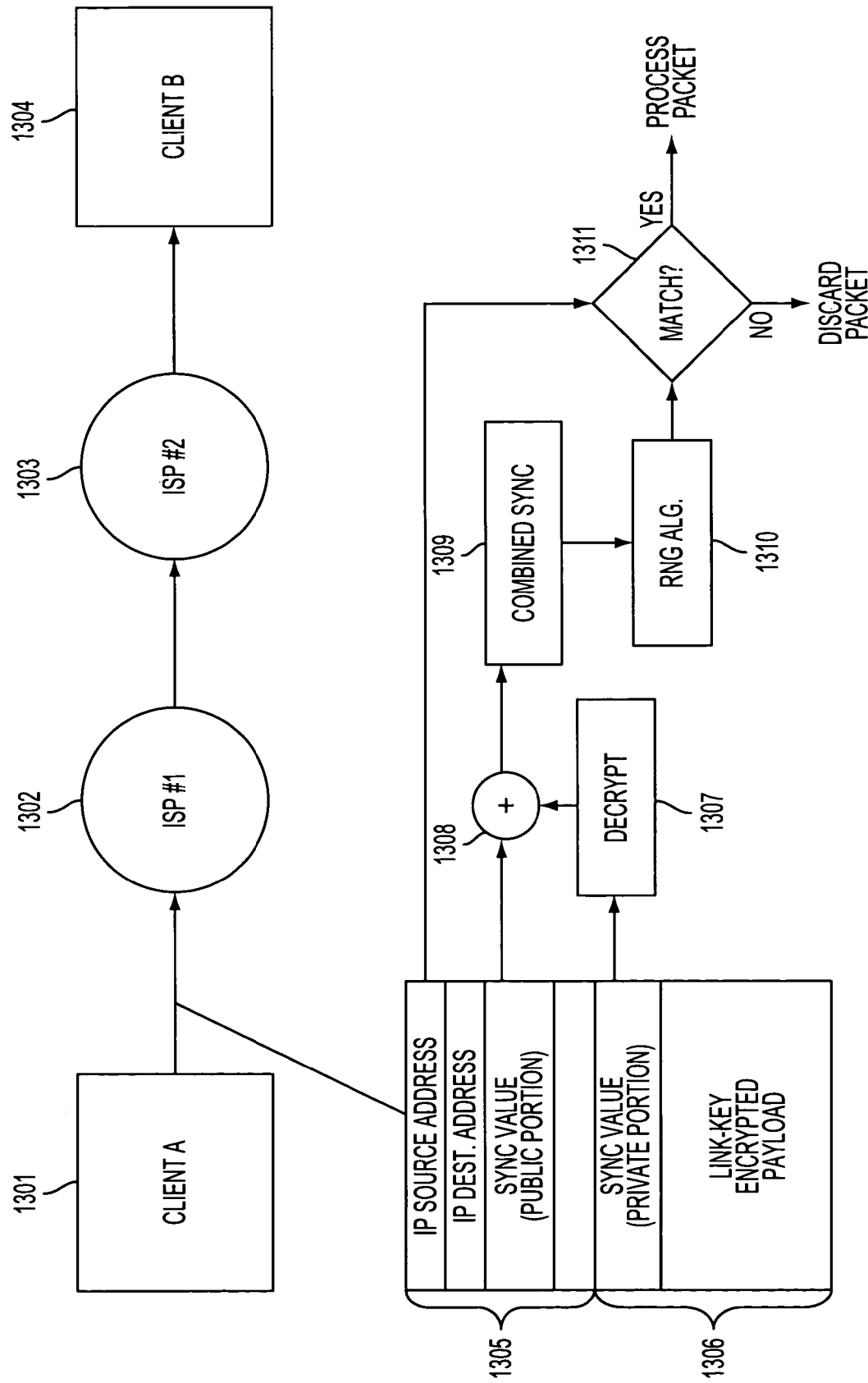


FIG. 13

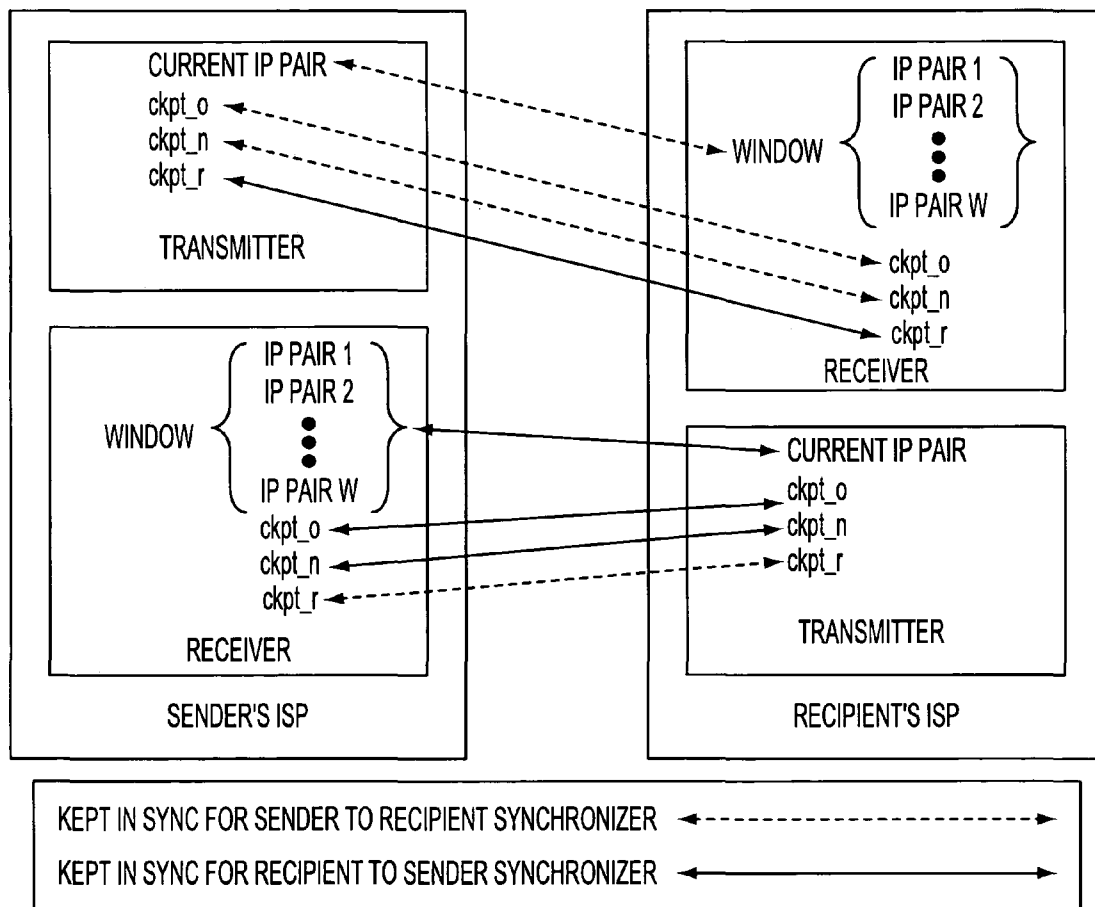


FIG. 14

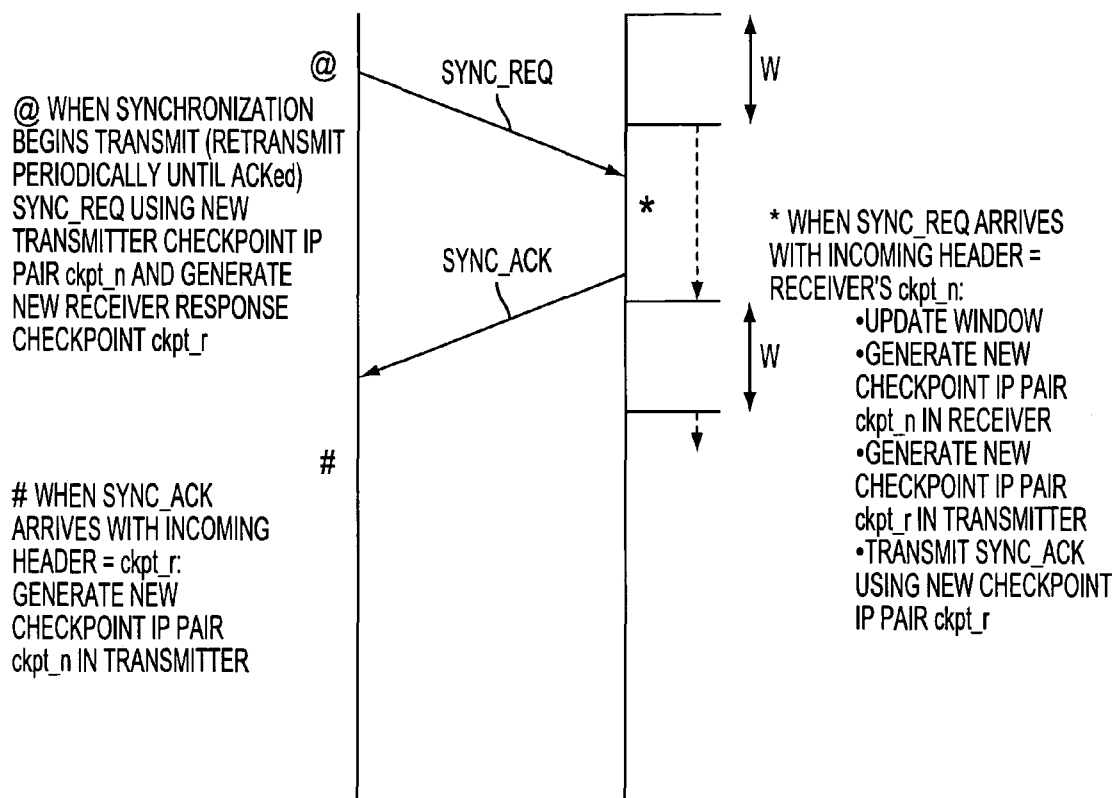


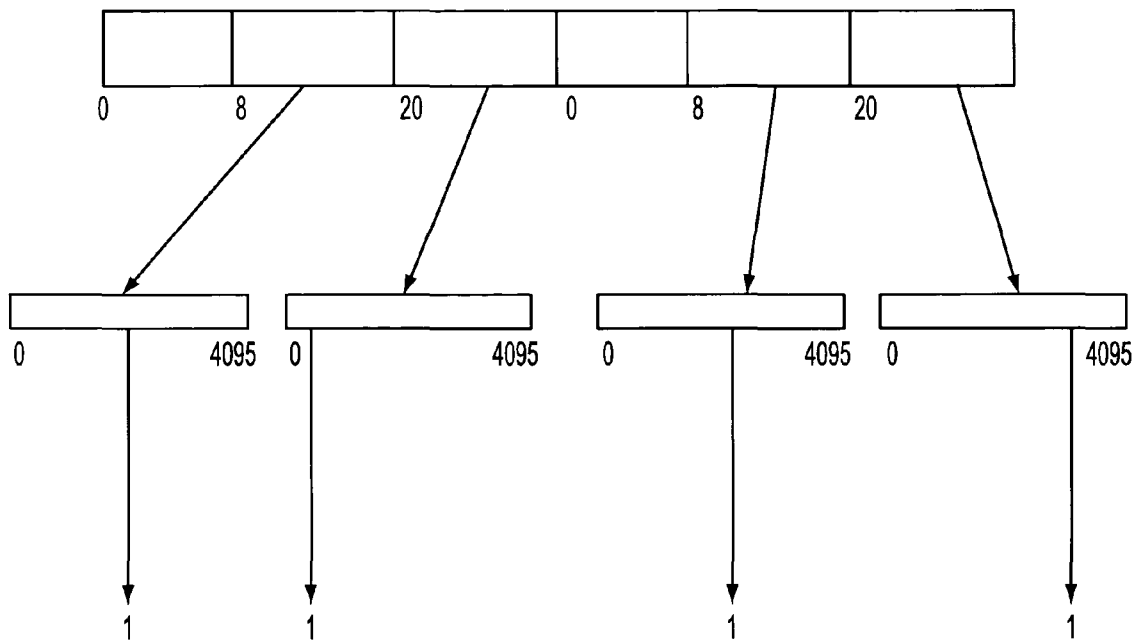
FIG. 15

**U.S. Patent**

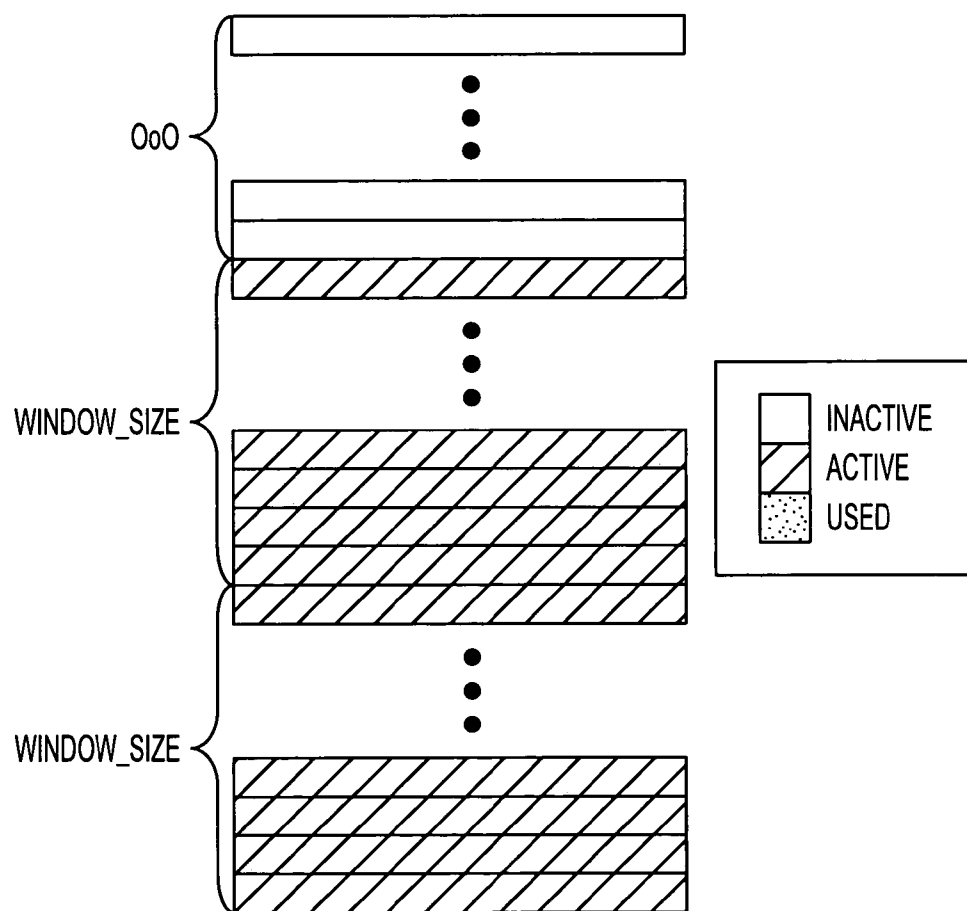
**Aug. 26, 2008**

**Sheet 18 of 40**

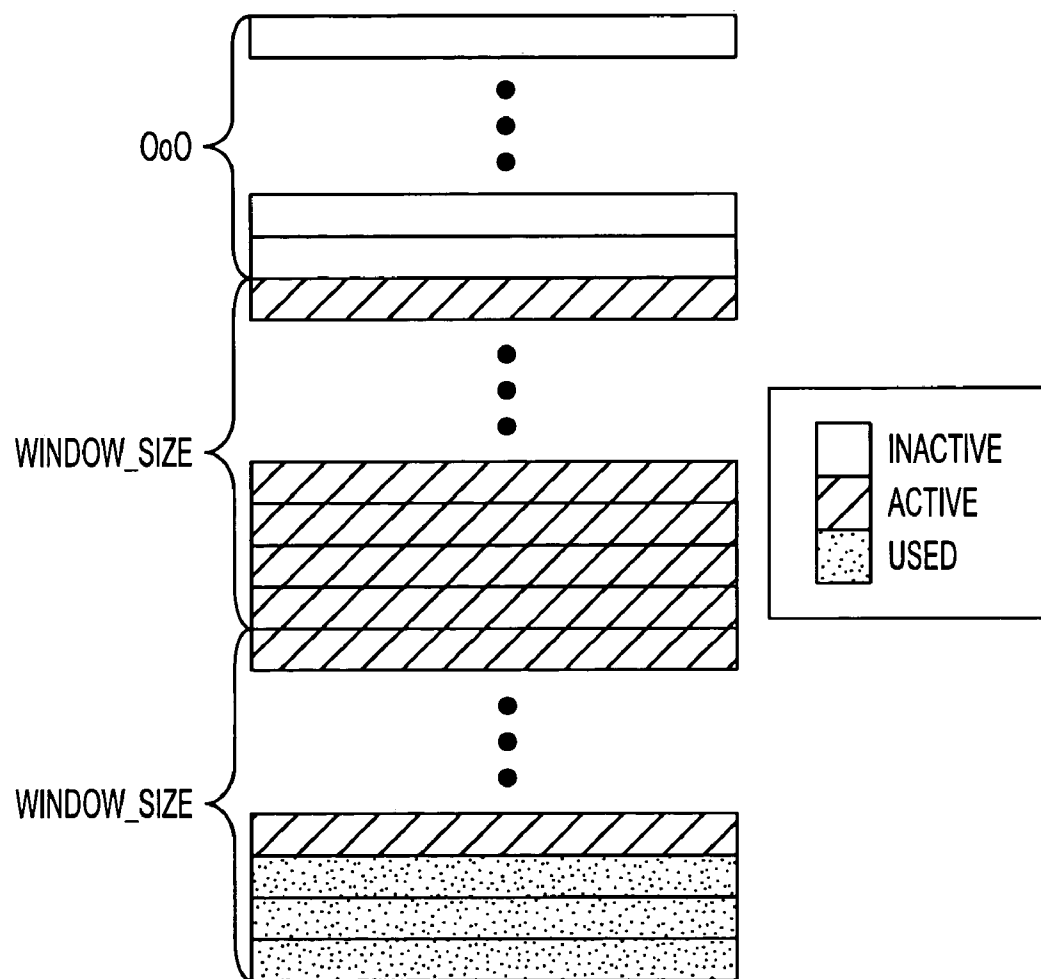
**US 7,418,504 B2**



**FIG. 16**



**FIG. 17**



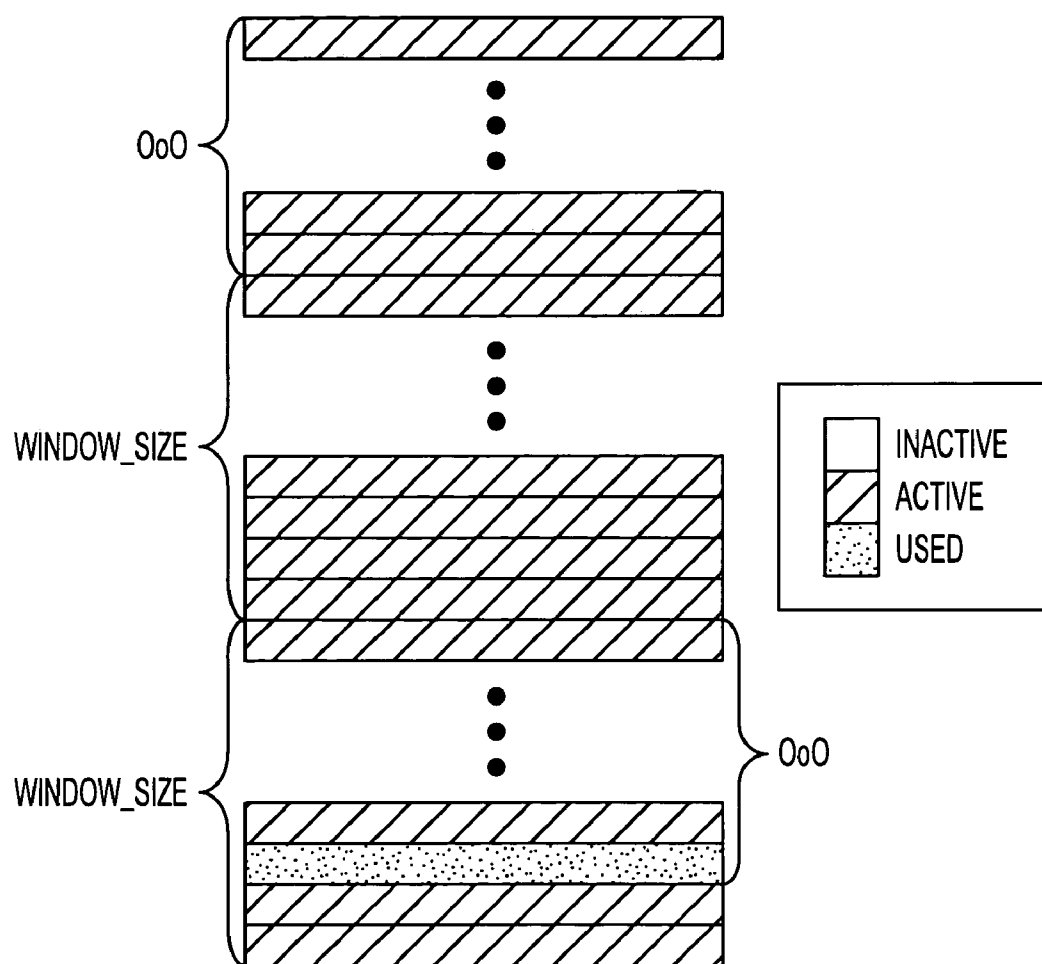
**FIG. 18**

**U.S. Patent**

Aug. 26, 2008

Sheet 21 of 40

**US 7,418,504 B2**



**FIG. 19**



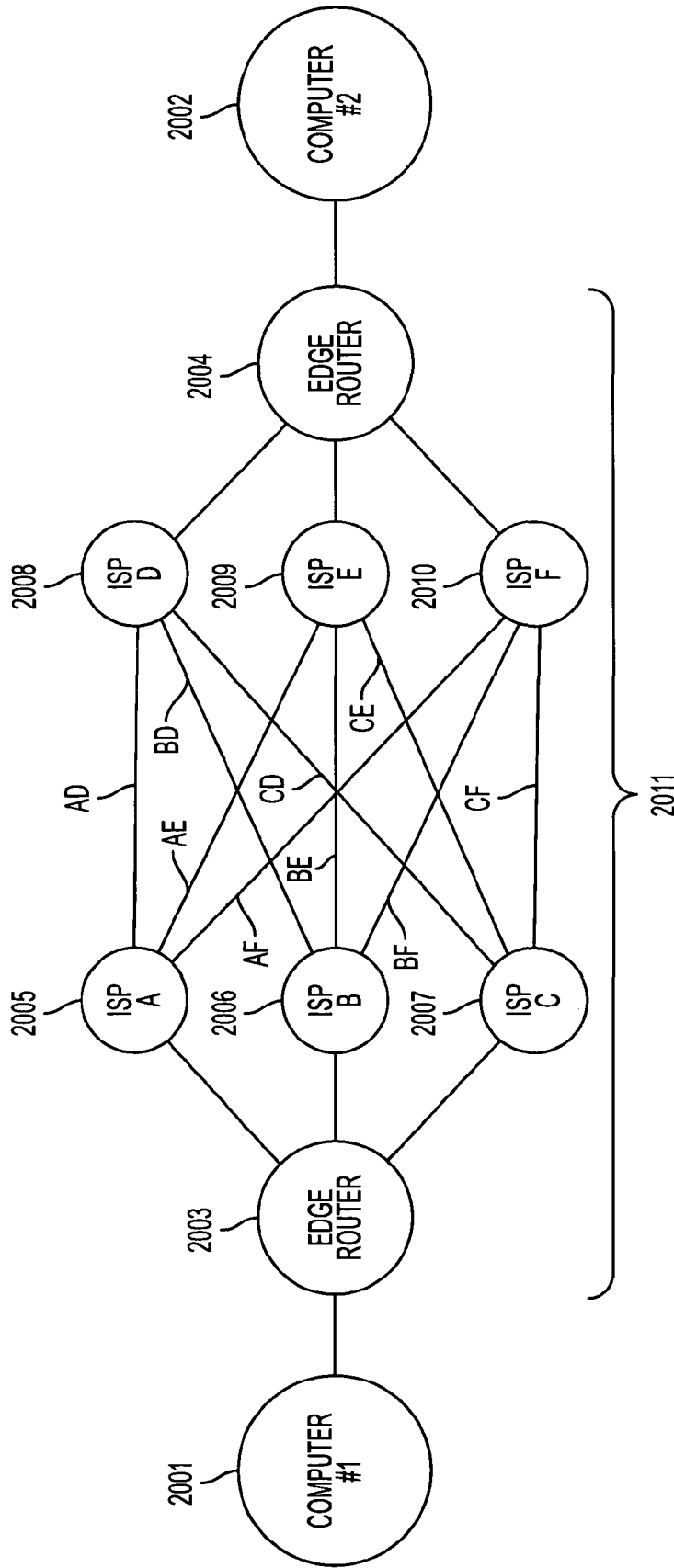
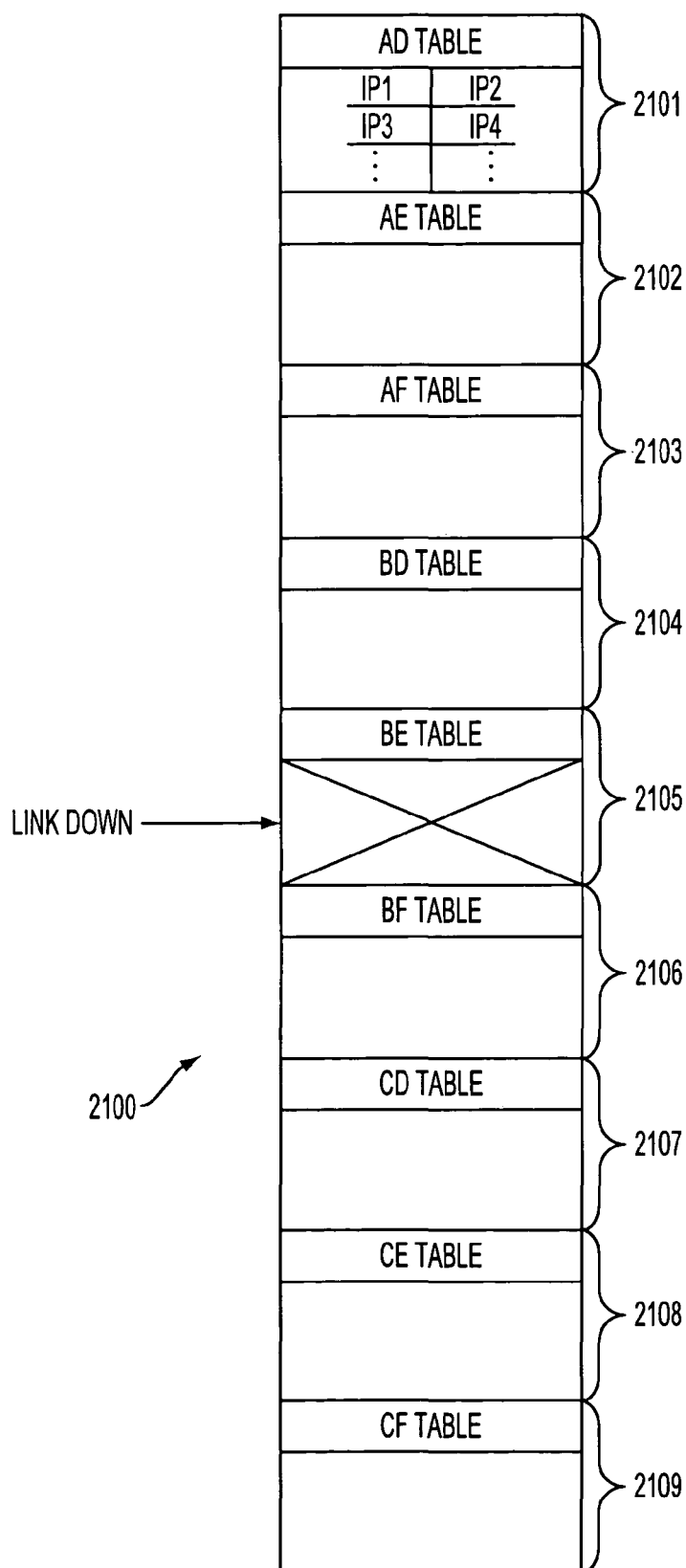


FIG. 20

**U.S. Patent**

Aug. 26, 2008

Sheet 23 of 40

**US 7,418,504 B2****FIG. 21****Appx217**

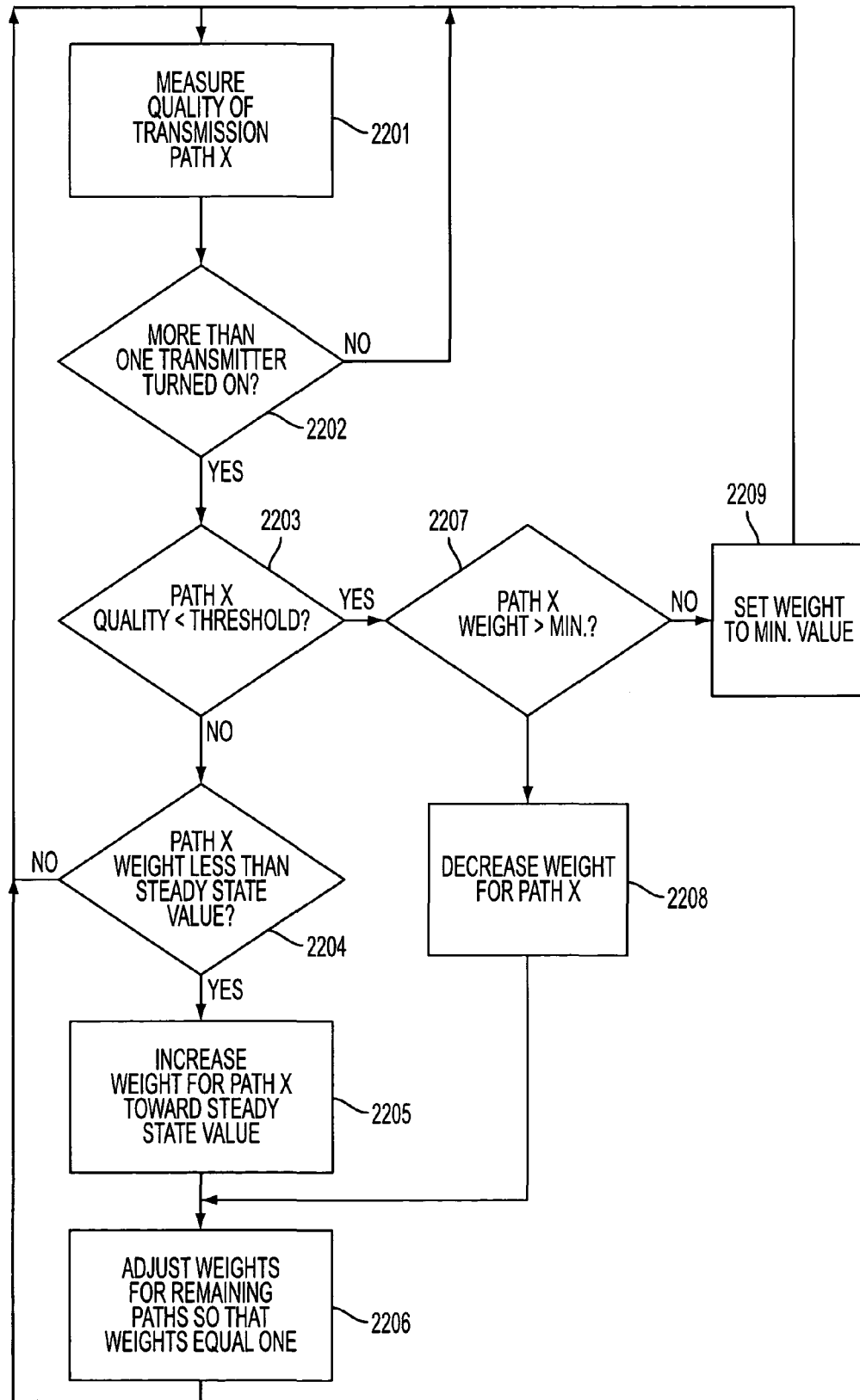


FIG. 22A

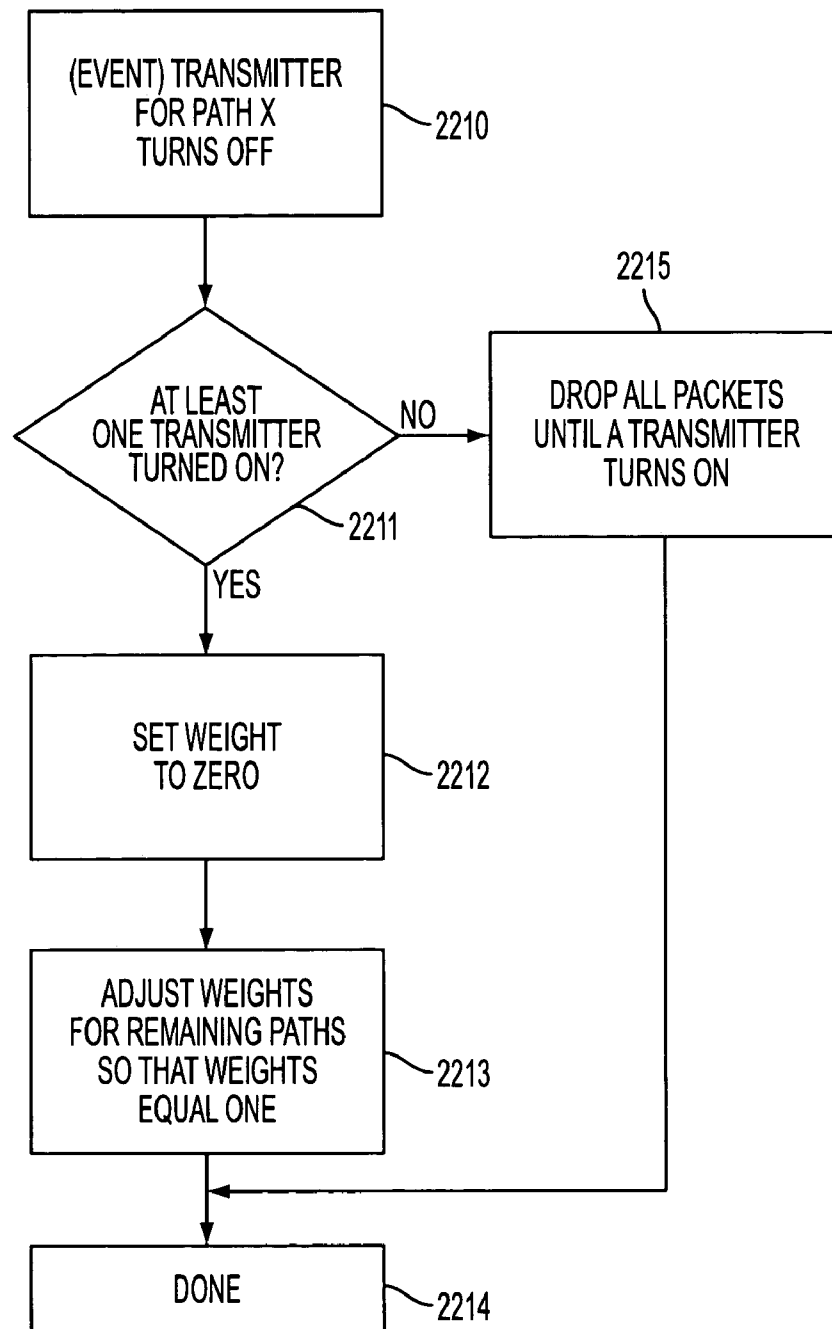


FIG. 22B

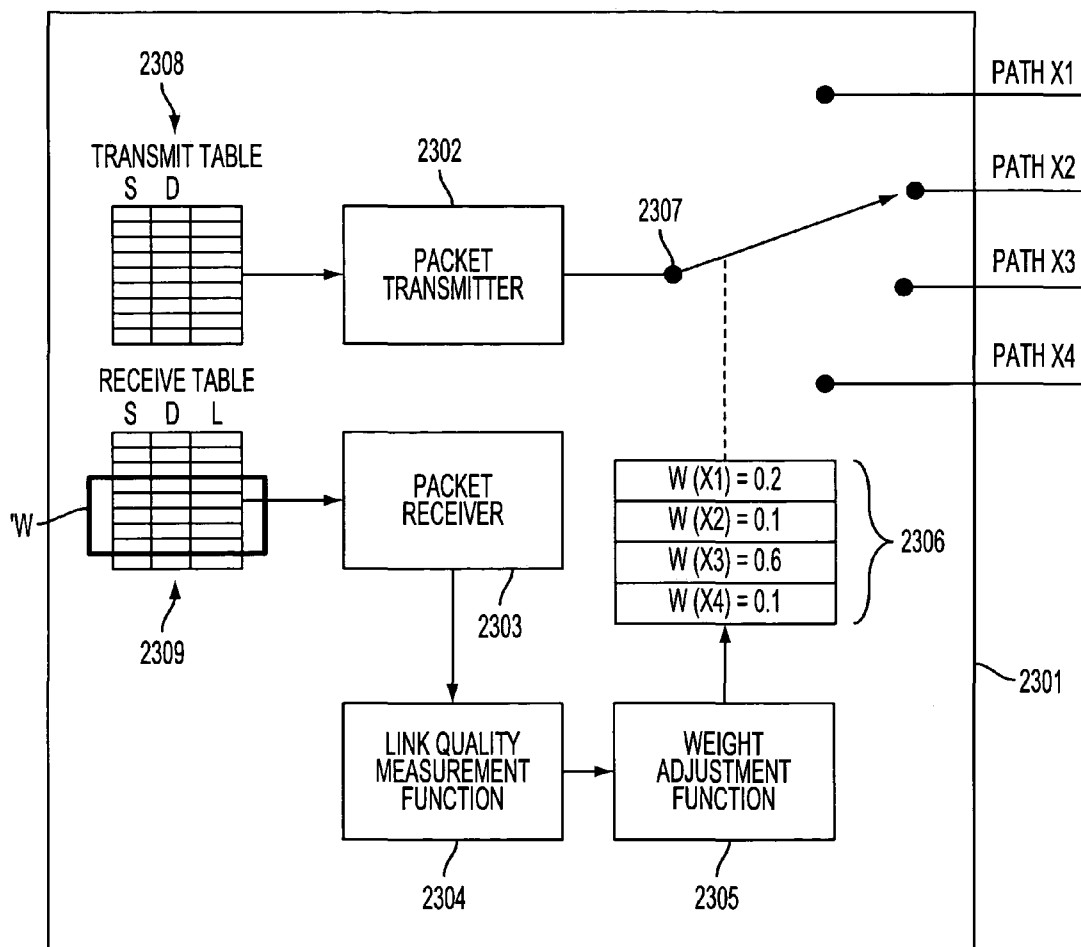


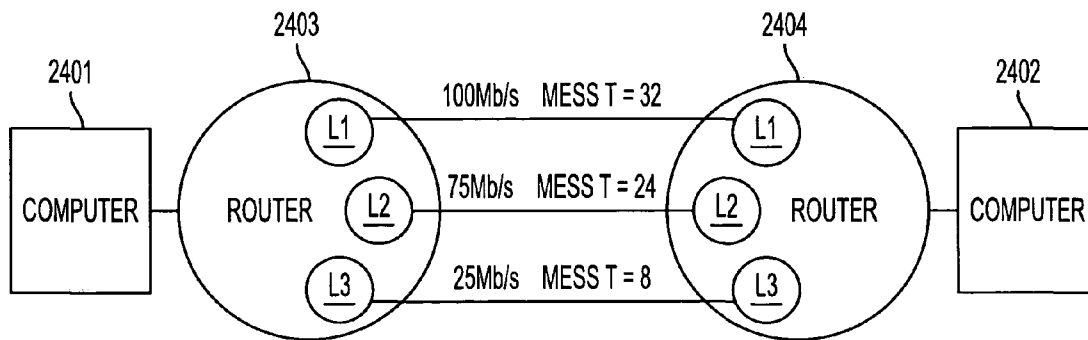
FIG. 23

**U.S. Patent**

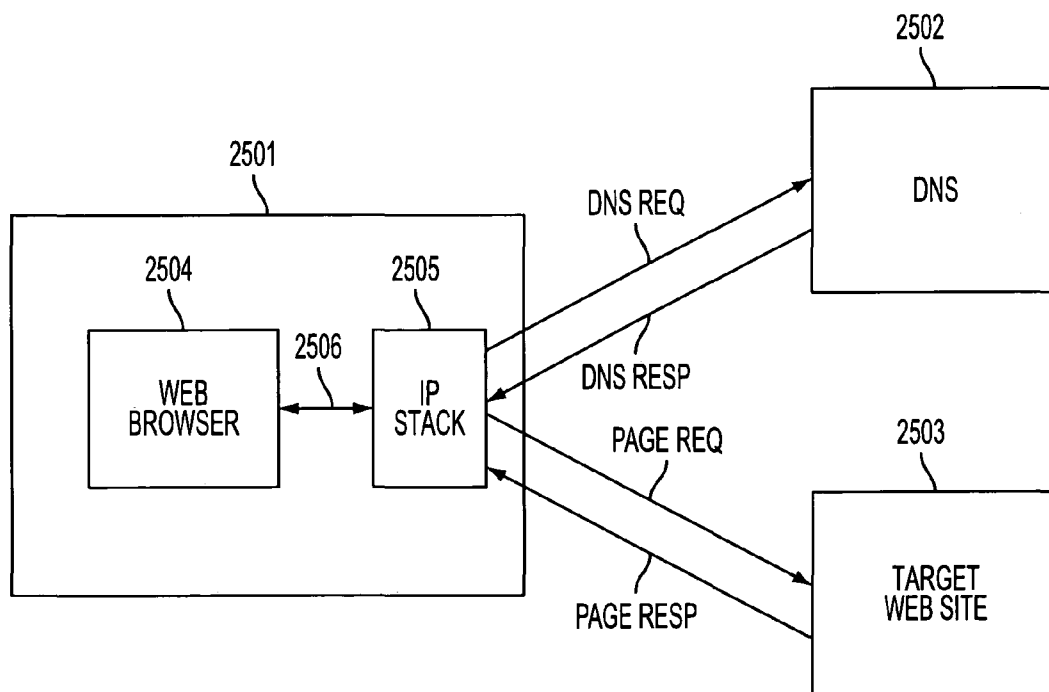
**Aug. 26, 2008**

**Sheet 27 of 40**

**US 7,418,504 B2**



**FIG. 24**



**FIG. 25**  
(PRIOR ART)

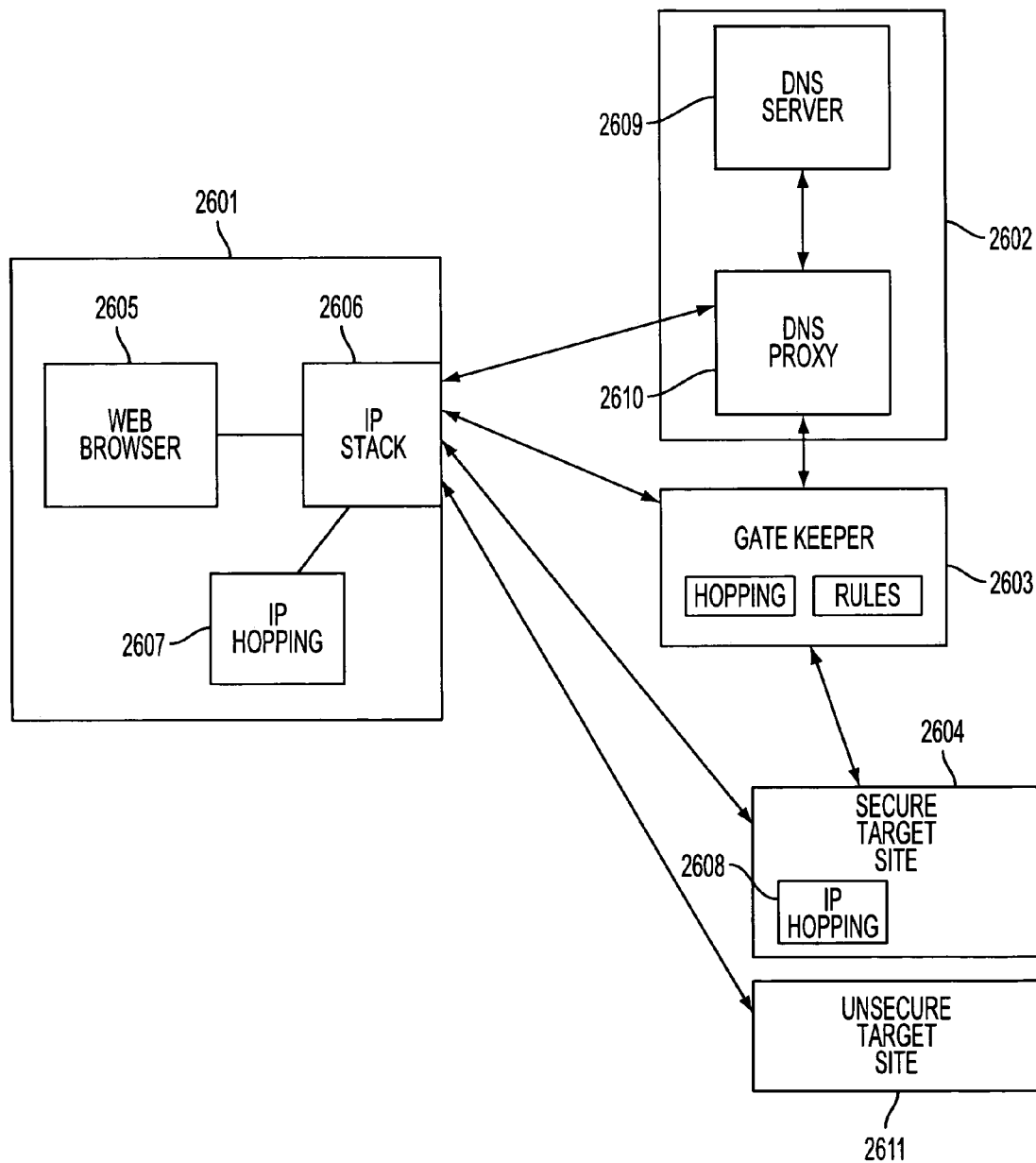


FIG. 26



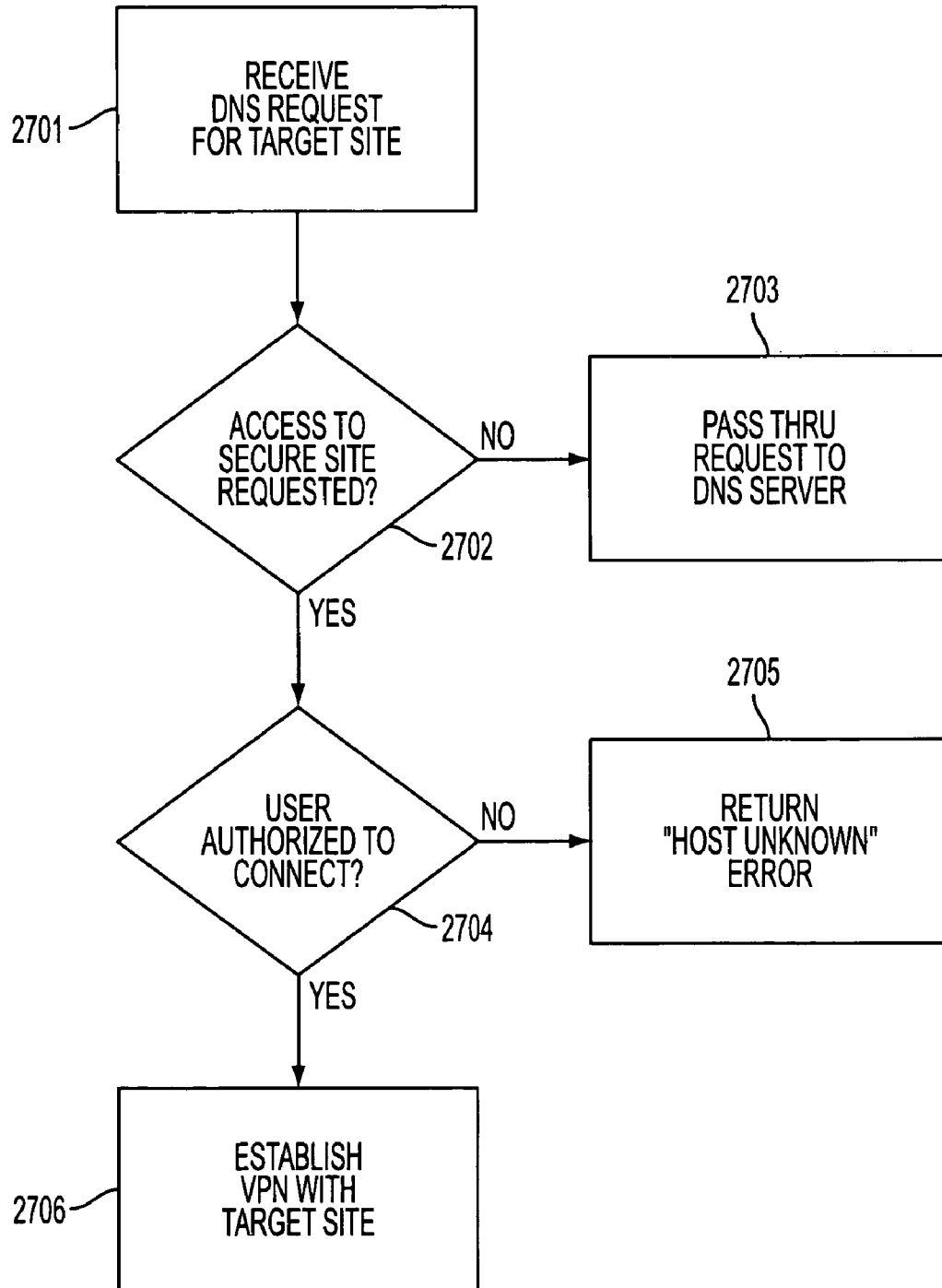
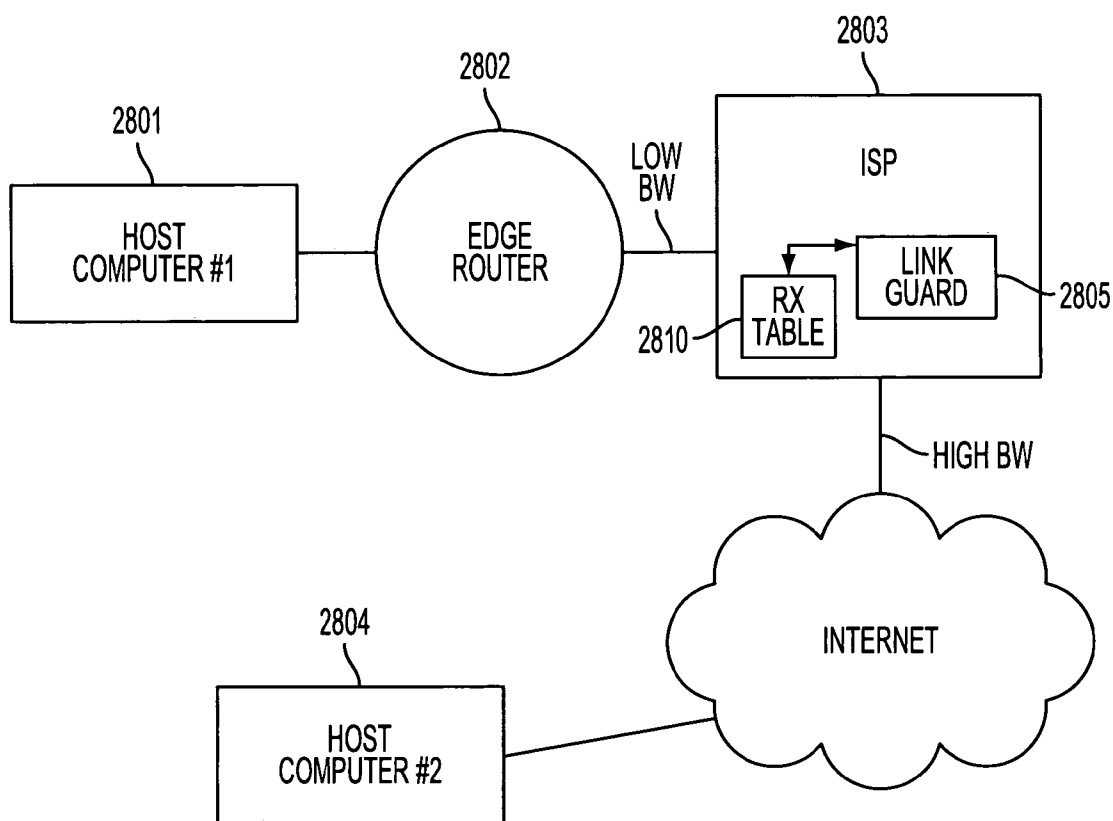


FIG. 27



**FIG. 28**

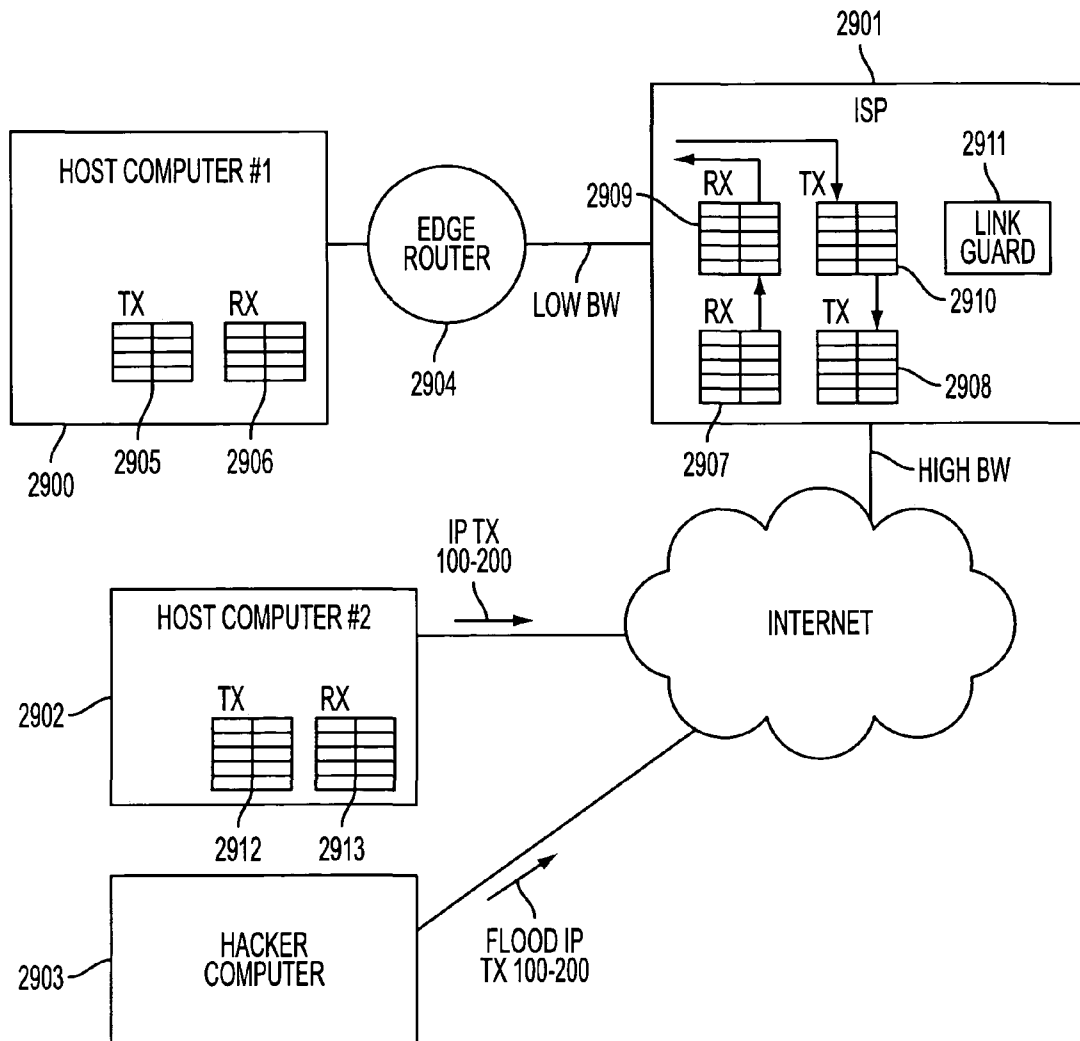


FIG. 29

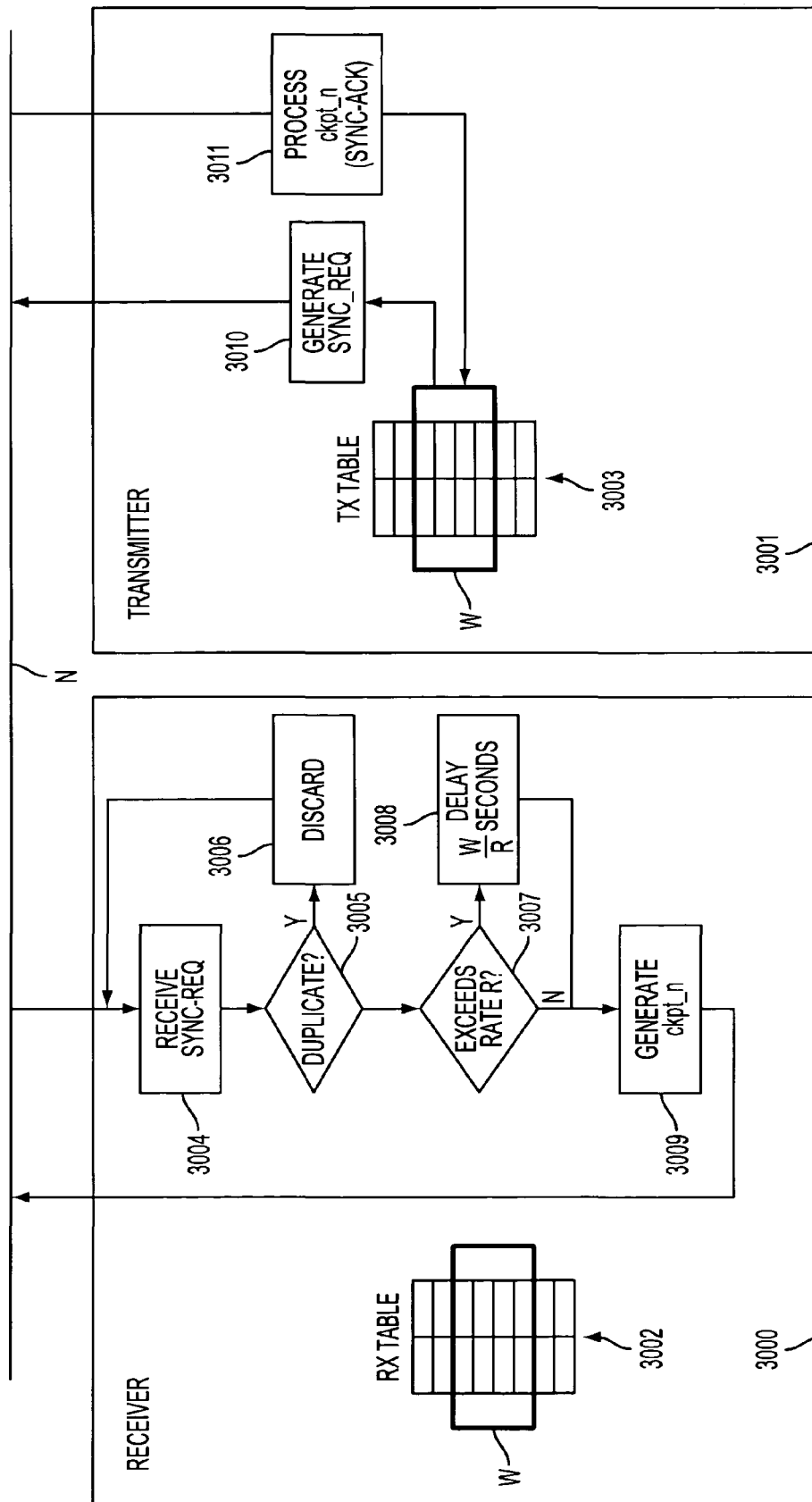


FIG. 30

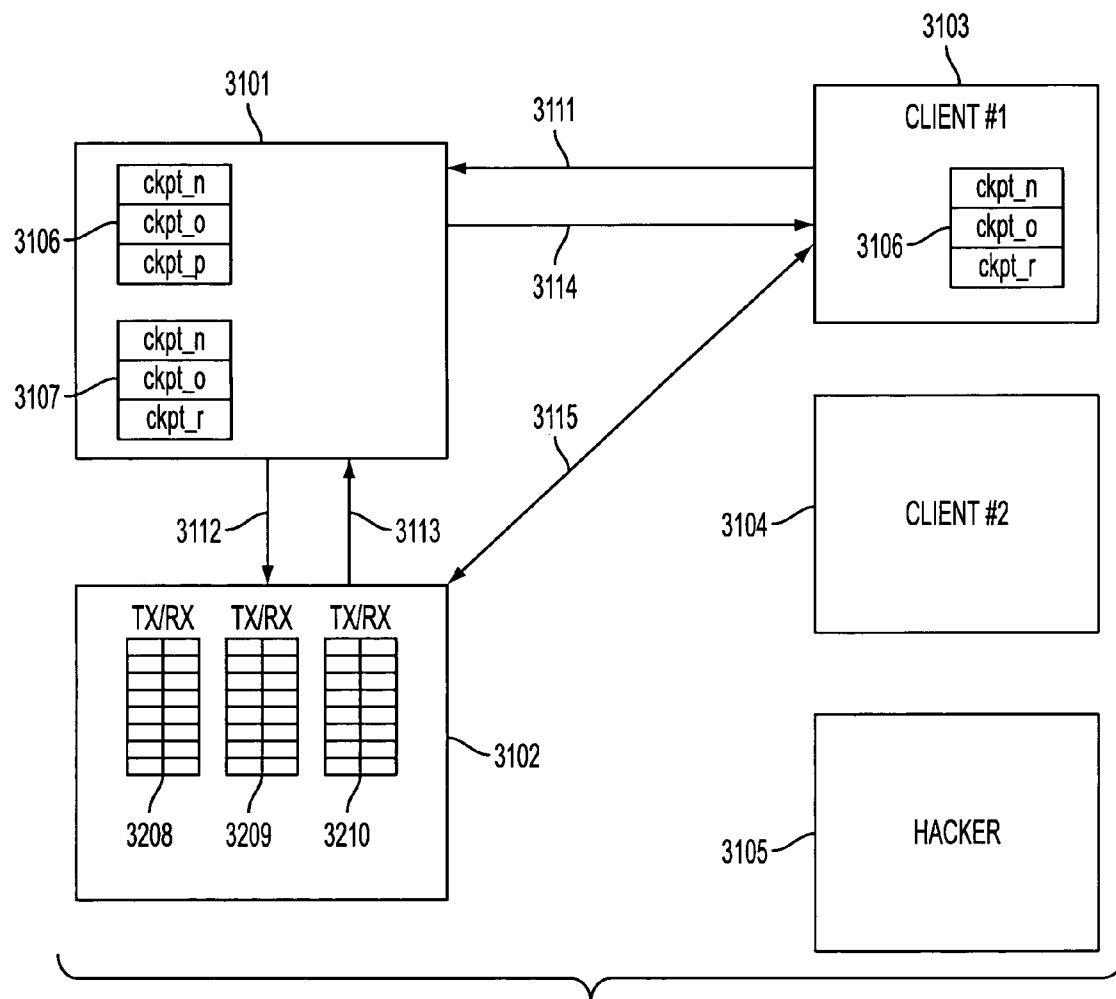


FIG. 31

**U.S. Patent**

Aug. 26, 2008

Sheet 35 of 40

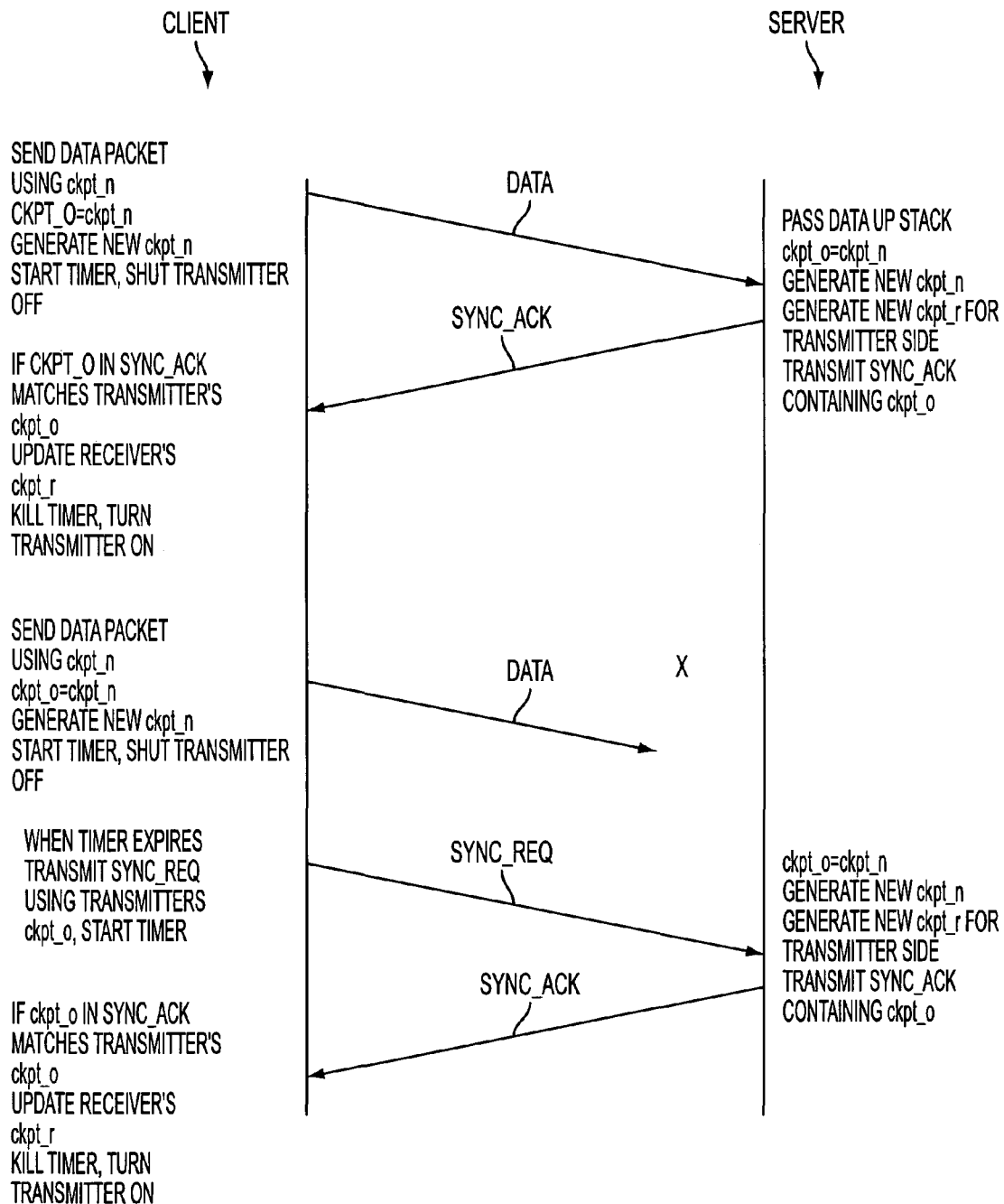
**US 7,418,504 B2**

FIG. 32

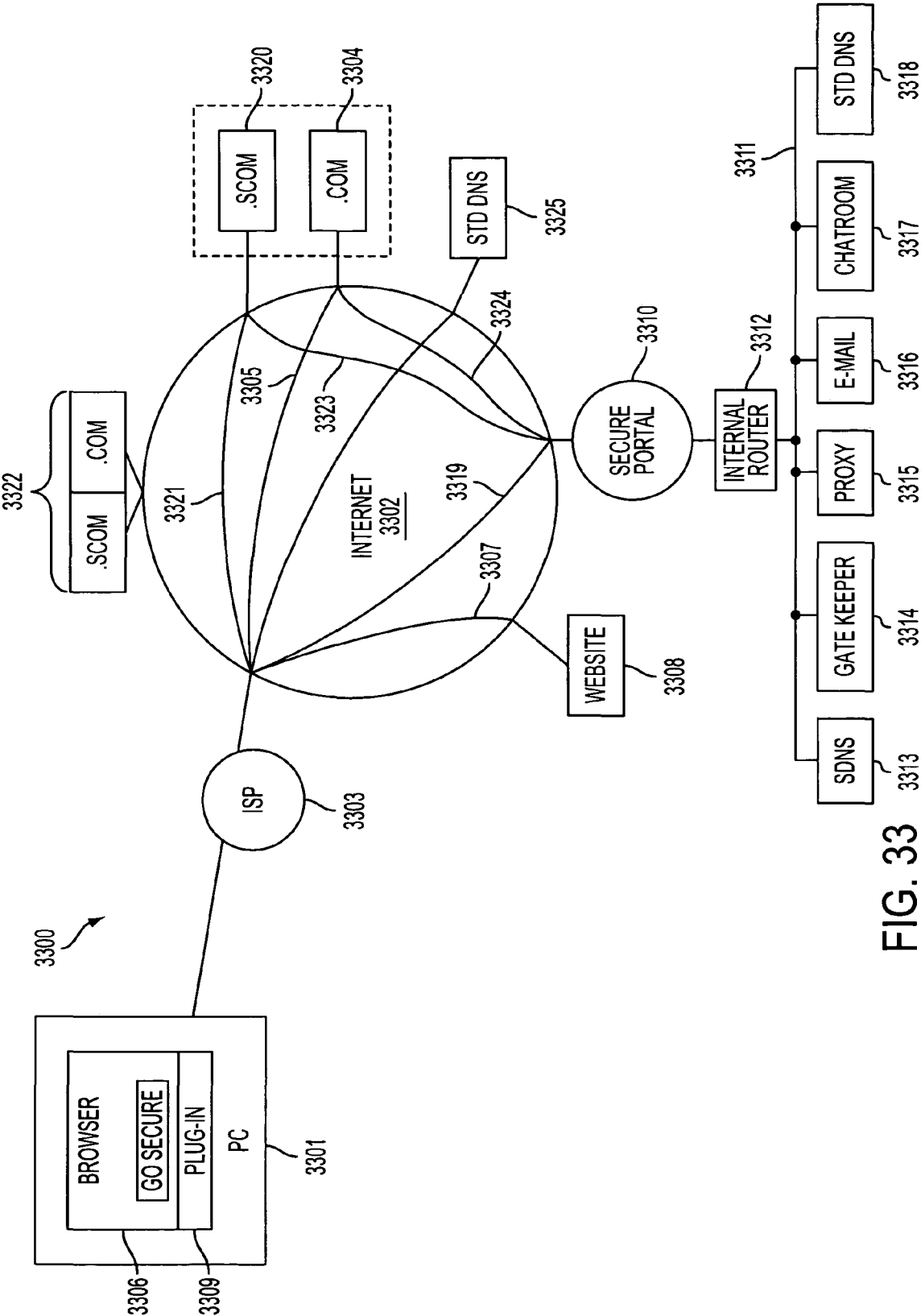


FIG. 33

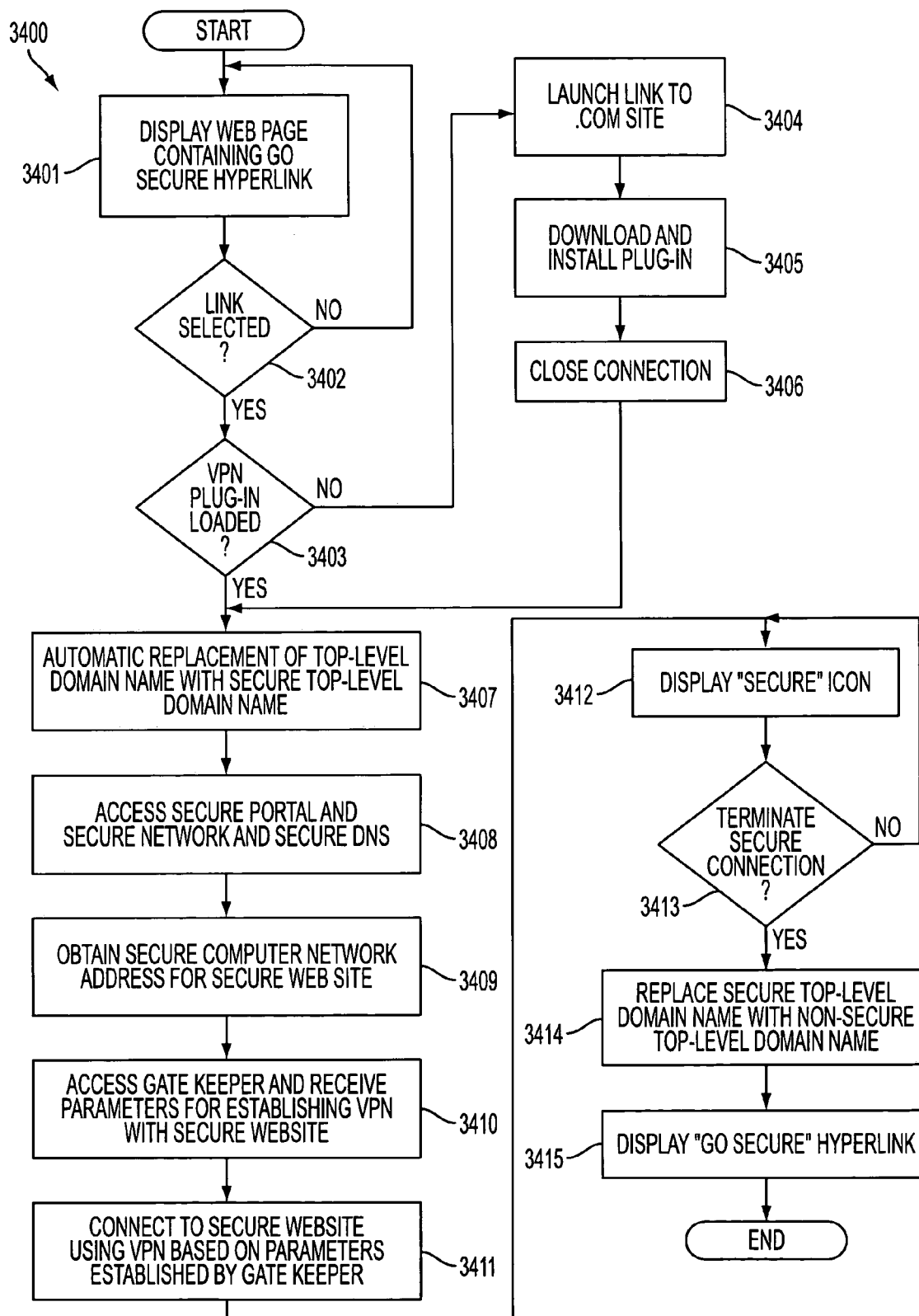


FIG. 34



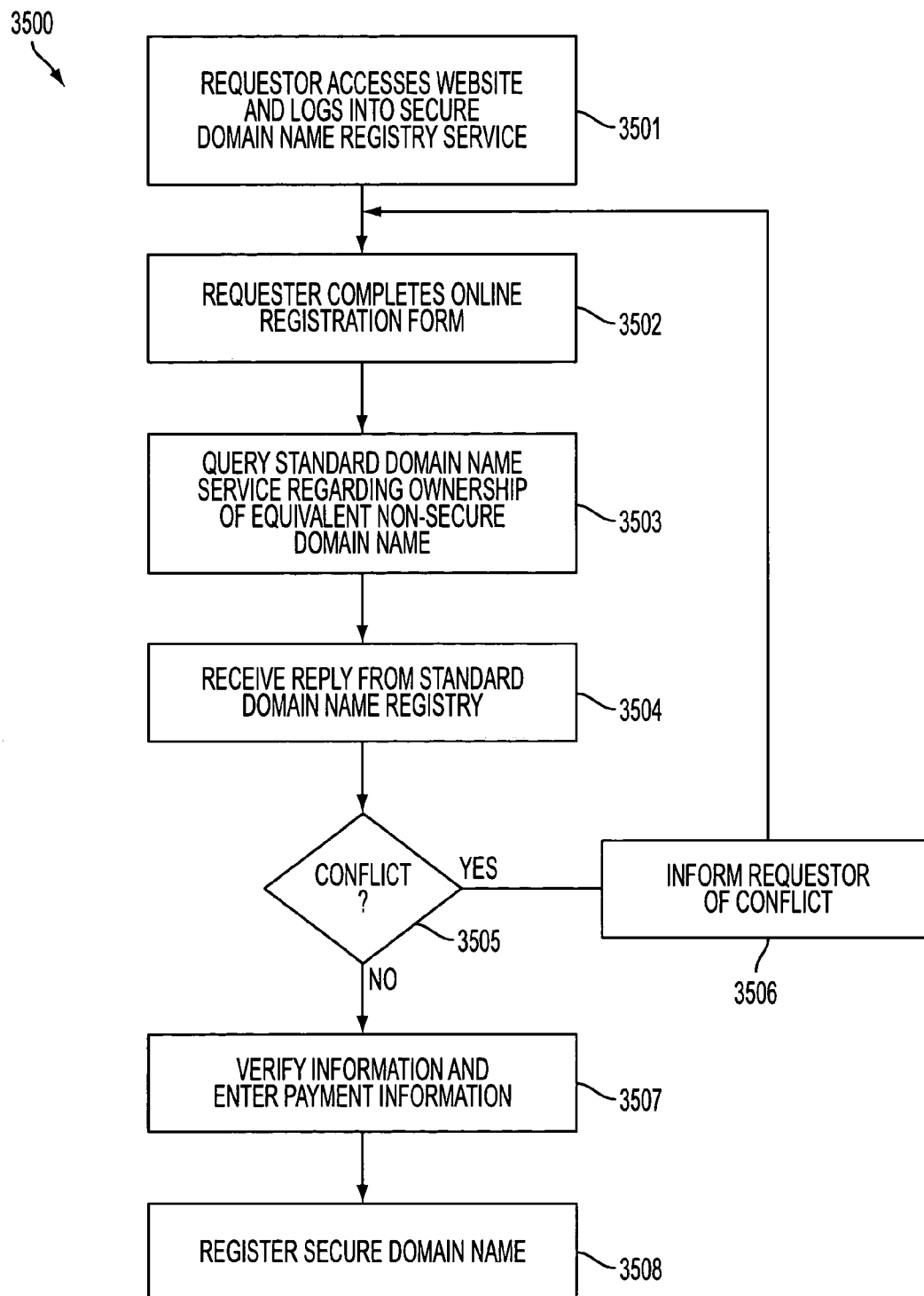


FIG. 35

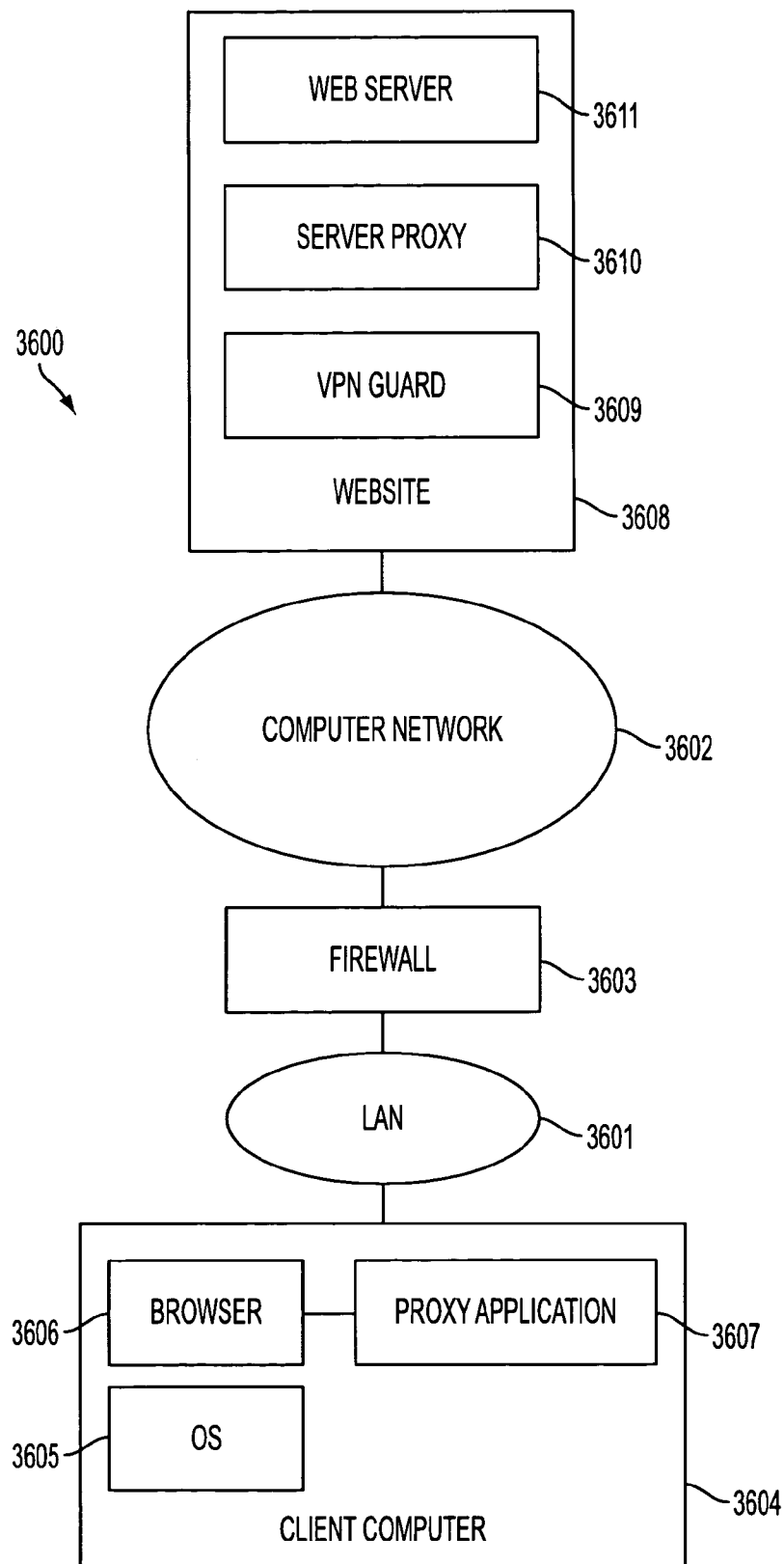


FIG. 36

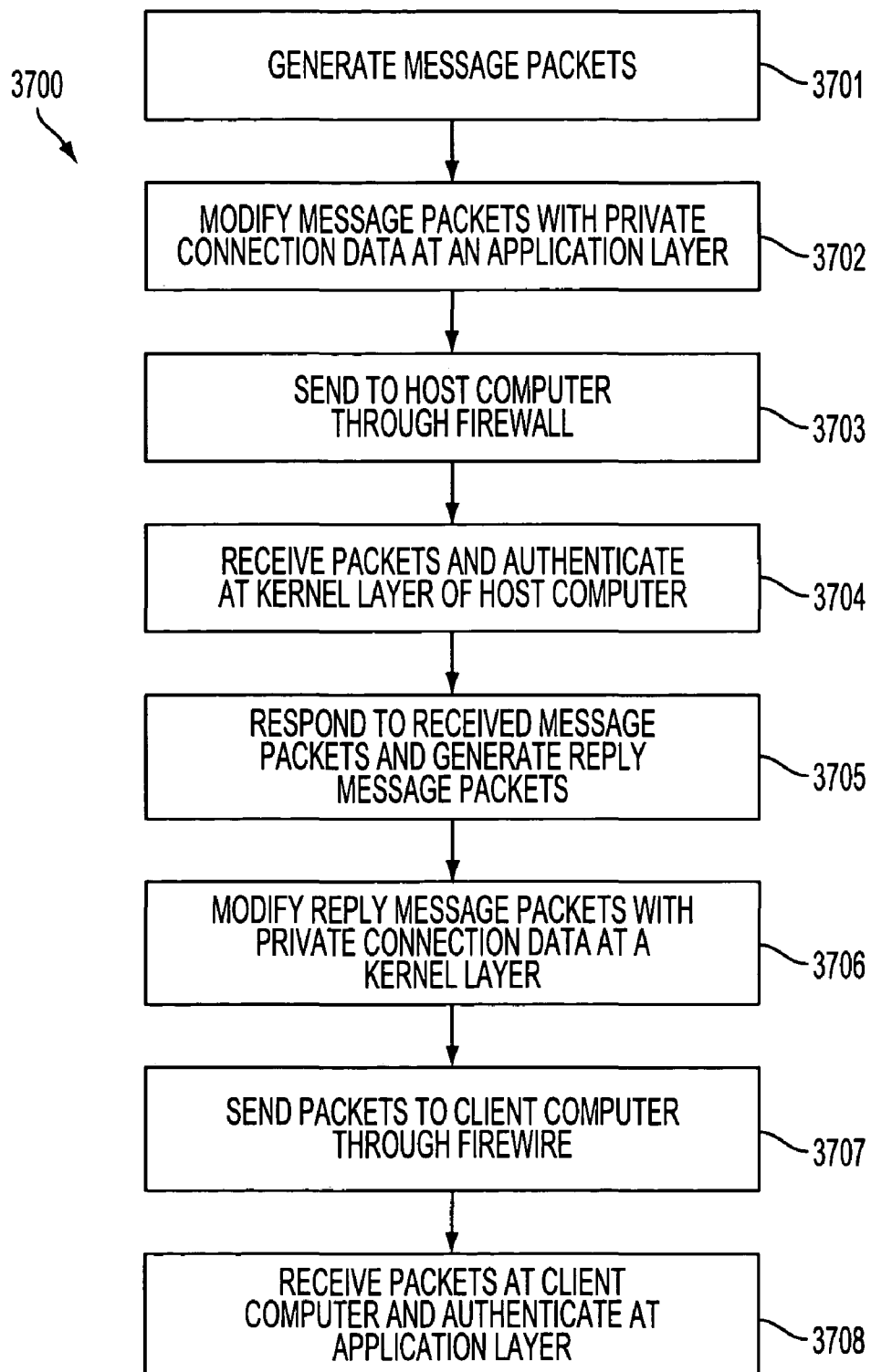


FIG. 37

## US 7,418,504 B2

1

**AGILE NETWORK PROTOCOL FOR SECURE  
COMMUNICATIONS USING SECURE  
DOMAIN NAMES****CROSS-REFERENCE TO RELATED  
APPLICATIONS**

This application claims priority from and is a continuation patent application of U.S. application Ser. No. 09/558,210, filed Apr. 26, 2000 now abandoned, which is a continuation-in-part patent application of previously-filed U.S. application Ser. No. 09/504,783, filed on Feb. 15, 2000, now U.S. Pat. No. 6,502,135, issued Dec. 31, 2002, which claims priority from and is a continuation-in-part patent application of previously-filed U.S. application Ser. No. 09/429,643, filed on Oct. 29, 1999 now U.S. Pat. No. 7,010,604. The subject matter of U.S. application Ser. No. 09/429,643, which is bodily incorporated herein, derives from provisional U.S. application Nos. 60/106,261 (filed Oct. 30, 1998) and 60/137,704 (filed Jun. 7, 1999). The present application is also related to U.S. application Ser. No. 09/558,209, filed Apr. 26, 2000, and which is incorporated by reference herein.

**GOVERNMENT CONTRACT RIGHTS**

This invention was made with Government support under Contract No. 360000-1999-000000-QC-000-000 awarded by the Central Intelligence Agency. The Government has certain rights in the invention.

**BACKGROUND OF THE INVENTION**

A tremendous variety of methods have been proposed and implemented to provide security and anonymity for communications over the Internet. The variety stems, in part, from the different needs of different Internet users. A basic heuristic framework to aid in discussing these different security techniques is illustrated in FIG. 1. Two terminals, an originating terminal 100 and a destination terminal 110 are in communication over the Internet. It is desired for the communications to be secure, that is, immune to eavesdropping. For example, terminal 100 may transmit secret information to terminal 110 over the Internet 107. Also, it may be desired to prevent an eavesdropper from discovering that terminal 100 is in communication with terminal 110. For example, if terminal 100 is a user and terminal 110 hosts a web site, terminal 100's user may not want anyone in the intervening networks to know what web sites he is "visiting." Anonymity would thus be an issue, for example, for companies that want to keep their market research interests private and thus would prefer to prevent outsiders from knowing which web-sites or other Internet resources they are "visiting." These two security issues may be called data security and anonymity, respectively.

Data security is usually tackled using some form of data encryption. An encryption key 48 is known at both the originating and terminating terminals 100 and 110. The keys may be private and public at the originating and destination terminals 100 and 110, respectively or they may be symmetrical keys (the same key is used by both parties to encrypt and decrypt). Many encryption methods are known and usable in this context.

To hide traffic from a local administrator or ISP, a user can employ a local proxy server in communicating over an encrypted channel with an outside proxy such that the local administrator or ISP only sees the encrypted traffic. Proxy servers prevent destination servers from determining the

2

identities of the originating clients. This system employs an intermediate server interposed between client and destination server. The destination server sees only the Internet Protocol (IP) address of the proxy server and not the originating client.

5 The target server only sees the address of the outside proxy. This scheme relies on a trusted outside proxy server. Also, proxy schemes are vulnerable to traffic analysis methods of determining identities of transmitters and receivers. Another important limitation of proxy servers is that the server knows the identities of both calling and called parties. In many instances, an originating terminal, such as terminal A, would prefer to keep its identity concealed from the proxy, for example, if the proxy server is provided by an Internet service provider (ISP).

15 To defeat traffic analysis, a scheme called Chaum's mixes employs a proxy server that transmits and receives fixed length messages, including dummy messages. Multiple originating terminals are connected through a mix (a server) to multiple target servers. It is difficult to tell which of the originating terminals are communicating to which of the connected target servers, and the dummy messages confuse eavesdroppers' efforts to detect communicating pairs by analyzing traffic. A drawback is that there is a risk that the mix server could be compromised. One way to deal with this risk is to spread the trust among multiple mixes. If one mix is compromised, the identities of the originating and target terminals may remain concealed. This strategy requires a number of alternative mixes so that the intermediate servers interposed between the originating and target terminals are not determinable except by compromising more than one mix. 25 The strategy wraps the message with multiple layers of encrypted addresses. The first mix in a sequence can decrypt only the outer layer of the message to reveal the next destination mix in sequence. The second mix can decrypt the message to reveal the next mix and so on. The target server receives the message and, optionally, a multi-layer encrypted payload containing return information to send data back in the same fashion. The only way to defeat such a mix scheme is to collude among mixes. If the packets are all fixed-length and intermixed with dummy packets, there is no way to do any kind of traffic analysis.

30 Still another anonymity technique, called 'crowds,' protects the identity of the originating terminal from the intermediate proxies by providing that originating terminals belong to groups of proxies called crowds. The crowd proxies are interposed between originating and target terminals. Each proxy through which the message is sent is randomly chosen by an upstream proxy. Each intermediate proxy can send the message either to another randomly chosen proxy in the "crowd" or to the destination. Thus, even crowd members cannot determine if a preceding proxy is the originator of the message or if it was simply passed from another proxy.

35 ZKS (Zero-Knowledge Systems) Anonymous IP Protocol allows users to select up to any of five different pseudonyms, while desktop software encrypts outgoing traffic and wraps it in User Datagram Protocol (UDP) packets. The first server in a 2+-hop system gets the UDP packets, strips off one layer of encryption to add another, then sends the traffic to the next server, which strips off yet another layer of encryption and adds a new one. The user is permitted to control the number of hops. At the final server, traffic is decrypted with an untraceable IP address. The technique is called onion-routing. This method can be defeated using traffic analysis. For a simple example, bursts of packets from a user during low-duty periods can reveal the identities of sender and receiver.

40 Firewalls attempt to protect LANs from unauthorized access and hostile exploitation or damage to computers con-

## US 7,418,504 B2

3

nected to the LAN. Firewalls provide a server through which all access to the LAN must pass. Firewalls are centralized systems that require administrative overhead to maintain. They can be compromised by virtual-machine applications ("applets"). They instill a false sense of security that leads to security breaches for example by users sending sensitive information to servers outside the firewall or encouraging use of modems to sidestep the firewall security. Firewalls are not useful for distributed systems such as business travelers, extranets, small teams, etc.

## SUMMARY OF THE INVENTION

A secure mechanism for communicating over the internet, including a protocol referred to as the Tunneled Agile Routing Protocol (TARP), uses a unique two-layer encryption format and special TARP routers. TARP routers are similar in function to regular IP routers. Each TARP router has one or more IP addresses and uses normal IP protocol to send IP packet messages ("packets" or "datagrams"). The IP packets exchanged between TARP terminals via TARP routers are actually encrypted packets whose true destination address is concealed except to TARP routers and servers. The normal or "clear" or "outside" IP header attached to TARP IP packets contains only the address of a next hop router or destination server. That is, instead of indicating a final destination in the destination field of the IP header, the TARP packet's IP header always points to a next-hop in a series of TARP router hops, or to the final destination. This means there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet since the destination could always be next-hop TARP router as well as the final destination.

Each TARP packet's true destination is concealed behind a layer of encryption generated using a link key. The link key is the encryption key used for encrypted communication between the hops intervening between an originating TARP terminal and a destination TARP terminal. Each TARP router can remove the outer layer of encryption to reveal the destination router for each TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal by the sender/receiver IP numbers in the cleartext IP header.

Once the outer layer of encryption is removed, the TARP router determines the final destination. Each TARP packet **140** undergoes a minimum number of hops to help foil traffic analysis. The hops may be chosen at random or by a fixed value. As a result, each TARP packet may make random trips among a number of geographically disparate routers before reaching its destination. Each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined. This feature is called agile routing. The fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. The associated advantages have to do with the inner layer of encryption discussed below. Agile routing is combined with another feature that furthers this purpose; a feature that ensures that any message is broken into multiple packets.

The IP address of a TARP router can be changed, a feature called IP agility. Each TARP router, independently or under direction from another TARP terminal or router, can change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be cor-

4

related at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs.

The message payload is hidden behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the intervening TARP routers. The session key is used to decrypt the payloads of the TARP packets permitting the data stream to be reconstructed.

Communication may be made private using link and session keys, which in turn may be shared and used according to any desired method. For example, public/private keys or symmetric keys may be used.

To transmit a data stream, a TARP originating terminal constructs a series of TARP packets from a series of IP packets generated by a network (IP) layer process. (Note that the terms "network layer," "data link layer," "application layer," etc. used in this specification correspond to the Open Systems Interconnection (OSI) network terminology.) The payloads of these packets are assembled into a block and chain-block encrypted using the session key. This assumes, of course, that all the IP packets are destined for the same TARP terminal. The block is then interleaved and the interleaved encrypted block is broken into a series of payloads, one for each TARP packet to be generated. Special TARP headers IPT are then added to each payload using the IP headers from the data stream packets. The TARP headers can be identical to normal IP headers or customized in some way. They should contain a formula or data for deinterleaving the data at the destination TARP terminal, a time-to-live (TTL) parameter to indicate the number of hops still to be executed, a data type identifier which indicates whether the payload contains, for example, TCP or UDP data, the sender's TARP address, the destination TARP address, and an indicator as to whether the packet contains real or decoy data or a formula for filtering out decoy data if decoy data is spread in some way through the TARP payload data.

Note that although chain-block encryption is discussed here with reference to the session key, any encryption method may be used. Preferably, as in chain block encryption, a method should be used that makes unauthorized decryption difficult without an entire result of the encryption process. Thus, by separating the encrypted block among multiple packets and making it difficult for an interloper to obtain access to all of such packets, the contents of the communications are provided an extra layer of security.

Decoy or dummy data can be added to a stream to help foil traffic analysis by reducing the peak-to-average network load. It may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to portion, or entirety, of a message, and that portion or entirety then interleaved into a number of separate packets. Considering the agile IP routing of the packets, and the attendant

## US 7,418,504 B2

5

difficulty of reconstructing an entire sequence of packets to form a single block-encrypted message element, decoy packets can significantly increase the difficulty of reconstructing an entire data stream.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Because the encryption system described above is insertable between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the Network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicating that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. In addition, it may create a subprocess that maintains the original IP address and continues interacting with the attacker in some manner.

Decoy packets may be generated by each TARP terminal on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis.

In various other embodiments of the invention, a scalable version of the system may be constructed in which a plurality of IP addresses are preassigned to each pair of communicating nodes in the network. Each pair of nodes agrees upon an algorithm for "hopping" between IP addresses (both sending and receiving), such that an eavesdropper sees apparently continuously random IP address pairs (source and destination) for packets transmitted between the pair. Overlapping or

6

"reusable" IP addresses may be allocated to different users on the same subnet, since each node merely verifies that a particular packet includes a valid source/destination pair from the agreed-upon algorithm. Source/destination pairs are preferably not reused between any two nodes during any given end-to-end session, though limited IP block sizes or lengthy sessions might require it.

Further improvements described in this continuation-in-part application include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities

The present invention provides key technologies for implementing a secure virtual Internet by using a new agile network protocol that is built on top of the existing Internet protocol (IP). The secure virtual Internet works over the existing Internet infrastructure, and interfaces with client applications the same way as the existing Internet. The key technologies provided by the present invention that support the secure virtual Internet include a "one-click" and "no-click" technique to become part of the secure virtual Internet, a secure domain name service (SDNS) for the secure virtual Internet, and a new approach for interfacing specific client applications onto the secure virtual Internet. According to the invention, the secure domain name service interfaces with existing applications, in addition to providing a way to register and serve domain names and addresses.

According to one aspect of the present invention, a user can conveniently establish a VPN using a "one-click" or a "no-click" technique without being required to enter user identification information, a password and/or an encryption key for establishing a VPN. The advantages of the present invention are provided by a method for establishing a secure communication link between a first computer and a second computer over a computer network, such as the Internet. In one embodiment, a secure communication mode is enabled at a first computer without a user entering any cryptographic information for establishing the secure communication mode of communication, preferably by merely selecting an icon displayed on the first computer. Alternatively, the secure communication mode of communication can be enabled by entering a command into the first computer. Then, a secure communication link is established between the first computer and a second computer over a computer network based on the enabled secure communication mode of communication. According to the invention, it is determined whether a secure communication software module is stored on the first computer in response to the step of enabling the secure communication mode of communication. A predetermined computer network address is then accessed for loading the secure communication software module when the software module is not stored on the first computer. Subsequently, the proxy software module is stored in the first computer. The secure communication link is a virtual private network communication link over the computer network. Preferably, the virtual private network can be based on inserting into each data packet one or more data values that vary according to a pseudo-random sequence. Alternatively, the virtual private network can be based on a computer network address hopping regime that is

## US 7,418,504 B2

7

used to pseudorandomly change computer network addresses or other data values in packets transmitted between the first computer and the second computer, such that the second computer compares the data values in each data packet transmitted between the first computer and the second computer to a moving window of valid values. Yet another alternative provides that the virtual private network can be based on a comparison between a discriminator field in each data packet to a table of valid discriminator fields maintained for the first computer.

According to another aspect of the invention, a command is entered to define a setup parameter associated with the secure communication link mode of communication. Consequently, the secure communication mode is automatically established when a communication link is established over the computer network.

The present invention also provides a computer system having a communication link to a computer network, and a display showing a hyperlink for establishing a virtual private network through the computer network. When the hyperlink for establishing the virtual private network is selected, a virtual private network is established over the computer network. A non-standard top-level domain name is then sent over the virtual private network communication to a predetermined computer network address, such as a computer network address for a secure domain name service (SDNS).

The present invention provides a domain name service that provides secure computer network addresses for secure, non-standard top-level domain names. The advantages of the present invention are provided by a secure domain name service for a computer network that includes a portal connected to a computer network, such as the Internet, and a domain name database connected to the computer network through the portal. According to the invention, the portal authenticates a query for a secure computer network address, and the domain name database stores secure computer network addresses for the computer network. Each secure computer network address is based on a non-standard top-level domain name, such as .scom, .sorg, .snet, .snet, .sedu, .smil and .sint.

The present invention provides a way to encapsulate existing application network traffic at the application layer of a client computer so that the client application can securely communicate with a server protected by an agile network protocol. The advantages of the present invention are provided by a method for communicating using a private communication link between a client computer and a server computer over a computer network, such as the Internet. According to the invention, an information packet is sent from the client computer to the server computer over the computer network. The information packet contains data that is inserted into the payload portion of the packet at the application layer of the client computer and is used for forming a virtual private connection between the client computer and the server computer. The modified information packet can be sent through a firewall before being sent over the computer network to the server computer and by working on top of existing protocols (i.e., UDP, ICMP and TCP), the present invention more easily penetrates the firewall. The information packet is received at a kernel layer of an operating system on the server side. It is then determined at the kernel layer of the operating system on the host computer whether the information packet contains the data that is used for forming the virtual private connection. The server side replies by sending an information packet to the client computer that has been modified at the kernel layer to containing virtual private connection information in the payload portion of the reply infor-

8

mation packet. Preferably, the information packet from the client computer and the reply information packet from the server side are each a UDP protocol information packet. Alternative, both information packets could be a TCP/IP protocol information packet, or an ICMP protocol information packet.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is an illustration of secure communications over the Internet according to a prior art embodiment.

FIG. 2 is an illustration of secure communications over the Internet according to an embodiment of the invention.

FIG. 3a is an illustration of a process of forming a tunneled IP packet according to an embodiment of the invention.

FIG. 3b is an illustration of a process of forming a tunneled IP packet according to another embodiment of the invention.

FIG. 4 is an illustration of an OSI layer location of processes that may be used to implement the invention.

FIG. 5 is a flow chart illustrating a process for routing a tunneled packet according to an embodiment of the invention.

FIG. 6 is a flow chart illustrating a process for forming a tunneled packet according to an embodiment of the invention.

FIG. 7 is a flow chart illustrating a process for receiving a tunneled packet according to an embodiment of the invention.

FIG. 8 shows how a secure session is established and synchronized between a client and a TARP router.

FIG. 9 shows an IP address hopping scheme between a client computer and TARP router using transmit and receive tables in each computer.

FIG. 10 shows physical link redundancy among three Internet Service Providers (ISPs) and a client computer.

FIG. 11 shows how multiple IP packets can be embedded into a single "frame" such as an Ethernet frame, and further shows the use of a discriminator field to camouflage true packet recipients.

FIG. 12A shows a system that employs hopped hardware addresses, hopped IP addresses, and hopped discriminator fields.

FIG. 12B shows several different approaches for hopping hardware addresses, IP addresses, and discriminator fields in combination.

FIG. 13 shows a technique for automatically re-establishing synchronization between sender and receiver through the use of a partially public sync value.

FIG. 14 shows a "checkpoint" scheme for regaining synchronization between a sender and recipient.

FIG. 15 shows further details of the checkpoint scheme of FIG. 14.

FIG. 16 shows how two addresses can be decomposed into a plurality of segments for comparison with presence vectors.

FIG. 17 shows a storage array for a receiver's active addresses.

FIG. 18 shows the receiver's storage array after receiving a sync request.

FIG. 19 shows the receiver's storage array after new addresses have been generated.

FIG. 20 shows a system employing distributed transmission paths.

FIG. 21 shows a plurality of link transmission tables that can be used to route packets in the system of FIG. 20.

FIG. 22A shows a flowchart for adjusting weight value distributions associated with a plurality of transmission links.

FIG. 22B shows a flowchart for setting a weight value to zero if a transmitter turns off.

## US 7,418,504 B2

9

FIG. 23 shows a system employing distributed transmission paths with adjusted weight value distributions for each path.

FIG. 24 shows an example using the system of FIG. 23.

FIG. 25 shows a conventional domain-name look-up service.

FIG. 26 shows a system employing a DNS proxy server with transparent VPN creation.

FIG. 27 shows steps that can be carried out to implement transparent VPN creation based on a DNS look-up function.

FIG. 28 shows a system including a link guard function that prevents packet overloading on a low-bandwidth link LOW BW.

FIG. 29 shows one embodiment of a system employing the principles of FIG. 28.

FIG. 30 shows a system that regulates packet transmission rates by throttling the rate at which synchronizations are performed.

FIG. 31 shows a signaling server 3101 and a transport server 3102 used to establish a VPN with a client computer.

FIG. 32 shows message flows relating to synchronization protocols of FIG. 31.

FIG. 33 shows a system block diagram of a computer network in which the "one-click" secure communication link of the present invention is suitable for use.

FIG. 34 shows a flow diagram for installing and establishing a "one-click" secure communication link over a computer network according to the present invention.

FIG. 35 shows a flow diagram for registering a secure domain name according to the present invention.

FIG. 36 shows a system block diagram of a computer network in which a private connection according to the present invention can be configured to more easily traverse a firewall between two computer networks.

FIG. 37 shows a flow diagram for establishing a virtual private connection that is encapsulated using an existing network protocol.

## DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. 2, a secure mechanism for communicating over the internet employs a number of special routers or servers, called TARP routers 122-127 that are similar to regular IP routers 128-132 in that each has one or more IP addresses and uses normal IP protocol to send normal-looking IP packet messages, called TARP packets 140. TARP packets 140 are identical to normal IP packet messages that are routed by regular IP routers 128-132 because each TARP packet 140 contains a destination address as in a normal IP packet. However, instead of indicating a final destination in the destination field of the IP header, the TARP packet's 140 IP header always points to a next-hop in a series of TARP router hops, or the final destination, TARP terminal 110. Because the header of the TARP packet contains only the next-hop destination, there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet 140 since the destination could always be the next-hop TARP router as well as the final destination, TARP terminal 110.

Each TARP packet's true destination is concealed behind an outer layer of encryption generated using a link key 146. The link key 146 is the encryption key used for encrypted communication between the end points (TARP terminals or TARP routers) of a single link in the chain of hops connecting the originating TARP terminal 100 and the destination TARP terminal 110. Each TARP router 122-127, using the link key 146 it uses to communicate with the previous hop in a chain,

10

can use the link key to reveal the true destination of a TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal (which may indicate the link key used) by the sender field of the clear IP header. Alternatively, this identity may be hidden behind another layer of encryption in available bits in the clear IP header. Each TARP router, upon receiving a TARP message, determines if the message is a TARP message by using authentication data in the TARP packet. This could be recorded in available bytes in the TARP packet's IP header. Alternatively, TARP packets could be authenticated by attempting to decrypt using the link key 146 and determining if the results are as expected. The former may have computational advantages because it does not involve a decryption process.

Once the outer layer of decryption is completed by a TARP router 122-127, the TARP router determines the final destination. The system is preferably designed to cause each TARP packet 140 to undergo a minimum number of hops to help foil traffic analysis. The time to live counter in the IP header of the TARP message may be used to indicate a number of TARP router hops yet to be completed. Each TARP router then would decrement the counter and determine from that whether it should forward the TARP packet 140 to another TARP router 122-127 or to the destination TARP terminal 110. If the time to live counter is zero or below zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to the destination TARP terminal 110. If the time to live counter is above zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to a TARP router 122-127 that the current TARP terminal chooses at random. As a result, each TARP packet 140 is routed through some minimum number of hops of TARP routers 122-127 which are chosen at random.

Thus, each TARP packet, irrespective of the traditional factors determining traffic in the Internet, makes random trips among a number of geographically disparate routers before reaching its destination and each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined as described above. This feature is called agile routing. For reasons that will become clear shortly, the fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. Agile routing is combined with another feature that furthers this purpose, a feature that ensures that any message is broken into multiple packets.

A TARP router receives a TARP packet when an IP address used by the TARP router coincides with the IP address in the TARP packet's IP header IP<sub>c</sub>. The IP address of a TARP router, however, may not remain constant. To avoid and manage attacks, each TARP router, independently or under direction from another TARP terminal or router, may change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs. In reality, whenever a TARP router looks up the address of a destination in the encrypted header, it must convert a TARP address to a real IP address using its LUT.



## US 7,418,504 B2

## 11

While every TARP router receiving a TARP packet has the ability to determine the packet's final destination, the message payload is embedded behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the TARP routers **122-127** intervening between the originating **100** and destination **110** TARP terminals. The session key is used to decrypt the payloads of the TARP packets **140** permitting an entire message to be reconstructed.

In one embodiment, communication may be made private using link and session keys, which in turn may be shared and used according any desired method. For example, a public key or symmetric keys may be communicated between link or session endpoints using a public key method. Any of a variety of other mechanisms for securing data to ensure that only authorized computers can have access to the private information in the TARP packets **140** may be used as desired.

Referring to FIG. 3a, to construct a series of TARP packets, a data stream **300** of IP packets **207a**, **207b**, **207c**, etc., such series of packets being formed by a network (IP) layer process, is broken into a series of small sized segments. In the present example, equal-sized segments **1-9** are defined and used to construct a set of interleaved data packets A, B, and C. Here it is assumed that the number of interleaved packets A, B, and C formed is three and that the number of IP packets **207a-207c** used to form the three interleaved packets A, B, and C is exactly three. Of course, the number of IP packets spread over a group of interleaved packets may be any convenient number as may be the number of interleaved packets over which the incoming data stream is spread. The latter, the number of interleaved packets over which the data stream is spread, is called the interleave window.

To create a packet, the transmitting software interleaves the normal IP packets **207a** et. seq. to form a new set of interleaved payload data **320**. This payload data **320** is then encrypted using a session key to form a set of session-key-encrypted payload data **330**, each of which, A, B, and C, will form the payload of a TARP packet. Using the IP header data, from the original packets **207a-207c**, new TARP headers  $IP_T$  are formed. The TARP headers  $IP_T$  can be identical to normal IP headers or customized in some way. In a preferred embodiment, the TARP headers  $IP_T$  are IP headers with added data providing the following information required for routing and reconstruction of messages, some of which data is ordinarily, or capable of being, contained in normal IP headers:

1. A window sequence number—an identifier that indicates where the packet belongs in the original message sequence.
2. An interleave sequence number—an identifier that indicates the interleaving sequence used to form the packet so that the packet can be deinterleaved along with other packets in the interleave window.
3. A time-to-live (TTL) datum—indicates the number of TARP-router-hops to be executed before the packet reaches its destination. Note that the TTL parameter may provide a datum to be used in a probabilistic formula for determining whether to route the packet to the destination or to another hop.
4. Data type identifier—indicates whether the payload contains, for example, TCP or UDP data.
5. Sender's address—indicates the sender's address in the TARP network.
6. Destination address—indicates the destination terminal's address in the TARP network.
7. Decoy/Real—an indicator of whether the packet contains real message data or dummy decoy data or a combination.

## 12

Obviously, the packets going into a single interleave window must include only packets with a common destination. Thus, it is assumed in the depicted example that the IP headers of IP packets **207a-207c** all contain the same destination address or at least will be received by the same terminal so that they can be deinterleaved. Note that dummy or decoy data or packets can be added to form a larger interleave window than would otherwise be required by the size of a given message. Decoy or dummy data can be added to a stream to help foil traffic analysis by leveling the load on the network. Thus, it may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or the entirety, of a message, and that portion or entirety then interleaved into a number of separate packets.

Referring to FIG. 3b, in an alternative mode of TARP packet construction, a series of IP packets are accumulated to make up a predefined interleave window. The payloads of the packets are used to construct a single block **520** for chain block encryption using the session key. The payloads used to form the block are presumed to be destined for the same terminal. The block size may coincide with the interleave window as depicted in the example embodiment of FIG. 3b. After encryption, the encrypted block is broken into separate payloads and segments which are interleaved as in the embodiment of FIG. 3a. The resulting interleaved packets A, B, and C, are then packaged as TARP packets with TARP headers as in the Example of FIG. 3a. The remaining process is as shown in, and discussed with reference to, FIG. 3a.

Once the TARP packets **340** are formed, each entire TARP packet **340**, including the TARP header  $IP_T$ , is encrypted using the link key for communication with the first-hop-TARP router. The first hop TARP router is randomly chosen. A final unencrypted IP header  $IP_C$  is added to each encrypted TARP packet **340** to form a normal IP packet **360** that can be transmitted to a TARP router. Note that the process of constructing the TARP packet **360** does not have to be done in stages as described. The above description is just a useful heuristic for describing the final product, namely, the TARP packet.

Note that, TARP header  $IP_T$  could be a completely custom header configuration with no similarity to a normal IP header except that it contain the information identified above. This is so since this header is interpreted by only TARP routers.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Referring to FIG. 4, a TARP transceiver **405** can be an originating terminal **100**, a destination terminal **110**, or a TARP router **122-127**. In each TARP Transceiver **405**, a transmitting process is generated to receive normal packets from the Network (IP) layer and generate TARP packets for communication over the network. A receiving process is generated to receive normal IP packets containing TARP packets and generate from these normal IP packets which are "passed

## US 7,418,504 B2

13

up” to the Network (IP) layer. Note that where the TARP Transceiver 405 is a router, the received TARP packets 140 are not processed into a stream of IP packets 415 because they need only be authenticated as proper TARP packets and then passed to another TARP router or a TARP destination terminal 110. The intervening process, a “TARP Layer” 420, could be combined with either the data link layer 430 or the Network layer 410. In either case, it would intervene between the data link layer 430 so that the process would receive regular IP packets containing embedded TARP packets and “hand up” a series of reassembled IP packets to the Network layer 410. As an example of combining the TARP layer 420 with the data link layer 430, a program may augment the normal processes running a communications card, for example, an Ethernet card. Alternatively, the TARP layer processes may form part of a dynamically loadable module that is loaded and executed to support communications between the network and data link layers.

Because the encryption system described above can be inserted between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

Note that IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of “attacks.” The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicates that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. To accomplish this, the TARP process will construct a TARP-formatted message, in the style of Internet Control Message Protocol (ICMP) datagrams as an example; this message will contain the machine’s TARP address, its previous IP address, and its new IP address. The TARP layer will transmit this packet to at least one known TARP router; then upon receipt and validation of the message, the TARP router will update its LUT with the new IP address for the stated TARP address. The TARP router will then format a similar message, and broadcast it to the other TARP routers so that they may update their LUTs. Since the total number of TARP routers on any given subnet is expected to be relatively small, this process of updating the LUTs should be relatively fast. It may not, however, work as well when there is a relatively large number of TARP routers and/or a relatively large number of clients; this has motivated a refinement of this architecture to provide scalability; this refinement has led to a second embodiment, which is discussed below.

14

Upon detection of an attack, the TARP process may also create a subprocess that maintains the original IP address and continues interacting with the attacker. The latter may provide an opportunity to trace the attacker or study the attacker’s methods (called “fishbowling” drawing upon the analogy of a small fish in a fish bowl that “thinks” it is in the ocean but is actually under captive observation). A history of the communication between the attacker and the abandoned (fish-bowled) IP address can be recorded or transmitted for human analysis or further synthesized for purposes of responding in some way.

As mentioned above, decoy or dummy data or packets can be added to outgoing data streams by TARP terminals or routers. In addition to making it convenient to spread data over a larger number of separate packets, such decoy packets can also help to level the load on inactive portions of the Internet to help foil traffic analysis efforts.

Decoy packets may be generated by each TARP terminal 100, 110 or each router 122-127 on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. That is, when a series of messages are received, the decoy packet generation rate may be increased. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis. The rate of reception of packets, decoy or otherwise, may be indicated to the decoy packet dropping and generating processes through perishable decoy and regular packet counters. (A perishable counter is one that resets or decrements its value in response to time so that it contains a high value when it is incremented in rapid succession and a small value when incremented either slowly or a small number of times in rapid succession.) Note that destination TARP terminal 110 may generate decoy packets equal in number and size to those TARP packets received to make it appear it is merely routing packets and is therefore not the destination terminal.

Referring to FIG. 5, the following particular steps may be employed in the above-described method for routing TARP packets.

- S0. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.
- S2. The TARP packet may be probed in some way to authenticate the packet before attempting to decrypt it using the link key. That is, the router may determine that the packet is an authentic TARP packet by performing a selected operation on some data included with the clear IP header attached to the encrypted TARP packet contained in the payload. This makes it possible to avoid performing decryption on packets that are not authentic TARP packets.

## US 7,418,504 B2

15

S3. The TARP packet is decrypted to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S4. If the packet is a decoy packet, the perishable decoy counter is incremented.

S5. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the router may choose to throw it away. If the received packet is a decoy packet and it is determined that it should be thrown away (S6), control returns to step S0.

S7. The TTL parameter of the TARP header is decremented and it is determined if the TTL parameter is greater than zero.

S8. If the TTL parameter is greater than zero, a TARP address is randomly chosen from a list of TARP addresses maintained by the router and the link key and IP address corresponding to that TARP address memorized for use in creating a new IP packet containing the TARP packet.

S9. If the TTL parameter is zero or less, the link key and IP address corresponding to the TARP address of the destination are memorized for use in creating the new IP packet containing the TARP packet.

S10. The TARP packet is encrypted using the memorized link key.

S11. An IP header is added to the packet that contains the stored IP address, the encrypted TARP packet wrapped with an IP header, and the completed packet transmitted to the next hop or destination.

Referring to FIG. 6, the following particular steps may be employed in the above-described method for generating TARP packets.

S20. A background loop operation applies an algorithm that determines the generation of decoy IP packets. The loop is interrupted when a data stream containing IP packets is received for transmission.

S21. The received IP packets are grouped into a set consisting of messages with a constant IP destination address. The set is further broken down to coincide with a maximum size of an interleave window. The set is encrypted, and interleaved into a set of payloads destined to become TARP packets.

S22. The TARP address corresponding to the IP address is determined from a lookup table and stored to generate the TARP header. An initial TTL count is generated and stored in the header. The TTL count may be random with minimum and maximum values or it may be fixed or determined by some other parameter.

S23. The window sequence numbers and interleave sequence numbers are recorded in the TARP headers of each packet.

S24. One TARP router address is randomly chosen for each TARP packet and the IP address corresponding to it stored for use in the clear IP header. The link key corresponding to this router is identified and used to encrypt TARP packets containing interleaved and encrypted data and TARP headers.

S25. A clear IP header with the first hop router's real IP address is generated and added to each of the encrypted TARP packets and the resulting packets.

Referring to FIG. 7, the following particular steps may be employed in the above-described method for receiving TARP packets.

S40. A background loop operation is performed which applies an algorithm which determines the generation of

16

decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S42. The TARP packet may be probed to authenticate the packet before attempting to decrypt it using the link key.

S43. The TARP packet is decrypted with the appropriate link key to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S44. If the packet is a decoy packet, the perishable decoy counter is incremented.

S45. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the receiver may choose to throw it away.

S46. The TARP packets are cached until all packets forming an interleave window are received.

S47. Once all packets of an interleave window are received, the packets are deinterleaved.

S48. The packets block of combined packets defining the interleave window is then decrypted using the session key.

S49. The decrypted block is then divided using the window sequence data and the IPT headers are converted into normal IPc headers. The window sequence numbers are integrated in the IPC headers.

S50. The packets are then handed up to the IP layer processes.

### 1. Scalability Enhancements

The IP agility feature described above relies on the ability to transmit IP address changes to all TARP routers. The embodiments including this feature will be referred to as "boutique" embodiments due to potential limitations in scaling these features up for a large network, such as the Internet. (The "boutique" embodiments would, however, be robust for use in smaller networks, such as small virtual private networks, for example). One problem with the boutique embodiments is that if IP address changes are to occur frequently, the message traffic required to update all routers sufficiently quickly creates a serious burden on the Internet when the TARP router and/or client population gets large. The bandwidth burden added to the networks, for example in ICMP packets, that would be used to update all the TARP routers could overwhelm the Internet for a large scale implementation that approached the scale of the Internet. In other words, the boutique system's scalability is limited.

A system can be constructed which trades some of the features of the above embodiments to provide the benefits of IP agility without the additional messaging burden. This is accomplished by IP address-hopping according to shared algorithms that govern IP addresses used between links participating in communications sessions between nodes such as TARP nodes. (Note that the IP hopping technique is also applicable to the boutique embodiment.) The IP agility feature discussed with respect to the boutique system can be modified so that it becomes decentralized under this scalable regime and governed by the above-described shared algorithm. Other features of the boutique system may be combined with this new type of IP-agility.

The new embodiment has the advantage of providing IP agility governed by a local algorithm and set of IP addresses exchanged by each communicating pair of nodes. This local governance is session-independent in that it may govern communications between a pair of nodes, irrespective of the session or end points being transferred between the directly communicating pair of nodes.

## US 7,418,504 B2

17

In the scalable embodiments, blocks of IP addresses are allocated to each node in the network. (This scalability will increase in the future, when Internet Protocol addresses are increased to 128-bit fields, vastly increasing the number of distinctly addressable nodes). Each node can thus use any of the IP addresses assigned to that node to communicate with other nodes in the network. Indeed, each pair of communicating nodes can use a plurality of source IP addresses and destination IP addresses for communicating with each other.

Each communicating pair of nodes in a chain participating in any session stores two blocks of IP addresses, called netblocks, and an algorithm and randomization seed for selecting, from each netblock, the next pair of source/destination IP addresses that will be used to transmit the next message. In other words, the algorithm governs the sequential selection of IP-address pairs, one sender and one receiver IP address, from each netblock. The combination of algorithm, seed, and netblock (IP address block) will be called a "hopblock." A router issues separate transmit and receive hopblocks to its clients. The send address and the receive address of the IP header of each outgoing packet sent by the client are filled with the send and receive IP addresses generated by the algorithm. The algorithm is "clocked" (indexed) by a counter so that each time a pair is used, the algorithm turns out a new transmit pair for the next packet to be sent.

The router's receive hopblock is identical to the client's transmit hopblock. The router uses the receive hopblock to predict what the send and receive IP address pair for the next expected packet from that client will be. Since packets can be received out of order, it is not possible for the router to predict with certainty what IP address pair will be on the next sequential packet. To account for this problem, the router generates a range of predictions encompassing the number of possible transmitted packet send/receive addresses, of which the next packet received could leap ahead. Thus, if there is a vanishingly small probability that a given packet will arrive at the router ahead of 5 packets transmitted by the client before the given packet, then the router can generate a series of 6 send/receive IP address pairs (or "hop window") to compare with the next received packet. When a packet is received, it is marked in the hop window as such, so that a second packet with the same IP address pair will be discarded. If an out-of-sequence packet does not arrive within a predetermined timeout period, it can be requested for retransmission or simply discarded from the receive table, depending upon the protocol in use for that communications session, or possibly by convention.

When the router receives the client's packet, it compares the send and receive IP addresses of the packet with the next N predicted send and receive IP address pairs and rejects the packet if it is not a member of this set. Received packets that do not have the predicted source/destination IP addresses falling within the window are rejected, thus thwarting possible hackers. (With the number of possible combinations, even a fairly large window would be hard to fall into at random.) If it is a member of this set, the router accepts the packet and processes it further. This link-based IP-hopping strategy, referred to as "IHOP," is a network element that stands on its own and is not necessarily accompanied by elements of the boutique system described above. If the routing agility feature described in connection with the boutique embodiment is combined with this link-based IP-hopping strategy, the router's next step would be to decrypt the TARP header to determine the destination TARP router for the packet and determine what should be the next hop for the packet. The TARP router would then forward the packet to a random TARP

18

router or the destination TARP router with which the source TARP router has a link-based IP hopping communication established.

FIG. 8 shows how a client computer **801** and a TARP router **811** can establish a secure session. When client **801** seeks to establish an IHOP session with TARP router **811**, the client **801** sends "secure synchronization" request ("SSYN") packet **821** to the TARP router **811**. This SYN packet **821** contains the client's **801** authentication token, and may be sent to the router **811** in an encrypted format. The source and destination IP numbers on the packet **821** are the client's **801** current fixed IP address, and a "known" fixed IP address for the router **811**. (For security purposes, it may be desirable to reject any packets from outside of the local network that are destined for the router's known fixed IP address.) Upon receipt and validation of the client's **801** SSYN packet **821**, the router **811** responds by sending an encrypted "secure synchronization acknowledgment" ("SSYN ACK") **822** to the client **801**. This SSYN ACK **822** will contain the transmit and receive hopblocks that the client **801** will use when communicating with the TARP router **811**. The client **801** will acknowledge the TARP router's **811** response packet **822** by generating an encrypted SSYN ACK ACK packet **823** which will be sent from the client's **801** fixed IP address and to the TARP router's **811** known fixed IP address. The client **801** will simultaneously generate a SSYN ACK ACK packet; this SSYN ACK packet, referred to as the Secure Session Initiation (SSI) packet **824**, will be sent with the first {sender, receiver} IP pair in the client's transmit table **921** (FIG. 9), as specified in the transmit hopblock provided by the TARP router **811** in the SSYN ACK packet **822**. The TARP router **811** will respond to the SSI packet **824** with an SSI ACK packet **825**, which will be sent with the first {sender, receiver} IP pair in the TARP router's transmit table **923**. Once these packets have been successfully exchanged, the secure communications session is established, and all further secure communications between the client **801** and the TARP router **811** will be conducted via this secure session, as long as synchronization is maintained. If synchronization is lost, then the client **801** and TARP router **802** may re-establish the secure session by the procedure outlined in FIG. 8 and described above.

While the secure session is active, both the client **901** and TARP router **911** (FIG. 9) will maintain their respective transmit tables **921**, **923** and receive tables **922**, **924**, as provided by the TARP router during session synchronization **822**. It is important that the sequence of IP pairs in the client's transmit table **921** be identical to those in the TARP router's receive table **924**; similarly, the sequence of IP pairs in the client's receive table **922** must be identical to those in the router's transmit table **923**. This is required for the session synchronization to be maintained. The client **901** need maintain only one transmit table **921** and one receive table **922** during the course of the secure session. Each sequential packet sent by the client **901** will employ the next {send, receive} IP address pair in the transmit table, regardless of TCP or UDP session. The TARP router **911** will expect each packet arriving from the client **901** to bear the next IP address pair shown in its receive table.

Since packets can arrive out of order, however, the router **911** can maintain a "look ahead" buffer in its receive table, and will mark previously-received IP pairs as invalid for future packets; any future packet containing an IP pair that is in the look-ahead buffer but is marked as previously received will be discarded. Communications from the TARP router **911** to the client **901** are maintained in an identical manner; in particular, the router **911** will select the next IP address pair

## US 7,418,504 B2

19

from its transmit table **923** when constructing a packet to send to the client **901**, and the client **901** will maintain a look-ahead buffer of expected IP pairs on packets that it is receiving. Each TARP router will maintain separate pairs of transmit and receive tables for each client that is currently engaged in a secure session with or through that TARP router.

While clients receive their hopblocks from the first server linking them to the Internet, routers exchange hopblocks. When a router establishes a link-based IP-hopping communication regime with another router, each router of the pair exchanges its transmit hopblock. The transmit hopblock of each router becomes the receive hopblock of the other router. The communication between routers is governed as described by the example of a client sending a packet to the first router.

While the above strategy works fine in the IP milieu, many local networks that are connected to the Internet are Ethernet systems. In Ethernet, the IP addresses of the destination devices must be translated into hardware addresses, and vice versa, using known processes ("address resolution protocol," and "reverse address resolution protocol"). However, if the link-based IP-hopping strategy is employed, the correlation process would become explosive and burdensome. An alternative to the link-based IP hopping strategy may be employed within an Ethernet network. The solution is to provide that the node linking the Internet to the Ethernet (call it the border node) use the link-based IP-hopping communication regime to communicate with nodes outside the Ethernet LAN. Within the Ethernet LAN, each TARP node would have a single IP address which would be addressed in the conventional way. Instead of comparing the {sender, receiver} IP address pairs to authenticate a packet, the intra-LAN TARP node would use one of the IP header extension fields to do so. Thus, the border node uses an algorithm shared by the intra-LAN TARP node to generate a symbol that is stored in the free field in the IP header, and the intra-LAN TARP node generates a range of symbols based on its prediction of the next expected packet to be received from that particular source IP address. The packet is rejected if it does not fall into the set of predicted symbols (for example, numerical values) or is accepted if it does. Communications from the intra-LAN TARP node to the border node are accomplished in the same manner, though the algorithm will necessarily be different for security reasons. Thus, each of the communicating nodes will generate transmit and receive tables in a similar manner to that of FIG. 9; the intra-LAN TARP nodes transmit table will be identical to the border node's receive table, and the intra-LAN TARP node's receive table will be identical to the border node's transmit table.

The algorithm used for IP address-hopping can be any desired algorithm. For example, the algorithm can be a given pseudo-random number generator that generates numbers of the range covering the allowed IP addresses with a given seed. Alternatively, the session participants can assume a certain type of algorithm and specify simply a parameter for applying the algorithm. For example the assumed algorithm could be a particular pseudo-random number generator and the session participants could simply exchange seed values.

Note that there is no permanent physical distinction between the originating and destination terminal nodes. Either device at either end point can initiate a synchronization of the pair. Note also that the authentication/synchronization-request (and acknowledgment) and hopblock-exchange may all be served by a single message so that separate message exchanges may not be required.

As another extension to the stated architecture, multiple physical paths can be used by a client, in order to provide link redundancy and further thwart attempts at denial of service

20

and traffic monitoring. As shown in FIG. 10, for example, client **1001** can establish three simultaneous sessions with each of three TARP routers provided by different ISPs **1011**, **1012**, **1013**. As an example, the client **1001** can use three different telephone lines **1021**, **1022**, **1023** to connect to the ISPs, or two telephone lines and a cable modem, etc. In this scheme, transmitted packets will be sent in a random fashion among the different physical paths. This architecture provides a high degree of communications redundancy, with improved immunity from denial-of-service attacks and traffic monitoring.

## 2. Further Extensions

The following describes various extensions to the techniques, systems, and methods described above. As described above, the security of communications occurring between computers in a computer network (such as the Internet, an Ethernet, or others) can be enhanced by using seemingly random source and destination Internet Protocol (IP) addresses for data packets transmitted over the network. This feature prevents eavesdroppers from determining which computers in the network are communicating with each other while permitting the two communicating computers to easily recognize whether a given received data packet is legitimate or not. In one embodiment of the above-described systems, an IP header extension field is used to authenticate incoming packets on an Ethernet.

Various extensions to the previously described techniques described herein include: (1) use of hopped hardware or "MAC" addresses in broadcast type network; (2) a self-synchronization technique that permits a computer to automatically regain synchronization with a sender; (3) synchronization algorithms that allow transmitting and receiving computers to quickly re-establish synchronization in the event of lost packets or other events; and (4) a fast-packet rejection mechanism for rejecting invalid packets. Any or all of these extensions can be combined with the features described above in any of various ways.

### A. Hardware Address Hopping

Internet protocol-based communications techniques on a LAN—or across any dedicated physical medium—typically embed the IP packets within lower-level packets, often referred to as "frames." As shown in FIG. 11, for example, a first Ethernet frame **1150** comprises a frame header **1101** and two embedded IP packets IP1 and IP2, while a second Ethernet frame **1160** comprises a different frame header **1104** and a single IP packet IP3. Each frame header generally includes a source hardware address **1101A** and a destination hardware address **1101B**; other well-known fields in frame headers are omitted from FIG. 11 for clarity. Two hardware nodes communicating over a physical communication channel insert appropriate source and destination hardware addresses to indicate which nodes on the channel or network should receive the frame.

It may be possible for a nefarious listener to acquire information about the contents of a frame and/or its communicants by examining frames on a local network rather than (or in addition to) the IP packets themselves. This is especially true in broadcast media, such as Ethernet, where it is necessary to insert into the frame header the hardware address of the machine that generated the frame and the hardware address of the machine to which frame is being sent. All nodes on the network can potentially "see" all packets transmitted across the network. This can be a problem for secure communica-

## US 7,418,504 B2

21

tions, especially in cases where the communicants do not want for any third party to be able to identify who is engaging in the information exchange. One way to address this problem is to push the address-hopping scheme down to the hardware layer. In accordance with various embodiments of the invention, hardware addresses are “hopped” in a manner similar to that used to change IP addresses, such that a listener cannot determine which hardware node generated a particular message nor which node is the intended recipient.

FIG. 12A shows a system in which Media Access Control (“MAC”) hardware addresses are “hopped” in order to increase security over a network such as an Ethernet. While the description refers to the exemplary case of an Ethernet environment, the inventive principles are equally applicable to other types of communications media. In the Ethernet case, the MAC address of the sender and receiver are inserted into the Ethernet frame and can be observed by anyone on the LAN who is within the broadcast range for that frame. For secure communications, it becomes desirable to generate frames with MAC addresses that are not attributable to any specific sender or receiver.

As shown in FIG. 12A, two computer nodes 1201 and 1202 communicate over a communication channel such as an Ethernet. Each node executes one or more application programs 1203 and 1218 that communicate by transmitting packets through communication software 1204 and 1217, respectively. Examples of application programs include video conferencing, e-mail, word processing programs, telephony, and the like. Communication software 1204 and 1217 can comprise, for example, an OSI layered architecture or “stack” that standardizes various services provided at different levels of functionality.

The lowest levels of communication software 1204 and 1217 communicate with hardware components 1206 and 1214 respectively, each of which can include one or more registers 1207 and 1215 that allow the hardware to be reconfigured or controlled in accordance with various communication protocols. The hardware components (an Ethernet network interface card, for example) communicate with each other over the communication medium. Each hardware component is typically pre-assigned a fixed hardware address or MAC number that identifies the hardware component to other nodes on the network. One or more interface drivers control the operation of each card and can, for example, be configured to accept or reject packets from certain hardware addresses. As will be described in more detail below, various embodiments of the inventive principles provide for “hopping” different addresses using one or more algorithms and one or more moving windows that track a range of valid addresses to validate received packets. Packets transmitted according to one or more of the inventive principles will be generally referred to as “secure” packets or “secure communications” to differentiate them from ordinary data packets that are transmitted in the clear using ordinary, machine-correlated addresses.

One straightforward method of generating non-attributable MAC addresses is an extension of the IP hopping scheme. In this scenario, two machines on the same LAN that desire to communicate in a secure fashion exchange random-number generators and seeds, and create sequences of quasi-random MAC addresses for synchronized hopping. The implementation and synchronization issues are then similar to that of IP hopping.

This approach, however, runs the risk of using MAC addresses that are currently active on the LAN—which, in turn, could interrupt communications for those machines. Since an Ethernet MAC address is at present 48 bits in length,

22

the chance of randomly misusing an active MAC address is actually quite small. However, if that figure is multiplied by a large number of nodes (as would be found on an extensive LAN), by a large number of frames (as might be the case with packet voice or streaming video), and by a large number of concurrent Virtual Private Networks (VPNs), then the chance that a non-secure machine’s MAC address could be used in an address-hopped frame can become non-trivial. In short, any scheme that runs even a small risk of interrupting communications for other machines on the LAN is bound to receive resistance from prospective system administrators. Nevertheless, it is technically feasible, and can be implemented without risk on a LAN on which there is a small number of machines, or if all of the machines on the LAN are engaging in MAC-hopped communications.

Synchronized MAC address hopping may incur some overhead in the course of session establishment, especially if there are multiple sessions or multiple nodes involved in the communications. A simpler method of randomizing MAC addresses is to allow each node to receive and process every incident frame on the network. Typically, each network interface driver will check the destination MAC address in the header of every incident frame to see if it matches that machine’s MAC address; if there is no match, then the frame is discarded. In one embodiment, however, these checks can be disabled, and every incident packet is passed to the TARP stack for processing. This will be referred to as “promiscuous” mode, since every incident frame is processed. Promiscuous mode allows the sender to use completely random, unsynchronized MAC addresses, since the destination machine is guaranteed to process the frame. The decision as to whether the packet was truly intended for that machine is handled by the TARP stack, which checks the source and destination IP addresses for a match in its IP synchronization tables. If no match is found, the packet is discarded; if there is a match, the packet is unwrapped, the inner header is evaluated, and if the inner header indicates that the packet is destined for that machine then the packet is forwarded to the IP stack—otherwise it is discarded.

One disadvantage of purely-random MAC address hopping is its impact on processing overhead; that is, since every incident frame must be processed, the machine’s CPU is engaged considerably more often than if the network interface driver is discriminating and rejecting packets unilaterally. A compromise approach is to select either a single fixed MAC address or a small number of MAC addresses (e.g., one for each virtual private network on an Ethernet) to use for MAC-hopped communications, regardless of the actual recipient for which the message is intended. In this mode, the network interface driver can check each incident frame against one (or a few) pre-established MAC addresses, thereby freeing the CPU from the task of physical-layer packet discrimination. This scheme does not betray any useful information to an interloper on the LAN; in particular, every secure packet can already be identified by a unique packet type in the outer header. However, since all machines engaged in secure communications would either be using the same MAC address, or be selecting from a small pool of predetermined MAC addresses, the association between a specific machine and a specific MAC address is effectively broken.

In this scheme, the CPU will be engaged more often than it would be in non-secure communications (or in synchronized MAC address hopping), since the network interface driver cannot always unilaterally discriminate between secure packets that are destined for that machine, and secure packets from other VPNs. However, the non-secure traffic is easily elimi-

## US 7,418,504 B2

23

nated at the network interface, thereby reducing the amount of processing required of the CPU. There are boundary conditions where these statements would not hold, of course—e.g., if all of the traffic on the LAN is secure traffic, then the CPU would be engaged to the same degree as it is in the purely-random address hopping case; alternatively, if each VPN on the LAN uses a different MAC address, then the network interface can perfectly discriminate secure frames destined for the local machine from those constituting other VPNs. These are engineering tradeoffs that might be best handled by providing administrative options for the users when installing the software and/or establishing VPNs.

Even in this scenario, however, there still remains a slight risk of selecting MAC addresses that are being used by one or more nodes on the LAN. One solution to this problem is to formally assign one address or a range of addresses for use in MAC-hopped communications. This is typically done via an assigned numbers registration authority; e.g., in the case of Ethernet, MAC address ranges are assigned to vendors by the Institute of Electrical and Electronics Engineers (IEEE). A formally-assigned range of addresses would ensure that secure frames do not conflict with any properly-configured and properly-functioning machines on the LAN.

Reference will now be made to FIGS. 12A and 12B in order to describe the many combinations and features that follow the inventive principles. As explained above, two computer nodes **1201** and **1202** are assumed to be communicating over a network or communication medium such as an Ethernet. A communication protocol in each node (**1204** and **1217**, respectively) contains a modified element **1205** and **1216** that performs certain functions that deviate from the standard communication protocols. In particular, computer node **1201** implements a first “hop” algorithm **1208X** that selects seemingly random source and destination IP addresses (and, in one embodiment, seemingly random IP header discriminator fields) in order to transmit each packet to the other computer node. For example, node **1201** maintains a transmit table **1208** containing triplets of source (S), destination (D), and discriminator fields (DS) that are inserted into outgoing IP packet headers. The table is generated through the use of an appropriate algorithm (e.g., a random number generator that is seeded with an appropriate seed) that is known to the recipient node **1202**. As each new IP packet is formed, the next sequential entry out of the sender’s transmit table **1208** is used to populate the IP source, IP destination, and IP header extension field (e.g., discriminator field). It will be appreciated that the transmit table need not be created in advance but could instead be created on-the-fly by executing the algorithm when each packet is formed.

At the receiving node **1202**, the same IP hop algorithm **1222X** is maintained and used to generate a receive table **1222** that lists valid triplets of source IP address, destination IP address, and discriminator field. This is shown by virtue of the first five entries of transmit table **1208** matching the second five entries of receive table **1222**. (The tables may be slightly offset at any particular time due to lost packets, mis-ordered packets, or transmission delays). Additionally, node **1202** maintains a receive window **W3** that represents a list of valid IP source, IP destination, and discriminator fields that will be accepted when received as part of an incoming IP packet. As packets are received, window **W3** slides down the list of valid entries, such that the possible valid entries change over time. Two packets that arrive out of order but are nevertheless matched to entries within window **W3** will be accepted; those falling outside of window **W3** will be rejected as invalid. The length of window **W3** can be adjusted as necessary to reflect network delays or other factors.

24

Node **1202** maintains a similar transmit table **1221** for creating IP packets and frames destined for node **1201** using a potentially different hopping algorithm **1221X**, and node **1201** maintains a matching receive table **1209** using the same algorithm **1209X**. As node **1202** transmits packets to node **1201** using seemingly random IP source, IP destination, and/or discriminator fields, node **1201** matches the incoming packet values to those falling within window **W1** maintained in its receive table. In effect, transmit table **1208** of node **1201** is synchronized (i.e., entries are selected in the same order) to receive table **1222** of receiving node **1202**. Similarly, transmit table **1221** of node **1202** is synchronized to receive table **1209** of node **1201**. It will be appreciated that although a common algorithm is shown for the source, destination and discriminator fields in FIG. 12A (using, e.g., a different seed for each of the three fields), an entirely different algorithm could in fact be used to establish values for each of these fields. It will also be appreciated that one or two of the fields can be “hopped” rather than all three as illustrated.

In accordance with another aspect of the invention, hardware or “MAC” addresses are hopped instead of or in addition to IP addresses and/or the discriminator field in order to improve security in a local area or broadcast-type network. To that end, node **1201** further maintains a transmit table **1210** using a transmit algorithm **1210X** to generate source and destination hardware addresses that are inserted into frame headers (e.g., fields **1101A** and **1101B** in FIG. 11) that are synchronized to a corresponding receive table **1224** at node **1202**. Similarly, node **1202** maintains a different transmit table **1223** containing source and destination hardware addresses that is synchronized with a corresponding receive table **1211** at node **1201**. In this manner, outgoing hardware frames appear to be originating from and going to completely random nodes on the network, even though each recipient can determine whether a given packet is intended for it or not. It will be appreciated that the hardware hopping feature can be implemented at a different level in the communications protocol than the IP hopping feature (e.g., in a card driver or in a hardware card itself to improve performance).

FIG. 12B shows three different embodiments or modes that can be employed using the aforementioned principles. In a first mode referred to as “promiscuous” mode, a common hardware address (e.g., a fixed address for source and another for destination) or else a completely random hardware address is used by all nodes on the network, such that a particular packet cannot be attributed to any one node. Each node must initially accept all packets containing the common (or random) hardware address and inspect the IP addresses or discriminator field to determine whether the packet is intended for that node. In this regard, either the IP addresses or the discriminator field or both can be varied in accordance with an algorithm as described above. As explained previously, this may increase each node’s overhead-since additional processing is involved to determine whether a given packet has valid source and destination hardware addresses.

In a second mode referred to as “promiscuous per VPN” mode, a small set of fixed hardware addresses are used, with a fixed source/destination hardware address used for all nodes communicating over a virtual private network. For example, if there are six nodes on an Ethernet, and the network is to be split up into two private virtual networks such that nodes on one VPN can communicate with only the other two nodes on its own VPN, then two sets of hardware addresses could be used: one set for the first VPN and a second set for the second VPN. This would reduce the amount of overhead involved in checking for valid frames since only packets arriving from the designated VPN would need to be checked. IP addresses and



## US 7,418,504 B2

25

one or more discriminator fields could still be hopped as before for secure communication within the VPN. Of course, this solution compromises the anonymity of the VPNs (i.e., an outsider can easily tell what traffic belongs in which VPN, though he cannot correlate it to a specific machine/person). It also requires the use of a discriminator field to mitigate the vulnerability to certain types of DoS attacks. (For example, without the discriminator field, an attacker on the LAN could stream frames containing the MAC addresses being used by the VPN; rejecting those frames could lead to excessive processing overhead. The discriminator field would provide a low-overhead means of rejecting the false packets.)

In a third mode referred to as “hardware hopping” mode, hardware addresses are varied as illustrated in FIG. 12A, such that hardware source and destination addresses are changed constantly in order to provide non-attributable addressing. Variations on these embodiments are of course possible, and the invention is not intended to be limited in any respect by these illustrative examples.

## B. Extending the Address Space

Address hopping provides security and privacy. However, the level of protection is limited by the number of addresses in the blocks being hopped. A hopblock denotes a field or fields modulated on a packet-wise basis for the purpose of providing a VPN. For instance, if two nodes communicate with IP address hopping using hopblocks of 4 addresses (2 bits) each, there would be 16 possible address-pair combinations. A window of size 16 would result in most address pairs being accepted as valid most of the time. This limitation can be overcome by using a discriminator field in addition to or instead of the hopped address fields. The discriminator field would be hopped in exactly the same fashion as the address fields and it would be used to determine whether a packet should be processed by a receiver.

Suppose that two clients, each using four-bit hopblocks, would like the same level of protection afforded to clients communicating via IP hopping between two A blocks (24 address bits eligible for hopping). A discriminator field of 20 bits, used in conjunction with the 4 address bits eligible for hopping in the IP address field, provides this level of protection. A 24-bit discriminator field would provide a similar level of protection if the address fields were not hopped or ignored. Using a discriminator field offers the following advantages: (1) an arbitrarily high level of protection can be provided, and (2) address hopping is unnecessary to provide protection. This may be important in environments where address hopping would cause routing problems.

## C. Synchronization Techniques

It is generally assumed that once a sending node and receiving node have exchanged algorithms and seeds (or similar information sufficient to generate quasi-random source and destination tables), subsequent communication between the two nodes will proceed smoothly. Realistically, however, two nodes may lose synchronization due to network delays or outages, or other problems. Consequently, it is desirable to provide means for re-establishing synchronization between nodes in a network that have lost synchronization.

One possible technique is to require that each node provide an acknowledgment upon successful receipt of each packet and, if no acknowledgment is received within a certain period of time, to re-send the unacknowledged packet. This approach, however, drives up overhead costs and may be

26

prohibitive in high-throughput environments such as streaming video or audio, for example.

A different approach is to employ an automatic synchronizing technique that will be referred to herein as “self-synchronization.” In this approach, synchronization information is embedded into each packet, thereby enabling the receiver to re-synchronize itself upon receipt of a single packet if it determines that it has lost synchronization with the sender. (If communications are already in progress, and the receiver determines that it is still in sync with the sender, then there is no need to re-synchronize.) A receiver could detect that it was out of synchronization by, for example, employing a “dead-man” timer that expires after a certain period of time, wherein the timer is reset with each valid packet. A time stamp could be hashed into the public sync field (see below) to preclude packet-retry attacks.

In one embodiment, a “sync field” is added to the header of each packet sent out by the sender. This sync field could appear in the clear or as part of an encrypted portion of the packet. Assuming that a sender and receiver have selected a random-number generator (RNG) and seed value, this combination of RNG and seed can be used to generate a random-number sequence (RNS). The RNS is then used to generate a sequence of source/destination IP pairs (and, if desired, discriminator fields and hardware source and destination addresses), as described above. It is not necessary, however, to generate the entire sequence (or the first N-1 values) in order to generate the Nth random number in the sequence; if the sequence index N is known, the random value corresponding to that index can be directly generated (see below). Different RNGs (and seeds) with different fundamental periods could be used to generate the source and destination IP sequences, but the basic concepts would still apply. For the sake of simplicity, the following discussion will assume that IP source and destination address pairs (only) are hopped using a single RNG sequencing mechanism.

In accordance with a “self-synchronization” feature, a sync field in each packet header provides an index (i.e., a sequence number) into the RNS that is being used to generate IP pairs. Plugging this index into the RNG that is being used to generate the RNS yields a specific random number value, which in turn yields a specific IP pair. That is, an IP pair can be generated directly from knowledge of the RNG, seed, and index number; it is not necessary, in this scheme, to generate the entire sequence of random numbers that precede the sequence value associated with the index number provided.

Since the communicants have presumably previously exchanged RNGs and seeds, the only new information that must be provided in order to generate an IP pair is the sequence number. If this number is provided by the sender in the packet header, then the receiver need only plug this number into the RNG in order to generate an IP pair—and thus verify that the IP pair appearing in the header of the packet is valid. In this scheme, if the sender and receiver lose synchronization, the receiver can immediately re-synchronize upon receipt of a single packet by simply comparing the IP pair in the packet header to the IP pair generated from the index number. Thus, synchronized communications can be resumed upon receipt of a single packet, making this scheme ideal for multicast communications. Taken to the extreme, it could obviate the need for synchronization tables entirely; that is, the sender and receiver could simply rely on the index number in the sync field to validate the IP pair on each packet, and thereby eliminate the tables entirely.

The aforementioned scheme may have some inherent security issues associated with it namely, the placement of the sync field. If the field is placed in the outer header, then an



## US 7,418,504 B2

27

interloper could observe the values of the field and their relationship to the IP stream. This could potentially compromise the algorithm that is being used to generate the IP-address sequence, which would compromise the security of the communications. If, however, the value is placed in the inner header, then the sender must decrypt the inner header before it can extract the sync value and validate the IP pair; this opens up the receiver to certain types of denial-of-service (DoS) attacks, such as packet replay. That is, if the receiver must decrypt a packet before it can validate the IP pair, then it could potentially be forced to expend a significant amount of processing on decryption if an attacker simply retransmits previously valid packets. Other attack methodologies are possible in this scenario.

A possible compromise between algorithm security and processing speed is to split up the sync value between an inner (encrypted) and outer (unencrypted) header. That is, if the sync value is sufficiently long, it could potentially be split into a rapidly-changing part that can be viewed in the clear, and a fixed (or very slowly changing) part that must be protected. The part that can be viewed in the clear will be called the “public sync” portion and the part that must be protected will be called the “private sync” portion.

Both the public sync and private sync portions are needed to generate the complete sync value. The private portion, however, can be selected such that it is fixed or will change only occasionally. Thus, the private sync value can be stored by the recipient, thereby obviating the need to decrypt the header in order to retrieve it. If the sender and receiver have previously agreed upon the frequency with which the private part of the sync will change, then the receiver can selectively decrypt a single header in order to extract the new private sync if the communications gap that has led to lost synchronization has exceeded the lifetime of the previous private sync. This should not represent a burdensome amount of decryption, and thus should not open up the receiver to denial-of-service attack simply based on the need to occasionally decrypt a single header.

One implementation of this is to use a hashing function with a one-to-one mapping to generate the private and public sync portions from the sync value. This implementation is shown in FIG. 13, where (for example) a first ISP 1302 is the sender and a second ISP 1303 is the receiver. (Other alternatives are possible from FIG. 13.) A transmitted packet comprises a public or “outer” header 1305 that is not encrypted, and a private or “inner” header 1306 that is encrypted using for example a link key. Outer header 1305 includes a public sync portion while inner header 1306 contains the private sync portion. A receiving node decrypts the inner header using a decryption function 1307 in order to extract the private sync portion. This step is necessary only if the lifetime of the currently buffered private sync has expired. (If the currently-buffered private sync is still valid, then it is simply extracted from memory and “added” (which could be an inverse hash) to the public sync, as shown in step 1308.) The public and decrypted private sync portions are combined in function 1308 in order to generate the combined sync 1309. The combined sync (1309) is then fed into the RNG (1310) and compared to the IP address pair (1311) to validate or reject the packet.

An important consideration in this architecture is the concept of “future” and “past” where the public sync values are concerned. Though the sync values, themselves, should be random to prevent spoofing attacks, it may be important that the receiver be able to quickly identify a sync value that has already been sent—even if the packet containing that sync value was never actually received by the receiver. One solu-

28

tion is to hash a time stamp or sequence number into the public sync portion, which could be quickly extracted, checked, and discarded, thereby validating the public sync portion itself.

In one embodiment, packets can be checked by comparing the source/destination IP pair generated by the sync field with the pair appearing in the packet header. If (1) they match, (2) the time stamp is valid, and (3) the dead-man timer has expired, then re-synchronization occurs; otherwise, the packet is rejected. If enough processing power is available, the dead-man timer and synchronization tables can be avoided altogether, and the receiver would simply resynchronize (e.g., validate) on every packet.

The foregoing scheme may require large-integer (e.g., 160-bit) math, which may affect its implementation. Without such large-integer registers, processing throughput would be affected, thus potentially affecting security from a denial-of-service standpoint. Nevertheless, as large-integer math processing features become more prevalent, the costs of implementing such a feature will be reduced.

#### D. Other Synchronization Schemes

As explained above, if W or more consecutive packets are lost between a transmitter and receiver in a VPN (where W is the window size), the receiver’s window will not have been updated and the transmitter will be transmitting packets not in the receiver’s window. The sender and receiver will not recover synchronization until perhaps the random pairs in the window are repeated by chance. Therefore, there is a need to keep a transmitter and receiver in synchronization whenever possible and to re-establish synchronization whenever it is lost.

A “checkpoint” scheme can be used to regain synchronization between a sender and a receiver that have fallen out of synchronization. In this scheme, a checkpoint message comprising a random IP address pair is used for communicating synchronization information. In one embodiment, two messages are used to communicate synchronization information between a sender and a recipient:

1. SYNC\_REQ is a message used by the sender to indicate that it wants to synchronize; and
2. SYNC\_ACK is a message used by the receiver to inform the transmitter that it has been synchronized.

According to one variation of this approach, both the transmitter and receiver maintain three checkpoints (see FIG. 14):

1. In the transmitter, ckpt\_o (“checkpoint old”) is the IP pair that was used to re-send the last SYNC\_REQ packet to the receiver. In the receiver, ckpt\_o (“checkpoint old”) is the IP pair that receives repeated SYNC\_REQ packets from the transmitter.
2. In the transmitter, ckpt\_n (“checkpoint new”) is the IP pair that will be used to send the next SYNC\_REQ packet to the receiver. In the receiver, ckpt\_n (“checkpoint new”) is the IP pair that receives a new SYNC\_REQ packet from the transmitter and which causes the receiver’s window to be re-aligned, ckpt\_o set to ckpt\_n, a new ckpt\_n to be generated and a new ckpt\_r to be generated.
3. In the transmitter, ckpt\_r is the IP pair that will be used to send the next SYNC\_ACK packet to the receiver. In the receiver, ckpt\_r is the IP pair that receives a new SYNC\_ACK packet from the transmitter and which

## US 7,418,504 B2

29

causes a new ckpt\_n to be generated. Since SYNC\_ACK is transmitted from the receiver ISP to the sender ISP, the transmitter ckpt\_r refers to the ckpt\_r of the receiver and the receiver ckpt\_r refers to the ckpt\_r of the transmitter (see FIG. 14).

When a transmitter initiates synchronization, the IP pair it will use to transmit the next data packet is set to a predetermined value and when a receiver first receives a SYNC\_REQ, the receiver window is updated to be centered on the transmitter's next IP pair. This is the primary mechanism for checkpoint synchronization.

Synchronization can be initiated by a packet counter (e.g., after every N packets transmitted, initiate a synchronization) or by a timer (every S seconds, initiate a synchronization) or a combination of both. See FIG. 15. From the transmitter's perspective, this technique operates as follows: (1) Each transmitter periodically transmits a "sync request" message to the receiver to make sure that it is in sync. (2) If the receiver is still in sync, it sends back a "sync ack" message. (If this works, no further action is necessary). (3) If no "sync ack" has been received within a period of time, the transmitter retransmits the sync request again. If the transmitter reaches the next checkpoint without receiving a "sync ack" response, then synchronization is broken, and the transmitter should stop transmitting. The transmitter will continue to send sync\_reqs until it receives a sync\_ack, at which point transmission is reestablished.

From the receiver's perspective, the scheme operates as follows: (1) when it receives a "sync request" request from the transmitter, it advances its window to the next checkpoint position (even skipping pairs if necessary), and sends a "sync ack" message to the transmitter. If sync was never lost, then the "jump ahead" really just advances to the next available pair of addresses in the table (i.e., normal advancement).

If an interloper intercepts the "sync request" messages and tries to interfere with communication by sending new ones, it will be ignored if the synchronization has been established or it will actually help to re-establish synchronization.

A window is realigned whenever a re-synchronization occurs. This realignment entails updating the receiver's window to straddle the address pairs used by the packet transmitted immediately after the transmission of the SYNC\_REQ packet. Normally, the transmitter and receiver are in synchronization with one another. However, when network events occur, the receiver's window may have to be advanced by many steps during resynchronization. In this case, it is desirable to move the window ahead without having to step through the intervening random numbers sequentially. (This feature is also desirable for the auto-sync approach discussed above).

#### E. Random Number Generator with a Jump-Ahead capability

An attractive method for generating randomly hopped addresses is to use identical random number generators in the transmitter and receiver and advance them as packets are transmitted and received. There are many random number generation algorithms that could be used. Each one has strengths and weaknesses for address hopping applications.

Linear congruential random number generators (LCRs) are fast, simple and well characterized random number generators that can be made to jump ahead n steps efficiently. An LCR generates random numbers  $X_1, X_2, X_3, \dots, X_k$  starting with seed  $X_0$  using a recurrence

$$X_i = (a X_{i-1} + b) \bmod c, \quad (1)$$

30

where a, b and c define a particular LCR. Another expression for  $X_i$ ,

$$X_i = ((a^i(X_0 + b) - b)/(a - 1)) \bmod c \quad (2)$$

enables the jump-ahead capability. The factor  $a^i$  can grow very large even for modest i if left unfettered. Therefore some special properties of the modulo operation can be used to control the size and processing time required to compute (2). (2) can be rewritten as:

$$X_i = (a^i(X_0(a-1)+b)-b)/(a-1) \bmod c. \quad (3)$$

It can be shown that:

$$(a^i(X_0(a-1)+b)-b)/(a-1) \bmod c = ((a^i \bmod ((a-1)c)(X_0(a-1)+b)-b)/(a-1)) \bmod c \quad (4)$$

( $X_0(a-1)+b$ ) can be stored as  $(X_0(a-1)+b) \bmod c$ , b as  $b \bmod c$  and compute  $a^i \bmod ((a-1)c)$  (this requires  $O(\log(i))$  steps).

A practical implementation of this algorithm would jump a fixed distance, n, between synchronizations; this is tantamount to synchronizing every n packets. The window would commence n IP pairs from the start of the previous window. Using  $X_j^w$ , the random number at the  $j^{th}$  checkpoint, as  $X_0$  and n as i, a node can store  $a^n \bmod ((a-1)c)$  once per LCR and set

$$X_{j+1}^w = X_{n(j+1)} = ((a^n \bmod ((a-1)c)(X_j^w(a-1)+b)-b)/(a-1)) \bmod c, \quad (5)$$

to generate the random number for the  $j+1^{th}$  synchronization. Using this construction, a node could jump ahead an arbitrary (but fixed) distance between synchronizations in a constant amount of time (independent of n).

Pseudo-random number generators, in general, and LCRs, in particular, will eventually repeat their cycles. This repetition may present vulnerability in the IP hopping scheme. An adversary would simply have to wait for a repeat to predict future sequences. One way of coping with this vulnerability is to create a random number generator with a known long cycle. A random sequence can be replaced by a new random number generator before it repeats. LCRs can be constructed with known long cycles. This is not currently true of many random number generators.

Random number generators can be cryptographically insecure. An adversary can derive the RNG parameters by examining the output or part of the output. This is true of LCGs. This vulnerability can be mitigated by incorporating an encryptor, designed to scramble the output as part of the random number generator. The random number generator prevents an adversary from mounting an attack—e.g., a known plaintext attack—against the encryptor.

#### F. Random Number Generator Example

Consider a RNG where  $a=31$ ,  $b=4$  and  $c=15$ . For this case equation (1) becomes:

$$X_i = (31X_{i-1} + 4) \bmod 15. \quad (6)$$

If one sets  $X_0=1$ , equation (6) will produce the sequence 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 0, 4, 8, 12. This sequence will repeat indefinitely. For a jump ahead of 3 numbers in this sequence  $a^3=31^3=29791$ ,  $c*(a-1)=15*30=450$  and  $a^n \bmod ((a-1)c)=31^3 \bmod (15*30)=29791 \bmod (450)=91$ . Equation (5) becomes:

$$((91(X_j 30 + 4) - 4)/30) \bmod 15 \quad (7)$$

Table 1 shows the jump ahead calculations from (7). The calculations start at 5 and jump ahead 3.

## US 7,418,504 B2

31

TABLE 1

I	X <sub>i</sub>	(X <sub>i</sub> 30 + 4)	91 (X <sub>i</sub> 30 + 4) - 4	((91 (X <sub>i</sub> 30 + 4) - 4)/30	X <sub>i+3</sub>
1	5	154	14010	467	2
4	2	64	5820	194	14
7	14	424	38580	1286	11
10	11	334	30390	1013	8
13	8	244	22200	740	5

## G. Fast Packet Filter

Address hopping VPNs must rapidly determine whether a packet has a valid header and thus requires further processing, or has an invalid header (a hostile packet) and should be immediately rejected. Such rapid determinations will be referred to as “fast packet filtering.” This capability protects the VPN from attacks by an adversary who streams hostile packets at the receiver at a high rate of speed in the hope of saturating the receiver’s processor (a so-called “denial of service” attack). Fast packet filtering is an important feature for implementing VPNs on shared media such as Ethernet.

Assuming that all participants in a VPN share an unsigned “A” block of addresses, one possibility is to use an experimental “A” block that will never be assigned to any machine that is not address hopping on the shared medium. “A” blocks have a 24 bits of address that can be hopped as opposed to the 8 bits in “C” blocks. In this case a hopblock will be the “A” block. The use of the experimental “A” block is a likely option on an Ethernet because:

1. The addresses have no validity outside of the Ethernet and will not be routed out to a valid outside destination by a gateway.
2. There are  $2^{24}$  (~16 million) addresses that can be hopped within each “A” block. This yields >280 trillion possible address pairs making it very unlikely that an adversary would guess a valid address. It also provides acceptably low probability of collision between separate VPNs (all VPNs on a shared medium independently generate random address pairs from the same “A” block).
3. The packets will not be received by someone on the Ethernet who is not on a VPN (unless the machine is in promiscuous mode) minimizing impact on non-VPN computers.

The Ethernet example will be used to describe one implementation of fast packet filtering. The ideal algorithm would quickly examine a packet header, determine whether the packet is hostile, and reject any hostile packets or determine which active IP pair the packet header matches. The problem is a classical associative memory problem. A variety of techniques have been developed to solve this problem (hashing, B-trees etc). Each of these approaches has its strengths and weaknesses. For instance, hash tables can be made to operate quite fast in a statistical sense, but can occasionally degenerate into a much slower algorithm. This slowness can persist for a period of time. Since there is a need to discard hostile packets quickly at all times, hashing would be unacceptable.

## H. Presence Vector Algorithm

A presence vector is a bit vector of length  $2^n$  that can be indexed by n-bit numbers (each ranging from 0 to  $2^n - 1$ ). One can indicate the presence of k n-bit numbers (not necessarily unique), by setting the bits in the presence vector indexed by each number to 1. Otherwise, the bits in the presence vector are 0. An n-bit number, x, is one of the k numbers if and only

32

if the  $x^{th}$  bit of the presence vector is 1. A fast packet filter can be implemented by indexing the presence vector and looking for a 1, which will be referred to as the “test.”

For example, suppose one wanted to represent the number 135 using a presence vector. The 135<sup>th</sup> bit of the vector would be set. Consequently, one could very quickly determine whether an address of 135 was valid by checking only one bit: the 135<sup>th</sup> bit. The presence vectors could be created in advance corresponding to the table entries for the IP addresses. In effect, the incoming addresses can be used as indices into a long vector, making comparisons very fast. As each RNG generates a new address, the presence vector is updated to reflect the information. As the window moves, the presence vector is updated to zero out addresses that are no longer valid.

There is a trade-off between efficiency of the test and the amount of memory required for storing the presence vector(s). For instance, if one were to use the 48 bits of hopping addresses as an index, the presence vector would have to be 35 terabytes. Clearly, this is too large for practical purposes. Instead, the 48 bits can be divided into several smaller fields. For instance, one could subdivide the 48 bits into four 12-bit fields (see FIG. 16). This reduces the storage requirement to 2048 bytes at the expense of occasionally having to process a hostile packet. In effect, instead of one long presence vector, the decomposed address portions must match all four shorter presence vectors before further processing is allowed. (If the first part of the address portion doesn’t match the first presence vector, there is no need to check the remaining three presence vectors).

A presence vector will have a 1 in the  $y^{th}$  bit if and only if one or more addresses with a corresponding field of y are active. An address is active only if each presence vector indexed by the appropriate sub-field of the address is 1.

Consider a window of 32 active addresses and 3 checkpoints. A hostile packet will be rejected by the indexing of one presence vector more than 99% of the time. A hostile packet will be rejected by the indexing of all 4 presence vectors more than 99.999995% of the time. On average, hostile packets will be rejected in less than 1.02 presence vector index operations.

The small percentage of hostile packets that pass the fast packet filter will be rejected when matching pairs are not found in the active window or are active checkpoints. Hostile packets that serendipitously match a header will be rejected when the VPN software attempts to decrypt the header. However, these cases will be extremely rare. There are many other ways this method can be configured to arbitrate the space/speed tradeoffs.

## I. Further Synchronization Enhancements

A slightly modified form of the synchronization techniques described above can be employed. The basic principles of the previously described checkpoint synchronization scheme remain unchanged. The actions resulting from the reception of the checkpoints are, however, slightly different. In this variation, the receiver will maintain between OoO (“Out of Order”) and  $2 \times \text{WINDOW\_SIZE} + \text{OoO}$  active addresses ( $1 \leq \text{OoO} \leq \text{WINDOW\_SIZE}$  and  $\text{WINDOW\_SIZE} \geq 2$ ). OoO and WINDOW\_SIZE are engineerable parameters, where OoO is the minimum number of addresses needed to accommodate lost packets due to events in the network or out of order arrivals and WINDOW\_SIZE is the number of packets transmitted before a SYNC\_REQ is issued. FIG. 17 depicts a storage array for a receiver’s active addresses.

## US 7,418,504 B2

33

The receiver starts with the first  $2 \times \text{WINDOW\_SIZE}$  addresses loaded and active (ready to receive data). As packets are received, the corresponding entries are marked as “used” and are no longer eligible to receive packets. The transmitter maintains a packet counter, initially set to 0, containing the number of data packets transmitted since the last initial transmission of a SYNC\_REQ for which SYNC\_ACK has been received. When the transmitter packet counter equals WINDOW\_SIZE, the transmitter generates a SYNC\_REQ and does its initial transmission. When the receiver receives a SYNC\_REQ corresponding to its current CKPT\_N, it generates the next WINDOW\_SIZE addresses and starts loading them in order starting at the first location after the last active address wrapping around to the beginning of the array after the end of the array has been reached. The receiver’s array might look like FIG. 18 when a SYNC\_REQ has been received. In this case a couple of packets have been either lost or will be received out of order when the SYNC\_REQ is received.

FIG. 19 shows the receiver’s array after the new addresses have been generated. If the transmitter does not receive a SYNC\_ACK, it will re-issue the SYNC\_REQ at regular intervals. When the transmitter receives a SYNC\_ACK, the packet counter is decremented by WINDOW\_SIZE. If the packet counter reaches  $2 \times \text{WINDOW\_SIZE} - \text{OoO}$  then the transmitter ceases sending data packets until the appropriate SYNC\_ACK is finally received. The transmitter then resumes sending data packets. Future behavior is essentially a repetition of this initial cycle. The advantages of this approach are:

1. There is no need for an efficient jump ahead in the random number generator,
2. No packet is ever transmitted that does not have a corresponding entry in the receiver side
3. No timer based re-synchronization is necessary. This is a consequence of 2.
4. The receiver will always have the ability to accept data messages transmitted within OoO messages of the most recently transmitted message.

#### J. Distributed Transmission Path Variant

Another embodiment incorporating various inventive principles is shown in FIG. 20. In this embodiment, a message transmission system includes a first computer 2001 in communication with a second computer 2002 through a network 2011 of intermediary computers. In one variant of this embodiment, the network includes two edge routers 2003 and 2004 each of which is linked to a plurality of Internet Service Providers (ISPs) 2005 through 2010. Each ISP is coupled to a plurality of other ISPs in an arrangement as shown in FIG. 20, which is a representative configuration only and is not intended to be limiting. Each connection between ISPs is labeled in FIG. 20 to indicate a specific physical transmission path (e.g., AD is a physical path that links ISP A (element 2005) to ISP D (element 2008)). Packets arriving at each edge router are selectively transmitted to one of the ISPs to which the router is attached on the basis of a randomly or quasi-randomly selected basis.

As shown in FIG. 21, computer 2001 or edge router 2003 incorporates a plurality of link transmission tables 2100 that identify, for each potential transmission path through the network, valid sets of IP addresses that can be used to transmit the packet. For example, AD table 2101 contains a plurality of IP source/destination pairs that are randomly or quasi-randomly generated. When a packet is to be transmitted from first computer 2001 to second computer 2002, one of the link tables is randomly (or quasi-randomly) selected, and the next

34

valid source/destination address pair from that table is used to transmit the packet through the network. If path AD is randomly selected, for example, the next source/destination IP address pair (which is pre-determined to transmit between ISP A (element 2005) and ISP B (element 2008)) is used to transmit the packet. If one of the transmission paths becomes degraded or inoperative, that link table can be set to a “down” condition as shown in table 2105, thus preventing addresses from being selected from that table. Other transmission paths would be unaffected by this broken link.

### 3. Continuation-In-Part Improvements

The following describes various improvements and features that can be applied to the embodiments described above. The improvements include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities. Each is discussed separately below.

#### A. Load Balancer

Various embodiments described above include a system in which a transmitting node and a receiving node are coupled through a plurality of transmission paths, and wherein successive packets are distributed quasi-randomly over the plurality of paths. See, for example, FIGS. 20 and 21 and accompanying description. The improvement extends this basic concept to encompass distributing packets across different paths in such a manner that the loads on the paths are generally balanced according to transmission link quality.

In one embodiment, a system includes a transmitting node and a receiving node that are linked via a plurality of transmission paths having potentially varying transmission quality. Successive packets are transmitted over the paths based on a weight value distribution function for each path. The rate that packets will be transmitted over a given path can be different for each path. The relative “health” of each transmission path is monitored in order to identify paths that have become degraded. In one embodiment, the health of each path is monitored in the transmitter by comparing the number of packets transmitted to the number of packet acknowledgements received. Each transmission path may comprise a physically separate path (e.g., via dial-up phone line, computer network, router, bridge, or the like), or may comprise logically separate paths contained within a broadband communication medium (e.g., separate channels in an FDM, TDM, CDMA, or other type of modulated or unmodulated transmission link).

When the transmission quality of a path falls below a predetermined threshold and there are other paths that can transmit packets, the transmitter changes the weight value used for that path, making it less likely that a given packet will be transmitted over that path. The weight will preferably be set no lower than a minimum value that keeps nominal traffic on the path. The weights of the other available paths are altered to compensate for the change in the affected path. When the quality of a path degrades to where the transmitter

## US 7,418,504 B2

35

is turned off by the synchronization function (i.e., no packets are arriving at the destination), the weight is set to zero. If all transmitters are turned off, no packets are sent.

Conventional TCP/IP protocols include a “throttling” feature that reduces the transmission rate of packets when it is determined that delays or errors are occurring in transmission. In this respect, timers are sometimes used to determine whether packets have been received. These conventional techniques for limiting transmission of packets, however, do not involve multiple transmission paths between two nodes wherein transmission across a particular path relative to the others is changed based on link quality.

According to certain embodiments, in order to damp oscillations that might otherwise occur if weight distributions are changed drastically (e.g., according to a step function), a linear or an exponential decay formula can be applied to gradually decrease the weight value over time that a degrading path will be used. Similarly, if the health of a degraded path improves, the weight value for that path is gradually increased.

Transmission link health can be evaluated by comparing the number of packets that are acknowledged within the transmission window (see embodiments discussed above) to the number of packets transmitted within that window and by the state of the transmitter (i.e., on or off). In other words, rather than accumulating general transmission statistics over time for a path, one specific implementation uses the “windowing” concepts described above to evaluate transmission path health.

The same scheme can be used to shift virtual circuit paths from an “unhealthy” path to a “healthy” one, and to select a path for a new virtual circuit.

FIG. 22A shows a flowchart for adjusting weight values associated with a plurality of transmission links. It is assumed that software executing in one or more computer nodes executes the steps shown in FIG. 22A. It is also assumed that the software can be stored on a computer-readable medium such as a magnetic or optical disk for execution by a computer.

Beginning in step 2201, the transmission quality of a given transmission path is measured. As described above, this measurement can be based on a comparison between the number of packets transmitted over a particular link to the number of packet acknowledgements received over the link (e.g., per unit time, or in absolute terms). Alternatively, the quality can be evaluated by comparing the number of packets that are acknowledged within the transmission window to the number of packets that were transmitted within that window. In yet another variation, the number of missed synchronization messages can be used to indicate link quality. Many other variations are of course possible.

In step 2202, a check is made to determine whether more than one transmitter (e.g., transmission path) is turned on. If not, the process is terminated and resumes at step 2201.

In step 2203, the link quality is compared to a given threshold (e.g., 50%, or any arbitrary number). If the quality falls below the threshold, then in step 2207 a check is made to determine whether the weight is above a minimum level (e.g., 1%). If not, then in step 2209 the weight is set to the minimum level and processing resumes at step 2201. If the weight is above the minimum level, then in step 2208 the weight is gradually decreased for the path, then in step 2206 the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are increased).

If in step 2203 the quality of the path was greater than or equal to the threshold, then in step 2204 a check is made to determine whether the weight is less than a steady-state value

36

for that path. If so, then in step 2205 the weight is increased toward the steady-state value, and in step 2206 the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are decreased). If in step 2204 the weight is not less than the steady-state value, then processing resumes at step 2201 without adjusting the weights.

The weights can be adjusted incrementally according to various functions, preferably by changing the value gradually. In one embodiment, a linearly decreasing function is used to adjust the weights; according to another embodiment, an exponential decay function is used. Gradually changing the weights helps to damp oscillators that might otherwise occur if the probabilities were abruptly.

Although not explicitly shown in FIG. 22A the process can be performed only periodically (e.g., according to a time schedule), or it can be continuously run, such as in a background mode of operation. In one embodiment, the combined weights of all potential paths should add up to unity (e.g., when the weighting for one path is decreased, the corresponding weights that the other paths will be selected will increase).

Adjustments to weight values for other paths can be prorated. For example, a decrease of 10% in weight value for one path could result in an evenly distributed increase in the weights for the remaining paths. Alternatively, weightings could be adjusted according to a weighted formula as desired (e.g., favoring healthy paths over less healthy paths). In yet another variation, the difference in weight value can be amortized over the remaining links in a manner that is proportional to their traffic weighting.

FIG. 22B shows steps that can be executed to shut down transmission links where a transmitter turns off. In step 2210, a transmitter shut-down event occurs. In step 2211, a test is made to determine whether at least one transmitter is still turned on. If not, then in step 2215 all packets are dropped until a transmitter turns on. If in step 2211 at least one transmitter is turned on, then in step 2212 the weight for the path is set to zero, and the weights for the remaining paths are adjusted accordingly.

FIG. 23 shows a computer node 2301 employing various principles of the above-described embodiments. It is assumed that two computer nodes of the type shown in FIG. 23 communicate over a plurality of separate physical transmission paths. As shown in FIG. 23, four transmission paths X1 through X4 are defined for communicating between the two nodes. Each node includes a packet transmitter 2302 that operates in accordance with a transmit table 2308 as described above. (The packet transmitter could also operate without using the IP-hopping features described above, but the following description assumes that some form of hopping is employed in conjunction with the path selection mechanism.). The computer node also includes a packet receiver 2303 that operates in accordance with a receive table 2309, including a moving window W that moves as valid packets are received. Invalid packets having source and destination addresses that do not fall within window W are rejected.

As each packet is readied for transmission, source and destination IP addresses (or other discriminator values) are selected from transmit table 2308 according to any of the various algorithms described above, and packets containing these source/destination address pairs, which correspond to the node to which the four transmission paths are linked, are generated to a transmission path switch 2307. Switch 2307, which can comprise a software function, selects from one of the available transmission paths according to a weight distribution table 2306. For example, if the weight for path X1 is 0.2, then every fifth packet will be transmitted on path X1. A similar regime holds true for the other paths as shown. Ini-

## US 7,418,504 B2

37

tially, each link's weight value can be set such that it is proportional to its bandwidth, which will be referred to as its "steady-state" value.

Packet receiver 2303 generates an output to a link quality measurement function 2304 that operates as described above to determine the quality of each transmission path. (The input to packet receiver 2303 for receiving incoming packets is omitted for clarity). Link quality measurement function 2304 compares the link quality to a threshold for each transmission link and, if necessary, generates an output to weight adjustment function 2305. If a weight adjustment is required, then the weights in table 2306 are adjusted accordingly, preferably according to a gradual (e.g., linearly or exponentially declining) function. In one embodiment, the weight values for all available paths are initially set to the same value, and only when paths degrade in quality are the weights changed to reflect differences.

Link quality measurement function 2304 can be made to operate as part of a synchronizer function as described above. That is, if resynchronization occurs and the receiver detects that synchronization has been lost (e.g., resulting in the synchronization window W being advanced out of sequence), that fact can be used to drive link quality measurement function 2304. According to one embodiment, load balancing is performed using information garnered during the normal synchronization, augmented slightly to communicate link health from the receiver to the transmitter. The receiver maintains a count, MESS\_R(W), of the messages received in synchronization window W. When it receives a synchronization request (SYNC\_REQ) corresponding to the end of window W, the receiver includes counter MESS\_R in the resulting synchronization acknowledgement (SYNC\_ACK) sent back to the transmitter. This allows the transmitter to compare messages sent to messages received in order to assess the health of the link.

If synchronization is completely lost, weight adjustment function 2305 decreases the weight value on the affected path to zero. When synchronization is regained, the weight value for the affected path is gradually increased to its original value. Alternatively, link quality can be measured by evaluating the length of time required for the receiver to acknowledge a synchronization request. In one embodiment, separate transmit and receive tables are used for each transmission path.

When the transmitter receives a SYNC\_ACK, the MESS\_R is compared with the number of messages transmitted in a window (MESS\_T). When the transmitter receives a SYNC\_ACK, the traffic probabilities will be examined and adjusted if necessary. MESS\_R is compared with the number of messages transmitted in a window (MESS\_T). There are two possibilities:

1. If MESS\_R is less than a threshold value, THRESH, then the link will be deemed to be unhealthy. If the transmitter was turned off, the transmitter is turned on and the weight P for that link will be set to a minimum value MIN. This will keep a trickle of traffic on the link for monitoring purposes until it recovers. If the transmitter was turned on, the weight P for that link will be set to:

$$P' = \alpha \times \text{MIN} + (1 - \alpha) \times P \quad (1)$$

Equation 1 will exponentially damp the traffic weight value to MIN during sustained periods of degraded service.

2. If MESS\_R for a link is greater than or equal to THRESH, the link will be deemed healthy. If the weight P for that link is greater than or equal to the steady state value S for

38

that link, then P is left unaltered. If the weight P for that link is less than THRESH then P will be set to:

$$P' = \beta \times S + (1 - \beta) \times P \quad (2)$$

where  $\beta$  is a parameter such that  $0 \leq \beta \leq 1$  that determines the damping rate of P.

Equation 2 will increase the traffic weight to S during sustained periods of acceptable service in a damped exponential fashion.

A detailed example will now be provided with reference to FIG. 24. As shown in FIG. 24, a first computer 2401 communicates with a second computer 2402 through two routers 2403 and 2404. Each router is coupled to the other router through three transmission links. As described above, these may be physically diverse links or logical links (including virtual private networks).

Suppose that a first link L1 can sustain a transmission bandwidth of 100 Mb/s and has a window size of 32; link L2 can sustain 75 Mb/s and has a window size of 24; and link L3 can sustain 25 Mb/s and has a window size of 8. The combined links can thus sustain 200 Mb/s. The steady state traffic weights are 0.5 for link L1; 0.375 for link L2, and 0.125 for link L3. MIN=1 Mb/s, THRESH=0.8 MESS\_T for each link,  $\alpha=0.75$  and  $\beta=0.5$ . These traffic weights will remain stable until a link stops for synchronization or reports a number of packets received less than its THRESH. Consider the following sequence of events:

1. Link L1 receives a SYNC\_ACK containing a MESS\_R of 24, indicating that only 75% of the MESS\_T (32) messages transmitted in the last window were successfully received. Link 1 would be below THRESH (0.8). Consequently, link L1's traffic weight value would be reduced to 0.12825, while link L2's traffic weight value would be increased to 0.65812 and link L3's traffic weight value would be increased to 0.217938.

2. Link L2 and L3 remained healthy and link L1 stopped to synchronize. Then link L1's traffic weight value would be set to 0, link L2's traffic weight value would be set to 0.75, and link L3's traffic weight value would be set to 0.25.

3. Link L1 finally received a SYNC\_ACK containing a MESS\_R of 0 indicating that none of the MESS\_T (32) messages transmitted in the last window were successfully received. Link L1 would be below THRESH. Link L1's traffic weight value would be increased to 0.005, link L2's traffic weight value would be decreased to 0.74625, and link L3's traffic weight value would be decreased to 0.24875.

4. Link L1 received a SYNC\_ACK containing a MESS\_R of 32 indicating that 100% of the MESS\_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.2525, while link L2's traffic weight value would be decreased to 0.560625 and link L3's traffic weight value would be decreased to 0.186875.

5. Link L1 received a SYNC\_ACK containing a MESS\_R of 32 indicating that 100% of the MESS\_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.37625; link L2's traffic weight value would be decreased to 0.4678125, and link L3's traffic weight value would be decreased to 0.1559375.

6. Link L1 remains healthy and the traffic probabilities approach their steady state traffic probabilities.

## US 7,418,504 B2

39

B. Use of a DNS Proxy to Transparently Create  
Virtual Private Networks

A second improvement concerns the automatic creation of a virtual private network (VPN) in response to a domain-name server look-up function.

Conventional Domain Name Servers (DNSs) provide a look-up function that returns the IP address of a requested computer or host. For example, when a computer user types in the web name "Yahoo.com," the user's web browser transmits a request to a DNS, which converts the name into a four-part IP address that is returned to the user's browser and then used by the browser to contact the destination web site.

This conventional scheme is shown in FIG. 25. A user's computer 2501 includes a client application 2504 (for example, a web browser) and an IP protocol stack 2505. When the user enters the name of a destination host, a request DNS REQ is made (through IP protocol stack 2505) to a DNS 2502 to look up the IP address associated with the name. The DNS returns the IP address DNS RESP to client application 2504, which is then able to use the IP address to communicate with the host 2503 through separate transactions such as PAGE REQ and PAGE RESP.

In the conventional architecture shown in FIG. 25, nefarious listeners on the Internet could intercept the DNS REQ and DNS RESP packets and thus learn what IP addresses the user was contacting. For example, if a user wanted to set up a secure communication path with a web site having the name "Target.com," when the user's browser contacted a DNS to find the IP address for that web site, the true IP address of that web site would be revealed over the Internet as part of the DNS inquiry. This would hamper anonymous communications on the Internet.

One conventional scheme that provides secure virtual private networks over the Internet provides the DNS server with the public keys of the machines that the DNS server has the addresses for. This allows hosts to retrieve automatically the public keys of a host that the host is to communicate with so that the host can set up a VPN without having the user enter the public key of the destination host. One implementation of this standard is presently being developed as part of the FreeS/WAN project(RFC 2535).

The conventional scheme suffers from certain drawbacks. For example, any user can perform a DNS request. Moreover, DNS requests resolve to the same value for all users.

According to certain aspects of the invention, a specialized DNS server traps DNS requests and, if the request is from a special type of user (e.g., one for which secure communication services are defined), the server does not return the true IP address of the target node, but instead automatically sets up a virtual private network between the target node and the user. The VPN is preferably implemented using the IP address "hopping" features of the basic invention described above, such that the true identity of the two nodes cannot be determined even if packets during the communication are intercepted. For DNS requests that are determined to not require secure services (e.g., an unregistered user), the DNS server transparently "passes through" the request to provide a normal look-up function and return the IP address of the target web server, provided that the requesting host has permissions to resolve unsecured sites. Different users who make an identical DNS request could be provided with different results.

FIG. 26 shows a system employing various principles summarized above. A user's computer 2601 includes a conventional client (e.g., a web browser) 2605 and an IP protocol stack 2606 that preferably operates in accordance with an IP hopping function 2607 as outlined above. A modified DNS

40

server 2602 includes a conventional DNS server function 2609 and a DNS proxy 2610. A gatekeeper server 2603 is interposed between the modified DNS server and a secure target site 2704. An "unsecure" target site 2611 is also accessible via conventional IP protocols.

According to one embodiment, DNS proxy 2610 intercepts all DNS lookup functions from client 2605 and determines whether access to a secure site has been requested. If access to a secure site has been requested (as determined, for example, by a domain name extension, or by reference to an internal table of such sites), DNS proxy 2610 determines whether the user has sufficient security privileges to access the site. If so, DNS proxy 2610 transmits a message to gatekeeper 2603 requesting that a virtual private network be created between user computer 2601 and secure target site 2604. In one embodiment, gatekeeper 2603 creates "hopblocks" to be used by computer 2601 and secure target site 2604 for secure communication. Then, gatekeeper 2603 communicates these to user computer 2601. Thereafter, DNS proxy 2610 returns to user computer 2601 the resolved address passed to it by the gatekeeper (this address could be different from the actual target computer) 2604, preferably using a secure administrative VPN. The address that is returned need not be the actual address of the destination computer.

Had the user requested lookup of a non-secure web site such as site 2611, DNS proxy would merely pass through to conventional DNS server 2609 the look-up request, which would be handled in a conventional manner, returning the IP address of non-secure web site 2611. If the user had requested lookup of a secure web site but lacked credentials to create such a connection, DNS proxy 2610 would return a "host unknown" error to the user. In this manner, different users requesting access to the same DNS name could be provided with different look-up results.

Gatekeeper 2603 can be implemented on a separate computer (as shown in FIG. 26) or as a function within modified DNS server 2602. In general, it is anticipated that gatekeeper 2703 facilitates the allocation and exchange of information needed to communicate securely, such as using "hopped" IP addresses. Secure hosts such as site 2604 are assumed to be equipped with a secure communication function such as an IP hopping function 2608.

It will be appreciated that the functions of DNS proxy 2610 and DNS server 2609 can be combined into a single server for convenience. Moreover, although element 2602 is shown as combining the functions of two servers, the two servers can be made to operate independently.

FIG. 27 shows steps that can be executed by DNS proxy server 2610 to handle requests for DNS look-up for secure hosts. In step 2701, a DNS look-up request is received for a target host. In step 2702, a check is made to determine whether access to a secure host was requested. If not, then in step 2703 the DNS request is passed to conventional DNS server 2609, which looks up the IP address of the target site and returns it to the user's application for further processing.

In step 2702, if access to a secure host was requested, then in step 2704 a further check is made to determine whether the user is authorized to connect to the secure host. Such a check can be made with reference to an internally stored list of authorized IP addresses, or can be made by communicating with gatekeeper 2603 (e.g., over an "administrative" VPN that is secure). It will be appreciated that different levels of security can also be provided for different categories of hosts. For example, some sites may be designated as having a certain security level, and the security level of the user requesting access must match that security level. The user's security



## US 7,418,504 B2

41

level can also be determined by transmitting a request message back to the user's computer requiring that it prove that it has sufficient privileges.

If the user is not authorized to access the secure site, then a "host unknown" message is returned (step 2705). If the user has sufficient security privileges, then in step 2706 a secure VPN is established between the user's computer and the secure target site. As described above, this is preferably done by allocating a hopping regime that will be carried out between the user's computer and the secure target site, and is preferably performed transparently to the user (i.e., the user need not be involved in creating the secure link). As described in various embodiments of this application, any of various fields can be "hopped" (e.g., IP source/destination addresses; a field in the header; etc.) in order to communicate securely.

Some or all of the security functions can be embedded in gatekeeper 2603, such that it handles all requests to connect to secure sites. In this embodiment, DNS proxy 2610 communicates with gatekeeper 2603 to determine (preferably over a secure administrative VPN) whether the user has access to a particular web site. Various scenarios for implementing these features are described by way of example below:

Scenario #1: Client has permission to access target computer, and gatekeeper has a rule to make a VPN for the client. In this scenario, the client's DNS request would be received by the DNS proxy server 2610, which would forward the request to gatekeeper 2603. The gatekeeper would establish a VPN between the client and the requested target. The gatekeeper would provide the address of the destination to the DNS proxy, which would then return the resolved name as a result. The resolved address can be transmitted back to the client in a secure administrative VPN.

Scenario #2: Client does not have permission to access target computer. In this scenario, the client's DNS request would be received by the DNS proxy server 2610, which would forward the request to gatekeeper 2603. The gatekeeper would reject the request, informing DNS proxy server 2610 that it was unable to find the target computer. The DNS proxy 2610 would then return a "host unknown" error message to the client.

Scenario #3: Client has permission to connect using a normal non-VPN link, and the gatekeeper does not have a rule to set up a VPN for the client to the target site. In this scenario, the client's DNS request is received by DNS proxy server 2610, which would check its rules and determine that no VPN is needed. Gatekeeper 2603 would then inform the DNS proxy server to forward the request to conventional DNS server 2609, which would resolve the request and return the result to the DNS proxy server and then back to the client.

Scenario #4: Client does not have permission to establish a normal/non-VPN link, and the gatekeeper does not have a rule to make a VPN for the client to the target site. In this scenario, the DNS proxy server would receive the client's DNS request and forward it to gatekeeper 2603. Gatekeeper 2603 would determine that no special VPN was needed, but that the client is not authorized to communicate with non-VPN members. The gatekeeper would reject the request, causing DNS proxy server 2610 to return an error message to the client.

#### C. Large Link to Small Link Bandwidth Management

One feature of the basic architecture is the ability to prevent so-called "denial of service" attacks that can occur if a computer hacker floods a known Internet node with packets, thus preventing the node from communicating with other nodes.

42

Because IP addresses or other fields are "hopped" and packets arriving with invalid addresses are quickly discarded, Internet nodes are protected against flooding targeted at a single IP address.

In a system in which a computer is coupled through a link having a limited bandwidth (e.g., an edge router) to a node that can support a much higher-bandwidth link (e.g., an Internet Service Provider), a potential weakness could be exploited by a determined hacker. Referring to FIG. 28, suppose that a first host computer 2801 is communicating with a second host computer 2804 using the IP address hopping principles described above. The first host computer is coupled through an edge router 2802 to an Internet Service Provider (ISP) 2803 through a low bandwidth link (LOW BW), and is in turn coupled to second host computer 2804 through parts of the Internet through a high bandwidth link (HIGH BW). In this architecture, the ISP is able to support a high bandwidth to the internet, but a much lower bandwidth to the edge router 2802.

Suppose that a computer hacker is able to transmit a large quantity of dummy packets addressed to first host computer 2801 across high bandwidth link HIGH BW. Normally, host computer 2801 would be able to quickly reject the packets since they would not fall within the acceptance window permitted by the IP address hopping scheme. However, because the packets must travel across low bandwidth link LOW BW, the packets overwhelm the lower bandwidth link before they are received by host computer 2801. Consequently, the link to host computer 2801 is effectively flooded before the packets can be discarded.

According to one inventive improvement, a "link guard" function 2805 is inserted into the high-bandwidth node (e.g., ISP 2803) that quickly discards packets destined for a low-bandwidth target node if they are not valid packets. Each packet destined for a low-bandwidth node is cryptographically authenticated to determine whether it belongs to a VPN. If it is not a valid VPN packet, the packet is discarded at the high-bandwidth node. If the packet is authenticated as belonging to a VPN, the packet is passed with high preference. If the packet is a valid non-VPN packet, it is passed with a lower quality of service (e.g., lower priority).

In one embodiment, the ISP distinguishes between VPN and non-VPN packets using the protocol of the packet. In the case of IPSEC [rfc 2401], the packets have IP protocols 420 and 421. In the case of the TARP VPN, the packets will have an IP protocol that is not yet defined. The ISP's link guard, 2805, maintains a table of valid VPNs which it uses to validate whether VPN packets are cryptographically valid. According to one embodiment, packets that do not fall within any hop windows used by nodes on the low-bandwidth link are rejected, or are sent with a lower quality of service. One approach for doing this is to provide a copy of the IP hopping tables used by the low-bandwidth nodes to the high-bandwidth node, such that both the high-bandwidth and low-bandwidth nodes track hopped packets (e.g., the high-bandwidth node moves its hopping window as valid packets are received). In such a scenario, the high-bandwidth node discards packets that do not fall within the hopping window before they are transmitted over the low-bandwidth link. Thus, for example, ISP 2903 maintains a copy 2910 of the receive table used by host computer 2901. Incoming packets that do not fall within this receive table are discarded. According to a different embodiment, link guard 2805 validates each VPN packet using a keyed hashed message authentication code (HMAC) [rfc 2104].

According to another embodiment, separate VPNs (using, for example, hopblocks) can be established for communicat-



## US 7,418,504 B2

43

ing between the low-bandwidth node and the high-bandwidth node (i.e., packets arriving at the high-bandwidth node are converted into different packets before being transmitted to the low-bandwidth node).

As shown in FIG. 29, for example, suppose that a first host computer 2900 is communicating with a second host computer 2902 over the Internet, and the path includes a high bandwidth link HIGH BW to an ISP 2901 and a low bandwidth link LOW BW through an edge router 2904. In accordance with the basic architecture described above, first host computer 2900 and second host computer 2902 would exchange hopblocks (or a hopblock algorithm) and would be able to create matching transmit and receive tables 2905, 2906, 2912 and 2913. Then in accordance with the basic architecture, the two computers would transmit packets having seemingly random IP source and destination addresses, and each would move a corresponding hopping window in its receive table as valid packets were received.

Suppose that a nefarious computer hacker 2903 was able to deduce that packets having a certain range of IP addresses (e.g., addresses 100 to 200 for the sake of simplicity) are being transmitted to ISP 2901, and that these packets are being forwarded over a low-bandwidth link. Hacker computer 2903 could thus “flood” packets having addresses falling into the range 100 to 200, expecting that they would be forwarded along low bandwidth link LOW BW, thus causing the low bandwidth link to become overwhelmed. The fast packet reject mechanism in first host computer 3000 would be of little use in rejecting these packets, since the low bandwidth link was effectively jammed before the packets could be rejected. In accordance with one aspect of the improvement, however, VPN link guard 2911 would prevent the attack from impacting the performance of VPN traffic because the packets would either be rejected as invalid VPN packets or given a lower quality of service than VPN traffic over the lower bandwidth link. A denial-of-service flood attack could, however, still disrupt non-VPN traffic.

According to one embodiment of the improvement, ISP 2901 maintains a separate VPN with first host computer 2900, and thus translates packets arriving at the ISP into packets having a different IP header before they are transmitted to host computer 2900. The cryptographic keys used to authenticate VPN packets at the link guard 2911 and the cryptographic keys used to encrypt and decrypt the VPN packets at host 2902 and host 2901 can be different, so that link guard 2911 does not have access to the private host data; it only has the capability to authenticate those packets.

According to yet a third embodiment, the low-bandwidth node can transmit a special message to the high-bandwidth node instructing it to shut down all transmissions on a particular IP address, such that only hopped packets will pass through to the low-bandwidth node. This embodiment would prevent a hacker from flooding packets using a single IP address. According to yet a fourth embodiment, the high-bandwidth node can be configured to discard packets transmitted to the low-bandwidth node if the transmission rate exceeds a certain predetermined threshold for any given IP address; this would allow hopped packets to go through. In this respect, link guard 2911 can be used to detect that the rate of packets on a given IP address are exceeding a threshold rate; further packets addressed to that same IP address would be dropped or transmitted at a lower priority (e.g., delayed).

#### D. Traffic Limiter

In a system in which multiple nodes are communicating using “hopping” technology, a treasonous insider could inter-

44

nally flood the system with packets. In order to prevent this possibility, one inventive improvement involves setting up “contracts” between nodes in the system, such that a receiver can impose a bandwidth limitation on each packet sender. One technique for doing this is to delay acceptance of a checkpoint synchronization request from a sender until a certain time period (e.g., one minute) has elapsed. Each receiver can effectively control the rate at which its hopping window moves by delaying “SYNC ACK” responses to “SYNC\_REQ” messages.

A simple modification to the checkpoint synchronizer will serve to protect a receiver from accidental or deliberate overload from an internally treasonous client. This modification is based on the observation that a receiver will not update its tables until a SYNC\_REQ is received on hopped address CKPT\_N. It is a simple matter of deferring the generation of a new CKPT\_N until an appropriate interval after previous checkpoints.

Suppose a receiver wished to restrict reception from a transmitter to 100 packets a second, and that checkpoint synchronization messages were triggered every 50 packets. A compliant transmitter would not issue new SYNC\_REQ messages more often than every 0.5 seconds. The receiver could delay a non-compliant transmitter from synchronizing by delaying the issuance of CKPT\_N for 0.5 second after the last SYNC\_REQ was accepted.

In general, if M receivers need to restrict N transmitters issuing new SYNC\_REQ messages after every W messages to sending R messages a second in aggregate, each receiver could defer issuing a new CKPT\_N until  $M \times N \times W / R$  seconds have elapsed since the last SYNC\_REQ has been received and accepted. If the transmitter exceeds this rate between a pair of checkpoints, it will issue the new checkpoint before the receiver is ready to receive it, and the SYNC\_REQ will be discarded by the receiver. After this, the transmitter will re-issue the SYNC\_REQ every  $T_i$  seconds until it receives a SYNC\_ACK. The receiver will eventually update CKPT\_N and the SYNC\_REQ will be acknowledged. If the transmission rate greatly exceeds the allowed rate, the transmitter will stop until it is compliant. If the transmitter exceeds the allowed rate by a little, it will eventually stop after several rounds of delayed synchronization until it is in compliance. Hacking the transmitter’s code to not shut off only permits the transmitter to lose the acceptance window. In this case it can recover the window and proceed only after it is compliant again.

Two practical issues should be considered when implementing the above scheme:

1. The receiver rate should be slightly higher than the permitted rate in order to allow for statistical fluctuations in traffic arrival times and non-uniform load balancing.

2. Since a transmitter will rightfully continue to transmit for a period after a SYNC\_REQ is transmitted, the algorithm above can artificially reduce the transmitter’s bandwidth. If events prevent a compliant transmitter from synchronizing for a period (e.g. the network dropping a SYNC\_REQ or a SYNC\_ACK) a SYNC\_REQ will be accepted later than expected. After this, the transmitter will transmit fewer than expected messages before encountering the next checkpoint. The new checkpoint will not have been activated and the transmitter will have to retransmit the SYNC\_REQ. This will appear to the receiver as if the transmitter is not compliant. Therefore, the next checkpoint will be accepted late from the transmitter’s perspective. This has the effect of reducing the transmitter’s allowed packet rate until the transmitter transmits at a packet rate below the agreed upon rate for a period of time.

## US 7,418,504 B2

45

To guard against this, the receiver should keep track of the times that the last C SYNC\_REQs were received and accepted and use the minimum of  $M \times N \times W/R$  seconds after the last SYNC\_REQ has been received and accepted,  $2 \times M \times N \times W/R$  seconds after next to the last SYNC\_REQ has been received and accepted,  $C \times M \times N \times W/R$  seconds after  $(C-1)^{th}$  to the last SYNC\_REQ has been received, as the time to activate CKPT\_N. This prevents the receiver from inappropriately limiting the transmitter's packet rate if at least one out of the last C SYNC\_REQs was processed on the first attempt.

FIG. 30 shows a system employing the above-described principles. In FIG. 30, two computers 3000 and 3001 are assumed to be communicating over a network N in accordance with the "hopping" principles described above (e.g., hopped IP addresses, discriminator values, etc.). For the sake of simplicity, computer 3000 will be referred to as the receiving computer and computer 3001 will be referred to as the transmitting computer, although full duplex operation is of course contemplated. Moreover, although only a single transmitter is shown, multiple transmitters can transmit to receiver 3000.

As described above, receiving computer 3000 maintains a receive table 3002 including a window W that defines valid IP address pairs that will be accepted when appearing in incoming data packets. Transmitting computer 3001 maintains a transmit table 3003 from which the next IP address pairs will be selected when transmitting a packet to receiving computer 3000. (For the sake of illustration, window W is also illustrated with reference to transmit table 3003). As transmitting computer moves through its table, it will eventually generate a SYNC\_REQ message as illustrated in function 3010. This is a request to receiver 3000 to synchronize the receive table 3002, from which transmitter 3001 expects a response in the form of a CKPT\_N (included as part of a SYNC\_ACK message). If transmitting computer 3001 transmits more messages than its allotment, it will prematurely generate the SYNC\_REQ message. (If it has been altered to remove the SYNC\_REQ message generation altogether, it will fall out of synchronization since receiver 3000 will quickly reject packets that fall outside of window W, and the extra packets generated by transmitter 3001 will be discarded).

In accordance with the improvements described above, receiving computer 3000 performs certain steps when a SYNC\_REQ message is received, as illustrated in FIG. 30. In step 3004, receiving computer 3000 receives the SYNC\_REQ message. In step 3005, a check is made to determine whether the request is a duplicate. If so, it is discarded in step 3006. In step 3007, a check is made to determine whether the SYNC\_REQ received from transmitter 3001 was received at a rate that exceeds the allowable rate R (i.e., the period between the time of the last SYNC\_REQ message). The value R can be a constant, or it can be made to fluctuate as desired. If the rate exceeds R, then in step 3008 the next activation of the next CKPT\_N hopping table entry is delayed by  $W/R$  seconds after the last SYNC\_REQ has been accepted.

Otherwise, if the rate has not been exceeded, then in step 3109 the next CKPT\_N value is calculated and inserted into the receiver's hopping table prior to the next SYNC\_REQ from the transmitter 3101. Transmitter 3101 then processes the SYNC\_REQ in the normal manner.

#### E. Signaling Synchronizer

In a system in which a large number of users communicate with a central node using secure hopping technology, a large amount of memory must be set aside for hopping tables and their supporting data structures. For example, if one million

46

subscribers to a web site occasionally communicate with the web site, the site must maintain one million hopping tables, thus using up valuable computer resources, even though only a small percentage of the users may actually be using the system at any one time. A desirable solution would be a system that permits a certain maximum number of simultaneous links to be maintained, but which would "recognize" millions of registered users at any one time. In other words, out of a population of a million registered users, a few thousand at a time could simultaneously communicate with a central server, without requiring that the server maintain one million hopping tables of appreciable size.

One solution is to partition the central node into two nodes: a signaling server that performs session initiation for user log-on and log-off (and requires only minimally sized tables), and a transport server that contains larger hopping tables for the users. The signaling server listens for the millions of known users and performs a fast-packet reject of other (bogus) packets. When a packet is received from a known user, the signaling server activates a virtual private link (VPL) between the user and the transport server, where hopping tables are allocated and maintained. When the user logs onto the signaling server, the user's computer is provided with hopping tables for communicating with the transport server, thus activating the VPL. The VPLs can be torn down when they become inactive for a time period, or they can be torn down upon user log-out. Communication with the signaling server to allow user log-on and log-off can be accomplished using a specialized version of the checkpoint scheme described above.

FIG. 31 shows a system employing certain of the above-described principles. In FIG. 31, a signaling server 3101 and a transport server 3102 communicate over a link. Signaling server 3101 contains a large number of small tables 3106 and 3107 that contain enough information to authenticate a communication request with one or more clients 3103 and 3104. As described in more detail below, these small tables may advantageously be constructed as a special case of the synchronizing checkpoint tables described previously. Transport server 3102, which is preferably a separate computer in communication with signaling server 3101, contains a smaller number of larger hopping tables 3108, 3109, and 3110 that can be allocated to create a VPN with one of the client computers.

According to one embodiment, a client that has previously registered with the system (e.g., via a system administration function, a user registration procedure, or some other method) transmits a request for information from a computer (e.g., a web site). In one variation, the request is made using a "hopped" packet, such that signaling server 3101 will quickly reject invalid packets from unauthorized computers such as hacker computer 3105. An "administrative" VPN can be established between all of the clients and the signaling server in order to ensure that a hacker cannot flood signaling server 3101 with bogus packets. Details of this scheme are provided below.

Signaling server 3101 receives the request 3111 and uses it to determine that client 3103 is a validly registered user. Next, signaling server 3101 issues a request to transport server 3102 to allocate a hopping table (or hopping algorithm or other regime) for the purpose of creating a VPN with client 3103. The allocated hopping parameters are returned to signaling server 3101 (path 3113), which then supplies the hopping parameters to client 3103 via path 3114, preferably in an encrypted form.

Thereafter, client 3103 communicates with transport server 3102 using the normal hopping techniques described

## US 7,418,504 B2

47

above. It will be appreciated that although signaling server **3101** and transport server **3102** are illustrated as being two separate computers, they could of course be combined into a single computer and their functions performed on the single computer. Alternatively, it is possible to partition the functions shown in FIG. **31** differently from as shown without departing from the inventive principles.

One advantage of the above-described architecture is that signaling server **3101** need only maintain a small amount of information on a large number of potential users, yet it retains the capability of quickly rejecting packets from unauthorized users such as hacker computer **3105**. Larger data tables needed to perform the hopping and synchronization functions are instead maintained in a transport server **3102**, and a smaller number of these tables are needed since they are only allocated for "active" links. After a VPN has become inactive for a certain time period (e.g., one hour), the VPN can be automatically torn down by transport server **3102** or signaling server **3101**.

A more detailed description will now be provided regarding how a special case of the checkpoint synchronization feature can be used to implement the signaling scheme described above.

The signaling synchronizer may be required to support many (millions) of standing, low bandwidth connections. It therefore should minimize per-VPL memory usage while providing the security offered by hopping technology. In order to reduce memory usage in the signaling server, the data hopping tables can be completely eliminated and data can be carried as part of the SYNC\_REQ message. The table used by the server side (receiver) and client side (transmitter) is shown schematically as element **3106** in FIG. **31**.

The meaning and behaviors of CKPT\_N, CKPT\_O and CKPT\_R remain the same from the previous description, except that CKPT\_N can receive a combined data and SYNC\_REQ message or a SYNC\_REQ message without the data.

The protocol is a straightforward extension of the earlier synchronizer. Assume that a client transmitter is on and the tables are synchronized. The initial tables can be generated "out of band." For example, a client can log into a web server to establish an account over the Internet. The client will receive keys etc encrypted over the Internet. Meanwhile, the server will set up the signaling VPN on the signaling server.

Assuming that a client application wishes to send a packet to the server on the client's standing signaling VPL:

1. The client sends the message marked as a data message on the inner header using the transmitter's CKPT\_N address. It turns the transmitter off and starts a timer TI noting CKPT\_O. Messages can be one of three types: DATA, SYNC\_REQ and SYNC\_ACK. In the normal algorithm, some potential problems can be prevented by identifying each message type as part of the encrypted inner header field. In this algorithm, it is important to distinguish a data packet and a SYNC\_REQ in the signaling synchronizer since the data and the SYNC\_REQ come in on the same address.

2. When the server receives a data message on its CKPT\_N, it verifies the message and passes it up the stack. The message can be verified by checking message type and other information (i.e., user credentials) contained in the inner header. It replaces its CKPT\_O with CKPT\_N and generates the next CKPT\_N. It updates its transmitter side CKPT\_R to correspond to the client's receiver side CKPT\_R and transmits a SYNC\_ACK containing CKPT\_O in its payload.

3. When the client side receiver receives a SYNC\_ACK on its CKPT\_R with a payload matching its transmitter side CKPT\_O and the transmitter is off, the transmitter is turned on and the receiver side CKPT\_R is updated. If the SYN-

48

C\_ACK's payload does not match the transmitter side CKPT\_O or the transmitter is on, the SYNC\_ACK is simply discarded.

4. TI expires: If the transmitter is off and the client's transmitter side CKPT\_O matches the CKPT\_O associated with the timer, it starts timer TI noting CKPT\_O again, and a SYNC\_REQ is sent using the transmitter's CKPT\_O address. Otherwise, no action is taken.

5. When the server receives a SYNC\_REQ on its CKPT\_N, it replaces its CKPT\_O with CKPT\_N and generates the next CKPT\_N. It updates its transmitter side CKPT\_R to correspond to the client's receiver side CKPT\_R and transmits a SYNC\_ACK containing CKPT\_O in its payload.

6. When the server receives a SYNC\_REQ on its CKPT\_O, it updates its transmitter side CKPT\_R to correspond to the client's receiver side CKPT\_R and transmits a SYNC\_ACK containing CKPT\_O in its payload.

FIG. **32** shows message flows to highlight the protocol. Reading from top to bottom, the client sends data to the server using its transmitter side CKPT\_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT\_N into CKPT\_O and updates CKPT\_N. This message is successfully received and a passed up the stack. It also synchronizes the receiver i.e., the server loads CKPT\_N into CKPT\_O and generates a new CKPT\_N, it generates a new CKPT\_R in the server side transmitter and transmits a SYNC\_ACK containing the server side receiver's CKPT\_O the server. The SYNC\_ACK is successfully received at the client. The client side receiver's CKPT\_R is updated, the transmitter is turned on and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

Next, the client sends data to the server using its transmitter side CKPT\_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT\_N into CKPT\_O and updates CKPT\_N. This message is lost. The client side timer expires and as a result a SYNC\_REQ is transmitted on the client side transmitter's CKPT\_O (this will keep happening until the SYNC\_ACK has been received at the client). The SYNC\_REQ is successfully received at the server. It synchronizes the receiver i.e., the server loads CKPT\_N into CKPT\_O and generates a new CKPT\_N, it generates a new CKPT\_R in the server side transmitter and transmits a SYNC\_ACK containing the server side receiver's CKPT\_O the server. The SYNC\_ACK is successfully received at the client. The client side receiver's CKPT\_R is updated, the transmitter is turned off and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

There are numerous other scenarios that follow this flow. For example, the SYNC\_ACK could be lost. The transmitter would continue to re-send the SYNC\_REQ until the receiver synchronizes and responds.

The above-described procedures allow a client to be authenticated at signaling server **3201** while maintaining the ability of signaling server **3201** to quickly reject invalid packets, such as might be generated by hacker computer **3205**. In various embodiments, the signaling synchronizer is really a derivative of the synchronizer. It provides the same protection as the hopping protocol, and it does so for a large number of low bandwidth connections.

## US 7,418,504 B2

49

## F. One-click Secure On-line Communications and Secure Domain Name Service

The present invention provides a technique for establishing a secure communication link between a first computer and a second computer over a computer network. Preferably, a user enables a secure communication link using a single click of a mouse, or a corresponding minimal input from another input device, such as a keystroke entered on a keyboard or a click entered through a trackball. Alternatively, the secure link is automatically established as a default setting at boot-up of the computer (i.e., no click). FIG. 33 shows a system block diagram 3300 of a computer network in which the one-click secure communication method of the present invention is suitable. In FIG. 33, a computer terminal or client computer 3301, such as a personal computer (PC), is connected to a computer network 3302, such as the Internet, through an ISP 3303. Alternatively, computer 3301 can be connected to computer network 3302 through an edge router. Computer 3301 includes an input device, such as a keyboard and/or mouse, and a display device, such as a monitor. Computer 3301 can communicate conventionally with another computer 3304 connected to computer network 3302 over a communication link 3305 using a browser 3306 that is installed and operates on computer 3301 in a well-known manner.

Computer 3304 can be, for example, a server computer that is used for conducting e-commerce. In the situation when computer network 3302 is the Internet, computer 3304 typically will have a standard top-level domain name such as .com, .net, .org, .edu, .mil or .gov.

FIG. 34 shows a flow diagram 3400 for installing and establishing a "one-click" secure communication link over a computer network according to the present invention. At step 3401, computer 3301 is connected to server computer 3304 over a non-VPN communication link 3305. Web browser 3306 displays a web page associated with server 3304 in a well-known manner. According to one variation of the invention, the display of computer 3301 contains a hyperlink, or an icon representing a hyperlink, for selecting a virtual private network (VPN) communication link ("go secure" hyperlink) through computer network 3302 between terminal 3301 and server 3304. Preferably, the "go secure" hyperlink is displayed as part of the web page downloaded from server computer 3304, thereby indicating that the entity providing server 3304 also provides VPN capability.

By displaying the "go secure" hyperlink, a user at computer 3301 is informed that the current communication link between computer 3301 and server computer 3304 is a non-secure, non-VPN communication link. At step 3402, it is determined whether a user of computer 3301 has selected the "go secure" hyperlink. If not, processing resumes using a non-secure (conventional) communication method (not shown). If, at step 3402, it is determined that the user has selected the "go secure" hyperlink, flow continues to step 3403 where an object associated with the hyperlink determines whether a VPN communication software module has already been installed on computer 3301. Alternatively, a user can enter a command into computer 3301 to "go secure."

If, at step 3403, the object determines that the software module has been installed, flow continues to step 3407. If, at step 3403, the object determines that the software module has not been installed, flow continues to step 3404 where a non-VPN communication link 3307 is launched between computer 3301 and a website 3308 over computer network 3302 in a well-known manner. Website 3308 is accessible by all computer terminals connected to computer network 3302 through a non-VPN communication link. Once connected to

50

website 3308, a software module for establishing a secure communication link over computer network 3302 can be downloaded and installed. Flow continues to step 3405 where, after computer 3301 connects to website 3308, the software module for establishing a communication link is downloaded and installed in a well-known manner on computer terminal 3301 as software module 3309. At step 3405, a user can optionally select parameters for the software module, such as enabling a secure communication link mode of communication for all communication links over computer network 3302. At step 3406, the communication link between computer 3301 and website 3308 is then terminated in a well-known manner.

By clicking on the "go secure" hyperlink, a user at computer 3301 has enabled a secure communication mode of communication between computer 3301 and server computer 3304. According to one variation of the invention, the user is not required to do anything more than merely click the "go secure" hyperlink. The user does not need to enter any user identification information, passwords or encryption keys for establishing a secure communication link. All procedures required for establishing a secure communication link between computer 3301 and server computer 3304 are performed transparently to a user at computer 3301.

At step 3407, a secure VPN communications mode of operation has been enabled and software module 3309 begins to establish a VPN communication link. In one embodiment, software module 3309 automatically replaces the top-level domain name for server 3304 within browser 3406 with a secure top-level domain name for server computer 3304. For example, if the top-level domain name for server 3304 is .com, software module 3309 replaces the .com top-level domain name with a scom top-level domain name, where the "s" stands for secure. Alternatively, software module 3409 can replace the top-level domain name of server 3304 with any other non-standard top-level domain name.

Because the secure top-level domain name is a non-standard domain name, a query to a standard domain name service (DNS) will return a message indicating that the universal resource locator (URL) is unknown. According to the invention, software module 3309 contains the URL for querying a secure domain name service (SDNS) for obtaining the URL for a secure top-level domain name. In this regard, software module 3309 accesses a secure portal 3310 that interfaces a secure network 3311 to computer network 3302. Secure network 3311 includes an internal router 3312, a secure domain name service (SDNS) 3313, a VPN gatekeeper 3314 and a secure proxy 3315. The secure network can include other network services, such as e-mail 3316, a plurality of chat-rooms (of which only one chatroom 3317 is shown), and a standard domain name service (STD DNS) 3318. Of course, secure network 3311 can include other resources and services that are not shown in FIG. 33.

When software module 3309 replaces the standard top-level domain name for server 3304 with the secure top-level domain name, software module 3309 sends a query to SDNS 3313 at step 3408 through secure portal 3310 preferably using an administrative VPN communication link 3319. In this configuration, secure portal 3310 can only be accessed using a VPN communication link. Preferably, such a VPN communication link can be based on a technique of inserting a source and destination IP address pair into each data packet that is selected according to a pseudo-random sequence; an IP address hopping regime that pseudorandomly changes IP addresses in packets transmitted between a client computer and a secure target computer; periodically changing at least one field in a series of data packets according to a known

## US 7,418,504 B2

51

sequence; an Internet Protocol (IP) address in a header of each data packet that is compared to a table of valid IP addresses maintained in a table in the second computer; and/or a comparison of the IP address in the header of each data packet to a moving window of valid IP addresses, and rejecting data packets having IP addresses that do not fall within the moving window. Other types of VPNs can alternatively be used. Secure portal 3310 authenticates the query from software module 3309 based on the particular information hopping technique used for VPN communication link 3319.

SDNS 3313 contains a cross-reference database of secure domain names and corresponding secure network addresses. That is, for each secure domain name, SDNS 3313 stores a computer network address corresponding to the secure domain name. An entity can register a secure domain name in SDNS 3313 so that a user who desires a secure communication link to the website of the entity can automatically obtain the secure computer network address for the secure website. Moreover, an entity can register several secure domain names, with each respective secure domain name representing a different priority level of access in a hierarchy of access levels to a secure website. For example, a securities trading website can provide users secure access so that a denial of service attack on the website will be ineffectual with respect to users subscribing to the secure website service. Different levels of subscription can be arranged based on, for example, an escalating fee, so that a user can select a desired level of guarantee for connecting to the secure securities trading website. When a user queries SDNS 3313 for the secure computer network address for the securities trading website, SDNS 3313 determines the particular secure computer network address based on the user's identity and the user's subscription level.

At step 3409, SDNS 3313 accesses VPN gatekeeper 3314 for establishing a VPN communication link between software module 3309 and secure server 3320. Server 3320 can only be accessed through a VPN communication link. VPN gatekeeper 3314 provisions computer 3301 and secure web server computer 3320, or a secure edge router for server computer 3320, thereby creating the VPN. Secure server computer 3320 can be a separate server computer from server computer 3304, or can be the same server computer having both non-VPN and VPN communication link capability, such as shown by server computer 3322. Returning to FIG. 34, in step 3410, SDNS 3313 returns a secure URL to software module 3309 for the .com server address for a secure server 3320 corresponding to server 3304.

Alternatively, SDNS 3313 can be accessed through secure portal 3310 "in the clear", that is, without using an administrative VPN communication link. In this situation, secure portal 3310 preferably authenticates the query using any well-known technique, such as a cryptographic technique, before allowing the query to proceed to SDNS 3313. Because the initial communication link in this situation is not a VPN communication link, the reply to the query can be "in the clear." The querying computer can use the clear reply for establishing a VPN link to the desired domain name. Alternatively, the query to SDNS 3313 can be in the clear, and SDNS 3313 and gatekeeper 3314 can operate to establish a VPN communication link to the querying computer for sending the reply.

At step 3411, software module 3309 accesses secure server 3320 through VPN communication link 3321 based on the VPN resources allocated by VPN gatekeeper 3314. At step 3412, web browser 3306 displays a secure icon indicating that the current communication link to server 3320 is a secure VPN communication link. Further communication between

52

computers 3301 and 3320 occurs via the VPN, e.g., using a "hopping" regime as discussed above. When VPN link 3321 is terminated at step 3413, flow continues to step 3414 where software module 3309 automatically replaces the secure top-level domain name with the corresponding non-secure top-level domain name for server 3304. Browser 3306 accesses a standard DNS 3325 for obtaining the non-secure URL for server 3304. Browser 3306 then connects to server 3304 in a well-known manner. At step 3415, browser 3306 displays the "go secure" hyperlink or icon for selecting a VPN communication link between terminal 3301 and server 3304. By again displaying the "go secure" hyperlink, a user is informed that the current communication link is a non-secure, non-VPN communication link.

When software module 3309 is being installed or when the user is off-line, the user can optionally specify that all communication links established over computer network 3302 are secure communication links. Thus, anytime that a communication link is established, the link is a VPN link. Consequently, software module 3309 transparently accesses SDNS 3313 for obtaining the URL for a selected secure website. In other words, in one embodiment, the user need not "click" on the secure option each time secure communication is to be effected.

Additionally, a user at computer 3301 can optionally select a secure communication link through proxy computer 3315. Accordingly, computer 3301 can establish a VPN communication link 3323 with secure server computer 3320 through proxy computer 3315. Alternatively, computer 3301 can establish a non-VPN communication link 3324 to a non-secure website, such as non-secure server computer 3304.

FIG. 35 shows a flow diagram 3500 for registering a secure domain name according to the present invention. At step 3501, a requester accesses website 3308 and logs into a secure domain name registry service that is available through website 3308. At step 3502, the requestor completes an online registration form for registering a secure domain name having a top-level domain name, such as .com, .net, .org, .edu, .mil or .gov. Of course, other secure top-level domain names can also be used. Preferably, the requestor must have previously registered a non-secure domain name corresponding to the equivalent secure domain name that is being requested. For example, a requestor attempting to register secure domain name "website.scom" must have previously registered the corresponding non-secure domain name "website.com".

At step 3503, the secure domain name registry service at website 3308 queries a non-secure domain name server database, such as standard DNS 3322, using, for example, a whois query, for determining ownership information relating to the non-secure domain name corresponding to the requested secure domain name. At step 3504, the secure domain name registry service at website 3308 receives a reply from standard DNS 3322 and at step 3505 determines whether there is conflicting ownership information for the corresponding non-secure domain name. If there is no conflicting ownership information, flow continues to step 3507, otherwise flow continues to step 3506 where the requestor is informed of the conflicting ownership information. Flow returns to step 3502.

When there is no conflicting ownership information at step 3505, the secure domain name registry service (website 3308) informs the requestor that there is no conflicting ownership information and prompts the requestor to verify the information entered into the online form and select an approved form of payment. After confirmation of the entered information and appropriate payment information, flow continues to step 3508 where the newly registered secure domain name sent to SDNS 3313 over communication link 3326.

US 7,418,504 B2

53

If, at step 3505, the requested secure domain name does not have a corresponding equivalent non-secure domain name, the present invention informs the requestor of the situation and prompts the requester for acquiring the corresponding equivalent non-secure domain name for an increased fee. By accepting the offer, the present invention automatically registers the corresponding equivalent non-secure domain name with standard DNS 3325 in a well-known manner. Flow then continues to step 3508.

#### G. Tunneling Secure Address Hopping Protocol Through Existing Protocol Using Web Proxy

The present invention also provides a technique for implementing the field hopping schemes described above in an application program on the client side of a firewall between two computer networks, and in the network stack on the server side of the firewall. The present invention uses a new secure connectionless protocol that provides good denial of service rejection capabilities by layering the new protocol on top of an existing IP protocol, such as the ICMP, UDP or TCP protocols. Thus, this aspect of the present invention does not require changes in the Internet infrastructure.

According to the invention, communications are protected by a client-side proxy application program that accepts unencrypted, unprotected communication packets from a local browser application. The client-side proxy application program tunnels the unencrypted, unprotected communication packets through a new protocol, thereby protecting the communications from a denial of service at the server side. Of course, the unencrypted, unprotected communication packets can be encrypted prior to tunneling.

The client-side proxy application program is not an operating system extension and does not involve any modifications to the operating system network stack and drivers. Consequently, the client is easier to install, remove and support in comparison to a VPN. Moreover, the client-side proxy application can be allowed through a corporate firewall using a much smaller "hole" in the firewall and is less of a security risk in comparison to allowing a protocol layer VPN through a corporate firewall.

The server-side implementation of the present invention authenticates valid field-hopped packets as valid or invalid very early in the server packet processing, similar to a standard virtual private network, for greatly minimizing the impact of a denial of service attempt in comparison to normal TCP/IP and HTTP communications, thereby protecting the server from invalid communications.

FIG. 36 shows a system block diagram of a computer network 3600 in which a virtual private connection according to the present invention can be configured to more easily traverse a firewall between two computer networks. FIG. 37 shows a flow diagram 3700 for establishing a virtual private connection that is encapsulated using an existing network protocol.

In FIG. 36 a local area network (LAN) 3601 is connected to another computer network 3602, such as the Internet, through a firewall arrangement 3603. Firewall arrangement operates in a well-known manner to interface LAN 3601 to computer network 3602 and to protect LAN 3601 from attacks initiated outside of LAN 3601.

A client computer 3604 is connected to LAN 3601 in a well-known manner. Client computer 3604 includes an operating system 3605 and a web browser 3606. Operating system 3605 provides kernel mode functions for operating client computer 3604. Browser 3606 is an application program for accessing computer network resources connected to LAN

54

3601 and computer network 3602 in a well-known manner. According to the present invention, a proxy application 3607 is also stored on client computer 3604 and operates at an application layer in conjunction with browser 3606. Proxy application 3607 operates at the application layer within client computer 3604 and when enabled, modifies unprotected, unencrypted message packets generated by browser 3606 by inserting data into the message packets that are used for forming a virtual private connection between client computer 3604 and a server computer connected to LAN 3601 or computer network 3602. According to the invention, a virtual private connection does not provide the same level of security to the client computer as a virtual private network. A virtual private connection can be conveniently authenticated so that, for example, a denial of service attack can be rapidly rejected, thereby providing different levels of service that can be subscribed to by a user.

Proxy application 3607 is conveniently installed and uninstalled by a user because proxy application 3607 operates at the application layer within client computer 3604. On installation, proxy application 3607 preferably configures browser 3606 to use proxy application for all web communications. That is, the payload portion of all message packets is modified with the data for forming a virtual private connection between client computer 3604 and a server computer. Preferably, the data for forming the virtual private connection contains field-hopping data, such as described above in connection with VPNs. Also, the modified message packets preferably conform to the UDP protocol. Alternatively, the modified message packets can conform to the TCP/IP protocol or the ICMP protocol. Alternatively, proxy application 3606 can be selected and enabled through, for example, an option provided by browser 3606. Additionally, proxy application 3607 can be enabled so that only the payload portion of specially designated message packets is modified with the data for forming a virtual private connection between client computer 3604 and a designated host computer. Specially designated message packets can be, for example, selected predetermined domain names.

Referring to FIG. 37, at step 3701, unprotected and unencrypted message packets are generated by browser 3606. At step 3702, proxy application 3607 modifies the payload portion of all message packets by tunneling the data for forming a virtual private connection between client computer 3604 and a destination server computer into the payload portion. At step, 3703, the modified message packets are sent from client computer 3604 to, for example, website (server computer) 3608 over computer network 3602.

Website 3608 includes a VPN guard portion 3609, a server proxy portion 3610 and a web server portion 3611. VPN guard portion 3609 is embedded within the kernel layer of the operating system of website 3608 so that large bandwidth attacks on website 3608 are rapidly rejected. When client computer 3604 initiates an authenticated connection to website 3608, VPN guard portion 3609 is keyed with the hopping sequence contained in the message packets from client computer 3604, thereby performing a strong authentication of the client packet streams entering website 3608 at step 3704. VPN guard portion 3609 can be configured for providing different levels of authentication and, hence, quality of service, depending upon a subscribed level of service. That is, VPN guard portion 3609 can be configured to let all message packets through until a denial of service attack is detected, in which case VPN guard portion 3609 would allow only client packet streams conforming to a keyed hopping sequence, such as that of the present invention.

## US 7,418,504 B2

55

Server proxy portion 3610 also operates at the kernel layer within website 3608 and catches incoming message packets from client computer 3604 at the VPN level. At step 3705, server proxy portion 3610 authenticates the message packets at the kernel level within host computer 3604 using the destination IP address, UDP ports and discriminator fields. The authenticated message packets are then forwarded to the authenticated message packets to web server portion 3611 as normal TCP web transactions.

At step 3705, web server portion 3611 responds to message packets received from client computer 3604 in accordance with the particular nature of the message packets by generating reply message packets. For example, when a client computer requests a webpage, web server portion 3611 generates message packets corresponding to the requested webpage. At step 3706, the reply message packets pass through server proxy portion 3610, which inserts data into the payload portion of the message packets that are used for forming the virtual private connection between host computer 3608 and client computer 3604 over computer network 3602. Preferably, the data for forming the virtual private connection is contains field-hopping data, such as described above in connection with VPNs. Server proxy portion 3610 operates at the kernel layer within host computer 3608 to insert the virtual private connection data into the payload portion of the reply message packets. Preferably, the modified message packets sent by host computer 3608 to client computer 3604 conform to the UDP protocol. Alternatively, the modified message packets can conform to the TCP/IP protocol or the ICMP protocol.

At step 3707, the modified packets are sent from host computer 3608 over computer network 3602 and pass through firewall 3603. Once through firewall 3603, the modified packets are directed to client computer 3604 over LAN 3601 and are received at step 3708 by proxy application 3607 at the application layer within client computer 3604. Proxy application 3607 operates to rapidly evaluate the modified message packets for determining whether the received packets should be accepted or dropped. If the virtual private connection data inserted into the received information packets conforms to expected virtual private connection data, then the received packets are accepted. Otherwise, the received packets are dropped.

While the present invention has been described in connection with the illustrated embodiments, it will be appreciated and understood that modifications may be made without departing from the true spirit and scope of the invention.

What is claimed is:

1. A system for providing a domain name service for establishing a secure communication link, the system comprising: a domain name service system configured to be connected to a communication network, to store a plurality of domain names and corresponding network addresses, to receive a query for a network address, and to comprise an indication that the domain name service system supports establishing a secure communication link.

2. The system of claim 1, wherein at least one of the plurality of domain names comprises a top-level domain name.

3. The system of claim 2, wherein the top-level domain name is a non-standard top-level domain name.

4. The system of claim 3, wherein the non-standard top-level domain name is one of .com, .org, .net, .gov, .edu, .mil and .int.

5. The system of claim 2, wherein the domain name service system is configured to authenticate the query using a cryptographic technique.

56

6. The system of claim 1, wherein the communication network includes the Internet.

7. The system of claim 1, wherein the domain name service system comprises an edge router.

8. The system of claim 1, wherein the domain name service system is connectable to a virtual private network through the communication network.

9. The system of claim 8, wherein the virtual private network is one of a plurality of secure communication links in a hierarchy of secure communication links.

10. The system of claim 8, wherein the virtual private network is based on inserting into each data packet communicated over a secure communication link one or more data values that vary according to a pseudo-random sequence.

11. The system of claim 8, wherein the virtual private network is based on a network address hopping regime that is used to pseudorandomly change network addresses in packets transmitted between a first device and a second device.

12. The system of claim 8, wherein the virtual private network is based on comparing a value in each data packet transmitted between a first device and a second device to a moving window of valid values.

13. The system of claim 8, wherein the virtual private network is based on a comparison of a discriminator field in a header of each data packet to a table of valid discriminator fields maintained for a first device.

14. The system of claim 1, wherein the domain name service system is configured to respond to the query for the network address.

15. The system of claim 1, wherein the domain name service system is configured to provide, in response to the query, the network address corresponding to a domain name from the plurality of domain names and the corresponding network addresses.

16. The system of claim 1, wherein the domain name service system is configured to receive the query initiated from a first location, the query requesting the network address associated with a domain name, wherein the domain name service system is configured to provide the network address associated with a second location, and wherein the domain name service system is configured to support establishing a secure communication link between the first location and the second location.

17. The system of claim 1, wherein the domain name service system is connected to a communication network, stores a plurality of domain names and corresponding network addresses, and comprises an indication that the domain name service system supports establishing a secure communication link.

18. The system of claim 1, wherein at least one of the plurality of domain names is reserved for secure communication links.

19. The system of claim 1, wherein the domain name service system comprises a server.

20. The system of claim 19, wherein the domain name service system further comprises a domain name database, and wherein the domain name database stores the plurality of domain names and the corresponding network addresses.

21. The system of claim 1, wherein the domain name service system comprises a server, wherein the server comprises a domain name database, and wherein the domain name database stores the plurality of domain names and the corresponding network addresses.

22. The system of claim 1, wherein the domain name service system is configured to store the corresponding network addresses for use in establishing secure communication links.



## US 7,418,504 B2

57

23. The system of claim 1, wherein the domain name service system is configured to authenticate the query for the network address.

24. The system of claim 1, wherein at least one of the plurality of domain names comprises an indication that the domain name service system supports establishing a secure communication link.

25. The system of claim 1, wherein at least one of the plurality of domain names comprises a secure name.

26. The system of claim 1, wherein at least one of the plurality of domain names enables establishment of a secure communication link.

27. The system of claim 1, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location transparently to a user at the first location.

28. The system of claim 1, wherein the secure communication link uses encryption.

29. The system of claim 1, wherein the secure communication link is capable of supporting a plurality of services.

30. The system of claim 29, wherein the plurality of services comprises a plurality of communication protocols, a plurality of application programs, multiple sessions, or a combination thereof.

31. The system of claim 30, wherein the plurality of application programs comprises items selected from a group consisting of the following: video conferencing, e-mail, a word processing program, and telephony.

32. The system of claim 29, wherein the plurality of services comprises audio, video, or a combination thereof.

33. The system of claim 1, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location.

34. The system of claim 33, wherein the query is initiated from the first location, wherein the second location comprises a computer, and wherein the network address is an address associated with the computer.

35. The system of claim 1, wherein the domain name service system comprises a domain name database connected to a communication network and storing a plurality of domain names and corresponding network addresses for communication,

wherein the domain name database is configured so as to provide a network address corresponding to a domain name in response to a query in order to establish a secure communication link.

36. A machine-readable medium comprising instructions executable in a domain name service system, the instructions comprising code for:

connecting the domain name service system to a communication network;

storing a plurality of domain names and corresponding network addresses;

receiving a query for a network address; and

supporting an indication that the domain name service system supports establishing a secure communication link.

37. The machine-readable medium of claim 36, wherein the instructions comprise code for storing the plurality of domain names and corresponding network addresses including at least one top-level domain name.

38. The machine-readable medium of claim 36, wherein the instructions comprise code for responding to the query for the network address.

39. The machine-readable medium of claim 36, wherein the instructions comprise code for providing, in response to

58

the query, the network address corresponding to a domain name from the plurality of domain names and the corresponding network addresses.

40. The machine-readable medium of claim 36, wherein the instructions comprise code for receiving the query for a network address associated with a domain name and initiated from a first location, and providing a network address associated with a second location, and establishing a secure communication link between the first location and the second location.

41. The machine-readable medium of claim 36, wherein the instructions comprise code for indicating that the domain name service system supports the establishment of a secure communication link.

42. The machine-readable medium of claim 36, wherein the instructions comprise code for reserving at least one of the plurality of domain names for secure communication links.

43. The machine-readable medium of claim 36, wherein the code resides on a server.

44. The machine-readable medium of claim 36, wherein the instructions comprise code for storing a plurality of domain names and corresponding network addresses so as to define a domain name database.

45. The machine-readable medium of claim 36, wherein the code resides on a server, and the instructions comprise code for creating a domain name database configured to store the plurality of domain names and the corresponding network addresses.

46. The machine-readable medium of claim 36, wherein the instructions comprise code for storing the corresponding network addresses for use in establishing secure communication links.

47. The machine-readable medium of claim 36, wherein the instructions comprise code for authenticating the query for the network address.

48. The machine-readable medium of claim 36, wherein at least one of the plurality of domain names includes an indication that the domain name service system supports the establishment of a secure communication link.

49. The machine-readable medium of claim 36, wherein at least one of the plurality of domain names includes a secure name.

50. The machine-readable medium of claim 36, wherein at least one of the plurality of domain names is configured so as to enable establishment of a secure communication link.

51. The machine-readable medium of claim 36, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location transparently to a user at the first location.

52. The machine-readable medium of claim 36, wherein the secure communication link uses encryption.

53. The machine-readable medium of claim 36, wherein the secure communication link is capable of supporting a plurality of services.

54. The machine-readable medium of claim 53, wherein the plurality of services comprises a plurality of communication protocols, a plurality of application programs, multiple sessions, or a combination thereof.

55. The machine-readable medium of claim 54, wherein the plurality of application programs comprises items selected from a group consisting of the following: video conferencing, e-mail, a word processing program, and telephony.

56. The machine-readable medium of claim 53, wherein the plurality of services comprises audio, video, or a combination thereof.



## US 7,418,504 B2

59

57. The machine-readable medium of claim 36, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location.

58. The machine-readable medium of claim 57, wherein the instructions include code for receiving a query initiated from the first location, wherein the second location comprises a computer, and wherein the network address is an address associated with the computer.

59. The machine-readable medium of claim 36, wherein the domain name service system comprises a domain name database connected to a communication network and storing a plurality of domain names and corresponding network addresses for communication,

wherein the domain name database is configured so as to provide a network address corresponding to a domain

60

name is response to the query in order to establish a secure communication link.

60. A method of providing a domain name service for establishing a secure communication link, the method comprising:

connecting a domain name service system to a communication network, the domain name service system comprising an indication that the domain name service system supports establishing a secure communication link; storing a plurality of domain names and corresponding network addresses; and

receiving a query for a network address for communication.

\* \* \* \* \*

(12) **United States Patent**  
**Munger et al.**

(10) **Patent No.:** **US 7,490,151 B2**  
(45) **Date of Patent:** **Feb. 10, 2009**

(54) **ESTABLISHMENT OF A SECURE COMMUNICATION LINK BASED ON A DOMAIN NAME SERVICE (DNS) REQUEST**

(75) Inventors: **Edward Colby Munger**, Crownsville, MD (US); **Robert Dunham Short, III**, Leesburg, VA (US); **Victor Larson**, Fairfax, VA (US); **Michael Williamson**, South Riding, VA (US)

(73) Assignee: **Virnetx Inc.**, Scotts Valley Drive, CA (US)

(\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 818 days.

(21) Appl. No.: **10/259,494**

(22) Filed: **Sep. 30, 2002**

(65) **Prior Publication Data**  
US 2003/0037142 A1 Feb. 20, 2003

#### **Related U.S. Application Data**

(60) Division of application No. 09/504,783, filed on Feb. 15, 2000, now Pat. No. 6,502,135, which is a continuation-in-part of application No. 09/429,643, filed on Oct. 29, 1999, now Pat. No. 7,010,604.

(60) Provisional application No. 60/137,704, filed on Jun. 7, 1999, provisional application No. 60/106,261, filed on Oct. 30, 1998.

(51) **Int. Cl.**  
**G06F 15/173** (2006.01)

(52) **U.S. Cl.** ..... **709/225; 709/229**

(58) **Field of Classification Search** ..... **709/217-225, 709/229; 713/201**

See application file for complete search history.

(56) **References Cited**

#### **U.S. PATENT DOCUMENTS**

4,933,846 A 6/1990 Humphrey et al.

(Continued)

#### **FOREIGN PATENT DOCUMENTS**

DE 199 24 575 12/1999

(Continued)

#### **OTHER PUBLICATIONS**

Search Report (dated Aug. 23, 2002), International Application No. PCT/US01/13260.

(Continued)

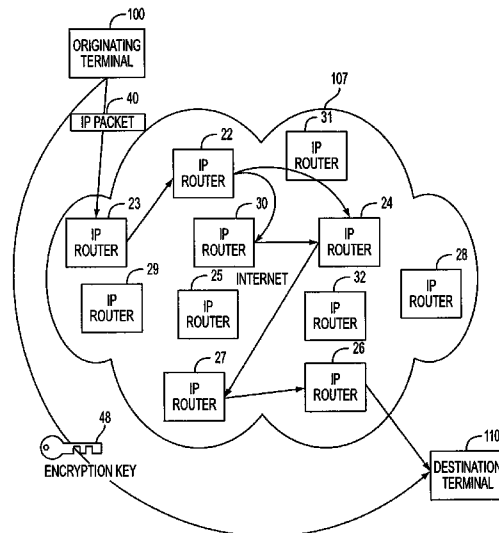
*Primary Examiner*—Krisna Lim

(74) *Attorney, Agent, or Firm*—McDermott Will & Emery

(57) **ABSTRACT**

A plurality of computer nodes communicate using seemingly random Internet Protocol source and destination addresses. Data packets matching criteria defined by a moving window of valid addresses are accepted for further processing, while those that do not meet the criteria are quickly rejected. Improvements to the basic design include (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities.

**16 Claims, 35 Drawing Sheets**



## US 7,490,151 B2

Page 2

## U.S. PATENT DOCUMENTS

4,988,990 A	1/1991	Warrior	
5,164,986 A *	11/1992	Bright	380/273
5,276,735 A	1/1994	Boebert et al.	
5,311,593 A	5/1994	Carmi	
5,329,521 A	7/1994	Walsh et al.	
5,341,426 A	8/1994	Barney et al.	
5,367,643 A	11/1994	Chang et al.	
5,559,883 A	9/1996	Williams	
5,561,669 A	10/1996	Lenney et al.	
5,588,060 A	12/1996	Aziz	
5,625,626 A	4/1997	Umekita	
5,654,695 A	8/1997	Olnowich et al.	
5,682,480 A	10/1997	Nakagawa	
5,689,566 A	11/1997	Nguyen	
5,740,375 A	4/1998	Dunne et al.	
5,774,660 A	6/1998	Brendel et al.	
5,787,172 A	7/1998	Arnold	
5,790,548 A *	8/1998	Sistanizadeh et al.	370/401
5,796,942 A	8/1998	Esbensen	
5,805,801 A	9/1998	Holloway et al.	
5,842,040 A	11/1998	Hughes et al.	
5,845,091 A	12/1998	Dunne et al.	
5,867,650 A	2/1999	Osterman	
5,870,610 A	2/1999	Beyda et al.	
5,878,231 A	3/1999	Baehr et al.	
5,892,903 A	4/1999	Klaus	
5,898,830 A *	4/1999	Wesinger et al.	726/15
5,905,859 A	5/1999	Holloway et al.	
5,918,019 A	6/1999	Valencia	
5,996,016 A	11/1999	Thalheimer et al.	
6,006,259 A	12/1999	Adelman et al.	
6,006,272 A	12/1999	Aravamudan et al.	
6,016,318 A	1/2000	Tomoike	
6,016,512 A	1/2000	Huitema	
6,041,342 A	3/2000	Yamaguchi	
6,052,788 A	4/2000	Wesinger, Jr. et al.	
6,055,574 A	4/2000	Smorodinsky et al.	
6,061,736 A	5/2000	Rochberger et al.	
6,079,020 A *	6/2000	Liu	713/201
6,092,200 A	7/2000	Muniyappa et al.	
6,101,182 A *	8/2000	Sistanizadeh et al.	370/352
6,119,171 A	9/2000	Alkhatib	
6,119,234 A *	9/2000	Aziz et al.	713/201
6,147,976 A	11/2000	Shand et al.	
6,157,957 A	12/2000	Berthaud	
6,158,011 A	12/2000	Chen et al.	
6,168,409 B1	1/2001	Fare	
6,175,867 B1	1/2001	Taghadoss	
6,178,409 B1	1/2001	Weber et al.	
6,178,505 B1	1/2001	Schneider et al.	
6,179,102 B1	1/2001	Weber et al.	
6,222,842 B1	4/2001	Sasyan et al.	
6,226,751 B1	5/2001	Arrow et al.	
6,233,618 B1	5/2001	Shannon	
6,243,360 B1	6/2001	Basilico	
6,243,749 B1	6/2001	Sitaraman et al.	
6,243,754 B1	6/2001	Guerin et al.	
6,256,671 B1 *	7/2001	Strentzsch et al.	709/227
6,263,445 B1	7/2001	Blumenau	
6,286,047 B1	9/2001	Ramanathan et al.	
6,301,223 B1	10/2001	Hrastar et al.	
6,308,274 B1	10/2001	Swift	
6,311,207 B1	10/2001	Mighdoll et al.	
6,324,161 B1	11/2001	Kirch	
6,330,562 B1	12/2001	Boden et al.	
6,332,158 B1 *	12/2001	Risley et al.	709/219
6,353,614 B1	3/2002	Borella et al.	
6,425,003 B1 *	7/2002	Herzog et al.	709/223
6,430,155 B1	8/2002	Davie et al.	
6,430,610 B1	8/2002	Carter	
6,487,598 B1	11/2002	Valencia	
6,502,135 B1 *	12/2002	Munger et al.	709/225
6,505,232 B1	1/2003	Mighdoll et al.	
6,510,154 B1	1/2003	Mayes et al.	
6,549,516 B1	4/2003	Albert et al.	
6,557,037 B1	4/2003	Provino	
6,571,296 B1	5/2003	Dillon	
6,571,338 B1	5/2003	Shaio et al.	
6,581,166 B1	6/2003	Hirst et al.	
6,606,708 B1 *	8/2003	Devine et al.	713/201
6,618,761 B2	9/2003	Munger et al.	
6,671,702 B2	12/2003	Kruglikov et al.	
6,687,551 B2	2/2004	Steindl	
6,714,970 B1	3/2004	Fiveash et al.	
6,717,949 B1	4/2004	Boden et al.	
6,751,738 B2 *	6/2004	Wesinger et al.	713/201
6,760,766 B1	7/2004	Sahlqvist	
6,826,616 B2	11/2004	Larson et al.	
6,839,759 B2	1/2005	Larson et al.	
7,010,604 B1	3/2006	Munger et al.	
7,133,930 B2	11/2006	Munger et al.	
7,188,180 B2	3/2007	Larson et al.	
7,197,563 B2	3/2007	Sheymov et al.	
2002/0004898 A1	1/2002	Droge	
2003/0196122 A1 *	10/2003	Wesinger et al.	713/201
2005/0055306 A1	3/2005	Miller et al.	
2006/0059337 A1 *	3/2006	Poyhonen et al.	713/165

## FOREIGN PATENT DOCUMENTS

EP	0 814 589	12/1997
EP	0 814 589 A	12/1997
EP	0 838 930	4/1998
EP	0 838 930 A	4/1998
EP	836306 A1	4/1998
EP	0 858 189	8/1998
GB	2 317 792	4/1998
GB	2 317 792 A	4/1998
GB	2 334 181 A	8/1999
GB	2334181 A	8/1999
WO	9827783 A	6/1998
WO	WO 98/27783	6/1998
WO	WO 9827783 A	6/1998
WO	WO 98 55930	12/1998
WO	WO 98 59470	12/1998
WO	WO 99 38081	7/1999
WO	WO 99 48303	9/1999
WO	WO 00/17775	3/2000
WO	WO 00/70458	11/2000
WO	WO 01 50688	7/2001

## OTHER PUBLICATIONS

Donald E. Eastlake, 3<sup>rd</sup>, "Domain Name System Security Extensions", Internet Draft, Apr. 1998, pp. 1-51.

D. B. Chapman et al., "Building Internet Firewalls", Nov. 1995, pp. 278-375.

P. Srisuresh et al., "DNA extensions to Network address Translators (DNS\_ALG)", Internet Draft, Jul. 1998, pp. 1-27.

James E. Bellaire, "New Statement of Rules—Naming Internet Domains", Internet Newsgroup, Jul. 30, 1995, 1 page.

D. Clark, "US Calls for Private Domain-Name System", Computer Society, Aug. 1, 1998, pp. 22-25.

August Bequai, "Balancing Legal Concerns Over Crime and Security in Cyberspace", Computer & Security, vol. 17, No. 4, 1998, pp. 293-298.

Rich Winkel, "CAQ: Networking With Spooks: The NET & The Control Of Information", Internet Newsgroup, Jun. 21, 1997, 4 pages.

Search Report (dated Jun. 18, 2002), International Application No. PCT/US01/13260.

Search Report (dated Jun. 28, 2002), International Application No. PCT/US01/13261.

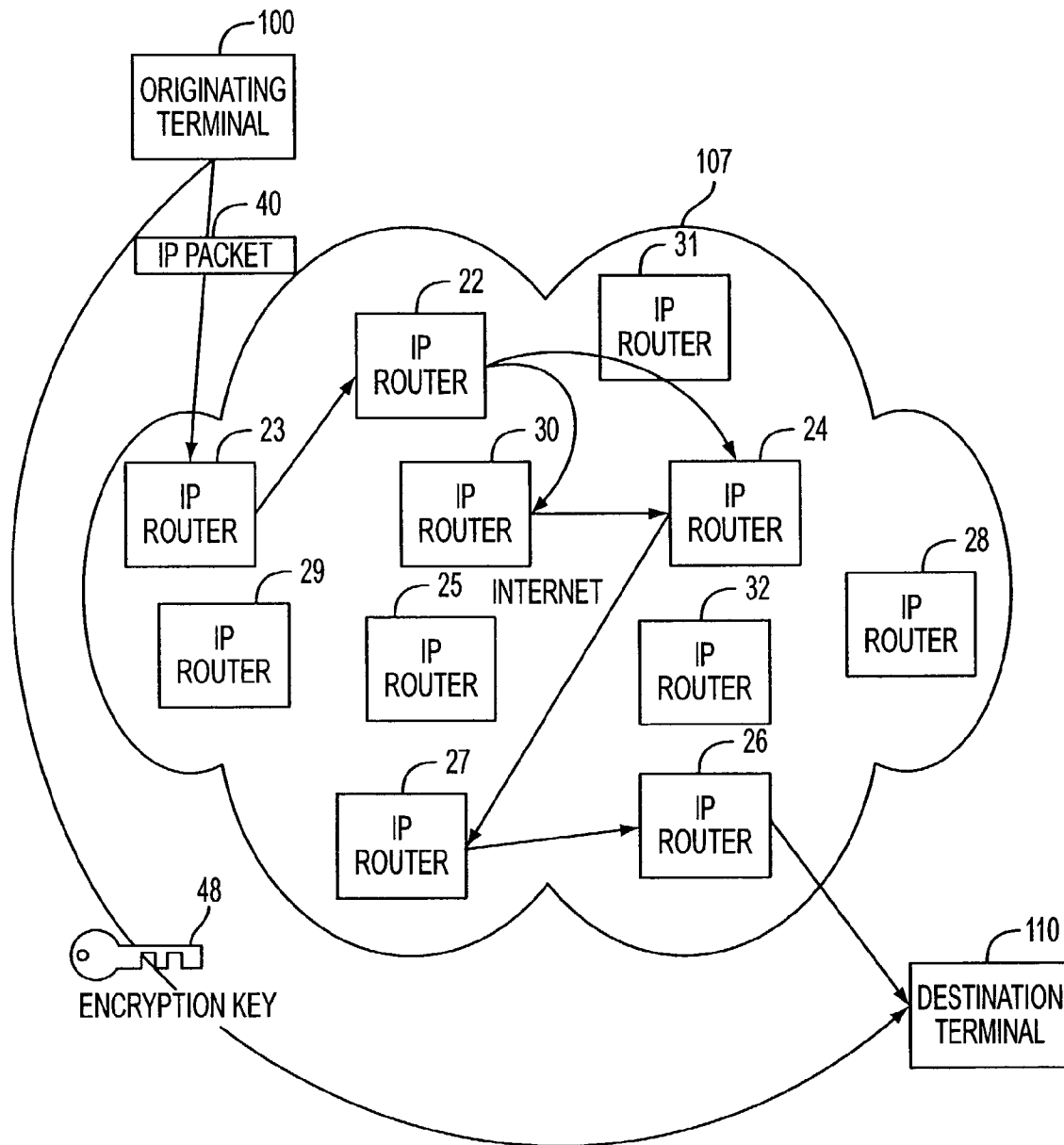
Donald E. Eastlake, "Domain Name System Security Extensions", DNS Security Working Group, Apr. 1998, 51 pages.

**US 7,490,151 B2**

Page 3

- D. B. Chapman et al., "Building Internet Firewalls", Nov. 1995, pp. 278-297 and pp. 351-375.
- P. Srisuresh et al., "DNS extensions to Network Address Translators", Jul. 1998, 27 pages.
- Laurie Wells, "Security Icon", Oct. 19, 1998, 1 page.
- W. Stallings, "Cryptography And Network Security", 2<sup>nd</sup> Edition, Chapter 13, IP Security, Jun. 8, 1998, pp. 399-440.
- W. Stallings, "New Cryptography and Network Security Book", Jun. 8, 1998, 3 pages.
- Search Report (dated Aug. 20, 2002), International Application No. PCT/US01/04340.
- Shree Murthy et al., "Congestion-Oriented Shortest Multipath Routing", Proceedings of IEEE Infocom, 1996, pp. 1028-1036.
- Jim Jones et al., "Distributed Denial of Service Attacks: Defenses", Global Integrity Corporation, 2000, pp. 1-14.
- Fasbender, Kesdogan, and Kubitz: "Variable and Scalable Security: Protection of Location Information in Mobile IP", IEEE publication, 1996, pp. 963-967.
- Laurie Wells (Lancasterbibelmail MSN COM); "Subject: Security Icon" Usenet Newsgroup, Oct. 19, 1998, XP002200606.
- Davila J et al, "Implementation of Virtual Private Networks at the Transport Layer", Information Security, Second International Workshop, ISW '99. Proceedings (Lecture Springer-Verlag Berlin, Germany, [Online] 1999, pp. 85-102, XP002399276, ISBN 3-540-66695-B, retrieved from the Internet: URL: <http://www.springerlink.com/content/4uac0tb0heccma89/fulltext.pdf> (Abstract).
- Alan O. Frier et al., "The SSL Protocol Version 3.0", Nov. 18, 1996, printed from <http://www.netscape.com/eng/ssl13/draft302.txt> on Feb. 4, 2002, 56 pages.
- Davila J et al, "Implementation of Virtual Private Networks at the Transport Layer", Information Security, Second International Workshop, ISW'99. Proceedings (Lecture Springer-Verlag Berlin, Germany, [Online] 1999, pp. 85-102, XP002399276, ISBN 3-540-66695-B, retrieved from the Internet: URL: <http://www.springerlink.com/content/4uac0tb0heccma89/fulltext.pdf>).
- Dolev, Shlomi and Ostrovsky, Rafil, Efficient Anonymous Multicast and Reception (Extended Abstract), 16 pages.
- F. Halsall, "Data Communications, Computer Networks and Open Systems", Chapter 4, Protocol Basics, 1996, pp. 198-203.
- Glossary for the Linux FreeS/WAN project, printed from [http://liberty.freesswan.org/freeswan\\_trees/freeswan-1.3/doc/glossary.html](http://liberty.freesswan.org/freeswan_trees/freeswan-1.3/doc/glossary.html) on Feb. 21, 2002, 25 pages.
- J. Gilmore, "Swan: Securing the Internet against Wiretapping", printed from [http://liberty.freesswan.org/freeswan\\_trees/freeswan-1.3.doc/rationale.html](http://liberty.freesswan.org/freeswan_trees/freeswan-1.3.doc/rationale.html) on Feb. 21, 2002, 4 pages.
- Linux FreeS/WAN Index File, printed from [http://liberty.freesswan.org/freeswan\\_trees/freeswan-1.3/doc/](http://liberty.freesswan.org/freeswan_trees/freeswan-1.3/doc/) on Feb. 21, 2002, 3 pages.
- Reiter, Michael K. and Rubin, Aviel D. (AT&T Labs—Research), Crowds: Anonymity for Web Transactions, pp. 1-23.
- RFC 2401-Security Architecture for the Internet Protocol (RTP).
- RFC 2543-SIP: Session Initiation Protocol (SIP or SIPS).
- Rubin, Aviel D., Geer, Daniel, and Ranum, Marcus J. (Wiley Computer Publishing), "Web Security Sourcebook", pp. 82-94.
- Search Report, IPER (dated Nov. 13, 2002), International Application No. PCT/US01/04340.
- Search Report, IPER (dated Feb. 6, 2002), International Application No. PCT/US01/13261.
- Search Report, IPER (dated Jan. 14, 2003), International Application No. PCT/US01/13260.
- Shankar, A.U. "A verified sliding window protocol with variable flow control". Proceedings of ACM SIGCOMM conference on Communications architectures & protocols. pp. 84-91, ACM Press, NY,NY 1986.

\* cited by examiner



**FIG. 1**

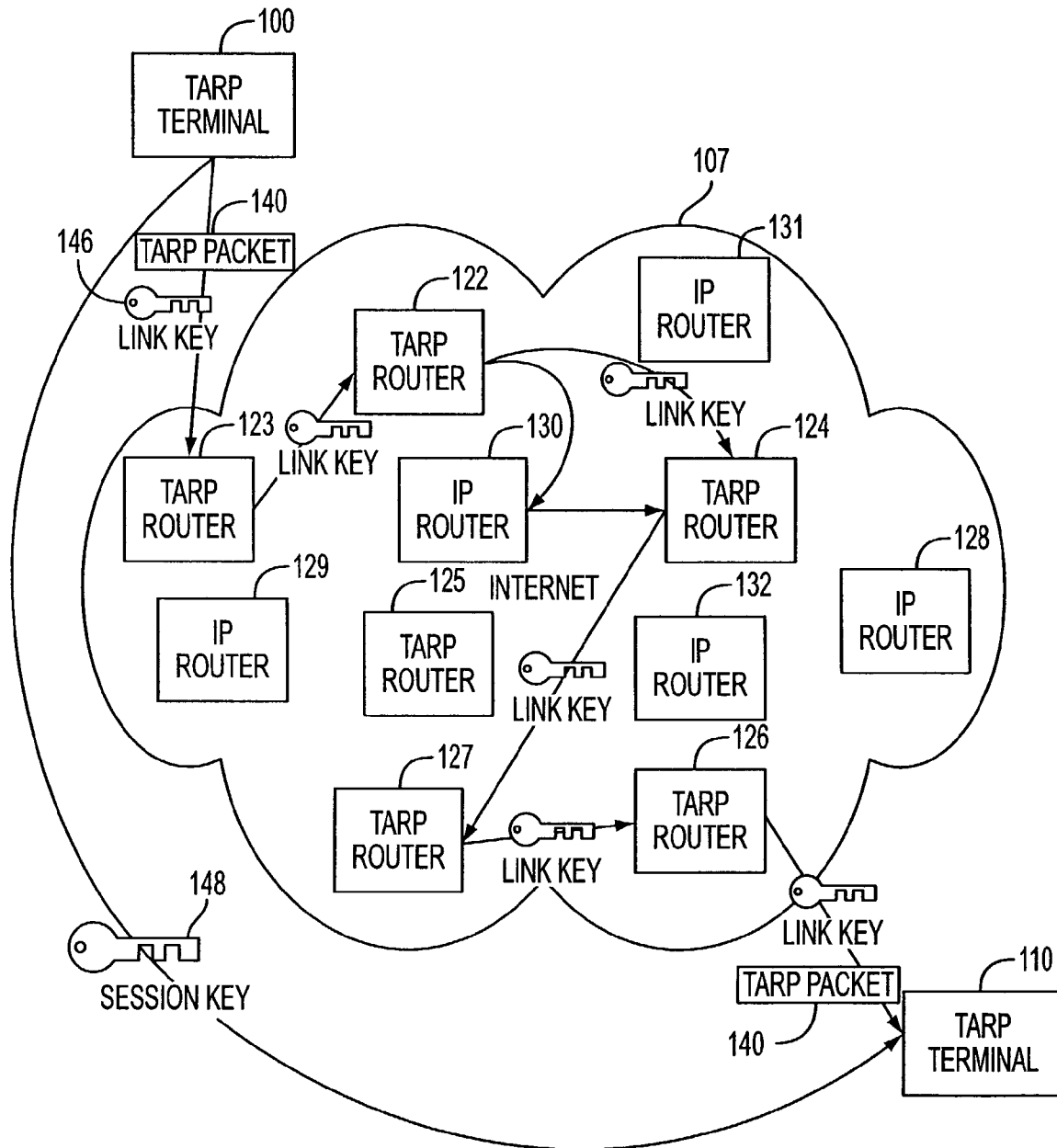


FIG. 2

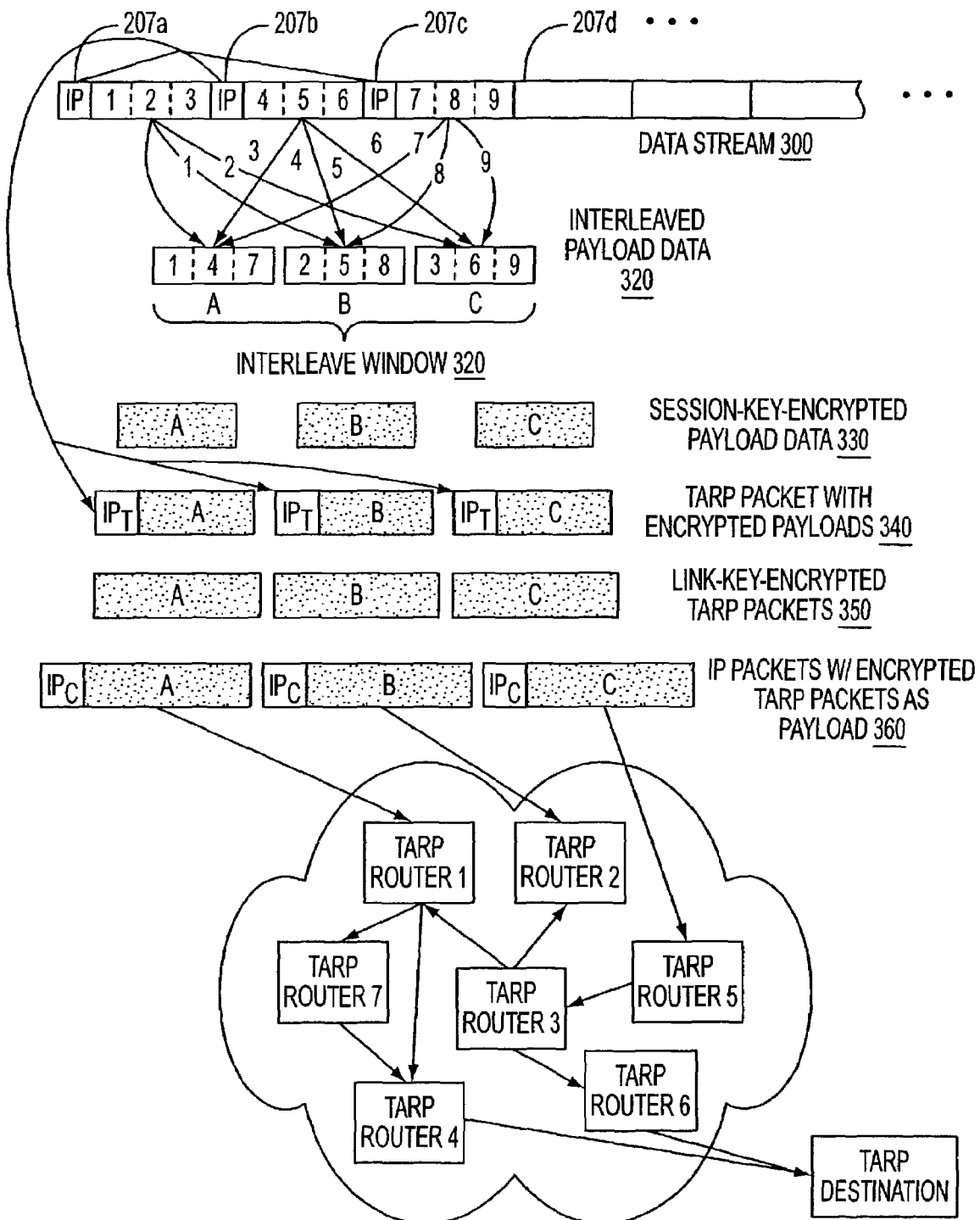


FIG. 3A

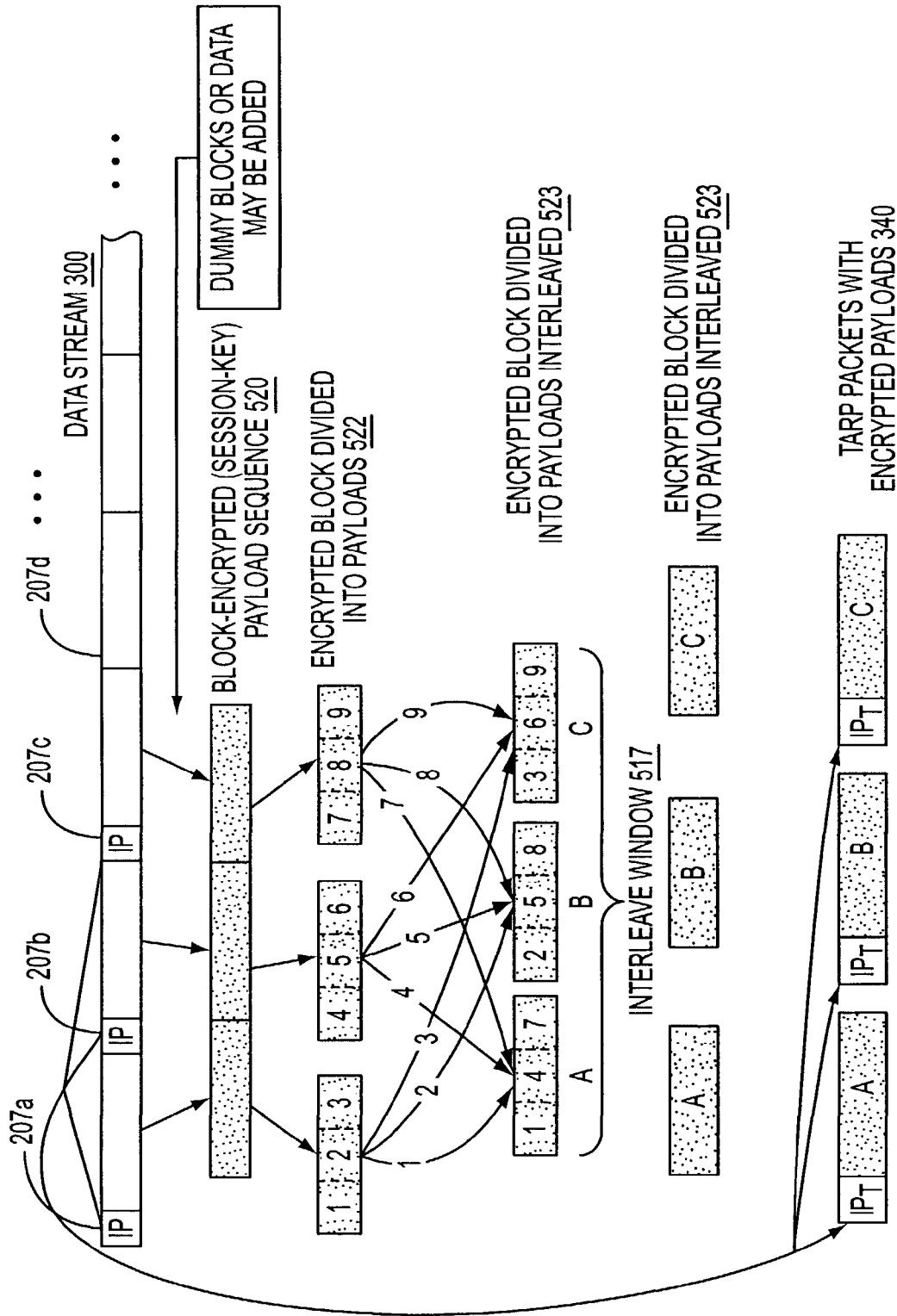
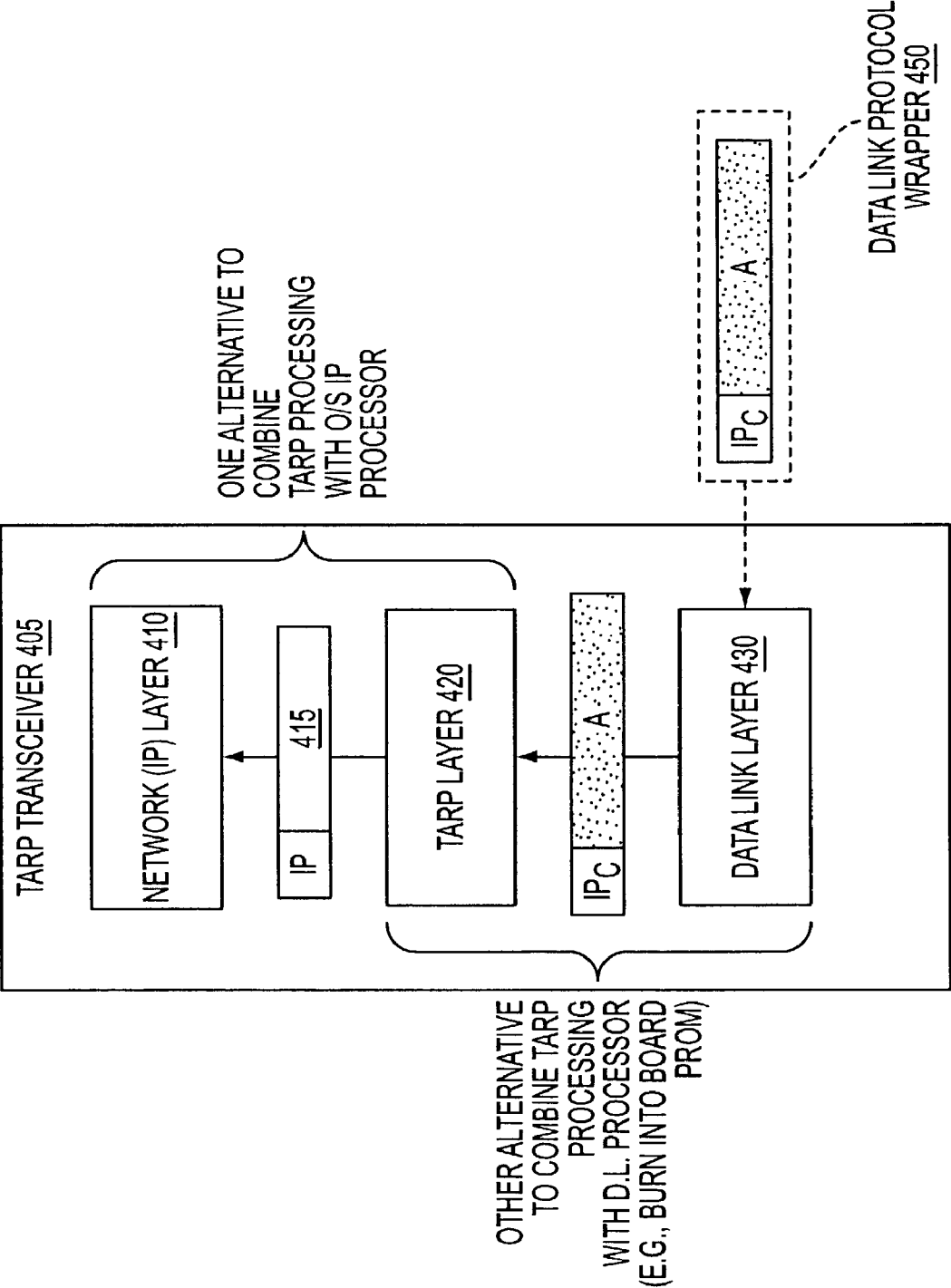


FIG. 3B

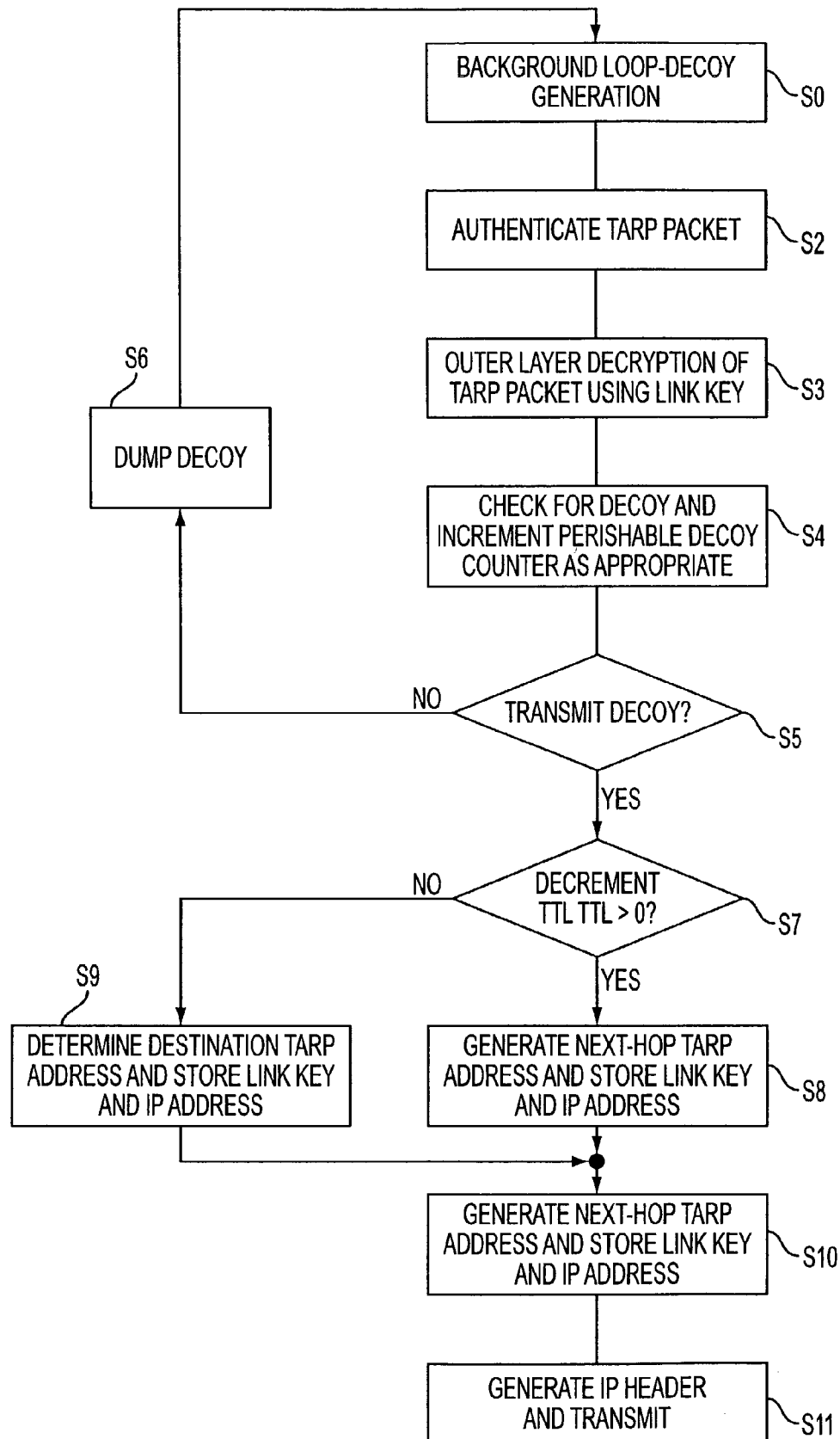




**U.S. Patent**

Feb. 10, 2009

Sheet 6 of 35

**US 7,490,151 B2****FIG. 5****Appx273**

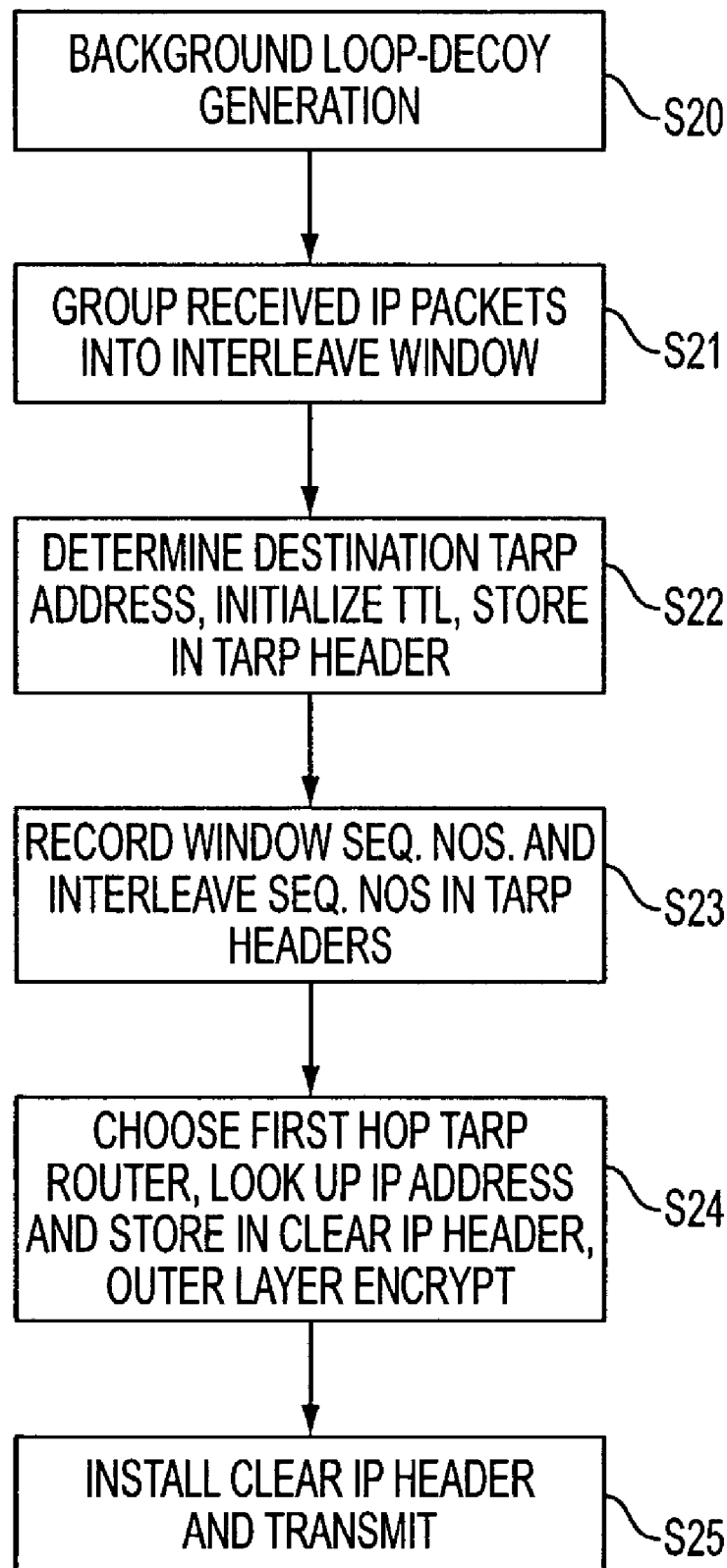


FIG. 6

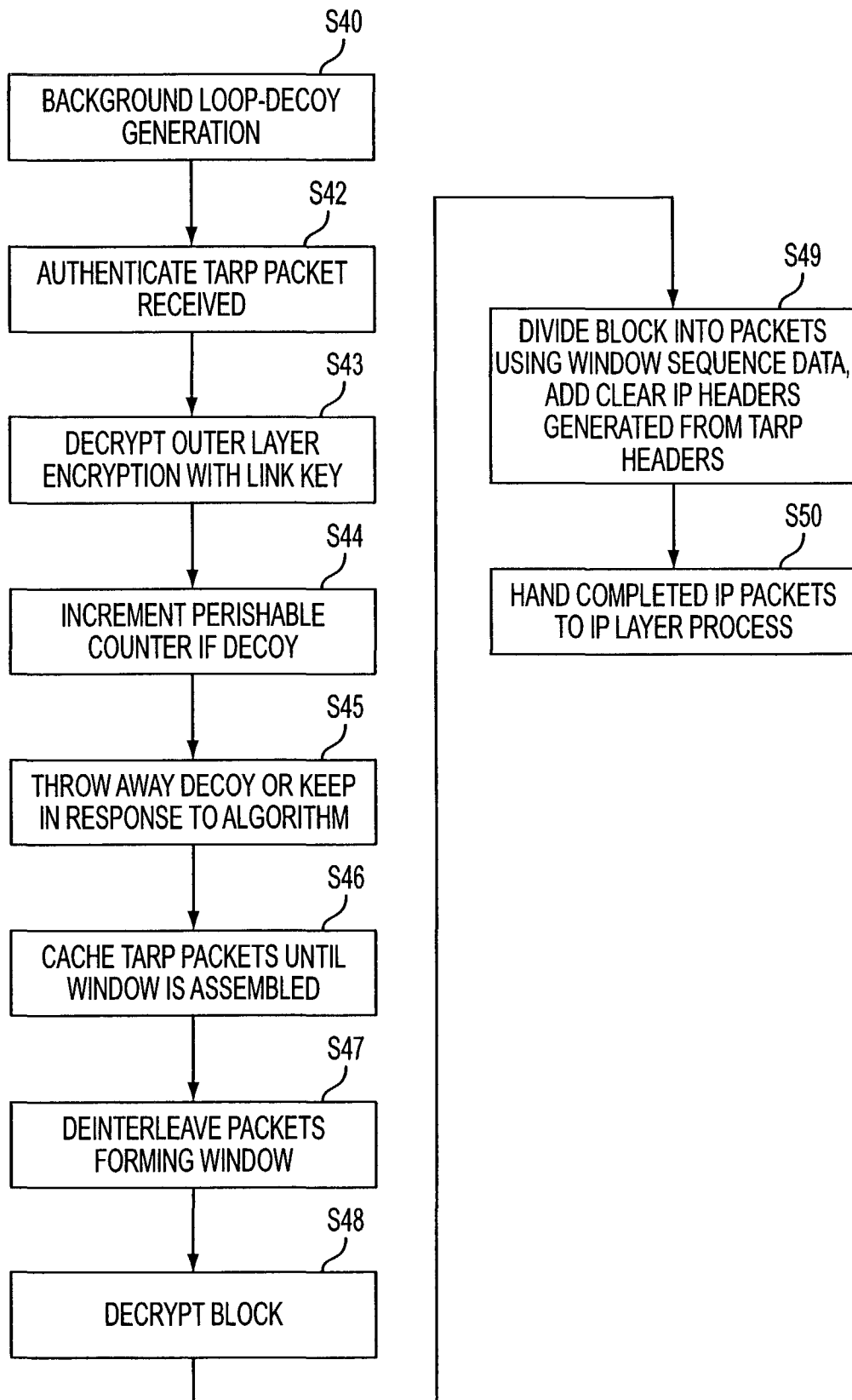


FIG. 7

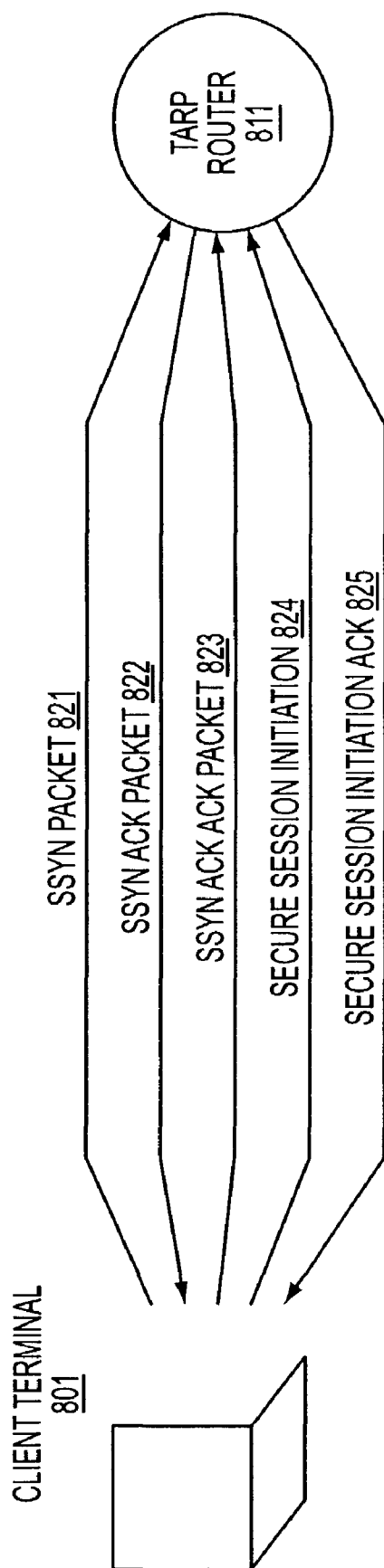
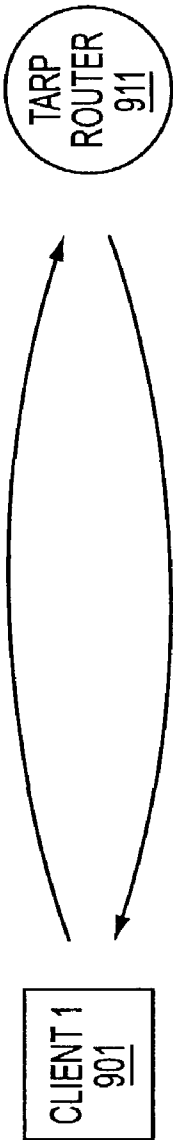


FIG. 8



TRANSMIT TABLE 921		RECEIVE TABLE 924	
131.218.204.98	, 131.218.204.65	131.218.204.98	, 131.218.204.65
131.218.204.221	, 131.218.204.97	131.218.204.221	, 131.218.204.97
131.218.204.139	, 131.218.204.186	131.218.204.139	, 131.218.204.186
131.218.204.12	, 131.218.204.55	131.218.204.12	, 131.218.204.55
.	.	.	.
.	.	.	.
.	.	.	.

RECEIVE TABLE 922		TRANSMIT TABLE 923	
131.218.204.161	, 131.218.204.89	131.218.204.161	, 131.218.204.89
131.218.204.66	, 131.218.204.212	131.218.204.66	, 131.218.204.212
131.218.204.201	, 131.218.204.127	131.218.204.201	, 131.218.204.127
131.218.204.119	, 131.218.204.49	131.218.204.119	, 131.218.204.49
.	.	.	.
.	.	.	.
.	.	.	.

FIG. 9

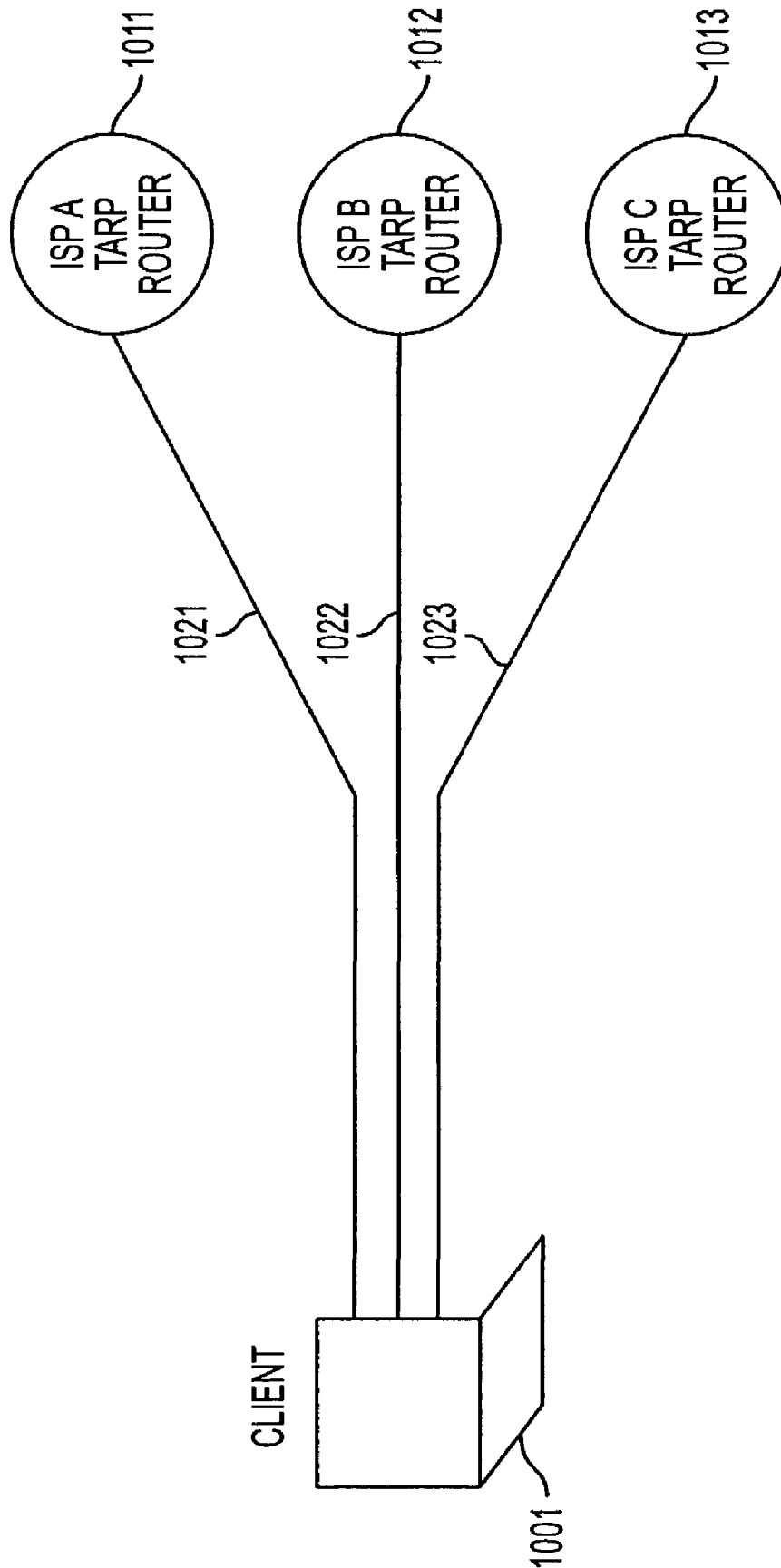


FIG. 10

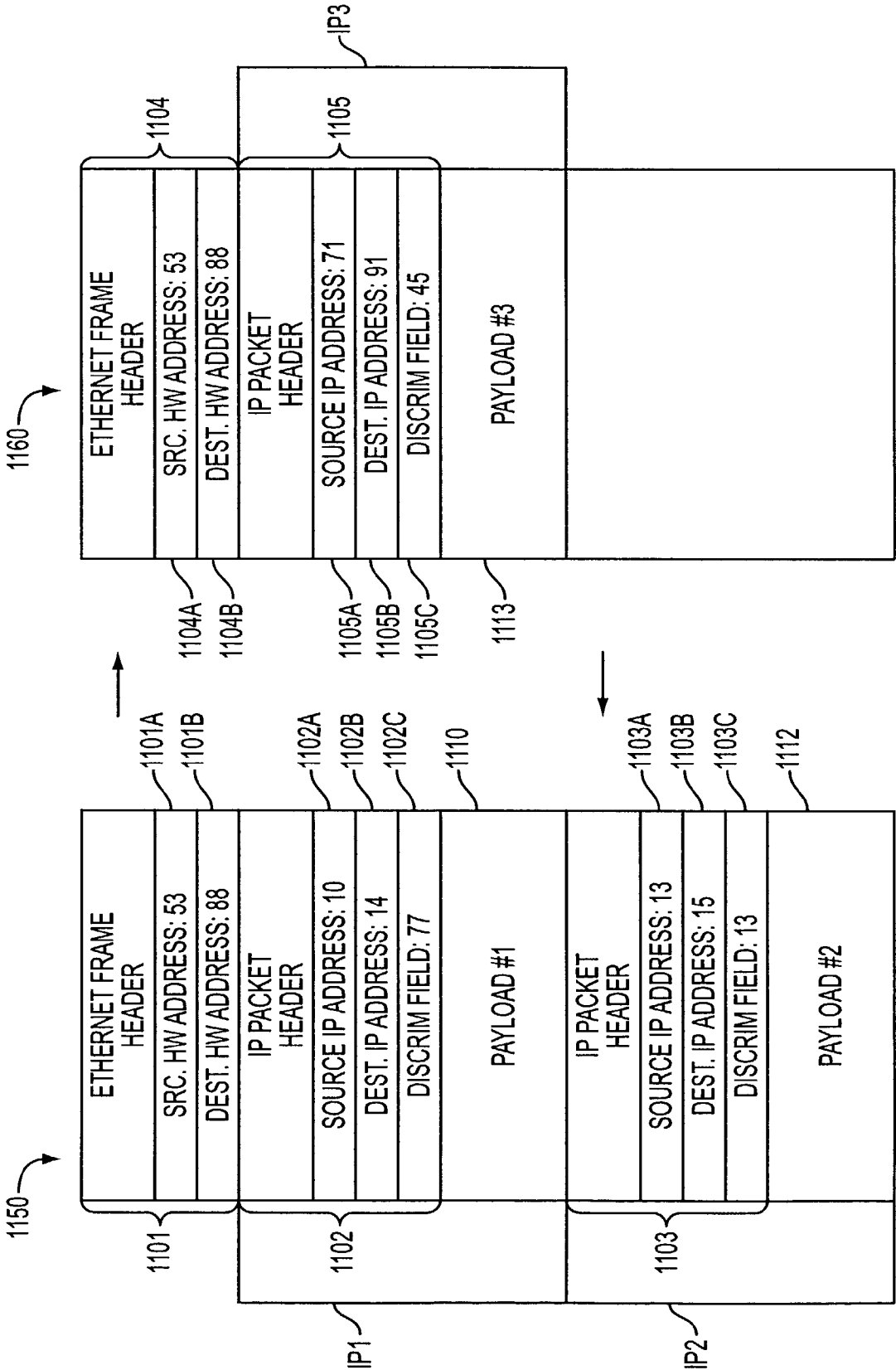


FIG. 11



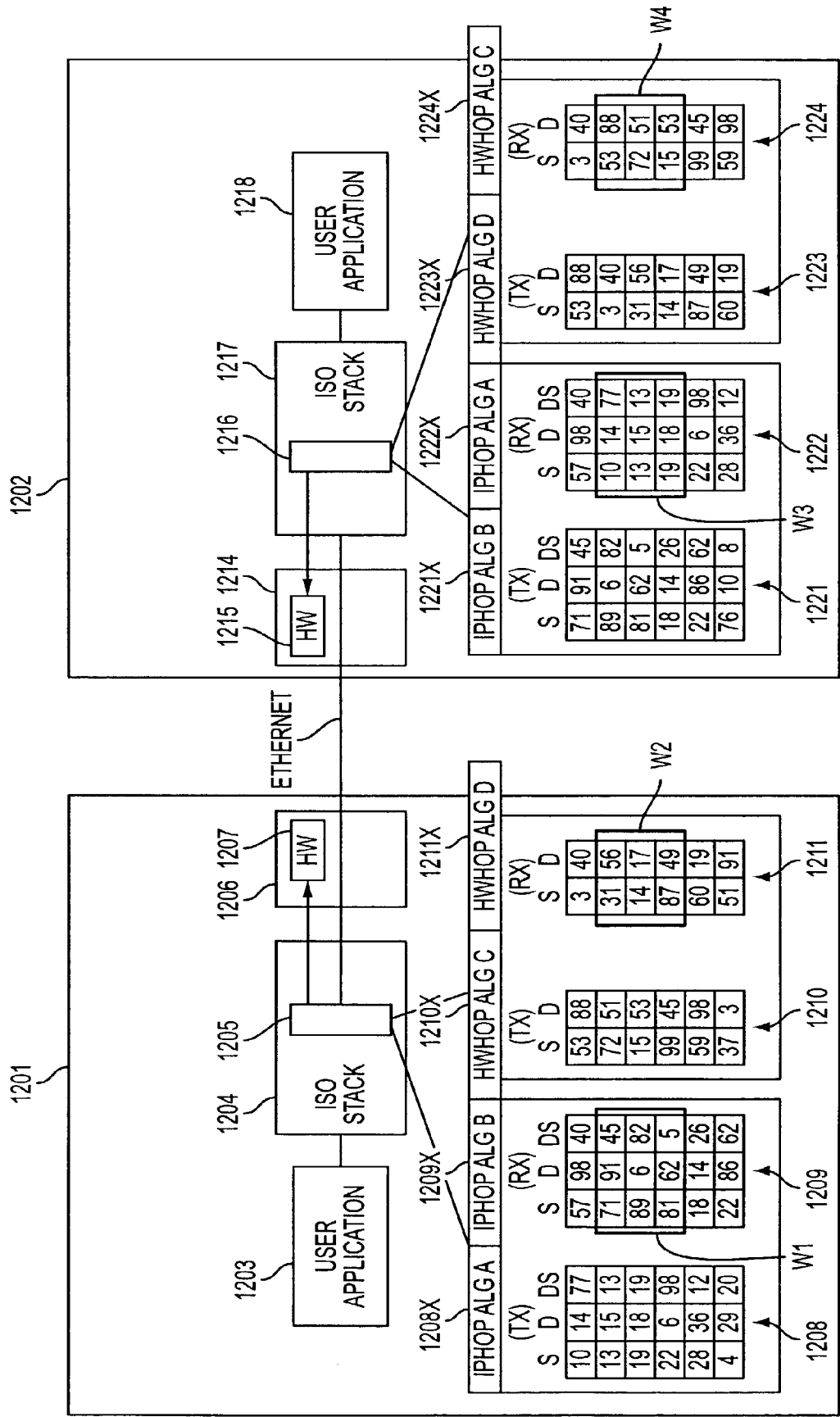


FIG. 12A

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

FIG. 12B

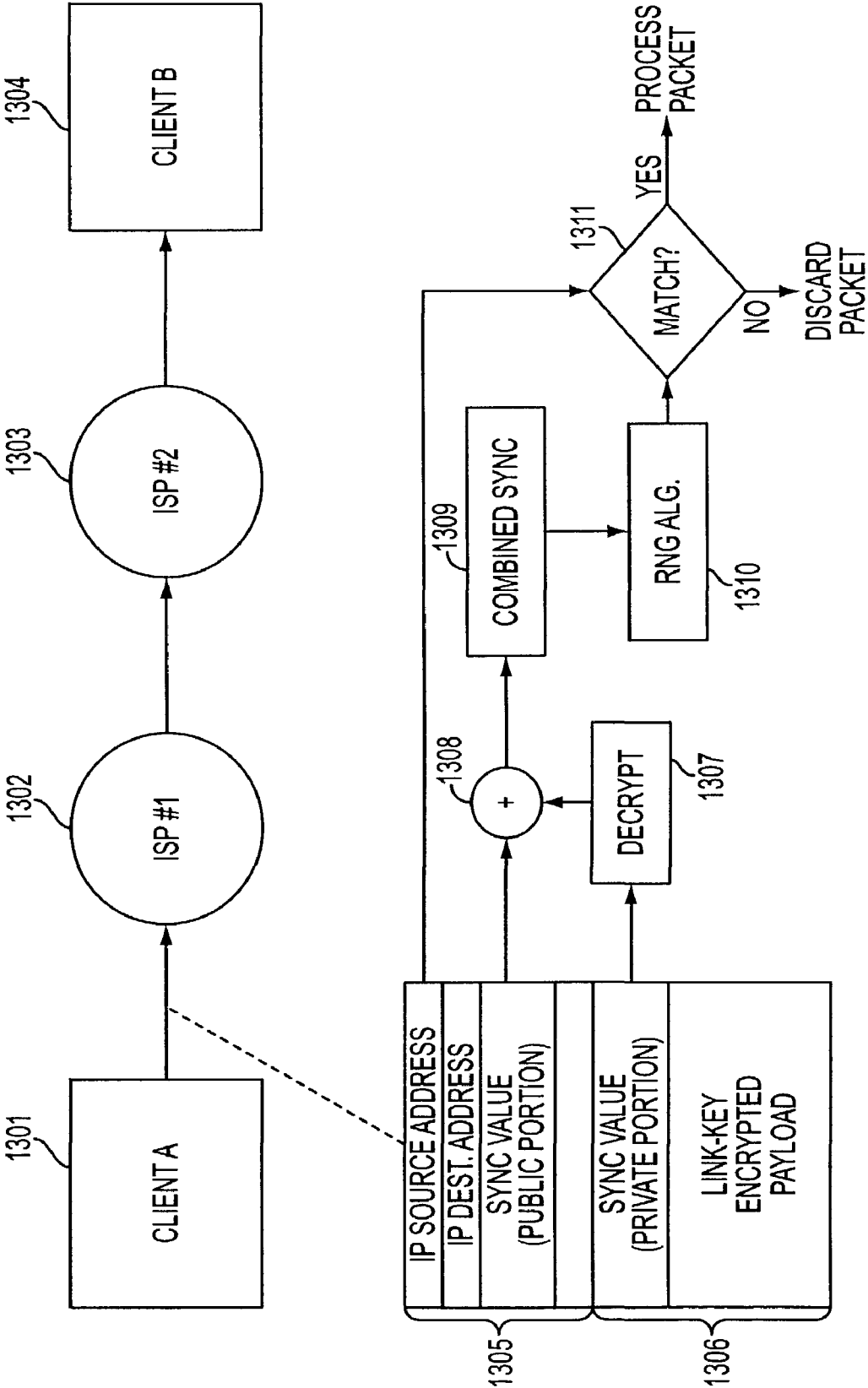


FIG. 13

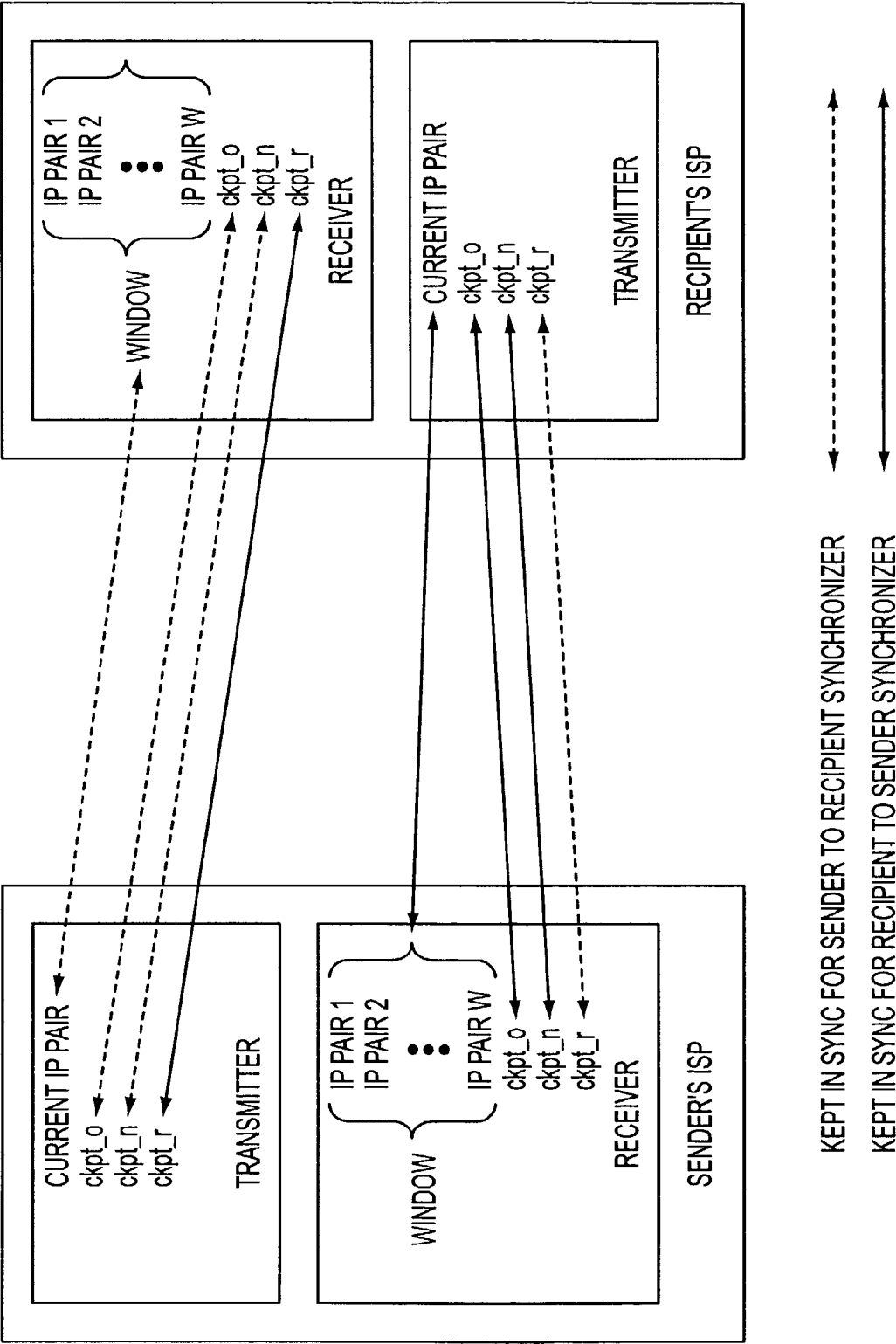


FIG. 14

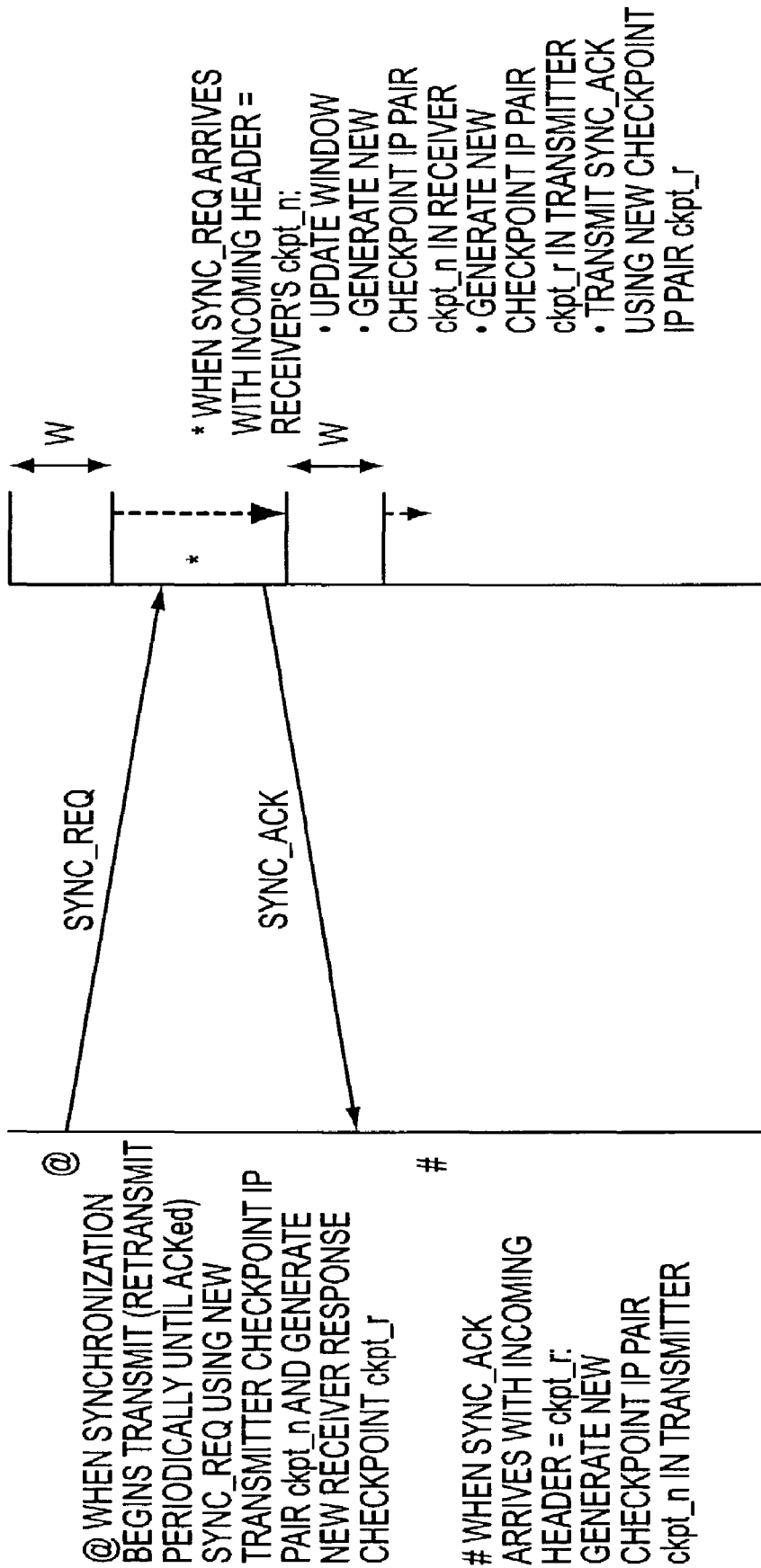


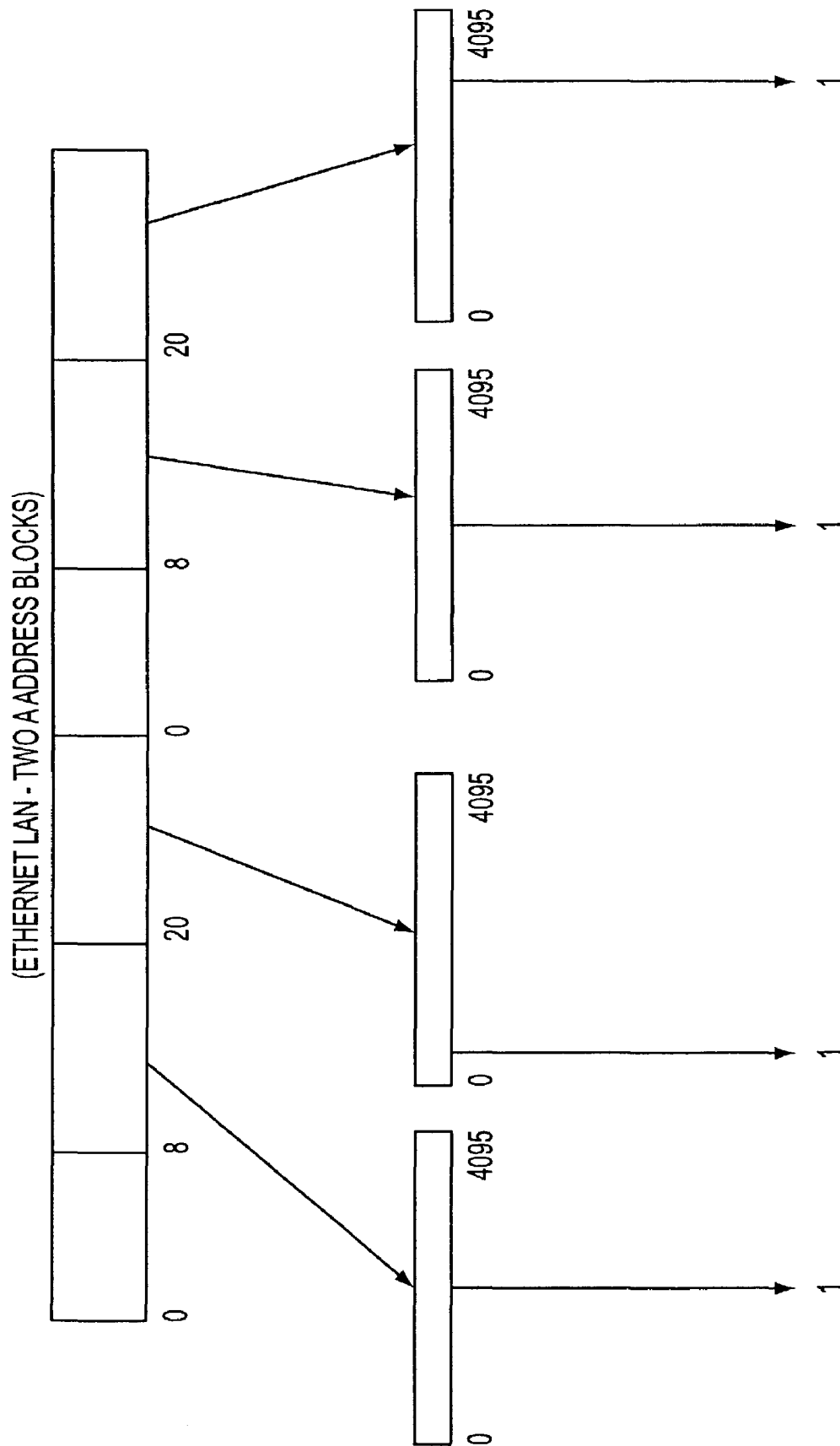
FIG. 15

**U.S. Patent**

**Feb. 10, 2009**

**Sheet 18 of 35**

**US 7,490,151 B2**



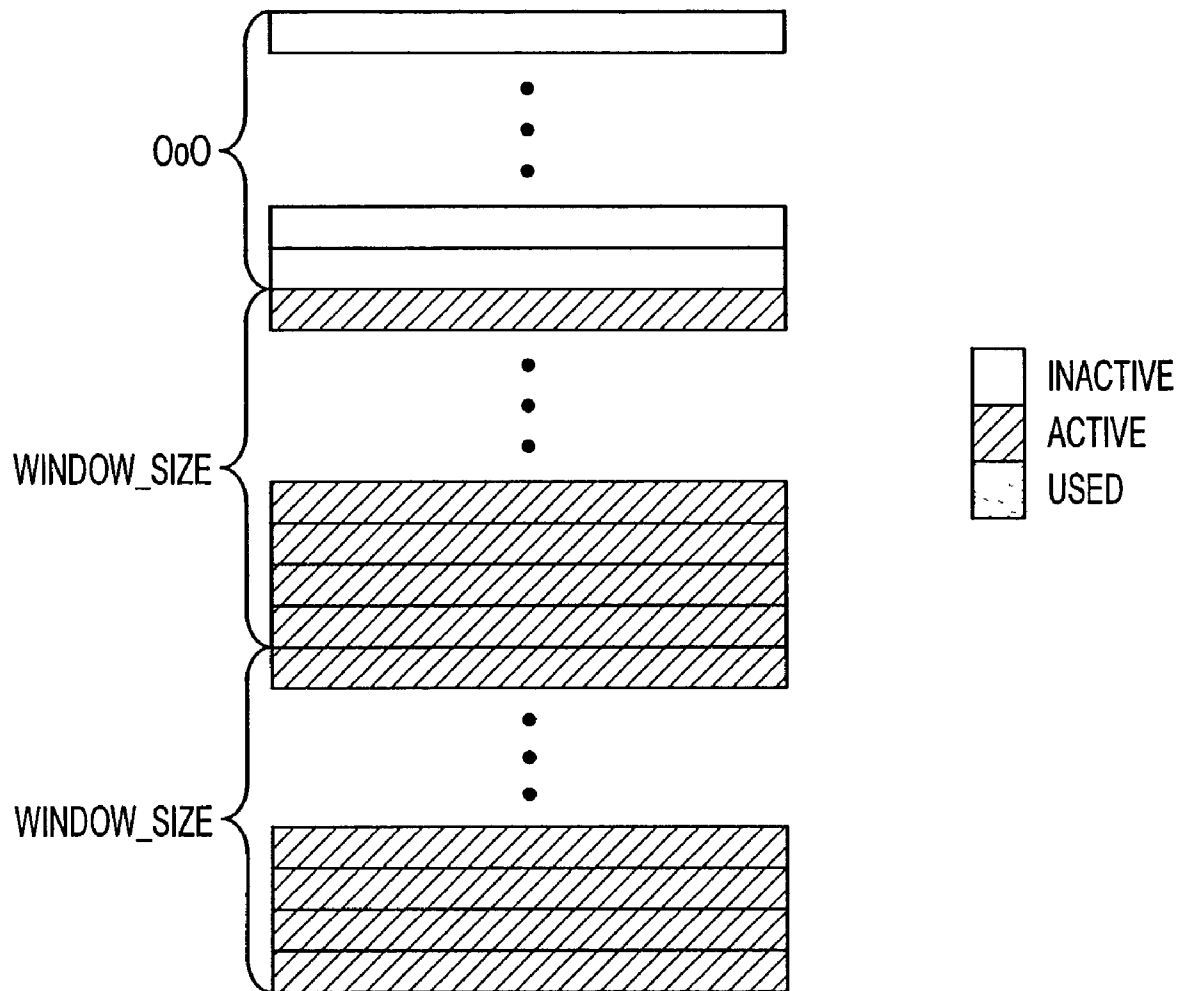
**FIG. 16**

**U.S. Patent**

**Feb. 10, 2009**

**Sheet 19 of 35**

**US 7,490,151 B2**



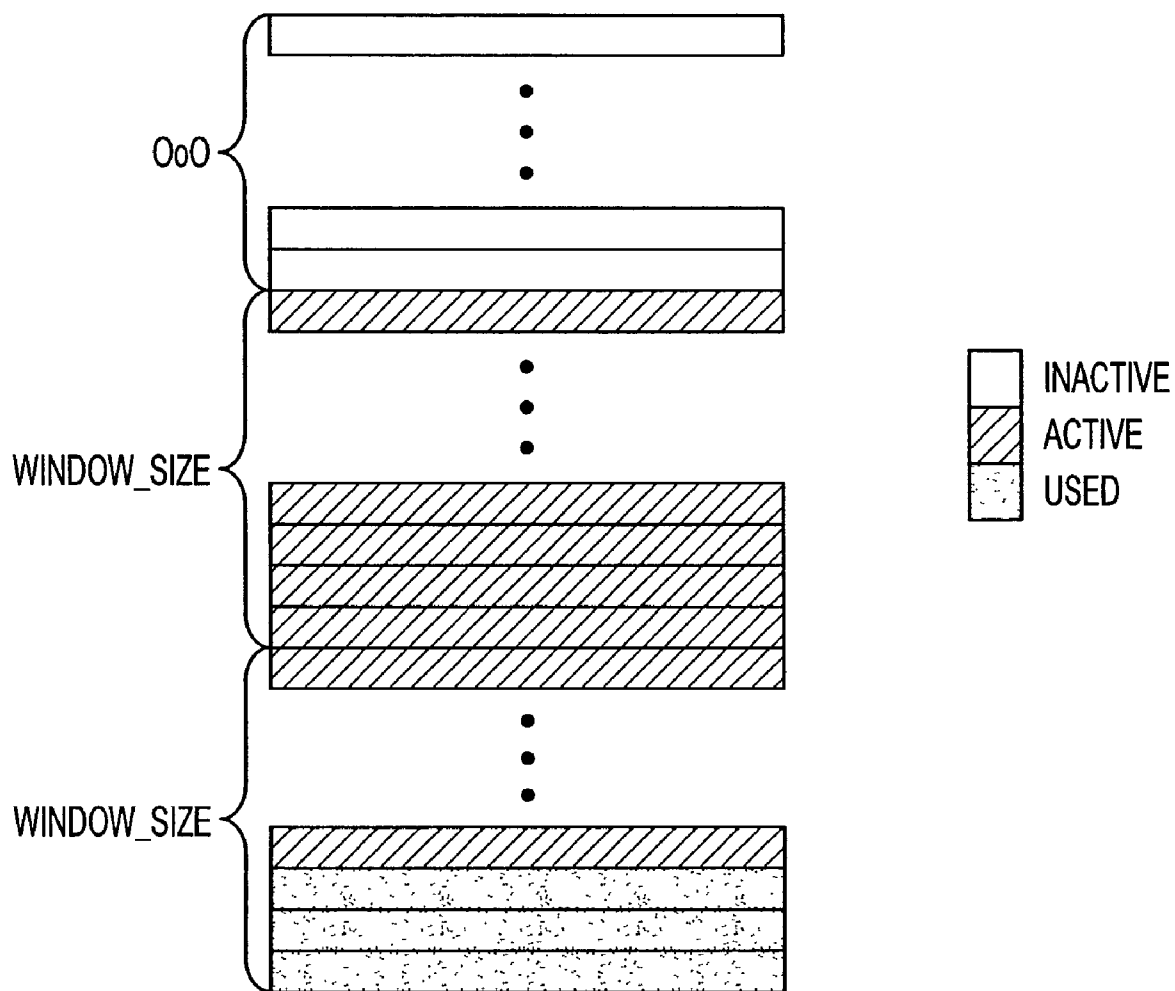
**FIG. 17**

**U.S. Patent**

**Feb. 10, 2009**

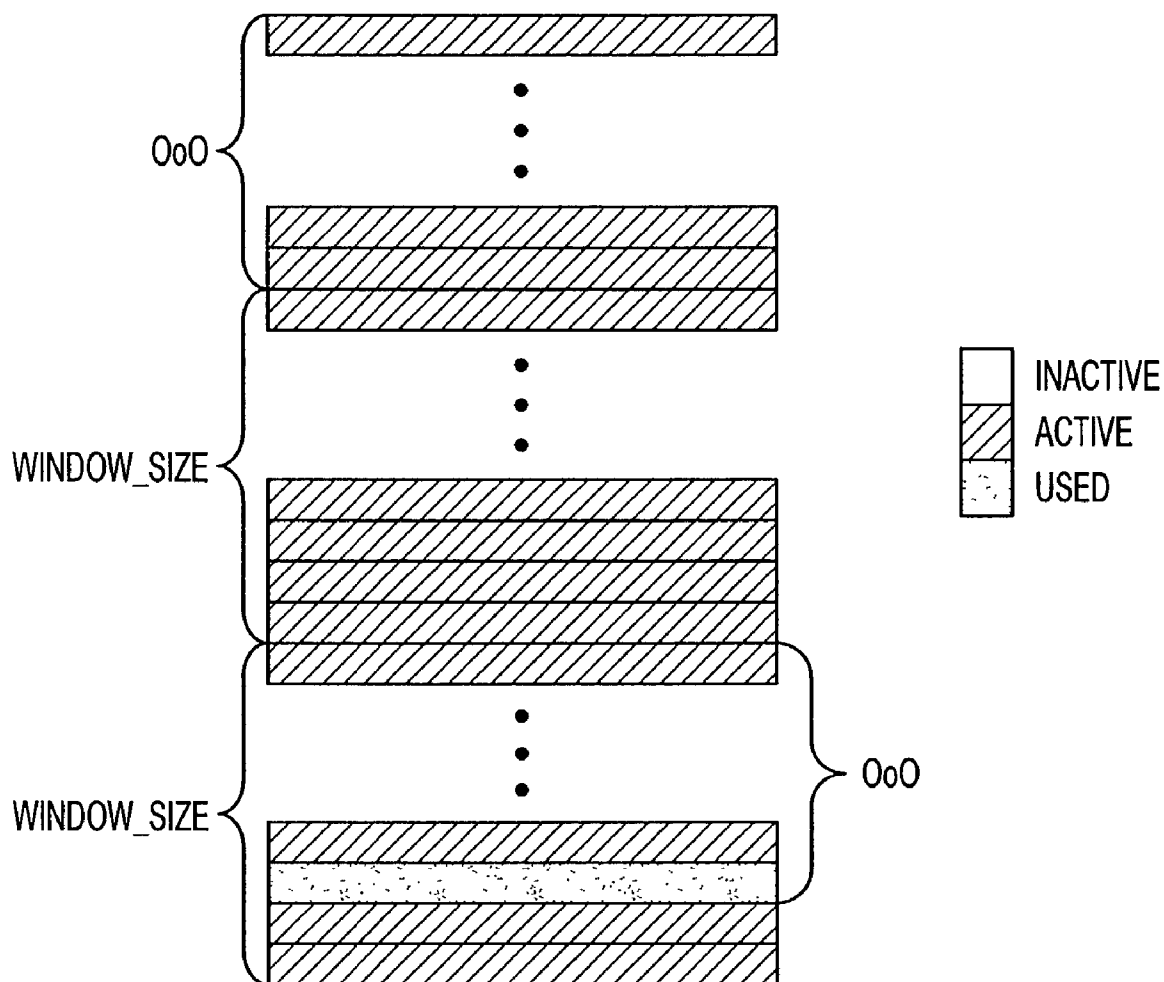
**Sheet 20 of 35**

**US 7,490,151 B2**



**FIG. 18**





**FIG. 19**

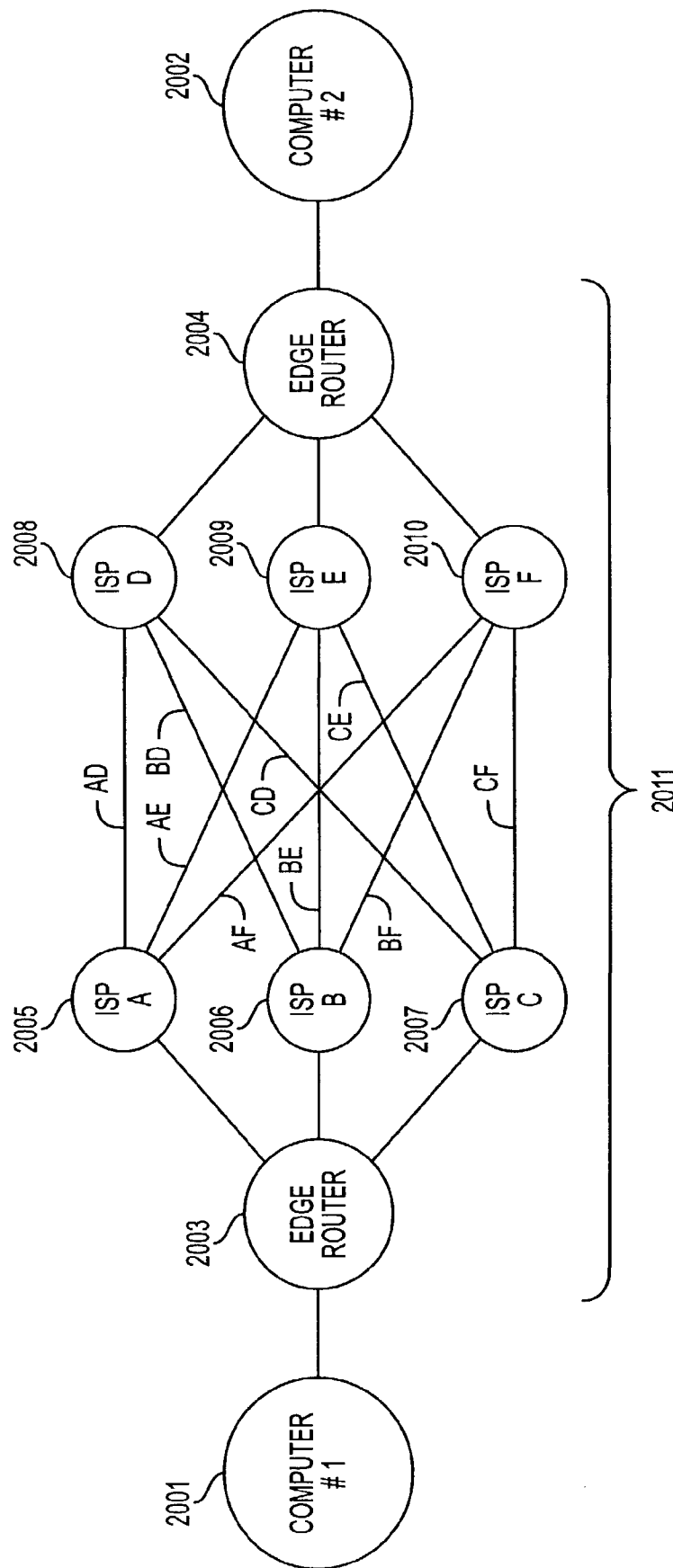
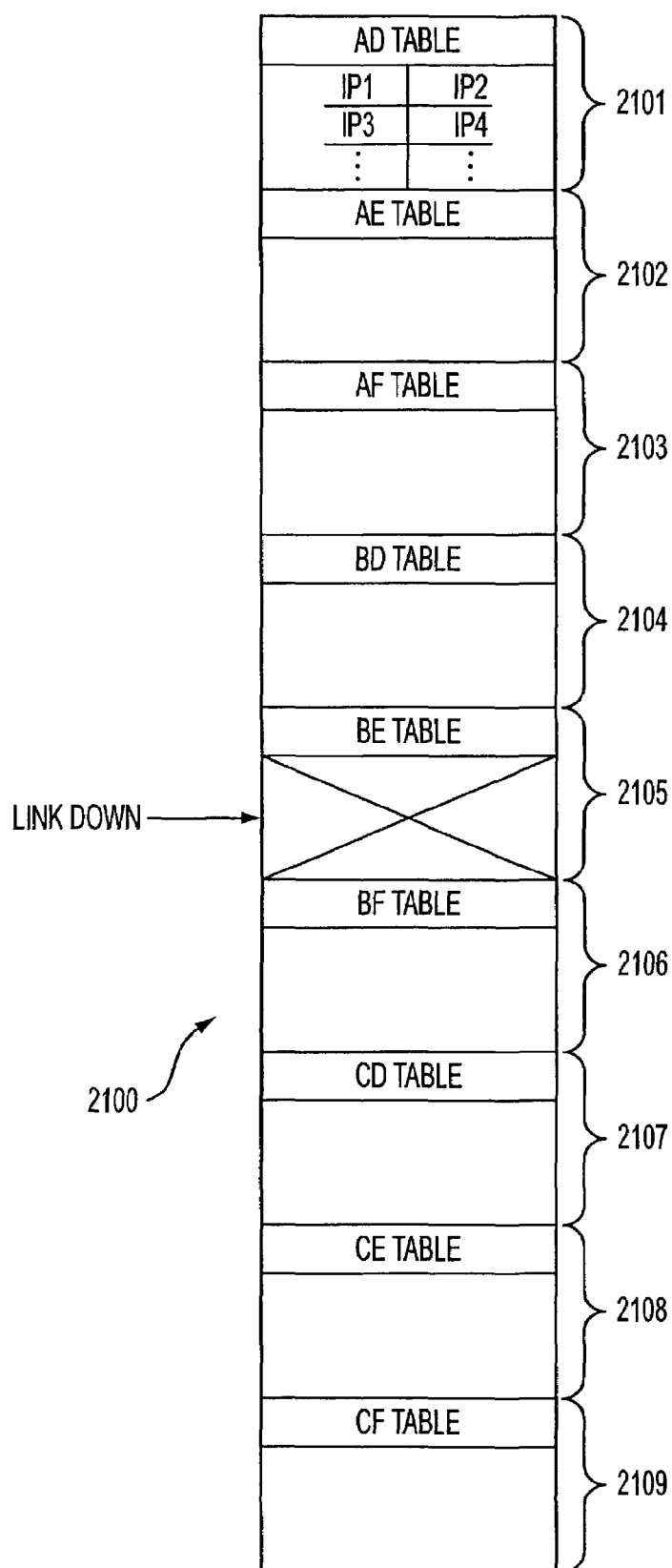


FIG. 20

**U.S. Patent**

Feb. 10, 2009

Sheet 23 of 35

**US 7,490,151 B2****FIG. 21****Appx290**

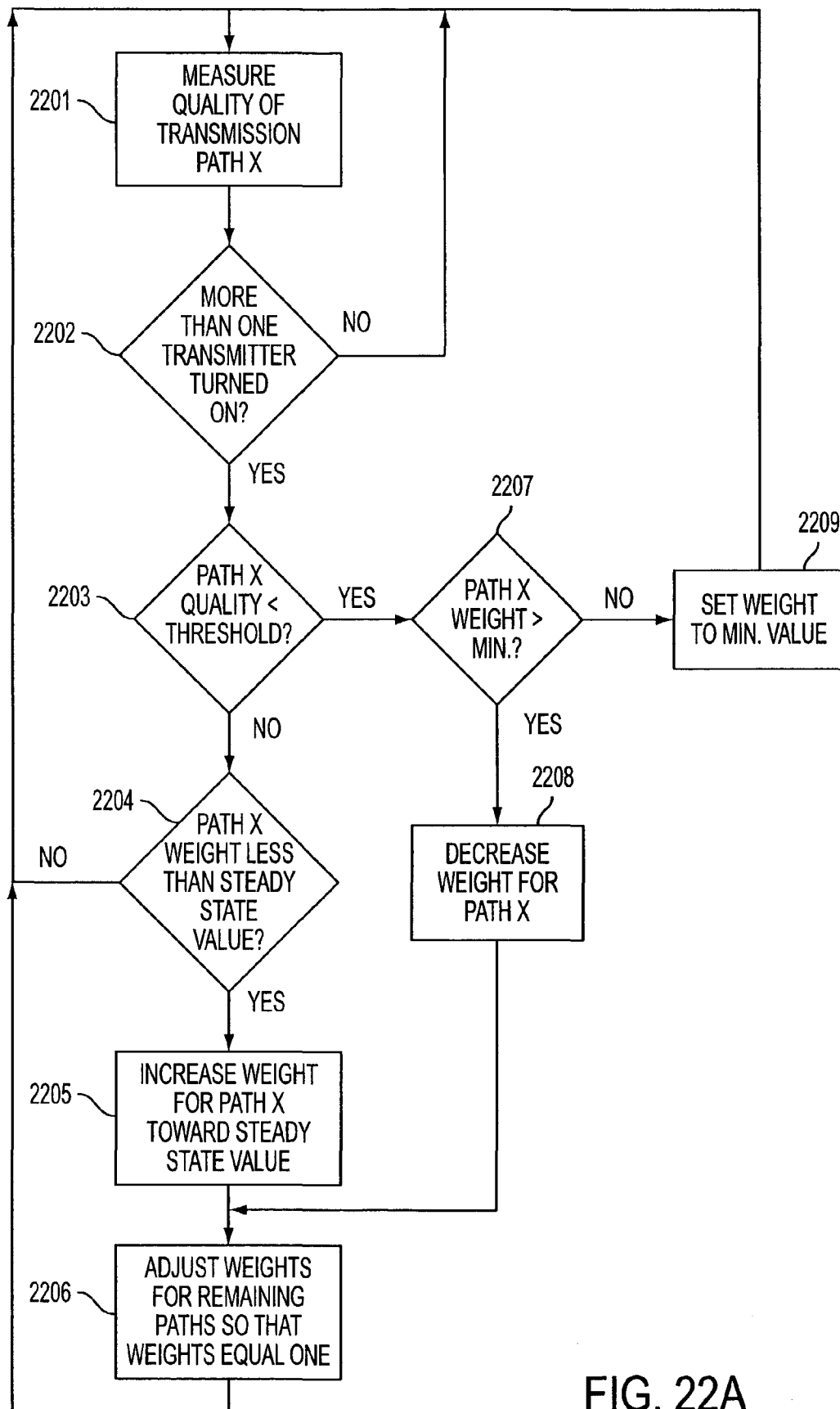


FIG. 22A

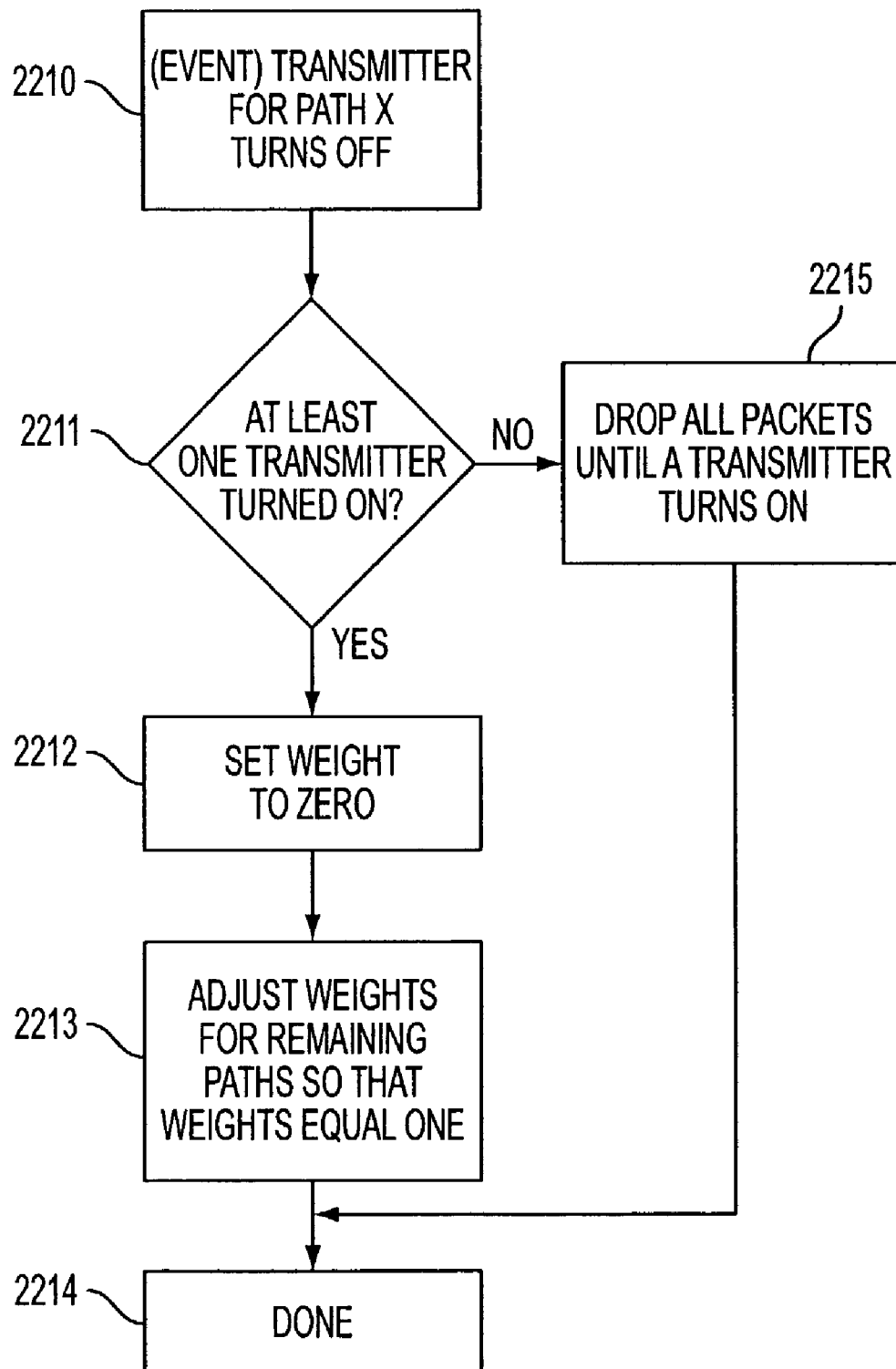


FIG. 22B

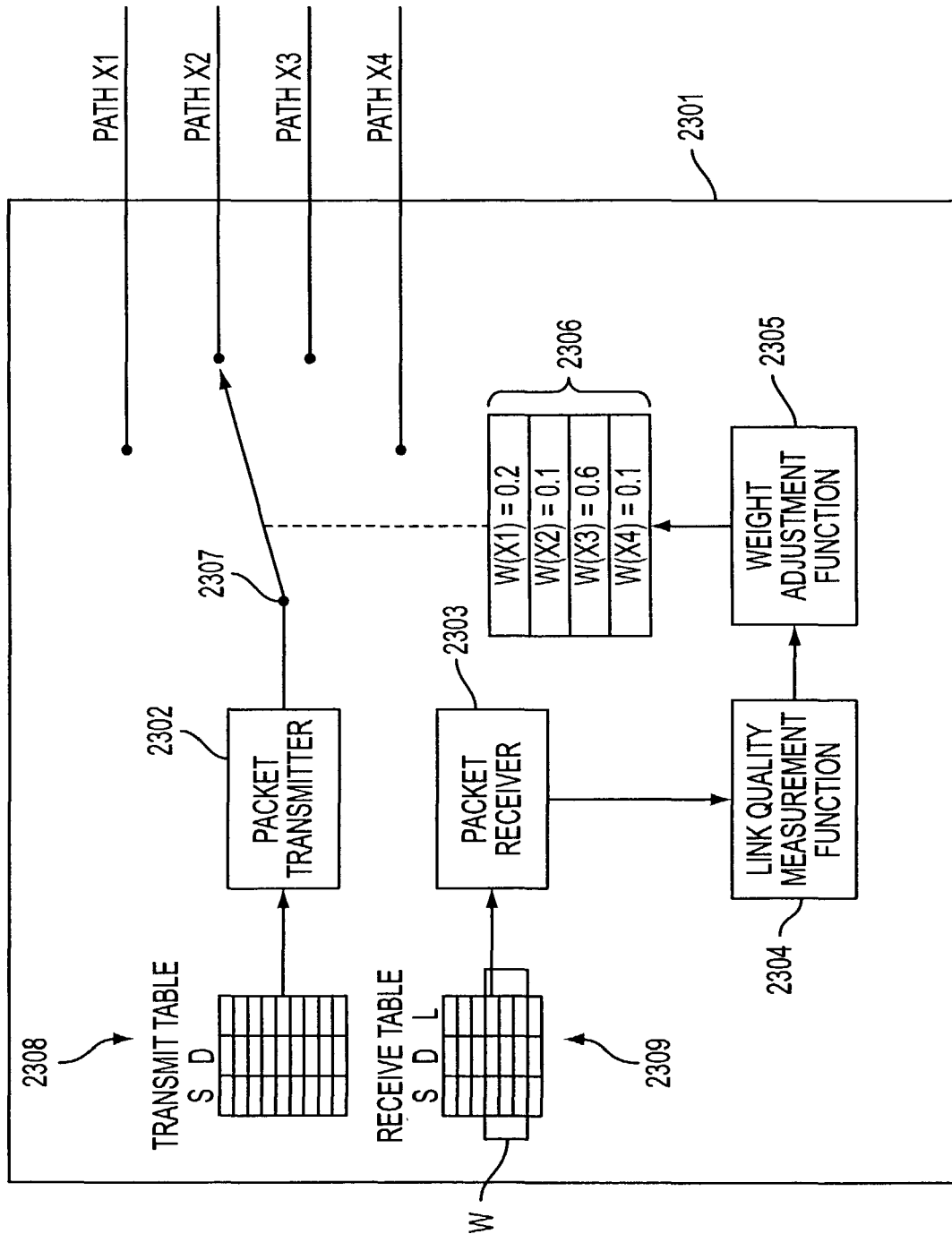


FIG. 23

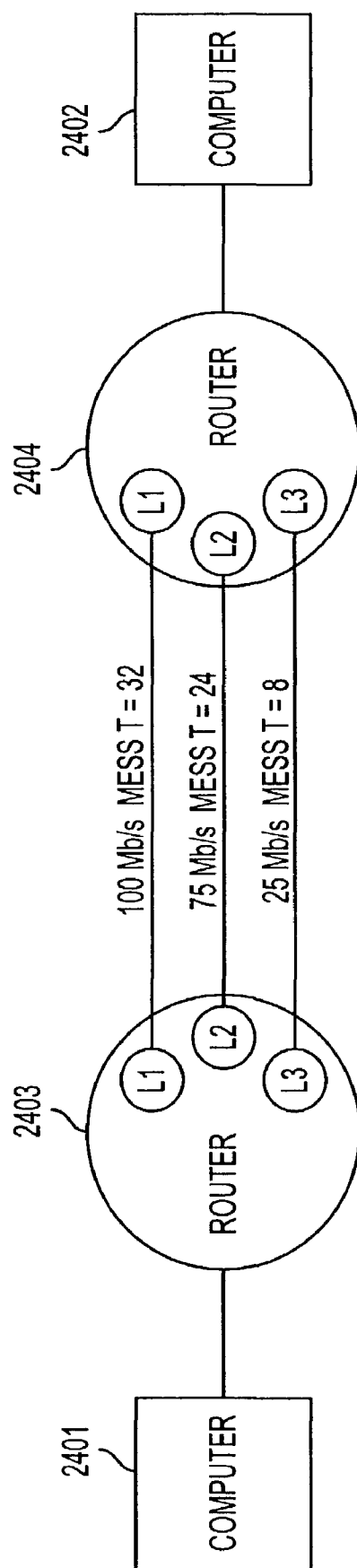


FIG. 24

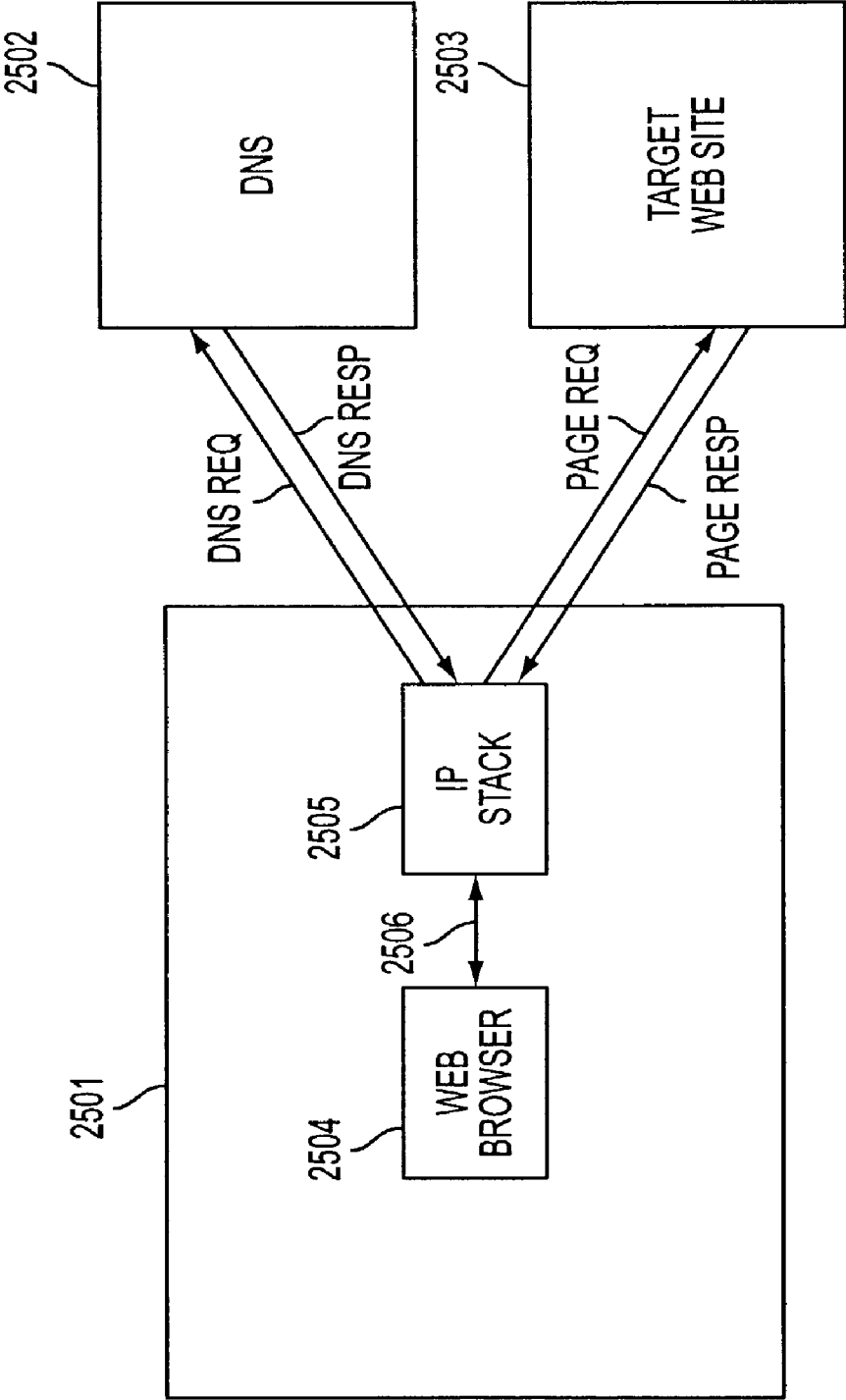


FIG. 25  
(PRIOR ART)



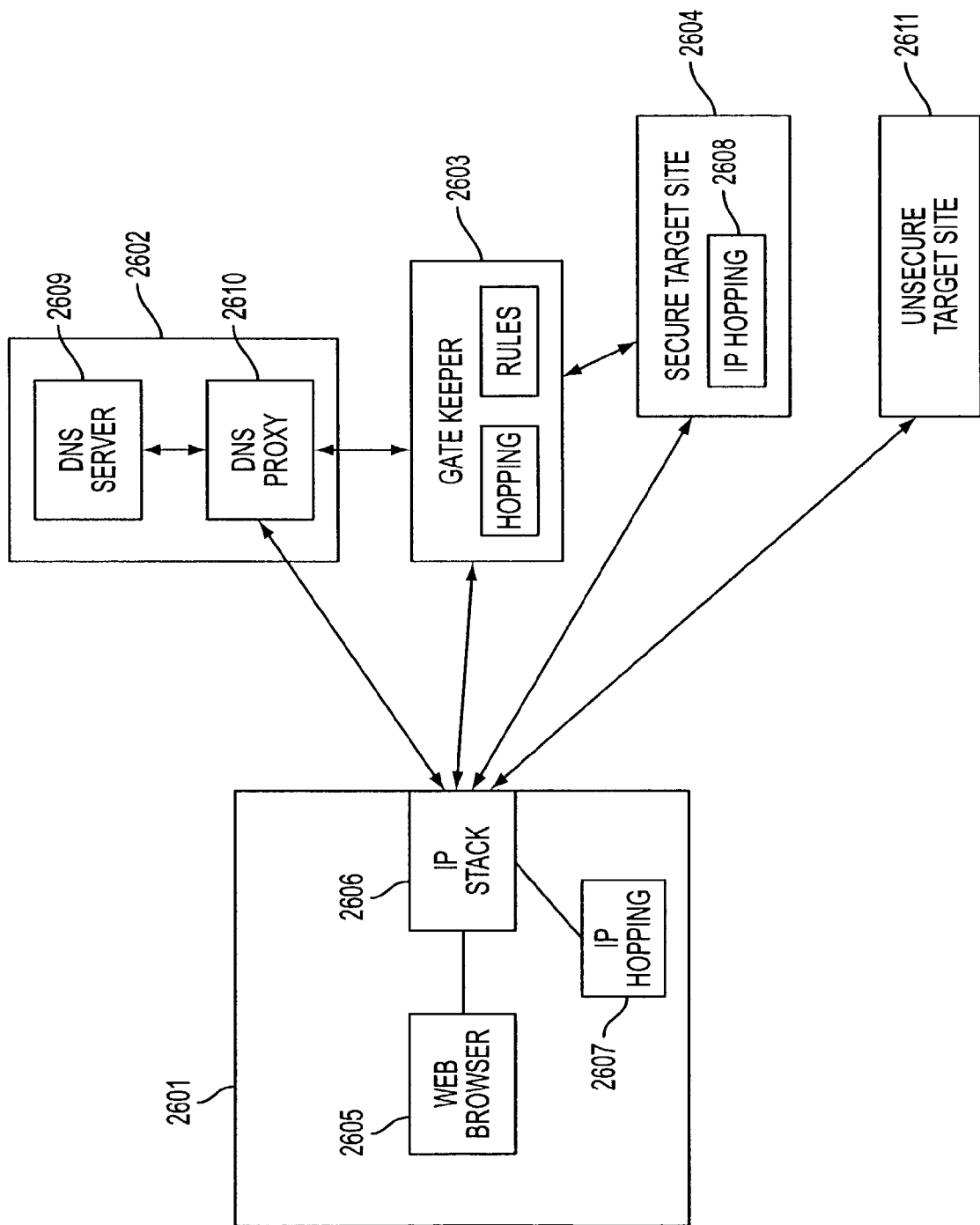


FIG. 26

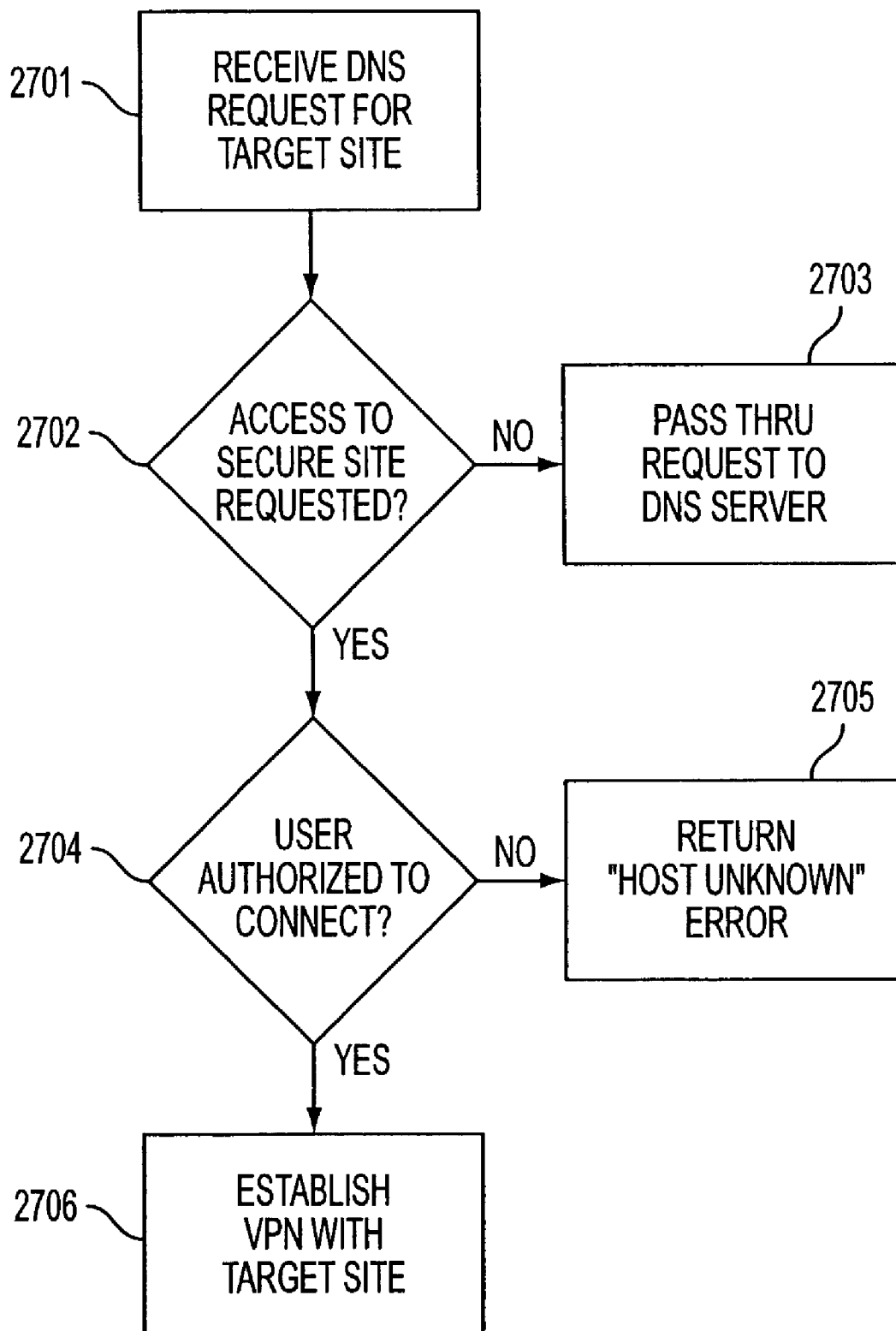


FIG. 27

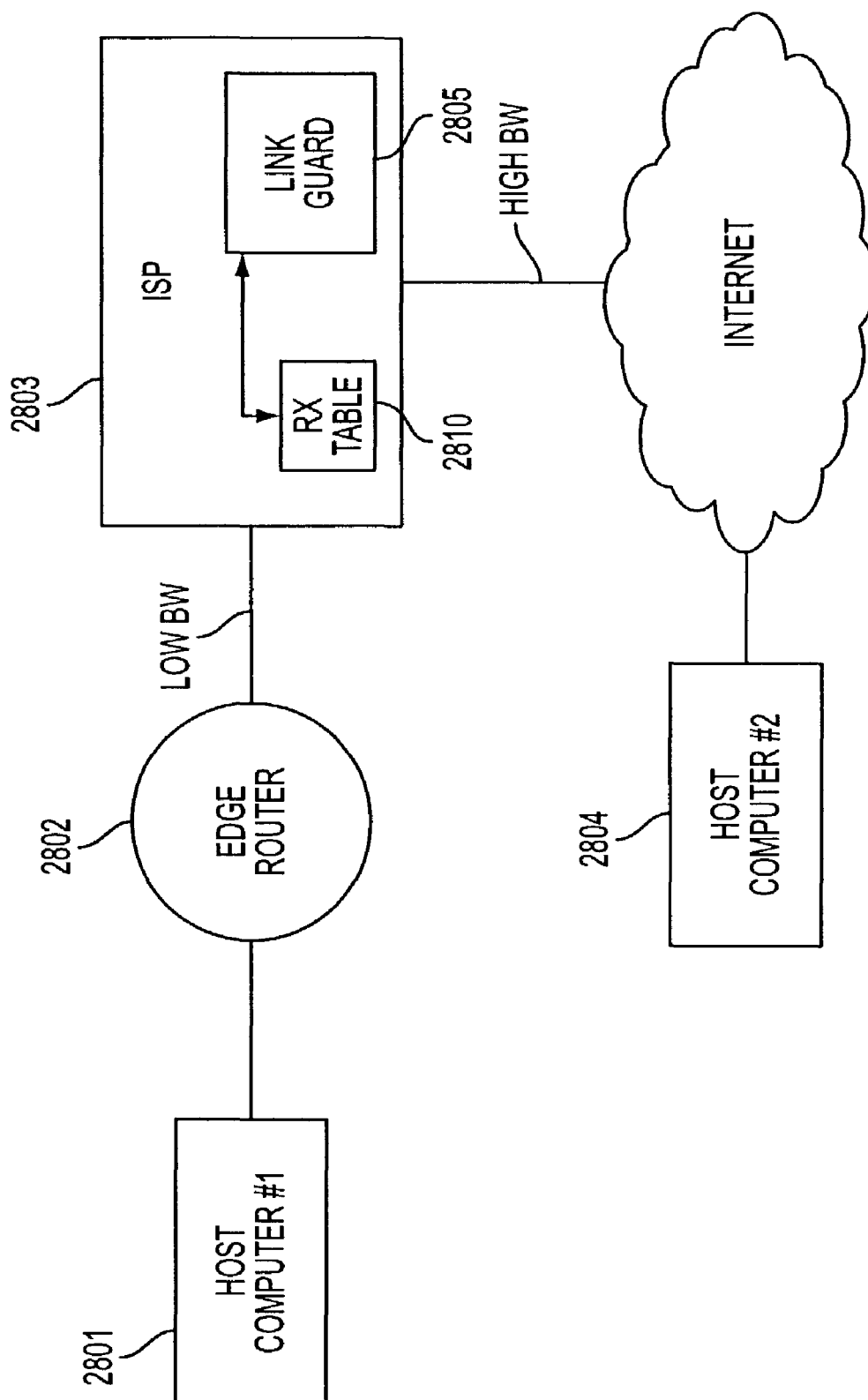


FIG. 28

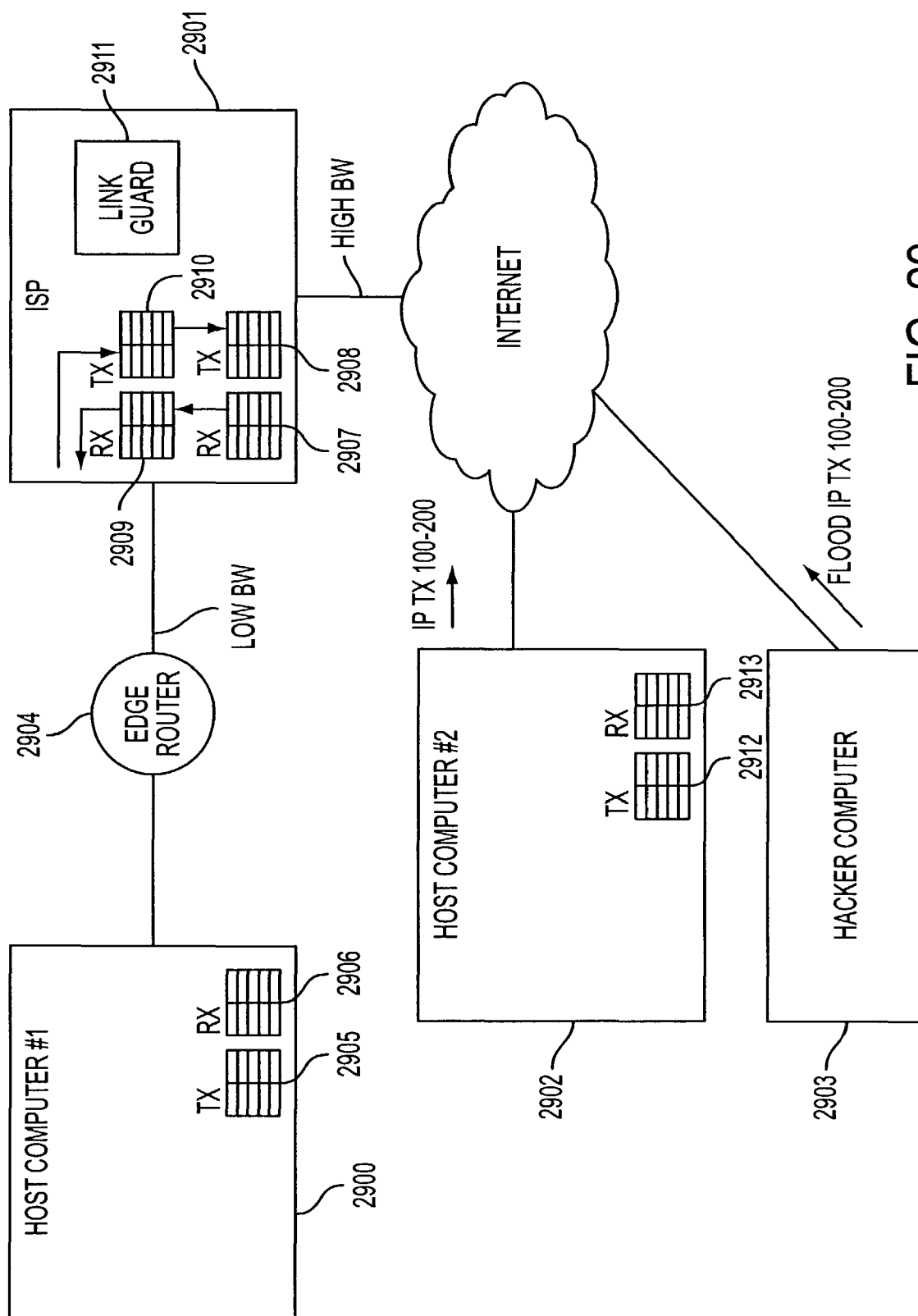


FIG. 29

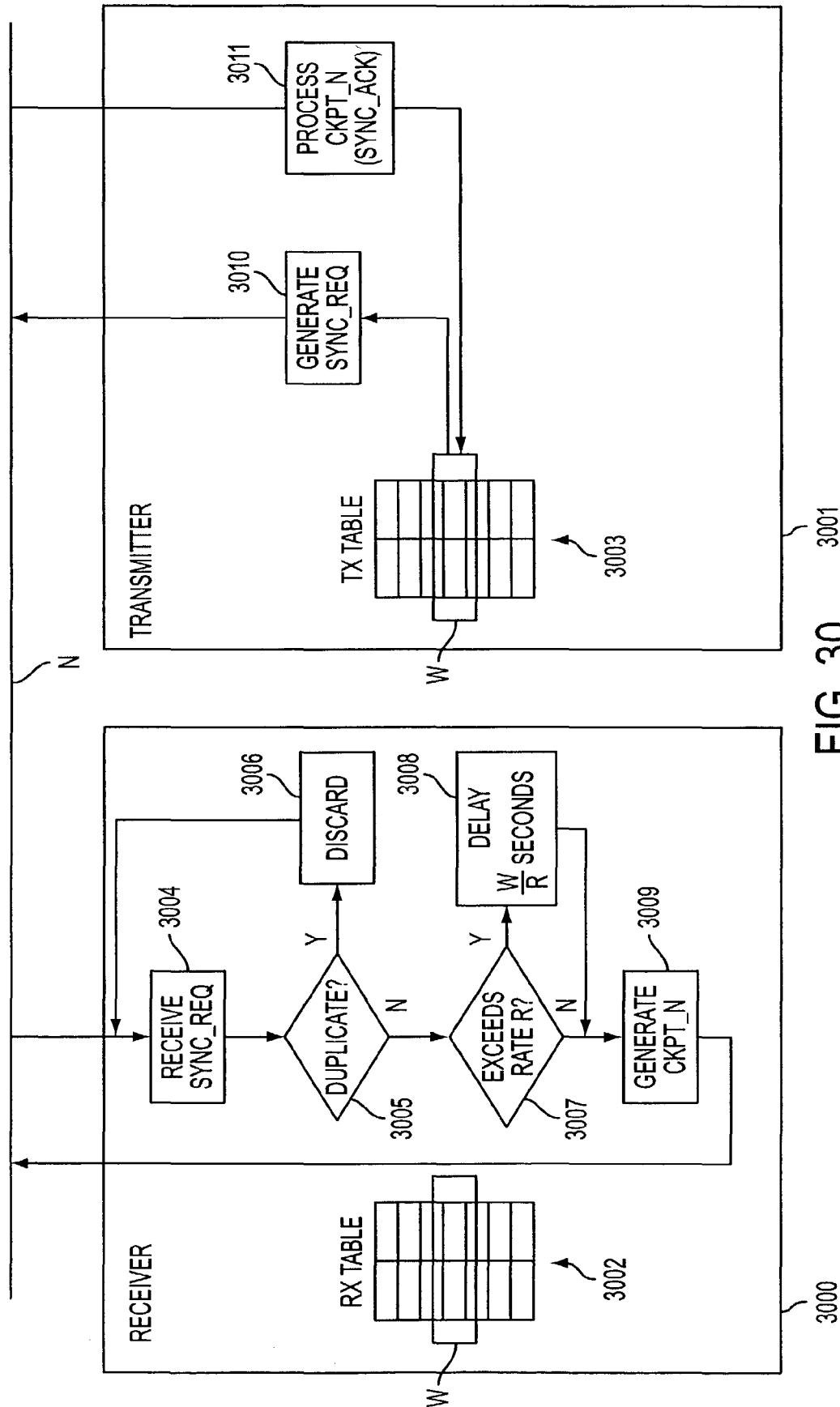


FIG. 30

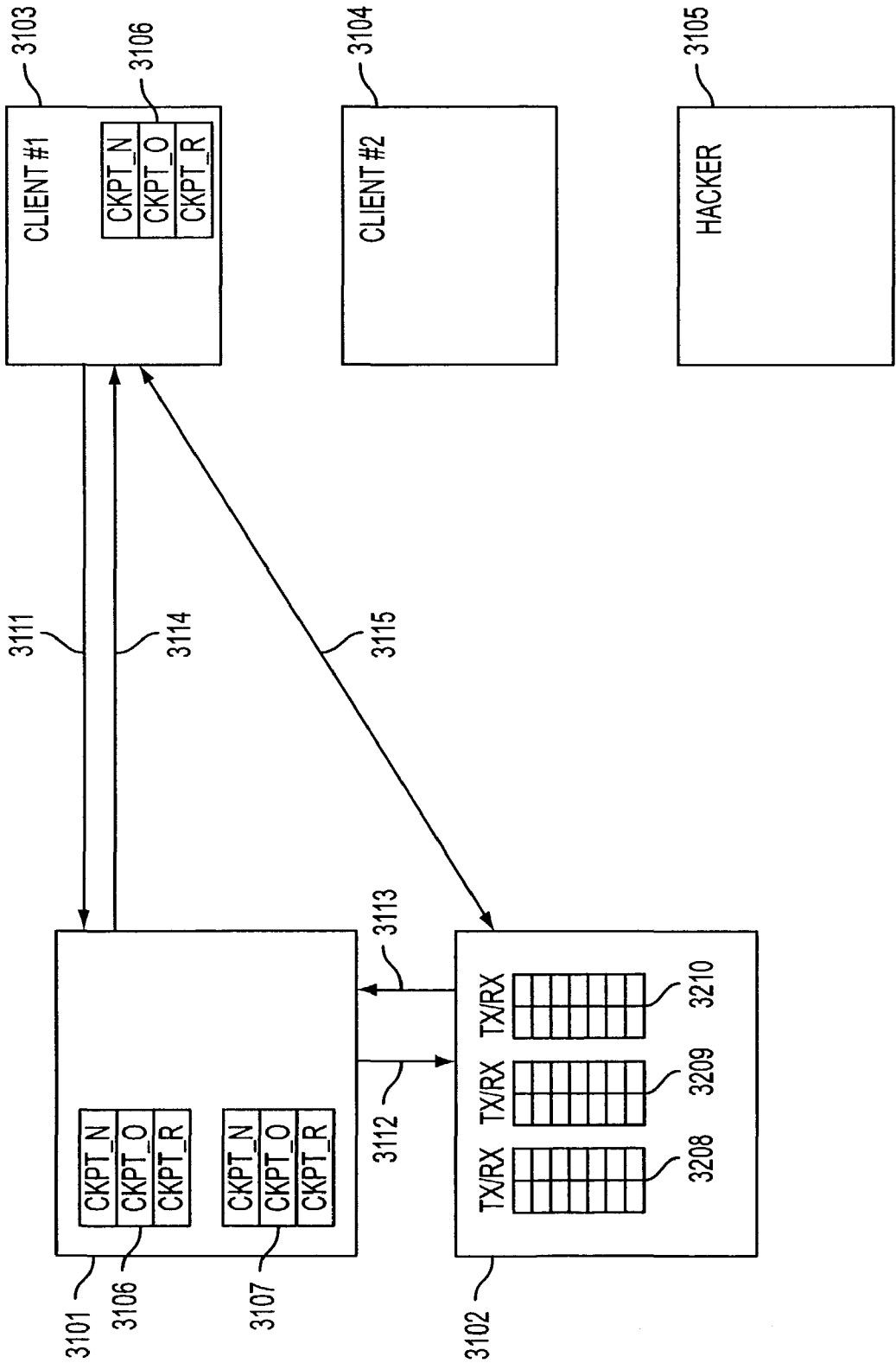
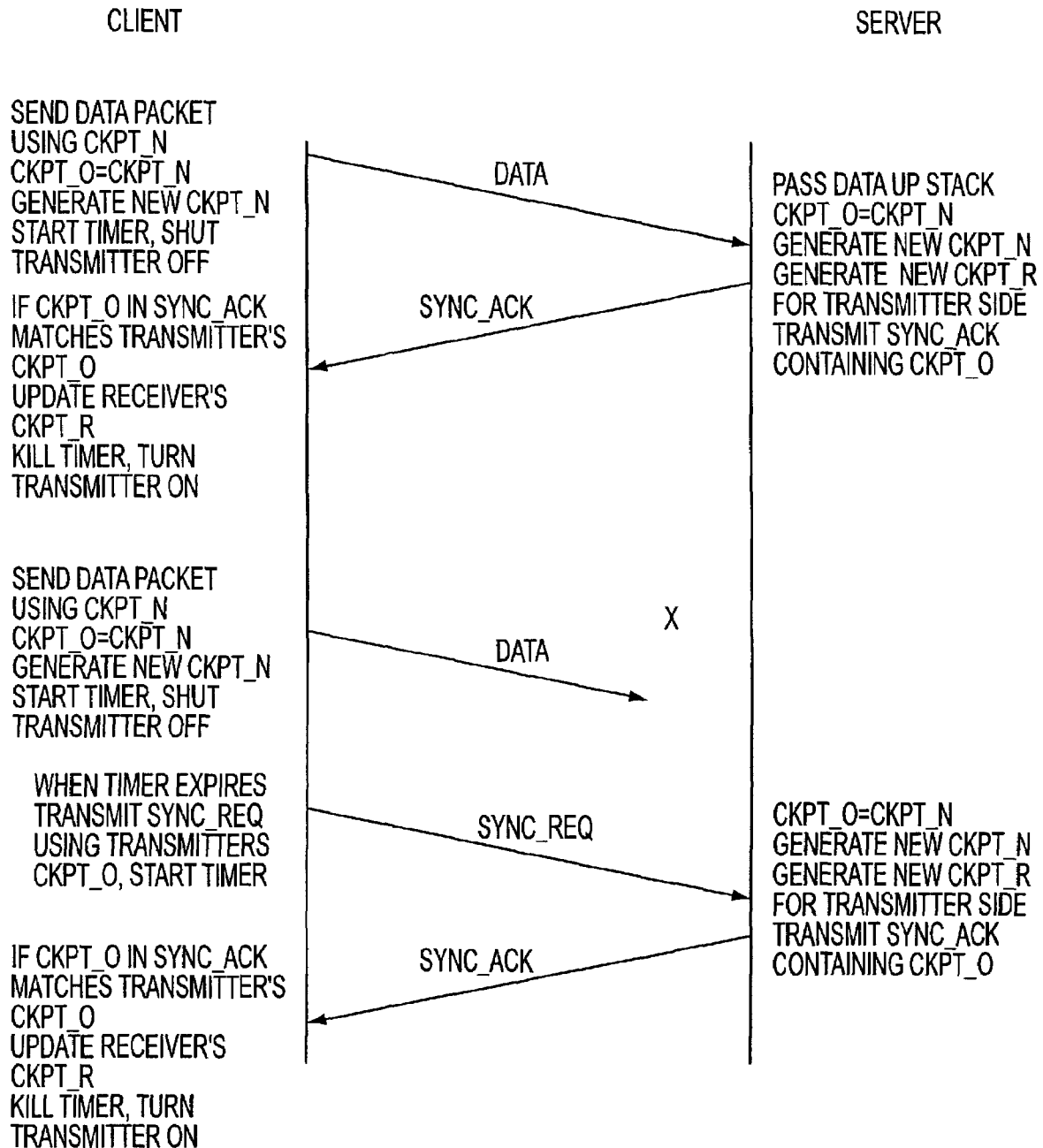


FIG. 31

**U.S. Patent**

Feb. 10, 2009

Sheet 35 of 35

**US 7,490,151 B2****FIG. 32**

US 7,490,151 B2

1

# ESTABLISHMENT OF A SECURE COMMUNICATION LINK BASED ON A DOMAIN NAME SERVICE (DNS) REQUEST

## CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a divisional application of 09/504,783 (filed Feb. 15, 2000), now U.S. Pat. No. 6,502,135, issued Dec. 31, 2002, which claims priority from and is a continuation-in-part of previously filed U.S. application Ser. No. 09/429,643 (filed Oct. 29, 1999) now U.S. Pat. No. 7,010,604. The subject matter of the '643 application, which is bodily incorporated herein, derives from provisional U.S. application No. 60/106,261 (filed Oct. 30, 1998) and 60/137,704 (filed Jun. 7, 1999).

## GOVERNMENT CONTRACT RIGHTS

This invention was made with Government support under Contract No. 360000-1999-000000-QC-000-000 awarded by the Central Intelligence Agency. The Government has certain rights in the invention.

## BACKGROUND OF THE INVENTION

A tremendous variety of methods have been proposed and implemented to provide security and anonymity for communications over the Internet. The variety stems, in part, from the different needs of different Internet users. A basic heuristic framework to aid in discussing these different security techniques is illustrated in FIG. 1. Two terminals, an originating terminal 100 and a destination terminal 110 are in communication over the Internet. It is desired for the communications to be secure, that is, immune to eavesdropping. For example, terminal 100 may transmit secret information to terminal 110 over the Internet 107. Also, it may be desired to prevent an eavesdropper from discovering that terminal 100 is in communication with terminal 110. For example, if terminal 100 is a user and terminal 110 hosts a web site, terminal 100's user may not want anyone in the intervening networks to know what web sites he is "visiting." Anonymity would thus be an issue, for example, for companies that want to keep their market research interests private and thus would prefer to prevent outsiders from knowing which web-sites or other Internet resources they are "visiting." These two security issues may be called data security and anonymity, respectively.

Data security is usually tackled using some form of data encryption. An encryption key 48 is known at both the originating and terminating terminals 100 and 110. The keys may be private and public at the originating and destination terminals 100 and 110, respectively or they may be symmetrical keys (the same key is used by both parties to encrypt and decrypt). Many encryption methods are known and usable in this context.

To hide traffic from a local administrator or ISP, a user can employ a local proxy server in communicating over an encrypted channel with an outside proxy such that the local administrator or ISP only sees the encrypted traffic. Proxy servers prevent destination servers from determining the identities of the originating clients. This system employs an intermediate server interposed between client and destination server. The destination server sees only the Internet Protocol (IP) address of the proxy server and not the originating client. The target server only sees the address of the outside proxy. This scheme relies on a trusted outside proxy server. Also,

2

proxy schemes are vulnerable to traffic analysis methods of determining identities of transmitters and receivers. Another important limitation of proxy servers is that the server knows the identities of both calling and called parties. In many instances, an originating terminal, such as terminal A, would prefer to keep its identity concealed from the proxy, for example, if the proxy server is provided by an Internet service provider (ISP).

To defeat traffic analysis, a scheme called Chaum's mixes employs a proxy server that transmits and receives fixed length messages, including dummy messages. Multiple originating terminals are connected through a mix (a server) to multiple target servers. It is difficult to tell which of the originating terminals are communicating to which of the connected target servers, and the dummy messages confuse eavesdroppers' efforts to detect communicating pairs by analyzing traffic. A drawback is that there is a risk that the mix server could be compromised. One way to deal with this risk is to spread the trust among multiple mixes. If one mix is compromised, the identities of the originating and target terminals may remain concealed. This strategy requires a number of alternative mixes so that the intermediate servers interposed between the originating and target terminals are not determinable except by compromising more than one mix. The strategy wraps the message with multiple layers of encrypted addresses. The first mix in a sequence can decrypt only the outer layer of the message to reveal the next destination mix in sequence. The second mix can decrypt the message to reveal the next mix and so on. The target server receives the message and, optionally, a multi-layer encrypted payload containing return information to send data back in the same fashion. The only way to defeat such a mix scheme is to collude among mixes. If the packets are all fixed-length and intermixed with dummy packets, there is no way to do any kind of traffic analysis.

Still another anonymity technique, called 'crowds,' protects the identity of the originating terminal from the intermediate proxies by providing that originating terminals belong to groups of proxies called crowds. The crowd proxies are interposed between originating and target terminals. Each proxy through which the message is sent is randomly chosen by an upstream proxy. Each intermediate proxy can send the message either to another randomly chosen proxy in the "crowd" or to the destination. Thus, even crowd members cannot determine if a preceding proxy is the originator of the message or if it was simply passed from another proxy.

ZKS (Zero-Knowledge Systems) Anonymous IP Protocol allows users to select up to any of five different pseudonyms, while desktop software encrypts outgoing traffic and wraps it in User Datagram Protocol (UDP) packets. The first server in a 2+-hop system gets the UDP packets, strips off one layer of encryption to add another, then sends the traffic to the next server, which strips off yet another layer of encryption and adds a new one. The user is permitted to control the number of hops. At the final server, traffic is decrypted with an untraceable IP address. The technique is called onion-routing. This method can be defeated using traffic analysis. For a simple example, bursts of packets from a user during low-duty periods can reveal the identities of sender and receiver.

Firewalls attempt to protect LANs from unauthorized access and hostile exploitation or damage to computers connected to the LAN. Firewalls provide a server through which all access to the LAN must pass. Firewalls are centralized systems that require administrative overhead to maintain. They can be compromised by virtual-machine applications ("applets"). They instill a false sense of security that leads to security breaches for example by users sending sensitive



## US 7,490,151 B2

3

information to servers outside the firewall or encouraging use of modems to sidestep the firewall security. Firewalls are not useful for distributed systems such as business travelers, extranets, small teams, etc.

## SUMMARY OF THE INVENTION

A secure mechanism for communicating over the internet, including a protocol referred to as the Tunneled Agile Routing Protocol (TARP), uses a unique two-layer encryption format and special TARP routers. TARP routers are similar in function to regular IP routers. Each TARP router has one or more IP addresses and uses normal IP protocol to send IP packet messages ("packets" or "datagrams"). The IP packets exchanged between TARP terminals via TARP routers are actually encrypted packets whose true destination address is concealed except to TARP routers and servers. The normal or "clear" or "outside" IP header attached to TARP IP packets contains only the address of a next hop router or destination server. That is, instead of indicating a final destination in the destination field of the IP header, the TARP packet's IP header always points to a next-hop in a series of TARP router hops, or to the final destination. This means there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet since the destination could always be next-hop TARP router as well as the final destination.

Each TARP packet's true destination is concealed behind a layer of encryption generated using a link key. The link key is the encryption key used for encrypted communication between the hops intervening between an originating TARP terminal and a destination TARP terminal. Each TARP router can remove the outer layer of encryption to reveal the destination router for each TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal by the sender/receiver IP numbers in the cleartext IP header.

Once the outer layer of encryption is removed, the TARP router determines the final destination. Each TARP packet **140** undergoes a minimum number of hops to help foil traffic analysis. The hops may be chosen at random or by a fixed value. As a result, each TARP packet may make random trips among a number of geographically disparate routers before reaching its destination. Each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined. This feature is called agile routing. The fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. The associated advantages have to do with the inner layer of encryption discussed below. Agile routing is combined with another feature that furthers this purpose; a feature that ensures that any message is broken into multiple packets.

The IP address of a TARP router can be changed, a feature called IP agility. Each TARP router, independently or under direction from another TARP terminal or router, can change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs.

The message payload is hidden behind an inner layer of encryption in the TARP packet that can only be unlocked

4

using a session key. The session key is not available to any of the intervening TARP routers. The session key is used to decrypt the payloads of the TARP packets permitting the data stream to be reconstructed.

Communication may be made private using link and session keys, which in turn may be shared and used according to any desired method. For example, public/private keys or symmetric keys may be used.

To transmit a data stream, a TARP originating terminal constructs a series of TARP packets from a series of IP packets generated by a network (IP) layer process. (Note that the terms "network layer," "data link layer," "application layer," etc. used in this specification correspond to the Open Systems Interconnection (OSI) network terminology.) The payloads of these packets are assembled into a block and chain-block encrypted using the session key. This assumes, of course, that all the IP packets are destined for the same TARP terminal. The block is then interleaved and the interleaved encrypted block is broken into a series of payloads, one for each TARP packet to be generated. Special TARP headers IPT are then added to each payload using the IP headers from the data stream packets. The TARP headers can be identical to normal IP headers or customized in some way. They should contain a formula or data for deinterleaving the data at the destination TARP terminal, a time-to-live (TTL) parameter to indicate the number of hops still to be executed, a data type identifier which indicates whether the payload contains, for example, TCP or UDP data, the sender's TARP address, the destination TARP address, and an indicator as to whether the packet contains real or decoy data or a formula for filtering out decoy data if decoy data is spread in some way through the TARP payload data.

Note that although chain-block encryption is discussed here with reference to the session key, any encryption method may be used. Preferably, as in chain block encryption, a method should be used that makes unauthorized decryption difficult without an entire result of the encryption process. Thus, by separating the encrypted block among multiple packets and making it difficult for an interloper to obtain access to all of such packets, the contents of the communications are provided an extra layer of security.

Decoy or dummy data can be added to a stream to help foil traffic analysis by reducing the peak-to-average network load. It may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to portion, or entirety, of a message, and that portion or entirety then interleaved into a number of separate packets. Considering the agile IP routing of the packets, and the attendant difficulty of reconstructing an entire sequence of packets to form a single block-encrypted message element, decoy packets can significantly increase the difficulty of reconstructing an entire data stream.

The above scheme may be implemented entirely by processes operating between the data link layer and the network

## US 7,490,151 B2

5

layer of each server or terminal participating in the TARP system. Because the encryption system described above is insertable between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the Network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicating that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. In addition, it may create a subprocess that maintains the original IP address and continues interacting with the attacker in some manner.

Decoy packets may be generated by each TARP terminal on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis.

In various other embodiments of the invention, a scalable version of the system may be constructed in which a plurality of IP addresses are preassigned to each pair of communicating nodes in the network. Each pair of nodes agrees upon an algorithm for "hopping" between IP addresses (both sending and receiving), such that an eavesdropper sees apparently continuously random IP address pairs (source and destination) for packets transmitted between the pair. Overlapping or "reusable" IP addresses may be allocated to different users on the same subnet, since each node merely verifies that a particular packet includes a valid source/destination pair from the agreed-upon algorithm. Source/destination pairs are pref-

6

erably not reused between any two nodes during any given end-to-end session, though limited IP block sizes or lengthy sessions might require it.

Further improvements described in this continuation-in-part application include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is an illustration of secure communications over the Internet according to a prior art embodiment.

FIG. 2 is an illustration of secure communications over the Internet according to an embodiment of the invention.

FIG. 3a is an illustration of a process of forming a tunneled IP packet according to an embodiment of the invention.

FIG. 3b is an illustration of a process of forming a tunneled IP packet according to another embodiment of the invention.

FIG. 4 is an illustration of an OSI layer location of processes that may be used to implement the invention.

FIG. 5 is a flow chart illustrating a process for routing a tunneled packet according to an embodiment of the invention.

FIG. 6 is a flow chart illustrating a process for forming a tunneled packet according to an embodiment of the invention.

FIG. 7 is a flow chart illustrating a process for receiving a tunneled packet according to an embodiment of the invention.

FIG. 8 shows how a secure session is established and synchronized between a client and a TARP router.

FIG. 9 shows an IP address hopping scheme between a client computer and TARP router using transmit and receive tables in each computer.

FIG. 10 shows physical link redundancy among three Internet Service Providers (ISPs) and a client computer.

FIG. 11 shows how multiple IP packets can be embedded into a single "frame" such as an Ethernet frame, and further shows the use of a discriminator field to camouflage true packet recipients.

FIG. 12A shows a system that employs hopped hardware addresses, hopped IP addresses, and hopped discriminator fields.

FIG. 12B shows several different approaches for hopping hardware addresses, IP addresses, and discriminator fields in combination.

FIG. 13 shows a technique for automatically re-establishing synchronization between sender and receiver through the use of a partially public sync value.

FIG. 14 shows a "checkpoint" scheme for regaining synchronization between a sender and recipient.

FIG. 15 shows further details of the checkpoint scheme of FIG. 14.

FIG. 16 shows how two addresses can be decomposed into a plurality of segments for comparison with presence vectors.

FIG. 17 shows a storage array for a receiver's active addresses.

FIG. 18 shows the receiver's storage array after receiving a sync request.

FIG. 19 shows the receiver's storage array after new addresses have been generated.

US 7,490,151 B2

7

FIG. 20 shows a system employing distributed transmission paths.

FIG. 21 shows a plurality of link transmission tables that can be used to route packets in the system of FIG. 20.

FIG. 22A shows a flowchart for adjusting weight value distributions associated with a plurality of transmission links.

FIG. 22B shows a flowchart for setting a weight value to zero if a transmitter turns off.

FIG. 23 shows a system employing distributed transmission paths with adjusted weight value distributions for each path.

FIG. 24 shows an example using the system of FIG. 23.

FIG. 25 shows a conventional domain-name look-up service.

FIG. 26 shows a system employing a DNS proxy server with transparent VPN creation.

FIG. 27 shows steps that can be carried out to implement transparent VPN creation based on a DNS look-up function.

FIG. 28 shows a system including a link guard function that prevents packet overloading on a low-bandwidth link LOW BW.

FIG. 29 shows one embodiment of a system employing the principles of FIG. 28.

FIG. 30 shows a system that regulates packet transmission rates by throttling the rate at which synchronizations are performed.

FIG. 31 shows a signaling server 3101 and a transport server 3102 used to establish a VPN with a client computer.

FIG. 32 shows message flows relating to synchronization protocols of FIG. 31.

## DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. 2, a secure mechanism for communicating over the internet employs a number of special routers or servers, called TARP routers 122-127 that are similar to regular IP routers 128-132 in that each has one or more IP addresses and uses normal IP protocol to send normal-looking IP packet messages, called TARP packets 140. TARP packets 140 are identical to normal IP packet messages that are routed by regular IP routers 128-132 because each TARP packet 140 contains a destination address as in a normal IP packet. However, instead of indicating a final destination in the destination field of the IP header, the TARP packet's 140 IP header always points to a next-hop in a series of TARP router hops, or the final destination, TARP terminal 110. Because the header of the TARP packet contains only the next-hop destination, there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet 140 since the destination could always be the next-hop TARP router as well as the final destination, TARP terminal 110.

Each TARP packet's true destination is concealed behind an outer layer of encryption generated using a link key 146. The link key 146 is the encryption key used for encrypted communication between the end points (TARP terminals or TARP routers) of a single link in the chain of hops connecting the originating TARP terminal 100 and the destination TARP terminal 110. Each TARP router 122-127, using the link key 146 it uses to communicate with the previous hop in a chain, can use the link key to reveal the true destination of a TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal (which may indicate the link key used) by the sender field of the clear IP header. Alternatively, this identity may be hidden behind another layer of encryption in available bits in the

8

clear IP header. Each TARP router, upon receiving a TARP message, determines if the message is a TARP message by using authentication data in the TARP packet. This could be recorded in available bytes in the TARP packet's IP header. Alternatively, TARP packets could be authenticated by attempting to decrypt using the link key 146 and determining if the results are as expected. The former may have computational advantages because it does not involve a decryption process.

Once the outer layer of decryption is completed by a TARP router 122-127, the TARP router determines the final destination. The system is preferably designed to cause each TARP packet 140 to undergo a minimum number of hops to help foil traffic analysis. The time to live counter in the IP header of the TARP message may be used to indicate a number of TARP router hops yet to be completed. Each TARP router then would decrement the counter and determine from that whether it should forward the TARP packet 140 to another TARP router 122-127 or to the destination TARP terminal 110. If the time to live counter is zero or below zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to the destination TARP terminal 110. If the time to live counter is above zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to a TARP router 122-127 that the current TARP terminal chooses at random. As a result, each TARP packet 140 is routed through some minimum number of hops of TARP routers 122-127 which are chosen at random.

Thus, each TARP packet, irrespective of the traditional factors determining traffic in the Internet, makes random trips among a number of geographically disparate routers before reaching its destination and each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined as described above. This feature is called agile routing. For reasons that will become clear shortly, the fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. Agile routing is combined with another feature that furthers this purpose, a feature that ensures that any message is broken into multiple packets.

A TARP router receives a TARP packet when an IP address used by the TARP router coincides with the IP address in the TARP packet's IP header IP<sub>C</sub>. The IP address of a TARP router, however, may not remain constant. To avoid and manage attacks, each TARP router, independently or under direction from another TARP terminal or router, may change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs. In reality, whenever a TARP router looks up the address of a destination in the encrypted header, it must convert a TARP address to a real IP address using its LUT.

While every TARP router receiving a TARP packet has the ability to determine the packet's final destination, the message payload is embedded behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the TARP routers 122-127 intervening between the originating 100 and destination 110 TARP terminals. The session key is used to

## US 7,490,151 B2

9

decrypt the payloads of the TARP packets **140** permitting an entire message to be reconstructed.

In one embodiment, communication may be made private using link and session keys, which in turn may be shared and used according any desired method. For example, a public key or symmetric keys may be communicated between link or session endpoints using a public key method. Any of a variety of other mechanisms for securing data to ensure that only authorized computers can have access to the private information in the TARP packets **140** may be used as desired.

Referring to FIG. **3a**, to construct a series of TARP packets, a data stream **300** of IP packets **207a**, **207b**, **207c**, etc., such series of packets being formed by a network (IP) layer process, is broken into a series of small sized segments. In the present example, equal-sized segments **1-9** are defined and used to construct a set of interleaved data packets A, B, and C. Here it is assumed that the number of interleaved packets A, B, and C formed is three and that the number of IP packets **207a-207c** used to form the three interleaved packets A, B, and C is exactly three. Of course, the number of IP packets spread over a group of interleaved packets may be any convenient number as may be the number of interleaved packets over which the incoming data stream is spread. The latter, the number of interleaved packets over which the data stream is spread, is called the interleave window.

To create a packet, the transmitting software interleaves the normal IP packets **207a** et. seq. to form a new set of interleaved payload data **320**. This payload data **320** is then encrypted using a session key to form a set of session-key-encrypted payload data **330**, each of which, A, B, and C, will form the payload of a TARP packet. Using the IP header data, from the original packets **207a-207c**, new TARP headers IPT are formed. The TARP headers IPT can be identical to normal IP headers or customized in some way. In a preferred embodiment, the TARP headers IPT are IP headers with added data providing the following information required for routing and reconstruction of messages, some of which data is ordinarily, or capable of being, contained in normal IP headers:

1. A window sequence number—an identifier that indicates where the packet belongs in the original message sequence.

2. An interleave sequence number—an identifier that indicates the interleaving sequence used to form the packet so that the packet can be deinterleaved along with other packets in the interleave window.

3. A time-to-live (TTL) datum—indicates the number of TARP-router-hops to be executed before the packet reaches its destination. Note that the TTL parameter may provide a datum to be used in a probabilistic formula for determining whether to route the packet to the destination or to another hop.

4. Data type identifier—indicates whether the payload contains, for example, TCP or UDP data.

5. Sender's address—indicates the sender's address in the TARP network.

6. Destination address—indicates the destination terminal's address in the TARP network.

7. Decoy/Real—an indicator of whether the packet contains real message data or dummy decoy data or a combination.

Obviously, the packets going into a single interleave window must include only packets with a common destination. Thus, it is assumed in the depicted example that the IP headers of IP packets **207a-207c** all contain the same destination address or at least will be received by the same terminal so that they can be deinterleaved. Note that dummy or decoy data or packets can be added to form a larger interleave window than would otherwise be required by the size of a

10

given message. Decoy or dummy data can be added to a stream to help foil traffic analysis by leveling the load on the network. Thus, it may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or the entirety, of a message, and that portion or entirety then interleaved into a number of separate packets.

Referring to FIG. **3b**, in an alternative mode of TARP packet construction, a series of IP packets are accumulated to make up a predefined interleave window. The payloads of the packets are used to construct a single block **520** for chain block encryption using the session key. The payloads used to form the block are presumed to be destined for the same terminal. The block size may coincide with the interleave window as depicted in the example embodiment of FIG. **3b**. After encryption, the encrypted block is broken into separate payloads and segments which are interleaved as in the embodiment of FIG. **3a**. The resulting interleaved packets A, B, and C, are then packaged as TARP packets with TARP headers as in the Example of FIG. **3a**. The remaining process is as shown in, and discussed with reference to, FIG. **3a**.

Once the TARP packets **340** are formed, each entire TARP packet **340**, including the TARP header  $IP_T$ , is encrypted using the link key for communication with the first-hop-TARP router. The first hop TARP router is randomly chosen. A final unencrypted IP header  $IP_C$  is added to each encrypted TARP packet **340** to form a normal IP packet **360** that can be transmitted to a TARP router. Note that the process of constructing the TARP packet **360** does not have to be done in stages as described. The above description is just a useful heuristic for describing the final product, namely, the TARP packet.

Note that, TARP header  $IP_T$  could be a completely custom header configuration with no similarity to a normal IP header except that it contain the information identified above. This is so since this header is interpreted by only TARP routers.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Referring to FIG. **4**, a TARP transceiver **405** can be an originating terminal **100**, a destination terminal **110**, or a TARP router **122-127**. In each TARP Transceiver **405**, a transmitting process is generated to receive normal packets from the Network (IP) layer and generate TARP packets for communication over the network. A receiving process is generated to receive normal IP packets containing TARP packets and generate from these normal IP packets which are "passed up" to the Network (IP) layer. Note that where the TARP Transceiver **405** is a router, the received TARP packets **140** are not processed into a stream of IP packets **415** because they need only be authenticated as proper TARP packets and then passed to another TARP router or a TARP destination terminal **110**. The intervening process, a "TARP Layer" **420**, could be combined with either the data link layer **430** or the Network layer **410**. In either case, it would intervene between the

## US 7,490,151 B2

11

data link layer **430** so that the process would receive regular IP packets containing embedded TARP packets and “hand up” a series of reassembled IP packets to the Network layer **410**. As an example of combining the TARP layer **420** with the data link layer **430**, a program may augment the normal processes running a communications card, for example, an Ethernet card. Alternatively, the TARP layer processes may form part of a dynamically loadable module that is loaded and executed to support communications between the network and data link layers.

Because the encryption system described above can be inserted between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

Note that IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of “attacks.” The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicates that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. To accomplish this, the TARP process will construct a TARP-formatted message, in the style of Internet Control Message Protocol (ICMP) datagrams as an example; this message will contain the machine’s TARP address, its previous IP address, and its new IP address. The TARP layer will transmit this packet to at least one known TARP router; then upon receipt and validation of the message, the TARP router will update its LUT with the new IP address for the stated TARP address. The TARP router will then format a similar message, and broadcast it to the other TARP routers so that they may update their LUTs. Since the total number of TARP routers on any given subnet is expected to be relatively small, this process of updating the LUTs should be relatively fast. It may not, however, work as well when there is a relatively large number of TARP routers and/or a relatively large number of clients; this has motivated a refinement of this architecture to provide scalability; this refinement has led to a second embodiment, which is discussed below.

Upon detection of an attack, the TARP process may also create a subprocess that maintains the original IP address and continues interacting with the attacker. The latter may provide an opportunity to trace the attacker or study the attacker’s methods (called “fishbowling” drawing upon the analogy of a small fish in a fish bowl that “thinks” it is in the ocean but is actually under captive observation). A history of the communication between the attacker and the abandoned (fish-

12

bowled) IP address can be recorded or transmitted for human analysis or further synthesized for purposes of responding in some way.

As mentioned above, decoy or dummy data or packets can be added to outgoing data streams by TARP terminals or routers. In addition to making it convenient to spread data over a larger number of separate packets, such decoy packets can also help to level the load on inactive portions of the Internet to help foil traffic analysis efforts.

Decoy packets may be generated by each TARP terminal **100, 110** or each router **122-127** on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. That is, when a series of messages are received, the decoy packet generation rate may be increased. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis. The rate of reception of packets, decoy or otherwise, may be indicated to the decoy packet dropping and generating processes through perishable decoy and regular packet counters. (A perishable counter is one that resets or decrements its value in response to time so that it contains a high value when it is incremented in rapid succession and a small value when incremented either slowly or a small number of times in rapid succession.) Note that destination TARP terminal **110** may generate decoy packets equal in number and size to those TARP packets received to make it appear it is merely routing packets and is therefore not the destination terminal.

Referring to FIG. 5, the following particular steps may be employed in the above-described method for routing TARP packets.

- S0. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.
- S2. The TARP packet may be probed in some way to authenticate the packet before attempting to decrypt it using the link key. That is, the router may determine that the packet is an authentic TARP packet by performing a selected operation on some data included with the clear IP header attached to the encrypted TARP packet contained in the payload. This makes it possible to avoid performing decryption on packets that are not authentic TARP packets.
- S3. The TARP packet is decrypted to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.
- S4. If the packet is a decoy packet, the perishable decoy counter is incremented.
- S5. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the router may choose to throw it away. If

## US 7,490,151 B2

13

the received packet is a decoy packet and it is determined that it should be thrown away (S6), control returns to step S0.

S7. The TTL parameter of the TARP header is decremented and it is determined if the TTL parameter is greater than zero. 5

S8. If the TTL parameter is greater than zero, a TARP address is randomly chosen from a list of TARP addresses maintained by the router and the link key and IP address corresponding to that TARP address memorized for use in creating a new IP packet containing the TARP packet. 10

S9. If the TTL parameter is zero or less, the link key and IP address corresponding to the TARP address of the destination are memorized for use in creating the new IP packet containing the TARP packet. 15

S10. The TARP packet is encrypted using the memorized link key.

S11. An IP header is added to the packet that contains the stored IP address, the encrypted TARP packet wrapped with an IP header, and the completed packet transmitted to the next hop or destination. 20

Referring to FIG. 6, the following particular steps may be employed in the above-described method for generating TARP packets. 25

S20. A background loop operation applies an algorithm that determines the generation of decoy IP packets. The loop is interrupted when a data stream containing IP packets is received for transmission.

S21. The received IP packets are grouped into a set consisting of messages with a constant IP destination address. The set is further broken down to coincide with a maximum size of an interleave window. The set is encrypted, and interleaved into a set of payloads destined to become TARP packets. 30

S22. The TARP address corresponding to the IP address is determined from a lookup table and stored to generate the TARP header. An initial TTL count is generated and stored in the header. The TTL count may be random with minimum and maximum values or it may be fixed or determined by some other parameter. 35

S23. The window sequence numbers and interleave sequence numbers are recorded in the TARP headers of each packet.

S24. One TARP router address is randomly chosen for each TARP packet and the IP address corresponding to it stored for use in the clear IP header. The link key corresponding to this router is identified and used to encrypt TARP packets containing interleaved and encrypted data and TARP headers. 40

S25. A clear IP header with the first hop router's real IP address is generated and added to each of the encrypted TARP packets and the resulting packets.

Referring to FIG. 7, the following particular steps may be employed in the above-described method for receiving TARP packets. 45

S40. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S42. The TARP packet may be probed to authenticate the packet before attempting to decrypt it using the link key.

S43. The TARP packet is decrypted with the appropriate link key to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message. 50

14

S44. If the packet is a decoy packet, the perishable decoy counter is incremented.

S45. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the receiver may choose to throw it away.

S46. The TARP packets are cached until all packets forming an interleave window are received.

S47. Once all packets of an interleave window are received, the packets are deinterleaved.

S48. The packets block of combined packets defining the interleave window is then decrypted using the session key.

S49. The decrypted block is then divided using the window sequence data and the IP<sub>T</sub> headers are converted into normal IP<sub>C</sub> headers. The window sequence numbers are integrated in the IP<sub>C</sub> headers.

S50. The packets are then handed up to the IP layer processes.

## 1. SCALABILITY ENHANCEMENTS

The IP agility feature described above relies on the ability to transmit IP address changes to all TARP routers. The embodiments including this feature will be referred to as "boutique" embodiments due to potential limitations in scaling these features up for a large network, such as the Internet. (The "boutique" embodiments would, however, be robust for use in smaller networks, such as small virtual private networks, for example). One problem with the boutique embodiments is that if IP address changes are to occur frequently, the message traffic required to update all routers sufficiently quickly creates a serious burden on the Internet when the TARP router and/or client population gets large. The bandwidth burden added to the networks, for example in ICMP packets, that would be used to update all the TARP routers could overwhelm the Internet for a large scale implementation that approached the scale of the Internet. In other words, the boutique system's scalability is limited.

A system can be constructed which trades some of the features of the above embodiments to provide the benefits of IP agility without the additional messaging burden. This is accomplished by IP address-hopping according to shared algorithms that govern IP addresses used between links participating in communications sessions between nodes such as TARP nodes. (Note that the IP hopping technique is also applicable to the boutique embodiment.) The IP agility feature discussed with respect to the boutique system can be modified so that it becomes decentralized under this scalable regime and governed by the above-described shared algorithm. Other features of the boutique system may be combined with this new type of IP-agility.

The new embodiment has the advantage of providing IP agility governed by a local algorithm and set of IP addresses exchanged by each communicating pair of nodes. This local governance is session-independent in that it may govern communications between a pair of nodes, irrespective of the session or end points being transferred between the directly communicating pair of nodes.

In the scalable embodiments, blocks of IP addresses are allocated to each node in the network. (This scalability will increase in the future, when Internet Protocol addresses are increased to 128-bit fields, vastly increasing the number of distinctly addressable nodes). Each node can thus use any of the IP addresses assigned to that node to communicate with other nodes in the network. Indeed, each pair of communicating nodes can use a plurality of source IP addresses and destination IP addresses for communicating with each other.

## US 7,490,151 B2

15

Each communicating pair of nodes in a chain participating in any session stores two blocks of IP addresses, called netblocks, and an algorithm and randomization seed for selecting, from each netblock, the next pair of source/destination IP addresses that will be used to transmit the next message. In other words, the algorithm governs the sequential selection of IP-address pairs, one sender and one receiver IP address, from each netblock. The combination of algorithm, seed, and netblock (IP address block) will be called a "hopblock." A router issues separate transmit and receive hopblocks to its clients. The send address and the receive address of the IP header of each outgoing packet sent by the client are filled with the send and receive IP addresses generated by the algorithm. The algorithm is "clocked" (indexed) by a counter so that each time a pair is used, the algorithm turns out a new transmit pair for the next packet to be sent.

The router's receive hopblock is identical to the client's transmit hopblock. The router uses the receive hopblock to predict what the send and receive IP address pair for the next expected packet from that client will be. Since packets can be received out of order, it is not possible for the router to predict with certainty what IP address pair will be on the next sequential packet. To account for this problem, the router generates a range of predictions encompassing the number of possible transmitted packet send/receive addresses, of which the next packet received could leap ahead. Thus, if there is a vanishingly small probability that a given packet will arrive at the router ahead of 5 packets transmitted by the client before the given packet, then the router can generate a series of 6 send/receive IP address pairs (or "hop window") to compare with the next received packet. When a packet is received, it is marked in the hop window as such, so that a second packet with the same IP address pair will be discarded. If an out-of-sequence packet does not arrive within a predetermined timeout period, it can be requested for retransmission or simply discarded from the receive table, depending upon the protocol in use for that communications session, or possibly by convention.

When the router receives the client's packet, it compares the send and receive IP addresses of the packet with the next N predicted send and receive IP address pairs and rejects the packet if it is not a member of this set. Received packets that do not have the predicted source/destination IP addresses falling within the window are rejected, thus thwarting possible hackers. (With the number of possible combinations, even a fairly large window would be hard to fall into at random.) If it is a member of this set, the router accepts the packet and processes it further. This link-based IP-hopping strategy, referred to as "IHOP," is a network element that stands on its own and is not necessarily accompanied by elements of the boutique system described above. If the routing agility feature described in connection with the boutique embodiment is combined with this link-based IP-hopping strategy, the router's next step would be to decrypt the TARP header to determine the destination TARP router for the packet and determine what should be the next hop for the packet. The TARP router would then forward the packet to a random TARP router or the destination TARP router with which the source TARP router has a link-based IP hopping communication established.

FIG. 8 shows how a client computer **801** and a TARP router **811** can establish a secure session. When client **801** seeks to establish an IHOP session with TARP router **811**, the client **801** sends "secure synchronization" request ("SSYN") packet **821** to the TARP router **811**. This SSYN packet **821** contains the client's **801** authentication token, and may be sent to the router **811** in an encrypted format. The source and

16

destination IP numbers on the packet **821** are the client's **801** current fixed IP address, and a "known" fixed IP address for the router **811**. (For security purposes, it may be desirable to reject any packets from outside of the local network that are destined for the router's known fixed IP address.) Upon receipt and validation of the client's **801** SSYN packet **821**, the router **811** responds by sending an encrypted "secure synchronization acknowledgment" ("SSYN ACK") **822** to the client **801**. This SSYN ACK **822** will contain the transmit and receive hopblocks that the client **801** will use when communicating with the TARP router **811**. The client **801** will acknowledge the TARP router's **811** response packet **822** by generating an encrypted SSYN ACK ACK packet **823** which will be sent from the client's **801** fixed IP address and to the TARP router's **811** known fixed IP address. The client **801** will simultaneously generate a SSYN ACK ACK packet; this SSYN ACK packet, referred to as the Secure Session Initiation (SSI) packet **824**, will be sent with the first {sender, receiver} IP pair in the client's transmit table **921** (FIG. 9), as specified in the transmit hopblock provided by the TARP router **811** in the SSYN ACK packet **822**. The TARP router **811** will respond to the SSI packet **824** with an SSI ACK packet **825**, which will be sent with the first {sender, receiver} IP pair in the TARP router's transmit table **923**. Once these packets have been successfully exchanged, the secure communications session is established, and all further secure communications between the client **801** and the TARP router **811** will be conducted via this secure session, as long as synchronization is maintained. If synchronization is lost, then the client **801** and TARP router **802** may re-establish the secure session by the procedure outlined in FIG. 8 and described above.

While the secure session is active, both the client **901** and TARP router **911** (FIG. 9) will maintain their respective transmit tables **921**, **923** and receive tables **922**, **924**, as provided by the TARP router during session synchronization **822**. It is important that the sequence of IP pairs in the client's transmit table **921** be identical to those in the TARP router's receive table **924**; similarly, the sequence of IP pairs in the client's receive table **922** must be identical to those in the router's transmit table **923**. This is required for the session synchronization to be maintained. The client **901** need maintain only one transmit table **921** and one receive table **922** during the course of the secure session. Each sequential packet sent by the client **901** will employ the next {send, receive} IP address pair in the transmit table, regardless of TCP or UDP session. The TARP router **911** will expect each packet arriving from the client **901** to bear the next IP address pair shown in its receive table.

Since packets can arrive out of order, however, the router **911** can maintain a "look ahead" buffer in its receive table, and will mark previously-received IP pairs as invalid for future packets; any future packet containing an IP pair that is in the look-ahead buffer but is marked as previously received will be discarded. Communications from the TARP router **911** to the client **901** are maintained in an identical manner; in particular, the router **911** will select the next IP address pair from its transmit table **923** when constructing a packet to send to the client **901**, and the client **901** will maintain a look-ahead buffer of expected IP pairs on packets that it is receiving. Each TARP router will maintain separate pairs of transmit and receive tables for each client that is currently engaged in a secure session with or through that TARP router.

While clients receive their hopblocks from the first server linking them to the Internet, routers exchange hopblocks. When a router establishes a link-based IP-hopping communication regime with another router, each router of the pair



US 7,490,151 B2

17

exchanges its transmit hopblock. The transmit hopblock of each router becomes the receive hopblock of the other router. The communication between routers is governed as described by the example of a client sending a packet to the first router.

While the above strategy works fine in the IP milieu, many local networks that are connected to the Internet are Ethernet systems. In Ethernet, the IP addresses of the destination devices must be translated into hardware addresses, and vice versa, using known processes (“address resolution protocol,” and “reverse address resolution protocol”). However, if the link-based IP-hopping strategy is employed, the correlation process would become explosive and burdensome. An alternative to the link-based IP hopping strategy may be employed within an Ethernet network. The solution is to provide that the node linking the Internet to the Ethernet (call it the border node) use the link-based IP-hopping communication regime to communicate with nodes outside the Ethernet LAN. Within the Ethernet LAN, each TARP node would have a single IP address which would be addressed in the conventional way. Instead of comparing the {sender, receiver} IP address pairs to authenticate a packet, the intra-LAN TARP node would use one of the IP header extension fields to do so. Thus, the border node uses an algorithm shared by the intra-LAN TARP node to generate a symbol that is stored in the free field in the IP header, and the intra-LAN TARP node generates a range of symbols based on its prediction of the next expected packet to be received from that particular source IP address. The packet is rejected if it does not fall into the set of predicted symbols (for example, numerical values) or is accepted if it does. Communications from the intra-LAN TARP node to the border node are accomplished in the same manner, though the algorithm will necessarily be different for security reasons. Thus, each of the communicating nodes will generate transmit and receive tables in a similar manner to that of FIG. 9; the intra-LAN TARP nodes transmit table will be identical to the border node’s receive table, and the intra-LAN TARP node’s receive table will be identical to the border node’s transmit table.

The algorithm used for IP address-hopping can be any desired algorithm. For example, the algorithm can be a given pseudo-random number generator that generates numbers of the range covering the allowed IP addresses with a given seed. Alternatively, the session participants can assume a certain type of algorithm and specify simply a parameter for applying the algorithm. For example the assumed algorithm could be a particular pseudo-random number generator and the session participants could simply exchange seed values.

Note that there is no permanent physical distinction between the originating and destination terminal nodes. Either device at either end point can initiate a synchronization of the pair. Note also that the authentication/synchronization-request (and acknowledgment) and hopblock-exchange may all be served by a single message so that separate message exchanges may not be required.

As another extension to the stated architecture, multiple physical paths can be used by a client, in order to provide link redundancy and further thwart attempts at denial of service and traffic monitoring. As shown in FIG. 10, for example, client 1001 can establish three simultaneous sessions with each of three TARP routers provided by different ISPs 1011, 1012, 1013. As an example, the client 1001 can use three different telephone lines 1021, 1022, 1023 to connect to the ISPs, or two telephone lines and a cable modem, etc. In this scheme, transmitted packets will be sent in a random fashion among the different physical paths. This architecture pro-

18

vides a high degree of communications redundancy, with improved immunity from denial-of-service attacks and traffic monitoring.

## 2. FURTHER EXTENSIONS

The following describes various extensions to the techniques, systems, and methods described above. As described above, the security of communications occurring between computers in a computer network (such as the Internet, an Ethernet, or others) can be enhanced by using seemingly random source and destination Internet Protocol (IP) addresses for data packets transmitted over the network. This feature prevents eavesdroppers from determining which computers in the network are communicating with each other while permitting the two communicating computers to easily recognize whether a given received data packet is legitimate or not. In one embodiment of the above-described systems, an IP header extension field is used to authenticate incoming packets on an Ethernet.

Various extensions to the previously described techniques described herein include: (1) use of hopped hardware or “MAC” addresses in broadcast type network; (2) a self-synchronization technique that permits a computer to automatically regain synchronization with a sender; (3) synchronization algorithms that allow transmitting and receiving computers to quickly re-establish synchronization in the event of lost packets or other events; and (4) a fast-packet rejection mechanism for rejecting invalid packets. Any or all of these extensions can be combined with the features described above in any of various ways.

### A. Hardware Address Hopping

Internet protocol-based communications techniques on a LAN—or across any dedicated physical medium—typically embed the IP packets within lower-level packets, often referred to as “frames.” As shown in FIG. 11, for example, a first Ethernet frame 1150 comprises a frame header 1101 and two embedded IP packets IP1 and IP2, while a second Ethernet frame 1160 comprises a different frame header 1104 and a single IP packet IP3. Each frame header generally includes a source hardware address 1101A and a destination hardware address 1101B; other well-known fields in frame headers are omitted from FIG. 11 for clarity. Two hardware nodes communicating over a physical communication channel insert appropriate source and destination hardware addresses to indicate which nodes on the channel or network should receive the frame.

It may be possible for a nefarious listener to acquire information about the contents of a frame and/or its communicants by examining frames on a local network rather than (or in addition to) the IP packets themselves. This is especially true in broadcast media, such as Ethernet, where it is necessary to insert into the frame header the hardware address of the machine that generated the frame and the hardware address of the machine to which frame is being sent. All nodes on the network can potentially “see” all packets transmitted across the network. This can be a problem for secure communications, especially in cases where the communicants do not want for any third party to be able to identify who is engaging in the information exchange. One way to address this problem is to push the address-hopping scheme down to the hardware layer. In accordance with various embodiments of the invention, hardware addresses are “hopped” in a manner similar to that used to change IP addresses, such that a listener cannot



## US 7,490,151 B2

19

determine which hardware node generated a particular message nor which node is the intended recipient.

FIG. 12A shows a system in which Media Access Control ("MAC") hardware addresses are "hopped" in order to increase security over a network such as an Ethernet. While the description refers to the exemplary case of an Ethernet environment, the inventive principles are equally applicable to other types of communications media. In the Ethernet case, the MAC address of the sender and receiver are inserted into the Ethernet frame and can be observed by anyone on the LAN who is within the broadcast range for that frame. For secure communications, it becomes desirable to generate frames with MAC addresses that are not attributable to any specific sender or receiver.

As shown in FIG. 12A, two computer nodes 1201 and 1202 communicate over a communication channel such as an Ethernet. Each node executes one or more application programs 1203 and 1218 that communicate by transmitting packets through communication software 1204 and 1217, respectively. Examples of application programs include video conferencing, e-mail, word processing programs, telephony, and the like. Communication software 1204 and 1217 can comprise, for example, an OSI layered architecture or "stack" that standardizes various services provided at different levels of functionality.

The lowest levels of communication software 1204 and 1217 communicate with hardware components 1206 and 1214 respectively, each of which can include one or more registers 1207 and 1215 that allow the hardware to be reconfigured or controlled in accordance with various communication protocols. The hardware components (an Ethernet network interface card, for example) communicate with each other over the communication medium. Each hardware component is typically pre-assigned a fixed hardware address or MAC number that identifies the hardware component to other nodes on the network. One or more interface drivers control the operation of each card and can, for example, be configured to accept or reject packets from certain hardware addresses. As will be described in more detail below, various embodiments of the inventive principles provide for "hopping" different addresses using one or more algorithms and one or more moving windows that track a range of valid addresses to validate received packets. Packets transmitted according to one or more of the inventive principles will be generally referred to as "secure" packets or "secure communications" to differentiate them from ordinary data packets that are transmitted in the clear using ordinary, machine-correlated addresses.

One straightforward method of generating non-attributable MAC addresses is an extension of the IP hopping scheme. In this scenario, two machines on the same LAN that desire to communicate in a secure fashion exchange random-number generators and seeds, and create sequences of quasi-random MAC addresses for synchronized hopping. The implementation and synchronization issues are then similar to that of IP hopping.

This approach, however, runs the risk of using MAC addresses that are currently active on the LAN—which, in turn, could interrupt communications for those machines. Since an Ethernet MAC address is at present 48 bits in length, the chance of randomly misusing an active MAC address is actually quite small. However, if that figure is multiplied by a large number of nodes (as would be found on an extensive LAN), by a large number of frames (as might be the case with packet voice or streaming video), and by a large number of concurrent Virtual Private Networks (VPNs), then the chance that a non-secure machine's MAC address could be used in an

20

address-hopped frame can become non-trivial. In short, any scheme that runs even a small risk of interrupting communications for other machines on the LAN is bound to receive resistance from prospective system administrators. Nevertheless, it is technically feasible, and can be implemented without risk on a LAN on which there is a small number of machines, or if all of the machines on the LAN are engaging in MAC-hopped communications.

Synchronized MAC address hopping may incur some overhead in the course of session establishment, especially if there are multiple sessions or multiple nodes involved in the communications. A simpler method of randomizing MAC addresses is to allow each node to receive and process every incident frame on the network. Typically, each network interface driver will check the destination MAC address in the header of every incident frame to see if it matches that machine's MAC address; if there is no match, then the frame is discarded. In one embodiment, however, these checks can be disabled, and every incident packet is passed to the TARP stack for processing. This will be referred to as "promiscuous" mode, since every incident frame is processed. Promiscuous mode allows the sender to use completely random, unsynchronized MAC addresses, since the destination machine is guaranteed to process the frame. The decision as to whether the packet was truly intended for that machine is handled by the TARP stack, which checks the source and destination IP addresses for a match in its IP synchronization tables. If no match is found, the packet is discarded; if there is a match, the packet is unwrapped, the inner header is evaluated, and if the inner header indicates that the packet is destined for that machine then the packet is forwarded to the IP stack—otherwise it is discarded.

One disadvantage of purely-random MAC address hopping is its impact on processing overhead; that is, since every incident frame must be processed, the machine's CPU is engaged considerably more often than if the network interface driver is discriminating and rejecting packets unilaterally. A compromise approach is to select either a single fixed MAC address or a small number of MAC addresses (e.g., one for each virtual private network on an Ethernet) to use for MAC-hopped communications, regardless of the actual recipient for which the message is intended. In this mode, the network interface driver can check each incident frame against one (or a few) pre-established MAC addresses, thereby freeing the CPU from the task of physical-layer packet discrimination. This scheme does not betray any useful information to an interloper on the LAN; in particular, every secure packet can already be identified by a unique packet type in the outer header. However, since all machines engaged in secure communications would either be using the same MAC address, or be selecting from a small pool of predetermined MAC addresses, the association between a specific machine and a specific MAC address is effectively broken.

In this scheme, the CPU will be engaged more often than it would be in non-secure communications (or in synchronized MAC address hopping), since the network interface driver cannot always unilaterally discriminate between secure packets that are destined for that machine, and secure packets from other VPNs. However, the non-secure traffic is easily eliminated at the network interface, thereby reducing the amount of processing required of the CPU. There are boundary conditions where these statements would not hold, of course—e.g., if all of the traffic on the LAN is secure traffic, then the CPU would be engaged to the same degree as it is in the purely-random address hopping case; alternatively, if each VPN on the LAN uses a different MAC address, then the

## US 7,490,151 B2

21

network interface can perfectly discriminate secure frames destined for the local machine from those constituting other VPNs. These are engineering tradeoffs that might be best handled by providing administrative options for the users when installing the software and/or establishing VPNs.

Even in this scenario, however, there still remains a slight risk of selecting MAC addresses that are being used by one or more nodes on the LAN. One solution to this problem is to formally assign one address or a range of addresses for use in MAC-hopped communications. This is typically done via an assigned numbers registration authority; e.g., in the case of Ethernet, MAC address ranges are assigned to vendors by the Institute of Electrical and Electronics Engineers (IEEE). A formally-assigned range of addresses would ensure that secure frames do not conflict with any properly-configured and properly-functioning machines on the LAN.

Reference will now be made to FIGS. 12A and 12B in order to describe the many combinations and features that follow the inventive principles. As explained above, two computer nodes **1201** and **1202** are assumed to be communicating over a network or communication medium such as an Ethernet. A communication protocol in each node (**1204** and **1217**, respectively) contains a modified element **1205** and **1216** that performs certain functions that deviate from the standard communication protocols. In particular, computer node **1201** implements a first "hop" algorithm **1208X** that selects seemingly random source and destination IP addresses (and, in one embodiment, seemingly random IP header discriminator fields) in order to transmit each packet to the other computer node. For example, node **1201** maintains a transmit table **1208** containing triplets of source (S), destination (D), and discriminator fields (DS) that are inserted into outgoing IP packet headers. The table is generated through the use of an appropriate algorithm (e.g., a random number generator that is seeded with an appropriate seed) that is known to the recipient node **1202**. As each new IP packet is formed, the next sequential entry out of the sender's transmit table **1208** is used to populate the IP source, IP destination, and IP header extension field (e.g., discriminator field). It will be appreciated that the transmit table need not be created in advance but could instead be created on-the-fly by executing the algorithm when each packet is formed.

At the receiving node **1202**, the same IP hop algorithm **1222X** is maintained and used to generate a receive table **1222** that lists valid triplets of source IP address, destination IP address, and discriminator field. This is shown by virtue of the first five entries of transmit table **1208** matching the second five entries of receive table **1222**. (The tables may be slightly offset at any particular time due to lost packets, mis-ordered packets, or transmission delays). Additionally, node **1202** maintains a receive window **W3** that represents a list of valid IP source, IP destination, and discriminator fields that will be accepted when received as part of an incoming IP packet. As packets are received, window **W3** slides down the list of valid entries, such that the possible valid entries change over time. Two packets that arrive out of order but are nevertheless matched to entries within window **W3** will be accepted; those falling outside of window **W3** will be rejected as invalid. The length of window **W3** can be adjusted as necessary to reflect network delays or other factors.

Node **1202** maintains a similar transmit table **1221** for creating IP packets and frames destined for node **1201** using a potentially different hopping algorithm **1221X**, and node **1201** maintains a matching receive table **1209** using the same algorithm **1209X**. As node **1202** transmits packets to node **1201** using seemingly random IP source, IP destination, and/or discriminator fields, node **1201** matches the incoming

22

packet values to those falling within window **W1** maintained in its receive table. In effect, transmit table **1208** of node **1201** is synchronized (i.e., entries are selected in the same order) to receive table **1222** of receiving node **1202**. Similarly, transmit table **1221** of node **1202** is synchronized to receive table **1209** of node **1201**. It will be appreciated that although a common algorithm is shown for the source, destination and discriminator fields in FIG. 12A (using, e.g., a different seed for each of the three fields), an entirely different algorithm could in fact be used to establish values for each of these fields. It will also be appreciated that one or two of the fields can be "hopped" rather than all three as illustrated.

In accordance with another aspect of the invention, hardware or "MAC" addresses are hopped instead of or in addition to IP addresses and/or the discriminator field in order to improve security in a local area or broadcast-type network. To that end, node **1201** further maintains a transmit table **1210** using a transmit algorithm **1210X** to generate source and destination hardware addresses that are inserted into frame headers (e.g., fields **1101A** and **1101B** in FIG. 11) that are synchronized to a corresponding receive table **1224** at node **1202**. Similarly, node **1202** maintains a different transmit table **1223** containing source and destination hardware addresses that is synchronized with a corresponding receive table **1211** at node **1201**. In this manner, outgoing hardware frames appear to be originating from and going to completely random nodes on the network, even though each recipient can determine whether a given packet is intended for it or not. It will be appreciated that the hardware hopping feature can be implemented at a different level in the communications protocol than the IP hopping feature (e.g., in a card driver or in a hardware card itself to improve performance).

FIG. 12B shows three different embodiments or modes that can be employed using the aforementioned principles. In a first mode referred to as "promiscuous" mode, a common hardware address (e.g., a fixed address for source and another for destination) or else a completely random hardware address is used by all nodes on the network, such that a particular packet cannot be attributed to any one node. Each node must initially accept all packets containing the common (or random) hardware address and inspect the IP addresses or discriminator field to determine whether the packet is intended for that node. In this regard, either the IP addresses or the discriminator field or both can be varied in accordance with an algorithm as described above. As explained previously, this may increase each node's overhead since additional processing is involved to determine whether a given packet has valid source and destination hardware addresses.

In a second mode referred to as "promiscuous per VPN" mode, a small set of fixed hardware addresses are used, with a fixed source/destination hardware address used for all nodes communicating over a virtual private network. For example, if there are six nodes on an Ethernet, and the network is to be split up into two private virtual networks such that nodes on one VPN can communicate with only the other two nodes on its own VPN, then two sets of hardware addresses could be used: one set for the first VPN and a second set for the second VPN. This would reduce the amount of overhead involved in checking for valid frames since only packets arriving from the designated VPN would need to be checked. IP addresses and one or more discriminator fields could still be hopped as before for secure communication within the VPN. Of course, this solution compromises the anonymity of the VPNs (i.e., an outsider can easily tell what traffic belongs in which VPN, though he cannot correlate it to a specific machine/person). It also requires the use of a discriminator field to mitigate the vulnerability to certain types of DoS attacks. (For example,

## US 7,490,151 B2

23

without the discriminator field, an attacker on the LAN could stream frames containing the MAC addresses being used by the VPN; rejecting those frames could lead to excessive processing overhead. The discriminator field would provide a low-overhead means of rejecting the false packets.)

In a third mode referred to as “hardware hopping” mode, hardware addresses are varied as illustrated in FIG. 12A, such that hardware source and destination addresses are changed constantly in order to provide non-attributable addressing. Variations on these embodiments are of course possible, and the invention is not intended to be limited in any respect by these illustrative examples.

#### B. Extending the Address Space

Address hopping provides security and privacy. However, the level of protection is limited by the number of addresses in the blocks being hopped. A hopblock denotes a field or fields modulated on a packet-wise basis for the purpose of providing a VPN. For instance, if two nodes communicate with IP address hopping using hopblocks of 4 addresses (2 bits) each, there would be 16 possible address-pair combinations. A window of size 16 would result in most address pairs being accepted as valid most of the time. This limitation can be overcome by using a discriminator field in addition to or instead of the hopped address fields. The discriminator field would be hopped in exactly the same fashion as the address fields and it would be used to determine whether a packet should be processed by a receiver.

Suppose that two clients, each using four-bit hopblocks, would like the same level of protection afforded to clients communicating via IP hopping between two A blocks (24 address bits eligible for hopping). A discriminator field of 20 bits, used in conjunction with the 4 address bits eligible for hopping in the IP address field, provides this level of protection. A 24-bit discriminator field would provide a similar level of protection if the address fields were not hopped or ignored. Using a discriminator field offers the following advantages: (1) an arbitrarily high level of protection can be provided, and (2) address hopping is unnecessary to provide protection. This may be important in environments where address hopping would cause routing problems.

#### C. Synchronization Techniques

It is generally assumed that once a sending node and receiving node have exchanged algorithms and seeds (or similar information sufficient to generate quasi-random source and destination tables), subsequent communication between the two nodes will proceed smoothly. Realistically, however, two nodes may lose synchronization due to network delays or outages, or other problems. Consequently, it is desirable to provide means for re-establishing synchronization between nodes in a network that have lost synchronization.

One possible technique is to require that each node provide an acknowledgment upon successful receipt of each packet and, if no acknowledgment is received within a certain period of time, to re-send the unacknowledged packet. This approach, however, drives up overhead costs and may be prohibitive in high-throughput environments such as streaming video or audio, for example.

A different approach is to employ an automatic synchronizing technique that will be referred to herein as “self-synchronization.” In this approach, synchronization information is embedded into each packet, thereby enabling the receiver to re-synchronize itself upon receipt of a single packet if it

24

determines that it has lost synchronization with the sender. (If communications are already in progress, and the receiver determines that it is still in sync with the sender, then there is no need to re-synchronize.) A receiver could detect that it was out of synchronization by, for example, employing a “dead-man” timer that expires after a certain period of time, wherein the timer is reset with each valid packet. A time stamp could be hashed into the public sync field (see below) to preclude packet-retry attacks.

In one embodiment, a “sync field” is added to the header of each packet sent out by the sender. This sync field could appear in the clear or as part of an encrypted portion of the packet. Assuming that a sender and receiver have selected a random-number generator (RNG) and seed value, this combination of RNG and seed can be used to generate a random-number sequence (RNS). The RNS is then used to generate a sequence of source/destination IP pairs (and, if desired, discriminator fields and hardware source and destination addresses), as described above. It is not necessary, however, to generate the entire sequence (or the first N-1 values) in order to generate the Nth random number in the sequence; if the sequence index N is known, the random value corresponding to that index can be directly generated (see below). Different RNGs (and seeds) with different fundamental periods could be used to generate the source and destination IP sequences, but the basic concepts would still apply. For the sake of simplicity, the following discussion will assume that IP source and destination address pairs (only) are hopped using a single RNG sequencing mechanism.

In accordance with a “self-synchronization” feature, a sync field in each packet header provides an index (i.e., a sequence number) into the RNS that is being used to generate IP pairs. Plugging this index into the RNG that is being used to generate the RNS yields a specific random number value, which in turn yields a specific IP pair. That is, an IP pair can be generated directly from knowledge of the RNG, seed, and index number; it is not necessary, in this scheme, to generate the entire sequence of random numbers that precede the sequence value associated with the index number provided.

Since the communicants have presumably previously exchanged RNGs and seeds, the only new information that must be provided in order to generate an IP pair is the sequence number. If this number is provided by the sender in the packet header, then the receiver need only plug this number into the RNG in order to generate an IP pair—and thus verify that the IP pair appearing in the header of the packet is valid. In this scheme, if the sender and receiver lose synchronization, the receiver can immediately re-synchronize upon receipt of a single packet by simply comparing the IP pair in the packet header to the IP pair generated from the index number. Thus, synchronized communications can be resumed upon receipt of a single packet, making this scheme ideal for multicast communications. Taken to the extreme, it could obviate the need for synchronization tables entirely; that is, the sender and receiver could simply rely on the index number in the sync field to validate the IP pair on each packet, and thereby eliminate the tables entirely.

The aforementioned scheme may have some inherent security issues associated with it—namely, the placement of the sync field. If the field is placed in the outer header, then an interloper could observe the values of the field and their relationship to the IP stream. This could potentially compromise the algorithm that is being used to generate the IP-address sequence, which would compromise the security of the communications. If, however, the value is placed in the inner header, then the sender must decrypt the inner header before it can extract the sync value and validate the IP pair;

## US 7,490,151 B2

25

this opens up the receiver to certain types of denial-of-service (DoS) attacks, such as packet replay. That is, if the receiver must decrypt a packet before it can validate the IP pair, then it could potentially be forced to expend a significant amount of processing on decryption if an attacker simply retransmits previously valid packets. Other attack methodologies are possible in this scenario.

A possible compromise between algorithm security and processing speed is to split up the sync value between an inner (encrypted) and outer (unencrypted) header. That is, if the sync value is sufficiently long, it could potentially be split into a rapidly-changing part that can be viewed in the clear, and a fixed (or very slowly changing) part that must be protected. The part that can be viewed in the clear will be called the “public sync” portion and the part that must be protected will be called the “private sync” portion.

Both the public sync and private sync portions are needed to generate the complete sync value. The private portion, however, can be selected such that it is fixed or will change only occasionally. Thus, the private sync value can be stored by the recipient, thereby obviating the need to decrypt the header in order to retrieve it. If the sender and receiver have previously agreed upon the frequency with which the private part of the sync will change, then the receiver can selectively decrypt a single header in order to extract the new private sync if the communications gap that has led to lost synchronization has exceeded the lifetime of the previous private sync. This should not represent a burdensome amount of decryption, and thus should not open up the receiver to denial-of-service attack simply based on the need to occasionally decrypt a single header.

One implementation of this is to use a hashing function with a one-to-one mapping to generate the private and public sync portions from the sync value. This implementation is shown in FIG. 13, where (for example) a first ISP 1302 is the sender and a second ISP 1303 is the receiver. (Other alternatives are possible from FIG. 13.) A transmitted packet comprises a public or “outer” header 1305 that is not encrypted, and a private or “inner” header 1306 that is encrypted using for example a link key. Outer header 1305 includes a public sync portion while inner header 1306 contains the private sync portion. A receiving node decrypts the inner header using a decryption function 1307 in order to extract the private sync portion. This step is necessary only if the lifetime of the currently buffered private sync has expired. (If the currently-buffered private sync is still valid, then it is simply extracted from memory and “added” (which could be an inverse hash) to the public sync, as shown in step 1308.) The public and decrypted private sync portions are combined in function 1308 in order to generate the combined sync 1309. The combined sync (1309) is then fed into the RNG (1310) and compared to the IP address pair (1311) to validate or reject the packet.

An important consideration in this architecture is the concept of “future” and “past” where the public sync values are concerned. Though the sync values, themselves, should be random to prevent spoofing attacks, it may be important that the receiver be able to quickly identify a sync value that has already been sent—even if the packet containing that sync value was never actually received by the receiver. One solution is to hash a time stamp or sequence number into the public sync portion, which could be quickly extracted, checked, and discarded, thereby validating the public sync portion itself.

In one embodiment, packets can be checked by comparing the source/destination IP pair generated by the sync field with the pair appearing in the packet header. If (1) they match, (2)

26

the time stamp is valid, and (3) the dead-man timer has expired, then re-synchronization occurs; otherwise, the packet is rejected. If enough processing power is available, the dead-man timer and synchronization tables can be avoided altogether, and the receiver would simply resynchronize (e.g., validate) on every packet.

The foregoing scheme may require large-integer (e.g., 160-bit) math, which may affect its implementation. Without such large-integer registers, processing throughput would be affected, thus potentially affecting security from a denial-of-service standpoint. Nevertheless, as large-integer math processing features become more prevalent, the costs of implementing such a feature will be reduced.

## D. Other Synchronization Schemes

As explained above, if W or more consecutive packets are lost between a transmitter and receiver in a VPN (where W is the window size), the receiver’s window will not have been updated and the transmitter will be transmitting packets not in the receiver’s window. The sender and receiver will not recover synchronization until perhaps the random pairs in the window are repeated by chance. Therefore, there is a need to keep a transmitter and receiver in synchronization whenever possible and to re-establish synchronization whenever it is lost.

A “checkpoint” scheme can be used to regain synchronization between a sender and a receiver that have fallen out of synchronization. In this scheme, a checkpoint message comprising a random IP address pair is used for communicating synchronization information. In one embodiment, two messages are used to communicate synchronization information between a sender and a recipient:

1. SYNC\_REQ is a message used by the sender to indicate that it wants to synchronize; and
2. SYNC\_ACK is a message used by the receiver to inform the transmitter that it has been synchronized.

According to one variation of this approach, both the transmitter and receiver maintain three checkpoints (see FIG. 14):

1. In the transmitter, ckpt\_o (“checkpoint old”) is the IP pair that was used to re-send the last SYNC\_REQ packet to the receiver. In the receiver, ckpt\_o (“checkpoint old”) is the IP pair that receives repeated SYNC\_REQ packets from the transmitter.
2. In the transmitter, ckpt\_n (“checkpoint new”) is the IP pair that will be used to send the next SYNC\_REQ packet to the receiver. In the receiver, ckpt\_n (“checkpoint new”) is the IP pair that receives a new SYNC\_REQ packet from the transmitter and which causes the receiver’s window to be re-aligned, ckpt\_o set to ckpt\_n, a new ckpt\_n to be generated and a new ckpt\_r to be generated.
3. In the transmitter, ckpt\_r is the IP pair that will be used to send the next SYNC\_ACK packet to the receiver. In the receiver, ckpt\_r is the IP pair that receives a new SYNC\_ACK packet from the transmitter and which causes a new ckpt\_n to be generated. Since SYNC\_ACK is transmitted from the receiver ISP to the sender ISP, the transmitter ckpt\_r refers to the ckpt\_r of the receiver and the receiver ckpt\_r refers to the ckpt\_r of the transmitter (see FIG. 14).

When a transmitter initiates synchronization, the IP pair it will use to transmit the next data packet is set to a predetermined value and when a receiver first receives a SYNC\_REQ, the receiver window is updated to be centered on the transmitter’s next IP pair. This is the primary mechanism for checkpoint synchronization.

## US 7,490,151 B2

27

Synchronization can be initiated by a packet counter (e.g., after every N packets transmitted, initiate a synchronization) or by a timer (every S seconds, initiate a synchronization) or a combination of both. See FIG. 15. From the transmitter's perspective, this technique operates as follows: (1) Each transmitter periodically transmits a "sync request" message to the receiver to make sure that it is in sync. (2) If the receiver is still in sync, it sends back a "sync ack" message. (If this works, no further action is necessary). (3) If no "sync ack" has been received within a period of time, the transmitter retransmits the sync request again. If the transmitter reaches the next checkpoint without receiving a "sync ack" response, then synchronization is broken, and the transmitter should stop transmitting. The transmitter will continue to send sync\_reqs until it receives a sync\_ack, at which point transmission is reestablished.

From the receiver's perspective, the scheme operates as follows: (1) when it receives a "sync request" request from the transmitter, it advances its window to the next checkpoint position (even skipping pairs if necessary), and sends a "sync ack" message to the transmitter. If sync was never lost, then the "jump ahead" really just advances to the next available pair of addresses in the table (i.e., normal advancement).

If an interloper intercepts the "sync request" messages and tries to interfere with communication by sending new ones, it will be ignored if the synchronization has been established or it it will actually help to re-establish synchronization.

A window is realigned whenever a re-synchronization occurs. This realignment entails updating the receiver's window to straddle the address pairs used by the packet transmitted immediately after the transmission of the SYNC\_REQ packet. Normally, the transmitter and receiver are in synchronization with one another. However, when network events occur, the receiver's window may have to be advanced by many steps during resynchronization. In this case, it is desirable to move the window ahead without having to step through the intervening random numbers sequentially. (This feature is also desirable for the auto-sync approach discussed above).

#### E. Random Number Generator with a Jump-Ahead capability

An attractive method for generating randomly hopped addresses is to use identical random number generators in the transmitter and receiver and advance them as packets are transmitted and received. There are many random number generation algorithms that could be used. Each one has strengths and weaknesses for address hopping applications.

Linear congruential random number generators (LCRs) are fast, simple and well characterized random number generators that can be made to jump ahead n steps efficiently. An LCR generates random numbers  $X_1, X_2, X_3 \dots X_k$  starting with seed  $X_0$  using a recurrence

$$X_i = (a X_{i-1} + b) \bmod c \quad (1)$$

where a, b and c define a particular LCR. Another expression for  $X_i$ ,

$$X_i = ((a^i(X_0 + b) - b) / (a - 1)) \bmod c \quad (2)$$

enables the jump-ahead capability. The factor  $a^i$  can grow very large even for modest i if left unfettered. Therefore some special properties of the modulo operation can be used to control the size and processing time required to compute (2). (2) can be rewritten as:

$$X_i = (a^i(X_0(a-1)+b)-b)/(a-1) \bmod c \quad (3)$$

28

It can be shown that:

$$\begin{aligned} (a^i(X_0(a-1)+b)-b)/(a-1) \bmod c &= ((a^i \bmod ((a-1)/\gcd(c, a-1))) \\ &\quad (X_0(a-1)+b)-b)/(a-1) \bmod c \end{aligned} \quad (4)$$

$(X_0(a-1)+b)$  can be stored as  $(X_0(a-1)+b) \bmod c$ , b as  $b \bmod c$  and compute  $a^i \bmod ((a-1)/\gcd(c, a-1))$  (this requires  $O(\log(i))$  steps).

A practical implementation of this algorithm would jump a fixed distance, n, between synchronizations; this is tantamount to synchronizing every n packets. The window would commence n IP pairs from the start of the previous window. Using  $X_j^w$ , the random number at the  $j^{\text{th}}$  checkpoint, as  $X_0$  and n as i, a node can store  $a^n \bmod ((a-1)/\gcd(c, a-1))$  once per LCR and set

$$X_{j+1}^w = X_{n(j+1)} = ((a^n \bmod ((a-1)/\gcd(c, a-1))) (X_j^w(a-1)+b)-b) / (a-1) \bmod c, \quad (5)$$

to generate the random number for the  $j+1^{\text{th}}$  synchronization. Using this construction, a node could jump ahead an arbitrary (but fixed) distance between synchronizations in a constant amount of time (independent of n).

Pseudo-random number generators, in general, and LCRs, in particular, will eventually repeat their cycles. This repetition may present vulnerability in the IP hopping scheme. An adversary would simply have to wait for a repeat to predict future sequences. One way of coping with this vulnerability is to create a random number generator with a known long cycle. A random sequence can be replaced by a new random number generator before it repeats. LCRs can be constructed with known long cycles. This is not currently true of many random number generators.

Random number generators can be cryptographically insecure. An adversary can derive the RNG parameters by examining the output or part of the output. This is true of LCGs. This vulnerability can be mitigated by incorporating an encryptor, designed to scramble the output as part of the random number generator. The random number generator prevents an adversary from mounting an attack—e.g., a known plaintext attack—against the encryptor.

#### F. Random Number Generator Example

Consider a RNG where  $a=31$ ,  $b=4$  and  $c=15$ . For this case equation (1) becomes:

$$X_i = (31 X_{i-1} + 4) \bmod 15. \quad (6)$$

If one sets  $X_0=1$ , equation (6) will produce the sequence 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 0, 4, 8, 12. This sequence will repeat indefinitely. For a jump ahead of 3 numbers in this sequence  $a^n=31^3=29791$ ,  $c*(a-1)=15*30=450$  and  $a^n \bmod ((a-1)c)=31^3 \bmod (15*30)=29791 \bmod (450)=91$ . Equation (5) becomes:

$$(91(X_i 30 + 4) - 4) / 30 \bmod 15 \quad (7)$$

Table 1 shows the jump ahead calculations from (7). The calculations start at 5 and jump ahead 3.

TABLE 1

I	$X_i$	$(X_i 30 + 4)$	$91(X_i 30 + 4) - 4$	$((91(X_i 30 + 4) - 4) / 30)$	$X_{i+3}$
1	5	154	14010	467	2
4	2	64	5820	194	14
7	14	424	38580	1286	11
10	11	334	30390	1013	8
13	8	244	22200	740	5

#### G. Fast Packet Filter

Address hopping VPNs must rapidly determine whether a packet has a valid header and thus requires further processing,

## US 7,490,151 B2

29

or has an invalid header (a hostile packet) and should be immediately rejected. Such rapid determinations will be referred to as “fast packet filtering.” This capability protects the VPN from attacks by an adversary who streams hostile packets at the receiver at a high rate of speed in the hope of saturating the receiver’s processor (a so-called “denial of service” attack). Fast packet filtering is an important feature for implementing VPNs on shared media such as Ethernet.

Assuming that all participants in a VPN share an unassigned “A” block of addresses, one possibility is to use an experimental “A” block that will never be assigned to any machine that is not address hopping on the shared medium. “A” blocks have a 24 bits of address that can be hopped as opposed to the 8 bits in “C” blocks. In this case a hopblock will be the “A” block. The use of the experimental “A” block is a likely option on an Ethernet because:

1. The addresses have no validity outside of the Ethernet and will not be routed out to a valid outside destination by a gateway.
2. There are  $2^{24}$  (~16 million) addresses that can be hopped within each “A” block. This yields >280 trillion possible address pairs making it very unlikely that an adversary would guess a valid address. It also provides acceptably low probability of collision between separate VPNs (all VPNs on a shared medium independently generate random address pairs from the same “A” block).
3. The packets will not be received by someone on the Ethernet who is not on a VPN (unless the machine is in promiscuous mode) minimizing impact on non-VPN computers.

The Ethernet example will be used to describe one implementation of fast packet filtering. The ideal algorithm would quickly examine a packet header, determine whether the packet is hostile, and reject any hostile packets or determine which active IP pair the packet header matches. The problem is a classical associative memory problem. A variety of techniques have been developed to solve this problem (hashing, B-trees etc). Each of these approaches has its strengths and weaknesses. For instance, hash tables can be made to operate quite fast in a statistical sense, but can occasionally degenerate into a much slower algorithm. This slowness can persist for a period of time. Since there is a need to discard hostile packets quickly at all times, hashing would be unacceptable.

#### H. Presence Vector Algorithm

A presence vector is a bit vector of length  $2^n$  that can be indexed by n-bit numbers (each ranging from 0 to  $2^n-1$ ). One can indicate the presence of k n-bit numbers (not necessarily unique), by setting the bits in the presence vector indexed by each number to 1. Otherwise, the bits in the presence vector are 0. An n-bit number, x, is one of the k numbers if and only if the  $x^{th}$  bit of the presence vector is 1. A fast packet filter can be implemented by indexing the presence vector and looking for a 1, which will be referred to as the “test.”

For example, suppose one wanted to represent the number 135 using a presence vector. The 135<sup>th</sup> bit of the vector would be set. Consequently, one could very quickly determine whether an address of 135 was valid by checking only one bit: the 135<sup>th</sup> bit. The presence vectors could be created in advance corresponding to the table entries for the IP addresses. In effect, the incoming addresses can be used as indices into a long vector, making comparisons very fast. As each RNG generates a new address, the presence vector is updated to reflect the information. As the window moves, the presence vector is updated to zero out addresses that are no longer valid.

30

There is a trade-off between efficiency of the test and the amount of memory required for storing the presence vector(s). For instance, if one were to use the 48 bits of hopping addresses as an index, the presence vector would have to be 35 terabytes. Clearly, this is too large for practical purposes. Instead, the 48 bits can be divided into several smaller fields. For instance, one could subdivide the 48 bits into four 12-bit fields (see FIG. 16). This reduces the storage requirement to 2048 bytes at the expense of occasionally having to process a hostile packet. In effect, instead of one long presence vector, the decomposed address portions must match all four shorter presence vectors before further processing is allowed. (If the first part of the address portion doesn’t match the first presence vector, there is no need to check the remaining three presence vectors).

A presence vector will have a 1 in the  $y^{th}$  bit if and only if one or more addresses with a corresponding field of y are active. An address is active only if each presence vector indexed by the appropriate sub-field of the address is 1.

Consider a window of 32 active addresses and 3 checkpoints. A hostile packet will be rejected by the indexing of one presence vector more than 99% of the time. A hostile packet will be rejected by the indexing of all 4 presence vectors more than 99.999995% of the time. On average, hostile packets will be rejected in less than 1.02 presence vector index operations.

The small percentage of hostile packets that pass the fast packet filter will be rejected when matching pairs are not found in the active window or are active checkpoints. Hostile packets that serendipitously match a header will be rejected when the VPN software attempts to decrypt the header. However, these cases will be extremely rare. There are many other ways this method can be configured to arbitrate the space/speed tradeoffs.

#### I. Further Synchronization Enhancements

A slightly modified form of the synchronization techniques described above can be employed. The basic principles of the previously described checkpoint synchronization scheme remain unchanged. The actions resulting from the reception of the checkpoints are, however, slightly different. In this variation, the receiver will maintain between OoO (“Out of Order”) and  $2 \times \text{WINDOW\_SIZE} + \text{OoO}$  active addresses ( $1 \leq \text{OoO} \leq \text{WINDOW\_SIZE}$  and  $\text{WINDOW\_SIZE} \geq 1$ ). OoO and WINDOW\_SIZE are engineerable parameters, where OoO is the minimum number of addresses needed to accommodate lost packets due to events in the network or out of order arrivals and WINDOW\_SIZE is the number of packets transmitted before a SYNC\_REQ is issued. FIG. 17 depicts a storage array for a receiver’s active addresses.

The receiver starts with the first  $2 \times \text{WINDOW\_SIZE}$  addresses loaded and active (ready to receive data). As packets are received, the corresponding entries are marked as “used” and are no longer eligible to receive packets. The transmitter maintains a packet counter, initially set to 0, containing the number of data packets transmitted since the last initial transmission of a SYNC\_REQ for which SYNC\_ACK has been received. When the transmitter packet counter equals WINDOW\_SIZE, the transmitter generates a SYNC\_REQ and does its initial transmission. When the receiver receives a SYNC\_REQ corresponding to its current CKPT\_N, it generates the next WINDOW\_SIZE addresses and starts loading them in order starting at the first location after the last active address wrapping around to the beginning of the array after the end of the array has been reached. The receiver’s array might look like FIG. 18 when a SYNC\_REQ

## US 7,490,151 B2

31

has been received. In this case a couple of packets have been either lost or will be received out of order when the SYNC\_REQ is received.

FIG. 19 shows the receiver's array after the new addresses have been generated. If the transmitter does not receive a SYNC\_ACK, it will re-issue the SYNC\_REQ at regular intervals. When the transmitter receives a SYNC\_ACK, the packet counter is decremented by WINDOW\_SIZE. If the packet counter reaches  $2 \times \text{WINDOW\_SIZE} - \text{OoO}$  then the transmitter ceases sending data packets until the appropriate SYNC\_ACK is finally received. The transmitter then resumes sending data packets. Future behavior is essentially a repetition of this initial cycle. The advantages of this approach are:

1. There is no need for an efficient jump ahead in the random number generator,
2. No packet is ever transmitted that does not have a corresponding entry in the receiver side
3. No timer based re-synchronization is necessary. This is a consequence of 2.
4. The receiver will always have the ability to accept data messages transmitted within OoO messages of the most recently transmitted message.

#### J. Distributed Transmission Path Variant

Another embodiment incorporating various inventive principles is shown in FIG. 20. In this embodiment, a message transmission system includes a first computer **2001** in communication with a second computer **2002** through a network **2011** of intermediary computers. In one variant of this embodiment, the network includes two edge routers **2003** and **2004** each of which is linked to a plurality of Internet Service Providers (ISPs) **2005** through **2010**. Each ISP is coupled to a plurality of other ISPs in an arrangement as shown in FIG. 20, which is a representative configuration only and is not intended to be limiting. Each connection between ISPs is labeled in FIG. 20 to indicate a specific physical transmission path (e.g., AD is a physical path that links ISP A (element **2005**) to ISP D (element **2008**)). Packets arriving at each edge router are selectively transmitted to one of the ISPs to which the router is attached on the basis of a randomly or quasi-randomly selected basis.

As shown in FIG. 21, computer **2001** or edge router **2003** incorporates a plurality of link transmission tables **2100** that identify, for each potential transmission path through the network, valid sets of IP addresses that can be used to transmit the packet. For example, AD table **2101** contains a plurality of IP source/destination pairs that are randomly or quasi-randomly generated. When a packet is to be transmitted from first computer **2001** to second computer **2002**, one of the link tables is randomly (or quasi-randomly) selected, and the next valid source/destination address pair from that table is used to transmit the packet through the network. If path AD is randomly selected, for example, the next source/destination IP address pair (which is pre-determined to transmit between ISP A (element **2005**) and ISP B (element **2008**)) is used to transmit the packet. If one of the transmission paths becomes degraded or inoperative, that link table can be set to a "down" condition as shown in table **2105**, thus preventing addresses from being selected from that table. Other transmission paths would be unaffected by this broken link.

### 3. CONTINUATION-IN-PART IMPROVEMENTS

The following describes various improvements and features that can be applied to the embodiments described above. The improvements include: (1) a load balancer that distrib-

32

utes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities. Each is discussed separately below.

#### A. Load Balancer

Various embodiments described above include a system in which a transmitting node and a receiving node are coupled through a plurality of transmission paths, and wherein successive packets are distributed quasi-randomly over the plurality of paths. See, for example, FIGS. 20 and 21 and accompanying description. The improvement extends this basic concept to encompass distributing packets across different paths in such a manner that the loads on the paths are generally balanced according to transmission link quality.

In one embodiment, a system includes a transmitting node and a receiving node that are linked via a plurality of transmission paths having potentially varying transmission quality. Successive packets are transmitted over the paths based on a weight value distribution function for each path. The rate that packets will be transmitted over a given path can be different for each path. The relative "health" of each transmission path is monitored in order to identify paths that have become degraded. In one embodiment, the health of each path is monitored in the transmitter by comparing the number of packets transmitted to the number of packet acknowledgements received. Each transmission path may comprise a physically separate path (e.g., via dial-up phone line, computer network, router, bridge, or the like), or may comprise logically separate paths contained within a broadband communication medium (e.g., separate channels in an FDM, TDM, CDMA, or other type of modulated or unmodulated transmission link).

When the transmission quality of a path falls below a predetermined threshold and there are other paths that can transmit packets, the transmitter changes the weight value used for that path, making it less likely that a given packet will be transmitted over that path. The weight will preferably be set no lower than a minimum value that keeps nominal traffic on the path. The weights of the other available paths are altered to compensate for the change in the affected path. When the quality of a path degrades to where the transmitter is turned off by the synchronization function (i.e., no packets are arriving at the destination), the weight is set to zero. If all transmitters are turned off, no packets are sent.

Conventional TCP/IP protocols include a "throttling" feature that reduces the transmission rate of packets when it is determined that delays or errors are occurring in transmission. In this respect, timers are sometimes used to determine whether packets have been received. These conventional techniques for limiting transmission of packets, however, do not involve multiple transmission paths between two nodes wherein transmission across a particular path relative to the others is changed based on link quality.

According to certain embodiments, in order to damp oscillations that might otherwise occur if weight distributions are changed drastically (e.g., according to a step function), a linear or an exponential decay formula can be applied to



## US 7,490,151 B2

33

gradually decrease the weight value over time that a degrading path will be used. Similarly, if the health of a degraded path improves, the weight value for that path is gradually increased.

Transmission link health can be evaluated by comparing the number of packets that are acknowledged within the transmission window (see embodiments discussed above) to the number of packets transmitted within that window and by the state of the transmitter (i.e., on or off). In other words, rather than accumulating general transmission statistics over time for a path, one specific implementation uses the “windowing” concepts described above to evaluate transmission path health.

The same scheme can be used to shift virtual circuit paths from an “unhealthy” path to a “healthy” one, and to select a path for a new virtual circuit.

FIG. 22A shows a flowchart for adjusting weight values associated with a plurality of transmission links. It is assumed that software executing in one or more computer nodes executes the steps shown in FIG. 22A. It is also assumed that the software can be stored on a computer-readable medium such as a magnetic or optical disk for execution by a computer.

Beginning in step 2201, the transmission quality of a given transmission path is measured. As described above, this measurement can be based on a comparison between the number of packets transmitted over a particular link to the number of packet acknowledgements received over the link (e.g., per unit time, or in absolute terms). Alternatively, the quality can be evaluated by comparing the number of packets that are acknowledged within the transmission window to the number of packets that were transmitted within that window. In yet another variation, the number of missed synchronization messages can be used to indicate link quality. Many other variations are of course possible.

In step 2202, a check is made to determine whether more than one transmitter (e.g., transmission path) is turned on. If not, the process is terminated and resumes at step 2201.

In step 2203, the link quality is compared to a given threshold (e.g., 50%, or any arbitrary number). If the quality falls below the threshold, then in step 2207 a check is made to determine whether the weight is above a minimum level (e.g., 1%). If not, then in step 2209 the weight is set to the minimum level and processing resumes at step 2201. If the weight is above the minimum level, then in step 2208 the weight is gradually decreased for the path, then in step 2206 the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are increased).

If in step 2203 the quality of the path was greater than or equal to the threshold, then in step 2204 a check is made to determine whether the weight is less than a steady-state value for that path. If so, then in step 2205 the weight is increased toward the steady-state value, and in step 2206 the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are decreased). If in step 2204 the weight is not less than the steady-state value, then processing resumes at step 2201 without adjusting the weights.

The weights can be adjusted incrementally according to various functions, preferably by changing the value gradually. In one embodiment, a linearly decreasing function is used to adjust the weights; according to another embodiment, an exponential decay function is used. Gradually changing the weights helps to damp oscillators that might otherwise occur if the probabilities were abruptly.

Although not explicitly shown in FIG. 22A the process can be performed only periodically (e.g., according to a time schedule), or it can be continuously run, such as in a back-

34

ground mode of operation. In one embodiment, the combined weights of all potential paths should add up to unity (e.g., when the weighting for one path is decreased, the corresponding weights that the other paths will be selected will increase).

Adjustments to weight values for other paths can be prorated. For example, a decrease of 10% in weight value for one path could result in an evenly distributed increase in the weights for the remaining paths. Alternatively, weightings could be adjusted according to a weighted formula as desired (e.g., favoring healthy paths over less healthy paths). In yet another variation, the difference in weight value can be amortized over the remaining links in a manner that is proportional to their traffic weighting.

FIG. 22B shows steps that can be executed to shut down transmission links where a transmitter turns off. In step 2210, a transmitter shut-down event occurs. In step 2211, a test is made to determine whether at least one transmitter is still turned on. If not, then in step 2215 all packets are dropped until a transmitter turns on. If in step 2211 at least one transmitter is turned on, then in step 2212 the weight for the path is set to zero, and the weights for the remaining paths are adjusted accordingly.

FIG. 23 shows a computer node 2301 employing various principles of the above-described embodiments. It is assumed that two computer nodes of the type shown in FIG. 23 communicate over a plurality of separate physical transmission paths. As shown in FIG. 23, four transmission paths X1 through X4 are defined for communicating between the two nodes. Each node includes a packet transmitter 2302 that operates in accordance with a transmit table 2308 as described above. (The packet transmitter could also operate without using the IP-hopping features described above, but the following description assumes that some form of hopping is employed in conjunction with the path selection mechanism.) The computer node also includes a packet receiver 2303 that operates in accordance with a receive table 2309, including a moving window W that moves as valid packets are received. Invalid packets having source and destination addresses that do not fall within window W are rejected.

As each packet is readied for transmission, source and destination IP addresses (or other discriminator values) are selected from transmit table 2308 according to any of the various algorithms described above, and packets containing these source/destination address pairs, which correspond to the node to which the four transmission paths are linked, are generated to a transmission path switch 2307. Switch 2307, which can comprise a software function, selects from one of the available transmission paths according to a weight distribution table 2306. For example, if the weight for path X1 is 0.2, then every fifth packet will be transmitted on path X1. A similar regime holds true for the other paths as shown. Initially, each link's weight value can be set such that it is proportional to its bandwidth, which will be referred to as its “steady-state” value.

Packet receiver 2303 generates an output to a link quality measurement function 2304 that operates as described above to determine the quality of each transmission path. (The input to packet receiver 2303 for receiving incoming packets is omitted for clarity). Link quality measurement function 2304 compares the link quality to a threshold for each transmission link and, if necessary, generates an output to weight adjustment function 2305. If a weight adjustment is required, then the weights in table 2306 are adjusted accordingly, preferably according to a gradual (e.g., linearly or exponentially declining) function. In one embodiment, the weight values for all



US 7,490,151 B2

35

available paths are initially set to the same value, and only when paths degrade in quality are the weights changed to reflect differences.

Link quality measurement function 2304 can be made to operate as part of a synchronizer function as described above. That is, if resynchronization occurs and the receiver detects that synchronization has been lost (e.g., resulting in the synchronization window W being advanced out of sequence), that fact can be used to drive link quality measurement function 2304. According to one embodiment, load balancing is performed using information garnered during the normal synchronization, augmented slightly to communicate link health from the receiver to the transmitter. The receiver maintains a count, MESS\_R(W), of the messages received in synchronization window W. When it receives a synchronization request (SYNC\_REQ) corresponding to the end of window W, the receiver includes counter MESS\_R in the resulting synchronization acknowledgement (SYNC\_ACK) sent back to the transmitter. This allows the transmitter to compare messages sent to messages received in order to assess the health of the link.

If synchronization is completely lost, weight adjustment function 2305 decreases the weight value on the affected path to zero. When synchronization is regained, the weight value for the affected path is gradually increased to its original value. Alternatively, link quality can be measured by evaluating the length of time required for the receiver to acknowledge a synchronization request. In one embodiment, separate transmit and receive tables are used for each transmission path.

When the transmitter receives a SYNC\_ACK, the MESS\_R is compared with the number of messages transmitted in a window (MESS\_T). When the transmitter receives a SYNC\_ACK, the traffic probabilities will be examined and adjusted if necessary. MESS\_R is compared with the number of messages transmitted in a window (MESS\_T). There are two possibilities:

1. If MESS\_R is less than a threshold value, THRESH, then the link will be deemed to be unhealthy. If the transmitter was turned off, the transmitter is turned on and the weight P for that link will be set to a minimum value MIN. This will keep a trickle of traffic on the link for monitoring purposes until it recovers. If the transmitter was turned on, the weight P for that link will be set to:

$$P' = \alpha \times \text{MIN} + (1 - \alpha) \times P \quad (1)$$

Equation 1 will exponentially damp the traffic weight value to MIN during sustained periods of degraded service.

2. If MESS\_R for a link is greater than or equal to THRESH, the link will be deemed healthy. If the weight P for that link is greater than or equal to the steady state value S for that link, then P is left unaltered. If the weight P for that link is less than THRESH then P will be set to:

$$P' = \beta \times S + (1 - \beta) \times P \quad (2)$$

where  $\beta$  is a parameter such that  $0 < \beta < 1$  that determines the damping rate of P.

Equation 2 will increase the traffic weight to S during sustained periods of acceptable service in a damped exponential fashion.

A detailed example will now be provided with reference to FIG. 24. As shown in FIG. 24, a first computer 2401 communicates with a second computer 2402 through two routers 2403 and 2404. Each router is coupled to the other router

36

through three transmission links. As described above, these may be physically diverse links or logical links (including virtual private networks).

Suppose that a first link L1 can sustain a transmission bandwidth of 100 Mb/s and has a window size of 32; link L2 can sustain 75 Mb/s and has a window size of 24; and link L3 can sustain 25 Mb/s and has a window size of 8. The combined links can thus sustain 200Mb/s. The steady state traffic weights are 0.5 for link L1; 0.375 for link L2, and 0.125 for link L3. MIN=1Mb/s, THRESH=0.8 MESS\_T for each link,  $\alpha=0.75$  and  $\beta=0.5$ . These traffic weights will remain stable until a link stops for synchronization or reports a number of packets received less than its THRESH. Consider the following sequence of events:

1. Link L1 receives a SYNC\_ACK containing a MESS\_R of 24, indicating that only 75% of the MESS\_T (32) messages transmitted in the last window were successfully received. Link L1 would be below THRESH (0.8). Consequently, link L1's traffic weight value would be reduced to 0.12825, while link L2's traffic weight value would be increased to 0.65812 and link L3's traffic weight value would be increased to 0.217938.
2. Link L2 and L3 remained healthy and link L1 stopped to synchronize. Then link L1's traffic weight value would be set to 0, link L2's traffic weight value would be set to 0.75, and link L3's traffic weight value would be set to 0.25.
3. Link L1 finally received a SYNC\_ACK containing a MESS\_R of 0 indicating that none of the MESS\_T (32) messages transmitted in the last window were successfully received. Link L1 would be below THRESH. Link L1's traffic weight value would be increased to 0.005, link L2's traffic weight value would be decreased to 0.74625, and link L3's traffic weight value would be decreased to 0.24875.
4. Link L1 received a SYNC\_ACK containing a MESS\_R of 32 indicating that 100% of the MESS\_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.2525, while link L2's traffic weight value would be decreased to 0.560625 and link L3's traffic weight value would be decreased to 0.186875.
5. Link L1 received a SYNC\_ACK containing a MESS\_R of 32 indicating that 100% of the MESS\_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.37625; link L2's traffic weight value would be decreased to 0.4678125, and link L3's traffic weight value would be decreased to 0.1559375.
6. Link L1 remains healthy and the traffic probabilities approach their steady state traffic probabilities.

#### B. Use of a DNS Proxy to Transparently Create Virtual Private Networks

A second improvement concerns the automatic creation of a virtual private network (VPN) in response to a domain-name server look-up function.

Conventional Domain Name Servers (DNSs) provide a look-up function that returns the IP address of a requested computer or host. For example, when a computer user types in the web name "Yahoo.com," the user's web browser transmits a request to a DNS, which converts the name into a four-part IP address that is returned to the user's browser and then used by the browser to contact the destination web site.

## US 7,490,151 B2

37

This conventional scheme is shown in FIG. 25. A user's computer 2501 includes a client application 2504 (for example, a web browser) and an IP protocol stack 2505. When the user enters the name of a destination host, a request DNS REQ is made (through IP protocol stack 2505) to a DNS 2502 to look up the IP address associated with the name. The DNS returns the IP address DNS RESP to client application 2504, which is then able to use the IP address to communicate with the host 2503 through separate transactions such as PAGE REQ and PAGE RESP.

In the conventional architecture shown in FIG. 25, nefarious listeners on the Internet could intercept the DNS REQ and DNS RESP packets and thus learn what IP addresses the user was contacting. For example, if a user wanted to set up a secure communication path with a web site having the name "Target.com," when the user's browser contacted a DNS to find the IP address for that web site, the true IP address of that web site would be revealed over the Internet as part of the DNS inquiry. This would hamper anonymous communications on the Internet.

One conventional scheme that provides secure virtual private networks over the Internet provides the DNS server with the public keys of the machines that the DNS server has the addresses for. This allows hosts to retrieve automatically the public keys of a host that the host is to communicate with so that the host can set up a VPN without having the user enter the public key of the destination host. One implementation of this standard is presently being developed as part of the FreeS/WAN project(RFC 2535).

The conventional scheme suffers from certain drawbacks. For example, any user can perform a DNS request. Moreover, DNS requests resolve to the same value for all users.

According to certain aspects of the invention, a specialized DNS server traps DNS requests and, if the request is from a special type of user (e.g., one for which secure communication services are defined), the server does not return the true IP address of the target node, but instead automatically sets up a virtual private network between the target node and the user. The VPN is preferably implemented using the IP address "hopping" features of the basic invention described above, such that the true identity of the two nodes cannot be determined even if packets during the communication are intercepted. For DNS requests that are determined to not require secure services (e.g., an unregistered user), the DNS server transparently "passes through" the request to provide a normal look-up function and return the IP address of the target web server, provided that the requesting host has permissions to resolve unsecured sites. Different users who make an identical DNS request could be provided with different results.

FIG. 26 shows a system employing various principles summarized above. A user's computer 2601 includes a conventional client (e.g., a web browser) 2605 and an IP protocol stack 2606 that preferably operates in accordance with an IP hopping function 2607 as outlined above. A modified DNS server 2602 includes a conventional DNS server function 2609 and a DNS proxy 2610. A gatekeeper server 2603 is interposed between the modified DNS server and a secure target site 2704. An "unsecure" target site 2611 is also accessible via conventional IP protocols.

According to one embodiment, DNS proxy 2610 intercepts all DNS lookup functions from client 2605 and determines whether access to a secure site has been requested. If access to a secure site has been requested (as determined, for example, by a domain name extension, or by reference to an internal table of such sites), DNS proxy 2610 determines whether the user has sufficient security privileges to access the site. If so, DNS proxy 2610 transmits a message to gatekeeper 2603

38

requesting that a virtual private network be created between user computer 2601 and secure target site 2604. In one embodiment, gatekeeper 2603 creates "hopblocks" to be used by computer 2601 and secure target site 2604 for secure communication. Then, gatekeeper 2603 communicates these to user computer 2601. Thereafter, DNS proxy 2610 returns to user computer 2601 the resolved address passed to it by the gatekeeper (this address could be different from the actual target computer) 2604, preferably using a secure administrative VPN. The address that is returned need not be the actual address of the destination computer.

Had the user requested lookup of a non-secure web site such as site 2611, DNS proxy would merely pass through to conventional DNS server 2609 the look-up request, which would be handled in a conventional manner, returning the IP address of non-secure web site 2611. If the user had requested lookup of a secure web site but lacked credentials to create such a connection, DNS proxy 2610 would return a "host unknown" error to the user. In this manner, different users requesting access to the same DNS name could be provided with different look-up results.

Gatekeeper 2603 can be implemented on a separate computer (as shown in FIG. 26) or as a function within modified DNS server 2602. In general, it is anticipated that gatekeeper 2703 facilitates the allocation and exchange of information needed to communicate securely, such as using "hopped" IP addresses. Secure hosts such as site 2604 are assumed to be equipped with a secure communication function such as an IP hopping function 2608.

It will be appreciated that the functions of DNS proxy 2610 and DNS server 2609 can be combined into a single server for convenience. Moreover, although element 2602 is shown as combining the functions of two servers, the two servers can be made to operate independently.

FIG. 27 shows steps that can be executed by DNS proxy server 2610 to handle requests for DNS look-up for secure hosts. In step 2701, a DNS look-up request is received for a target host. In step 2702, a check is made to determine whether access to a secure host was requested. If not, then in step 2703 the DNS request is passed to conventional DNS server 2609, which looks up the IP address of the target site and returns it to the user's application for further processing.

In step 2702, if access to a secure host was requested, then in step 2704 a further check is made to determine whether the user is authorized to connect to the secure host. Such a check can be made with reference to an internally stored list of authorized IP addresses, or can be made by communicating with gatekeeper 2603 (e.g., over an "administrative" VPN that is secure). It will be appreciated that different levels of security can also be provided for different categories of hosts. For example, some sites may be designated as having a certain security level, and the security level of the user requesting access must match that security level. The user's security level can also be determined by transmitting a request message back to the user's computer requiring that it prove that it has sufficient privileges.

If the user is not authorized to access the secure site, then a "host unknown" message is returned (step 2705). If the user has sufficient security privileges, then in step 2706 a secure VPN is established between the user's computer and the secure target site. As described above, this is preferably done by allocating a hopping regime that will be carried out between the user's computer and the secure target site, and is preferably performed transparently to the user (i.e., the user need not be involved in creating the secure link). As described in various embodiments of this application, any of various

fields can be “hopped” (e.g., IP source/destination addresses; a field in the header; etc.) in order to communicate securely.

Some or all of the security functions can be embedded in gatekeeper **2603**, such that it handles all requests to connect to secure sites. In this embodiment, DNS proxy **2610** communicates with gatekeeper **2603** to determine (preferably over a secure administrative VPN) whether the user has access to a particular web site. Various scenarios for implementing these features are described by way of example below:

Scenario #1: Client has permission to access target computer, and gatekeeper has a rule to make a VPN for the client. In this scenario, the client’s DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would establish a VPN between the client and the requested target. The gatekeeper would provide the address of the destination to the DNS proxy, which would then return the resolved name as a result. The resolved address can be transmitted back to the client in a secure administrative VPN.

Scenario #2: Client does not have permission to access target computer. In this scenario, the client’s DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would reject the request, informing DNS proxy server **2610** that it was unable to find the target computer. The DNS proxy **2610** would then return a “host unknown” error message to the client.

Scenario #3: Client has permission to connect using a normal non-VPN link, and the gatekeeper does not have a rule to set up a VPN for the client to the target site. In this scenario, the client’s DNS request is received by DNS proxy server **2610**, which would check its rules and determine that no VPN is needed. Gatekeeper **2603** would then inform the DNS proxy server to forward the request to conventional DNS server **2609**, which would resolve the request and return the result to the DNS proxy server and then back to the client.

Scenario #4: Client does not have permission to establish a normal/non-VPN link, and the gatekeeper does not have a rule to make a VPN for the client to the target site. In this scenario, the DNS proxy server would receive the client’s DNS request and forward it to gatekeeper **2603**. Gatekeeper **2603** would determine that no special VPN was needed, but that the client is not authorized to communicate with non-VPN members. The gatekeeper would reject the request, causing DNS proxy server **2610** to return an error message to the client.

### C. Large Link to Small Link Bandwidth Management

One feature of the basic architecture is the ability to prevent so-called “denial of service” attacks that can occur if a computer hacker floods a known Internet node with packets, thus preventing the node from communicating with other nodes. Because IP addresses or other fields are “hopped” and packets arriving with invalid addresses are quickly discarded, Internet nodes are protected against flooding targeted at a single IP address.

In a system in which a computer is coupled through a link having a limited bandwidth (e.g., an edge router) to a node that can support a much higher-bandwidth link (e.g., an Internet Service Provider), a potential weakness could be exploited by a determined hacker. Referring to FIG. **28**, suppose that a first host computer **2801** is communicating with a second host computer **2804** using the IP address hopping principles described above. The first host computer is coupled through an edge router **2802** to an Internet Service Provider

(ISP) **2803** through a low bandwidth link (LOW BW), and is in turn coupled to second host computer **2804** through parts of the Internet through a high bandwidth link (HIGH BW). In this architecture, the ISP is able to support a high bandwidth to the internet, but a much lower bandwidth to the edge router **2802**.

Suppose that a computer hacker is able to transmit a large quantity of dummy packets addressed to first host computer **2801** across high bandwidth link HIGH BW. Normally, host computer **2801** would be able to quickly reject the packets since they would not fall within the acceptance window permitted by the IP address hopping scheme. However, because the packets must travel across low bandwidth link LOW BW, the packets overwhelm the lower bandwidth link before they are received by host computer **2801**. Consequently, the link to host computer **2801** is effectively flooded before the packets can be discarded.

According to one inventive improvement, a “link guard” function **2805** is inserted into the high-bandwidth node (e.g., ISP **2803**) that quickly discards packets destined for a low-bandwidth target node if they are not valid packets. Each packet destined for a low-bandwidth node is cryptographically authenticated to determine whether it belongs to a VPN. If it is not a valid VPN packet, the packet is discarded at the high-bandwidth node. If the packet is authenticated as belonging to a VPN, the packet is passed with high preference. If the packet is a valid non-VPN packet, it is passed with a lower quality of service (e.g., lower priority).

In one embodiment, the ISP distinguishes between VPN and non-VPN packets using the protocol of the packet. In the case of IPSEC [rfc **2401**], the packets have IP protocols **420** and **421**. In the case of the TARP VPN, the packets will have an IP protocol that is not yet defined. The ISP’s link guard, **2805**, maintains a table of valid VPNs which it uses to validate whether VPN packets are cryptographically valid.

According to one embodiment, packets that do not fall within any hop windows used by nodes on the low-bandwidth link are rejected, or are sent with a lower quality of service. One approach for doing this is to provide a copy of the IP hopping tables used by the low-bandwidth nodes to the high-bandwidth node, such that both the high-bandwidth and low-bandwidth nodes track hopped packets (e.g., the high-bandwidth node moves its hopping window as valid packets are received). In such a scenario, the high-bandwidth node discards packets that do not fall within the hopping window before they are transmitted over the low-bandwidth link. Thus, for example, ISP **2903** maintains a copy **2910** of the receive table used by host computer **2901**. Incoming packets that do not fall within this receive table are discarded. According to a different embodiment, link guard **2805** validates each VPN packet using a keyed hashed message authentication code (HMAC) [rfc **2104**]. According to another embodiment, separate VPNs (using, for example, hopblocks) can be established for communicating between the low-bandwidth node and the high-bandwidth node (i.e., packets arriving at the high-bandwidth node are converted into different packets before being transmitted to the low-bandwidth node).

As shown in FIG. **29**, for example, suppose that a first host computer **2900** is communicating with a second host computer **2902** over the Internet, and the path includes a high bandwidth link HIGH BW to an ISP **2901** and a low bandwidth link LOW BW through an edge router **2904**. In accordance with the basic architecture described above, first host computer **2900** and second host computer **2902** would exchange hopblocks (or a hopblock algorithm) and would be able to create matching transmit and receive tables **2905**, **2906**, **2912** and **2913**. Then in accordance with the basic

## US 7,490,151 B2

41

architecture, the two computers would transmit packets having seemingly random IP source and destination addresses, and each would move a corresponding hopping window in its receive table as valid packets were received.

Suppose that a nefarious computer hacker **2903** was able to deduce that packets having a certain range of IP addresses (e.g., addresses 100 to 200 for the sake of simplicity) are being transmitted to ISP **2901**, and that these packets are being forwarded over a low-bandwidth link. Hacker computer **2903** could thus “flood” packets having addresses falling into the range 100 to 200, expecting that they would be forwarded along low bandwidth link LOW BW, thus causing the low bandwidth link to become overwhelmed. The fast packet reject mechanism in first host computer **3000** would be of little use in rejecting these packets, since the low bandwidth link was effectively jammed before the packets could be rejected. In accordance with one aspect of the improvement, however, VPN link guard **2911** would prevent the attack from impacting the performance of VPN traffic because the packets would either be rejected as invalid VPN packets or given a lower quality of service than VPN traffic over the lower bandwidth link. A denial-of-service flood attack could, however, still disrupt non-VPN traffic.

According to one embodiment of the improvement, ISP **2901** maintains a separate VPN with first host computer **2900**, and thus translates packets arriving at the ISP into packets having a different IP header before they are transmitted to host computer **2900**. The cryptographic keys used to authenticate VPN packets at the link guard **2911** and the cryptographic keys used to encrypt and decrypt the VPN packets at host **2902** and host **2901** can be different, so that link guard **2911** does not have access to the private host data; it only has the capability to authenticate those packets.

According to yet a third embodiment, the low-bandwidth node can transmit a special message to the high-bandwidth node instructing it to shut down all transmissions on a particular IP address, such that only hopped packets will pass through to the low-bandwidth node. This embodiment would prevent a hacker from flooding packets using a single IP address. According to yet a fourth embodiment, the high-bandwidth node can be configured to discard packets transmitted to the low-bandwidth node if the transmission rate exceeds a certain predetermined threshold for any given IP address; this would allow hopped packets to go through. In this respect, link guard **2911** can be used to detect that the rate of packets on a given IP address are exceeding a threshold rate; further packets addressed to that same IP address would be dropped or transmitted at a lower priority (e.g., delayed).

## D. Traffic Limiter

In a system in which multiple nodes are communicating using “hopping” technology, a treasonous insider could internally flood the system with packets. In order to prevent this possibility, one inventive improvement involves setting up “contracts” between nodes in the system, such that a receiver can impose a bandwidth limitation on each packet sender. One technique for doing this is to delay acceptance of a checkpoint synchronization request from a sender until a certain time period (e.g., one minute) has elapsed. Each receiver can effectively control the rate at which its hopping window moves by delaying “SYNC ACK” responses to “SYNC\_REQ” messages.

A simple modification to the checkpoint synchronizer will serve to protect a receiver from accidental or deliberate overload from an internally treasonous client. This modification is based on the observation that a receiver will not update its

42

tables until a SYNC\_REQ is received on hopped address CKPT\_N. It is a simple matter of deferring the generation of a new CKPT\_N until an appropriate interval after previous checkpoints.

Suppose a receiver wished to restrict reception from a transmitter to 100 packets a second, and that checkpoint synchronization messages were triggered every 50 packets. A compliant transmitter would not issue new SYNC\_REQ messages more often than every 0.5 seconds. The receiver could delay a non-compliant transmitter from synchronizing by delaying the issuance of CKPT\_N for 0.5 second after the last SYNC\_REQ was accepted.

In general, if M receivers need to restrict N transmitters issuing new SYNC\_REQ messages after every W messages to sending R messages a second in aggregate, each receiver could defer issuing a new CKPT\_N until  $M \times N \times W/R$  seconds have elapsed since the last SYNC\_REQ has been received and accepted. If the transmitter exceeds this rate between a pair of checkpoints, it will issue the new checkpoint before the receiver is ready to receive it, and the SYNC\_REQ will be discarded by the receiver. After this, the transmitter will re-issue the SYNC\_REQ every T1 seconds until it receives a SYNC\_ACK. The receiver will eventually update CKPT\_N and the SYNC\_REQ will be acknowledged. If the transmission rate greatly exceeds the allowed rate, the transmitter will stop until it is compliant. If the transmitter exceeds the allowed rate by a little, it will eventually stop after several rounds of delayed synchronization until it is in compliance. Hacking the transmitter’s code to not shut off only permits the transmitter to lose the acceptance window. In this case it can recover the window and proceed only after it is compliant again.

Two practical issues should be considered when implementing the above scheme:

1. The receiver rate should be slightly higher than the permitted rate in order to allow for statistical fluctuations in traffic arrival times and non-uniform load balancing.

2. Since a transmitter will rightfully continue to transmit for a period after a SYNC\_REQ is transmitted, the algorithm above can artificially reduce the transmitter’s bandwidth. If events prevent a compliant transmitter from synchronizing for a period (e.g. the network dropping a SYNC\_REQ or a SYNC\_ACK) a SYNC\_REQ will be accepted later than expected. After this, the transmitter will transmit fewer than expected messages before encountering the next checkpoint. The new checkpoint will not have been activated and the transmitter will have to retransmit the SYNC\_REQ. This will appear to the receiver as if the transmitter is not compliant. Therefore, the next checkpoint will be accepted late from the transmitter’s perspective. This has the effect of reducing the transmitter’s allowed packet rate until the transmitter transmits at a packet rate below the agreed upon rate for a period of time.

To guard against this, the receiver should keep track of the times that the last C SYNC\_REQs were received and accepted and use the minimum of  $M \times N \times W/R$  seconds after the last SYNC\_REQ has been received and accepted,  $2 \times M \times N \times W/R$  seconds after next to the last SYNC\_REQ has been received and accepted,  $C \times M \times N \times W/R$  seconds after  $(C-1)^{th}$  to the last SYNC\_REQ has been received, as the time to activate CKPT\_N. This prevents the receiver from inappropriately limiting the transmitter’s packet rate if at least one out of the last C SYNC\_REQs was processed on the first attempt.

FIG. 30 shows a system employing the above-described principles. In FIG. 30, two computers **3000** and **3001** are assumed to be communicating over a network N in accordance with the “hopping” principles described above (e.g.,

US 7,490,151 B2

43

hopped IP addresses, discriminator values, etc.). For the sake of simplicity, computer 3000 will be referred to as the receiving computer and computer 3001 will be referred to as the transmitting computer, although full duplex operation is of course contemplated. Moreover, although only a single transmitter is shown, multiple transmitters can transmit to receiver 3000.

As described above, receiving computer 3000 maintains a receive table 3002 including a window W that defines valid IP address pairs that will be accepted when appearing in incoming data packets. Transmitting computer 3001 maintains a transmit table 3003 from which the next IP address pairs will be selected when transmitting a packet to receiving computer 3000. (For the sake of illustration, window W is also illustrated with reference to transmit table 3003). As transmitting computer moves through its table, it will eventually generate a SYNC\_REQ message as illustrated in function 3010. This is a request to receiver 3000 to synchronize the receive table 3002, from which transmitter 3001 expects a response in the form of a CKPT\_N (included as part of a SYNC\_ACK message). If transmitting computer 3001 transmits more messages than its allotment, it will prematurely generate the SYNC\_REQ message. (If it has been altered to remove the SYNC\_REQ message generation altogether, it will fall out of synchronization since receiver 3000 will quickly reject packets that fall outside of window W, and the extra packets generated by transmitter 3001 will be discarded).

In accordance with the improvements described above, receiving computer 3000 performs certain steps when a SYNC\_REQ message is received, as illustrated in FIG. 30. In step 3004, receiving computer 3000 receives the SYNC\_REQ message. In step 3005, a check is made to determine whether the request is a duplicate. If so, it is discarded in step 3006. In step 3007, a check is made to determine whether the SYNC\_REQ received from transmitter 3001 was received at a rate that exceeds the allowable rate R (i.e., the period between the time of the last SYNC\_REQ message). The value R can be a constant, or it can be made to fluctuate as desired. If the rate exceeds R, then in step 3008 the next activation of the next CKPT\_N hopping table entry is delayed by W/R seconds after the last SYNC\_REQ has been accepted.

Otherwise, if the rate has not been exceeded, then in step 3109 the next CKPT\_N value is calculated and inserted into the receiver's hopping table prior to the next SYNC\_REQ from the transmitter 3101. Transmitter 3101 then processes the SYNC\_REQ in the normal manner.

#### E. Signaling Synchronizer

In a system in which a large number of users communicate with a central node using secure hopping technology, a large amount of memory must be set aside for hopping tables and their supporting data structures. For example, if one million subscribers to a web site occasionally communicate with the web site, the site must maintain one million hopping tables, thus using up valuable computer resources, even though only a small percentage of the users may actually be using the system at any one time. A desirable solution would be a system that permits a certain maximum number of simultaneous links to be maintained, but which would "recognize" millions of registered users at any one time. In other words, out of a population of a million registered users, a few thousand at a time could simultaneously communicate with a central server, without requiring that the server maintain one million hopping tables of appreciable size.

One solution is to partition the central node into two nodes: a signaling server that performs session initiation for user

44

log-on and log-off (and requires only minimally sized tables), and a transport server that contains larger hopping tables for the users. The signaling server listens for the millions of known users and performs a fast-packet reject of other (bogus) packets. When a packet is received from a known user, the signaling server activates a virtual private link (VPL) between the user and the transport server, where hopping tables are allocated and maintained. When the user logs onto the signaling server, the user's computer is provided with hopping tables for communicating with the transport server, thus activating the VPL. The VPLs can be torn down when they become inactive for a time period, or they can be torn down upon user log-out. Communication with the signaling server to allow user log-on and log-off can be accomplished using a specialized version of the checkpoint scheme described above.

FIG. 31 shows a system employing certain of the above-described principles. In FIG. 31, a signaling server 3101 and a transport server 3102 communicate over a link. Signaling server 3101 contains a large number of small tables 3106 and 3107 that contain enough information to authenticate a communication request with one or more clients 3103 and 3104. As described in more detail below, these small tables may advantageously be constructed as a special case of the synchronizing checkpoint tables described previously. Transport server 3102, which is preferably a separate computer in communication with signaling server 3101, contains a smaller number of larger hopping tables 3108, 3109, and 3110 that can be allocated to create a VPN with one of the client computers.

According to one embodiment, a client that has previously registered with the system (e.g., via a system administration function, a user registration procedure, or some other method) transmits a request for information from a computer (e.g., a web site). In one variation, the request is made using a "hopped" packet, such that signaling server 3101 will quickly reject invalid packets from unauthorized computers such as hacker computer 3105. An "administrative" VPN can be established between all of the clients and the signaling server in order to ensure that a hacker cannot flood signaling server 3101 with bogus packets. Details of this scheme are provided below.

Signaling server 3101 receives the request 3111 and uses it to determine that client 3103 is a validly registered user. Next, signaling server 3101 issues a request to transport server 3102 to allocate a hopping table (or hopping algorithm or other regime) for the purpose of creating a VPN with client 3103. The allocated hopping parameters are returned to signaling server 3101 (path 3113), which then supplies the hopping parameters to client 3103 via path 3114, preferably in encrypted form.

Thereafter, client 3103 communicates with transport server 3102 using the normal hopping techniques described above. It will be appreciated that although signaling server 3101 and transport server 3102 are illustrated as being two separate computers, they could of course be combined into a single computer and their functions performed on the single computer. Alternatively, it is possible to partition the functions shown in FIG. 31 differently from as shown without departing from the inventive principles.

One advantage of the above-described architecture is that signaling server 3101 need only maintain a small amount of information on a large number of potential users, yet it retains the capability of quickly rejecting packets from unauthorized users such as hacker computer 3105. Larger data tables needed to perform the hopping and synchronization functions are instead maintained in a transport server 3102, and a

## US 7,490,151 B2

45

smaller number of these tables are needed since they are only allocated for “active” links. After a VPN has become inactive for a certain time period (e.g., one hour), the VPN can be automatically torn down by transport server **3102** or signaling server **3101**.

A more detailed description will now be provided regarding how a special case of the checkpoint synchronization feature can be used to implement the signaling scheme described above.

The signaling synchronizer may be required to support many (millions) of standing, low bandwidth connections. It therefore should minimize per-VPL memory usage while providing the security offered by hopping technology. In order to reduce memory usage in the signaling server, the data hopping tables can be completely eliminated and data can be carried as part of the SYNC\_REQ message. The table used by the server side (receiver) and client side (transmitter) is shown schematically as element **3106** in FIG. **31**.

The meaning and behaviors of CKPT\_N, CKPT\_O and CKPT\_R remain the same from the previous description, except that CKPT\_N can receive a combined data and SYNC\_REQ message or a SYNC\_REQ message without the data.

The protocol is a straightforward extension of the earlier synchronizer. Assume that a client transmitter is on and the tables are synchronized. The initial tables can be generated “out of band.” For example, a client can log into a web server to establish an account over the Internet. The client will receive keys etc encrypted over the Internet. Meanwhile, the server will set up the signaling VPN on the signaling server.

Assuming that a client application wishes to send a packet to the server on the client’s standing signaling VPL:

1. The client sends the message marked as a data message on the inner header using the transmitter’s CKPT\_N address. It turns the transmitter off and starts a timer T1 noting CKPT\_O. Messages can be one of three types: DATA, SYNC\_REQ and SYNC\_ACK. In the normal algorithm, some potential problems can be prevented by identifying each message type as part of the encrypted inner header field. In this algorithm, it is important to distinguish a data packet and a SYNC\_REQ in the signaling synchronizer since the data and the SYNC\_REQ come in on the same address.
2. When the server receives a data message on its CKPT\_N, it verifies the message and passes it up the stack. The message can be verified by checking message type and other information (i.e user credentials) contained in the inner header. It replaces its CKPT\_O with CKPT\_N and generates the next CKPT\_N. It updates its transmitter side CKPT\_R to correspond to the client’s receiver side CKPT\_R and transmits a SYNC\_ACK containing CKPT\_O in its payload.
3. When the client side receiver receives a SYNC\_ACK on its CKPT\_R with a payload matching its transmitter side CKPT\_O and the transmitter is off, the transmitter is turned on and the receiver side CKPT\_R is updated. If the SYNC\_ACK’s payload does not match the transmitter side CKPT\_O or the transmitter is on, the SYNC\_ACK is simply discarded.
4. T1 expires: If the transmitter is off and the client’s transmitter side CKPT\_O matches the CKPT\_O associated with the timer, it starts timer T1 noting CKPT\_O again, and a SYNC\_REQ is sent using the transmitter’s CKPT\_O address. Otherwise, no action is taken.
5. When the server receives a SYNC\_REQ on its CKPT\_N, it replaces its CKPT\_O with CKPT\_N and generates the next CKPT\_N. It updates its transmitter side CKPT\_R to

46

correspond to the client’s receiver side CKPT\_R and transmits a SYNC\_ACK containing CKPT\_O in its payload.

6. When the server receives a SYNC\_REQ on its CKPT\_O, it updates its transmitter side CKPT\_R to correspond to the client’s receiver side CKPT\_R and transmits a SYNC\_ACK containing CKPT\_O in its payload.

FIG. **32** shows message flows to highlight the protocol. Reading from top to bottom, the client sends data to the server using its transmitter side CKPT\_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT\_N into CKPT\_O and updates CKPT\_N. This message is successfully received and a passed up the stack. It also synchronizes the receiver i.e, the server loads CKPT\_N into CKPT\_O and generates a new CKPT\_N, it generates a new CKPT\_R in the server side transmitter and transmits a SYNC\_ACK containing the server side receiver’s CKPT\_O the server. The SYNC\_ACK is successfully received at the client. The client side receiver’s CKPT\_R is updated, the transmitter is turned on and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

Next, the client sends data to the server using its transmitter side CKPT\_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT\_N into CKPT\_O and updates CKPT\_N. This message is lost. The client side timer expires and as a result a SYNC\_REQ is transmitted on the client side transmitter’s CKPT\_O (this will keep happening until the SYNC\_ACK has been received at the client). The SYNC\_REQ is successfully received at the server. It synchronizes the receiver i.e, the server loads CKPT\_N into CKPT\_O and generates a new CKPT\_N, it generates a new CKPT\_R in the server side transmitter and transmits a SYNC\_ACK containing the server side receiver’s CKPT\_O the server. The SYNC\_ACK is successfully received at the client. The client side receiver’s CKPT\_R is updated, the transmitter is turned off and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

There are numerous other scenarios that follow this flow. For example, the SYNC\_ACK could be lost. The transmitter would continue to re-send the SYNC\_REQ until the receiver synchronizes and responds.

The above-described procedures allow a client to be authenticated at signaling server **3201** while maintaining the ability of signaling server **3201** to quickly reject invalid packets, such as might be generated by hacker computer **3205**. In various embodiments, the signaling synchronizer is really a derivative of the synchronizer. It provides the same protection as the hopping protocol, and it does so for a large number of low bandwidth connections.

We claim:

1. A data processing device, comprising memory storing a domain name server (DNS) proxy module that intercepts DNS requests sent by a client and, for each intercepted DNS request, performs the steps of:
  - (i) determining whether the intercepted DNS request corresponds to a secure server;
  - (ii) when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer, and
  - (iii) when the intercepted DNS request corresponds to a secure server, automatically initiating an encrypted channel between the client and the secure server.

## US 7,490,151 B2

47

2. The data processing device of claim 1, wherein step (iii) comprises the steps of:

- (a) determining whether the client is authorized to access the secure server; and
- (b) when the client is authorized to access the secure server, sending a request to the secure server to establish an encrypted channel between the secure server and the client.

3. The data processing device of claim 2, wherein step (iii) further comprises the step of:

- (c) when the client is not authorized to access the secure server, returning a host unknown error message to the client.

4. The data processing device of claim 3, wherein the client comprises a web browser into which a user enters a URL resulting in the DNS request.

5. The data processing device of claim 1, wherein automatically initiating the encrypted channel between the client and the secure server comprises establishing an IP address hopping scheme between the client and the secure server.

6. The data processing device of claim 1, wherein automatically initiating the encrypted channel between the client and the secure server avoids sending a true IP address of the secure server to the client.

7. A computer readable medium storing a domain name server (DNS) proxy module comprised of computer readable instructions that, when executed, cause a data processing device to perform the steps of:

- (i) intercepting a DNS request sent by a client;
- (ii) determining whether the intercepted DNS request corresponds to a secure server;
- (iii) when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer; and
- (iv) when the intercepted DNS request corresponds to a secure server, automatically initiating an encrypted channel between the client and the secure server.

8. The computer readable medium of claim 7, wherein step (iv) comprises the steps of

- (a) determining whether the client is authorized to access the secure server, and
- (b) when the client is authorized to access the secure server, sending a request to the secure server to establish an encrypted channel between the secure server and the client.

48

9. The computer readable medium of claim 8, wherein step (iv) further comprises the step of:

- (c) when the client is not authorized to access the secure server, returning a host unknown error message to the client.

10. The computer readable medium of claim 9, wherein the client comprises a web browser into which a user enters a URL resulting in the DNS request.

11. The computer readable medium of claim 7, wherein automatically initiating the encrypted channel between the client and the secure server comprises establishing an IP address hopping scheme between the client and the secure server.

12. The computer readable medium of claim 7, wherein automatically initiating the encrypted channel between the client and the secure server avoids sending a true IP address of the secure server to the client.

13. A computer readable medium storing a domain name server (DNS) module comprised of computer readable instructions that, when executed, cause a data processing device to perform the steps of:

- (i) determining whether a DNS request sent by a client corresponds to a secure server;
- (ii) when the DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer; and
- (iii) when the intercepted DNS request corresponds to a secure server, automatically creating a secure channel between the client and the secure server.

14. The computer readable medium of claim 13, wherein step (iii) comprises the steps of

- (a) determining whether the client is authorized to access the secure server; and
- (b) when the client is authorized to access the secure server, sending a request to the secure server to establish a secure channel between the secure server and the client.

15. The computer readable medium of claim 14, wherein step (iii) further comprises the step of:

- (c) when the client is not authorized to access the secure server, returning a host unknown error message to the client.

16. The computer readable medium of claim 15, wherein the client comprises a web browser into which a user enters a URL resulting in the DNS request.

\* \* \* \* \*

(12) **United States Patent**  
**Larson et al.**

(10) **Patent No.:** **US 7,921,211 B2**  
(45) **Date of Patent:** **\*Apr. 5, 2011**

(54) **AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES**

(75) Inventors: **Victor Larson**, Fairfax, VA (US);  
**Robert Dunham Short, III**, Leesburg, VA (US); **Edmund Colby Munger**,  
Crownsville, MD (US); **Michael Williamson**, South Riding, VA (US)

(73) Assignee: **VirnetX, Inc.**, Scotts Valley, CA (US)

(\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 701 days.  
  
This patent is subject to a terminal disclaimer.

(21) Appl. No.: **11/840,560**

(22) Filed: **Aug. 17, 2007**

(65) **Prior Publication Data**

US 2008/0040792 A1 Feb. 14, 2008

**Related U.S. Application Data**

(63) Continuation of application No. 10/714,849, filed on Nov. 18, 2003, now Pat. No. 7,418,504, which is a continuation of application No. 09/558,210, filed on Apr. 26, 2000, now abandoned, which is a continuation-in-part of application No. 09/504,783, filed on Feb. 15, 2000, now Pat. No. 6,502,135, which is a continuation-in-part of application No. 09/429,643, filed on Oct. 29, 1999, now Pat. No. 7,010,604.

(60) Provisional application No. 60/106,261, filed on Oct. 30, 1998, provisional application No. 60/137,704, filed on Jun. 7, 1999.

(51) **Int. Cl.**  
**G06F 15/173** (2006.01)

(52) **U.S. Cl.** ..... **709/226**

(58) **Field of Classification Search** ..... 709/226,  
709/221; 726/15

See application file for complete search history.

(56) **References Cited**

**U.S. PATENT DOCUMENTS**

2,895,502 A 7/1959 Roper et al.  
5,303,302 A 4/1994 Burrows  
5,311,593 A 5/1994 Carmi

(Continued)

**FOREIGN PATENT DOCUMENTS**

EP 0838930 4/1988

(Continued)

**OTHER PUBLICATIONS**

Baumgartner et al, "Differentiated Services: A New Approach for Quality of Service in the Internet," International Conference on High Performance Networking, 255-273 (1998).

(Continued)

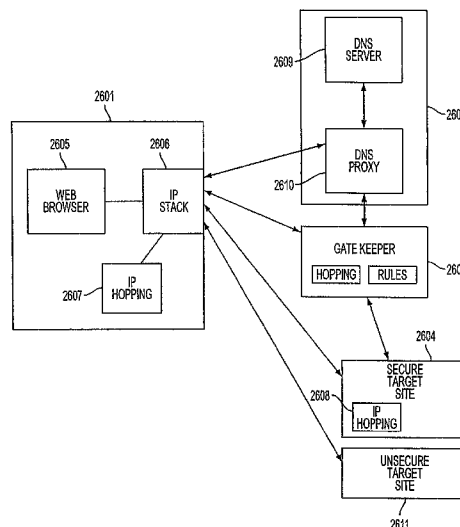
*Primary Examiner* — Krisna Lim

(74) *Attorney, Agent, or Firm* — McDermott Will & Emery LLP

(57) **ABSTRACT**

A secure domain name service for a computer network is disclosed that includes a portal connected to a computer network, such as the Internet, and a domain name database connected to the computer network through the portal. The portal authenticates a query for a secure computer network address, and the domain name database stores secure computer network addresses for the computer network. Each secure computer network address is based on a non-standard top-level domain name, such as .scom, .sorg, .snet, .snet, .sedu, .smil and .sint.

**60 Claims, 40 Drawing Sheets**





## US 7,921,211 B2

Page 2

## U.S. PATENT DOCUMENTS

5,384,848	A	1/1995	Kikuchi	
5,511,122	A	4/1996	Atkinson	
5,629,984	A	5/1997	McManis	
5,764,906	A	6/1998	Edelstein et al.	
5,771,239	A	6/1998	Moroney et al.	
5,805,803	A	9/1998	Birrell et al.	
5,822,434	A	10/1998	Caronni et al.	
5,864,666	A	1/1999	Shrader	726/15
5,870,610	A	2/1999	Beyda et al.	
5,898,830	A	4/1999	Wesinger, Jr. et al.	
5,950,195	A	9/1999	Stockwell et al.	
6,052,788	A	4/2000	Wesinger et al.	
6,055,574	A	4/2000	Smorodinsky et al.	
6,061,346	A	5/2000	Nordman	
6,079,020	A	6/2000	Liu	
6,081,900	A	6/2000	Subramaniam et al.	726/19
6,101,182	A	8/2000	Sistanizadeh et al.	
6,119,171	A	9/2000	Alkhatib	
6,173,399	B1	1/2001	Gilbrech	
6,199,112	B1	3/2001	Wilson	
6,202,081	B1	3/2001	Naudus	
6,223,287	B1	4/2001	Douglas et al.	
6,226,748	B1	5/2001	Bots et al.	
6,226,751	B1	5/2001	Arrow et al.	
6,246,670	B1	6/2001	Karlsson et al.	
6,262,987	B1	7/2001	Mogul	
6,298,341	B1	10/2001	Mann et al.	
6,314,463	B1	11/2001	Abbott et al.	
6,333,272	B1	12/2001	McMillin et al.	
6,338,082	B1	1/2002	Schneider	
6,502,135	B1	12/2002	Munger et al.	
6,557,037	B1	4/2003	Provino	
6,687,746	B1	2/2004	Shuster et al.	
6,701,437	B1	3/2004	Hoke et al.	
6,752,166	B2	6/2004	Lull et al.	
6,757,740	B1	6/2004	Parkh et al.	
6,937,597	B1	8/2005	Rosenberg et al.	
7,039,713	B1	5/2006	Van Gunter et al.	
7,072,964	B1	7/2006	Whittle et al.	
7,167,904	B1	1/2007	Devarajan et al.	
7,188,175	B1	3/2007	McKeeth	
7,353,841	B2	4/2008	Kono et al.	
7,461,334	B1	12/2008	Lu et al.	
7,490,151	B2	2/2009	Munger et al.	
7,493,403	B2	2/2009	Shull et al.	
2001/0049741	A1	12/2001	Skene et al.	
2004/0199493	A1	10/2004	Ruiz et al.	
2004/0199520	A1	10/2004	Ruiz et al.	
2004/0199608	A1	10/2004	Rechterman et al.	
2004/0199620	A1	10/2004	Ruiz et al.	
2007/0208869	A1	9/2007	Adelman et al.	
2007/0214284	A1	9/2007	King et al.	
2007/0266141	A1	11/2007	Norton	
2008/0235507	A1	9/2008	Ishikawa et al.	

## FOREIGN PATENT DOCUMENTS

EP	0814589	12/1997
GB	2317792	4/1998
GB	2334181	8/1999
GB	2340702	2/2000
JP	62-214744	9/1987
JP	04-363941	12/1992
JP	09-018492	1/1997
JP	10-070531	3/1998
WO	WO98/27783	6/1998
WO	WO99/11019	3/1999
WO	WO 00/17775	3/2000
WO	WO 00/70458	11/2000
WO	WO 01/16766	3/2001

## OTHER PUBLICATIONS

Chapman et al., "Domain Name System (DNS)," 278-296 (1995).  
 Davila et al., "Implementation of Virtual Private Networks at the Transport Layer," M. Mambo, Y. Zheng (Eds), Information Security (Second International) Workshop, ISW' 99. Lecture Notes in Computer Science (LNCS), vol. 1729; 85-102 (1999).  
 De Raadt et al., "Cryptography in OpenBSD," 10 pages (1999).

Eastlake, "Domain Name System Security Extensions," Internet Citation, Retrieved from the Internet: URL:ftp://ftp.inet.no/pub/ietf/internet-drafts/draft-ietf-dnssec-secext2-05.txt (1998).

Gunter et al., "An Architecture for Managing QoS-Enabled VRNs Over the Internet," Proceedings 24th Conference on Local Computer Networks. LCN' 99 IEEE Comput. Soc Los Alamitos, CA, pp. 122-131 (1999).

Shimizu, "Special Feature: Mastering the Internet with Windows 2000", Internet Magazine, 63:296-307 (2000).

Stallings, "Cryptography and Network Security," Principals and Practice, 2nd Edition, pp. 399-440 (1999).

Takata, "U.S. Vendors Take Serious Action to Act Against Crackers—A Tracking Tool and a Highly Safe DNS Software are Released", Nikkei Communications, 257:87(1997).

Wells, Email (Lancasterblbe@mail.msn.com), Subject: "Security Icon," (1998).

Fasbender, A., et al., Variable and Scalable Security: Protection of Location Information in Mobile IP, IEEE VTS, 46th, 1996, 5 pp.

DNS-related correspondence dated Sep. 7, 1993 to Sep. 20, 1993. (Pre KX, KX Records).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Dec. 2, 1996). (RFC 2543 Internet Draft 1).

Aventail Corp., "AutoSOCKS v. 2.1 Datasheet," available at <http://www.archive.org/web/19970212013409/www.aventail.com/prod/autosocks2ds.html> (1997). (AutoSOCKS, Aventail).

Aventail Corp., "Socks Version 5," Aventail Whitepaper, available at [http://web.archive.org/web/19970620030312/www.aventail.com/educate/whitepaper/soc\\_kswp.html](http://web.archive.org/web/19970620030312/www.aventail.com/educate/whitepaper/soc_kswp.html) (1997). (Socks, Aventail).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Mar. 27, 1997). (RFC 2543 Internet Draft 2).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jul. 31, 1997). (RFC 2543 Internet Draft 3).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Nov. 11, 1997). (RFC 2543 Internet Draft 4).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (May 14, 1998). (RFC 2543 Internet Draft 5).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jun. 17, 1998). (RFC 2543 Internet Draft 6).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jul. 16, 1998). (RFC 2543 Internet Draft 7).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Aug. 7, 1998). (RFC 2543 Internet Draft 8).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Sep. 18, 1998). (RFC 2543 Internet Draft 9).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Nov. 12, 1998). (RFC 2543 Internet Draft 10).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Dec. 15, 1998). (RFC 2543 Internet Draft 11).

Aventail Corp., "Aventail Connect 3.1/2.6 Administrator's Guide," (1999). (Aventail Administrator 3.1, Aventail).

Aventail Corp., "Aventail Connect 3.1/2.6 User's Guide," (1999). (Aventail User 3.1, Aventail).

Aventail Corp., "Aventail ExtraWeb Server v3.2 Administrator's Guide," (1999). (Aventail ExtraWeb 3.2, Aventail).

Check Point Software Technologies Ltd. (1999) (Check Point, Checkpoint FW).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jan. 15, 1999). (RFC 2543 Internet Draft 12).

Goncalves, et al. *Check Point Firewall—1 Administration Guide*, McGraw-Hill Companies (2000). (Goncalves, Checkpoint FW).

Assured Digital Products. (Assured Digital).

F-Secure, *F-Secure Evaluation Kit* (May 1999) (FSECURE 00000003) (Evaluation Kit 3).

F-Secure, *F-Secure Evaluation Kit* (Sep. 1998) (FSECURE 00000009) (Evaluation Kit 9).

IRE, Inc., *SafeNet/Soft-PK Version 4* (Mar. 28, 2000) (Soft-PK Version 4).

IRE/SafeNet Inc., *VPN Technologies Overview* (Mar. 28, 2000) (SafeNet VPN Overview).

IRE, Inc., *SafeNet/VPN Policy Manager Quick Start Guide Version 1* (1999) (SafeNet VPN Policy Manager).

Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation v.3* (2000).

## US 7,921,211 B2

Page 3

- U.S. Appl. No. 60/134,547, filed May 17, 1999, Victor Sheymov.
- U.S. Appl. No. 60/151,563, filed Aug. 31, 1999, Bryan Whittles.
- U.S. Appl. No. 09/399,753, filed Sep. 22, 1998, Graig Miller et al.
- Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009, *VirnetX Inc. and Science Applications International Corp. v. Microsoft Corporation*.
- Appendix A of the Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009.
- Concordance Table for the References Cited in Tables on pp. 6-15, 71-80 and 116-124 of the Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009.
- I. P. Mockapetris, "DNS Encoding of Network Names and Other Types," Network Working Group, RFC 1101 (Apr. 1989) (RFC1101, DNS SRV).
- R. Atkinson, "An Internetwork Authentication Architecture," Naval Research Laboratory, Center for High Assurance Computing Systems (Aug. 5, 1993). (Atkinson NRL, KX Records).
- Henning Schulzrinne, *Personal Mobility for Multimedia Services In The Internet*, Proceedings of the Interactive Distributed Multimedia Systems and Services European Workshop at 143 (1996). (Schulzrinne 96).
- Microsoft Corp., *Microsoft Virtual Private Networking: Using Point-to-Point Tunneling Protocol for Low-Cost, Secure, Remote Access Across the Internet* (1996) (printed from 1998 PDC DVD-ROM). (Point to Point, Microsoft Prior Art VPN Technology).
- "Safe Surfing: How to Build a Secure World Wide Web Connection," IBM Technical Support Organization, (Mar. 1996). (Safe Surfing, Website Art).
- Goldschlag, et al., "Hiding Routing Information," Workshop on Information Hiding, Cambridge, UK (May 1996). (Goldschlag II, Onion Routing).
- "IPSec Minutes From Montreal", IPSEC Working Group Meeting Notes, <http://www.sandleman.ca/ipsec/1996/08/msg00018.html> (Jun. 1996). (IPSec Minutes, FreeS/WAN).
- J. M. Galvin, "Public Key Distribution with Secure DNS," Proceedings of the Sixth USENIX UNIX Security Symposium, San Jose, California, Jul. 1996. (Galvin, DNSSEC).
- J. Gilmore, et al. "Re: Key Management, anyone? (DNS Keying)," IPsec Working Group Mailing List Archives (Aug. 1996). (Gilmore DNS, FreeS/WAN).
- H. Orman, et al. "Re: 'Re: DNS? was Re: Key Management, anyone?'" IETF IPsec Working Group Mailing List Archive (Aug. 1996-Sep. 1996). (Orman DNS, FreeS/WAN).
- Arnt Gulbrandsen & Paul Vixie, *A DNSRR for specifying the location of services (DNS SRV)*, IETF RFC 2052 (Oct. 1996). (RFC 2052, DNS SRV).
- Freier, et al. "The SSL Protocol Version 3.0," Transport Layer Security Working Group (Nov. 18, 1996). (SSL, Underlying Security Technology).
- M.G. Reed, et al. "Proxies for Anonymous Routing," 12th Annual Computer Security Applications Conference, San Diego, CA, Dec. 9-13, 1996. (Reed, Onion Routing).
- Kenneth F. Alden & Edward P. Wobber, *The AltaVista Tunnel: Using the Internet to Extend Corporate Networks*, Digital Technical Journal (1997) (Alden, AltaVista).
- Automotive Industry Action Group, "ANX Release 1 Document Publication," AIAG (1997). (AIAG, ANX).
- Automotive Industry Action Group, "ANX Release 1 Draft Document Publication," AIAG Publications (1997). (AIAG Release, ANX).
- Aventail Corp. "Aventail VPN Data Sheet," available at <http://www.archive.org/web/19970212013043/www.aventail.com/prod/vpndata.html> (1997). (Data Sheet, Aventail).
- Aventail Corp., "Directed VPN Vs. Tunnel," available at <http://web.archive.org/web/19970620030312/www.aventail.com/educate/directvpn.html> (1997). (Directed VPN, Aventail).
- Aventail Corp., "Managing Corporate Access to the Internet," Aventail AutoSOCKS White Paper available at <http://web.archive.org/web/19970620030312/www.aventail.com/educate/whitepaper/ipmwp.html> (1997). (Corporate Access, Aventail).
- Aventail Corp., "VPN Server V2.0 Administration Guide," (1997). (VPN, Aventail).
- Goldschlag, et al. "Privacy on the Internet," Naval Research Laboratory, Center for High Assurance Computer Systems (1997). (Goldschlag I, Onion Routing).
- Microsoft Corp., *Installing Configuring and Using PPTP with Microsoft Clients and Servers* (1997). (Using PPTP, Microsoft Prior Art VPN Technology).
- Microsoft Corp., *IP Security for Microsoft Windows NT Server 5.0* (1997) (printed from 1998 PDC DVD-ROM). (IP Security, Microsoft Prior Art VPN Technology).
- Microsoft Corp., *Microsoft Windows NT Active Directory: An Introduction to the Next Generation Directory Services* (1997) (printed from 1998 PDC DVD-ROM). (Directory, Microsoft Prior Art VPN Technology).
- Microsoft Corp., *Routing and Remote Access Service for Windows NT Server New Opportunities Today and Looking Ahead* (1997) (printed from 1998 PDC DVD-ROM). (Routing, Microsoft Prior Art VPN Technology).
- Microsoft Corp., *Understanding Point-to-Point Tunneling Protocol PPTP* (1997) (printed from 1998 PDC DVD-ROM). (Understanding PPTP, Microsoft Prior Art VPN Technology).
- J. Mark Smith et al., *Protecting a Private Network: The AltaVista Firewall*, Digital Technical Journal (1997). (Smith, AltaVista).
- Naganand Doraswamy *Implementation of Virtual Private Networks (VPNs) with IPsec*, <draft-ietf-ipsec-vpn-00.txt> (Mar. 12, 1997). (Doraswamy).
- Aventail Corp., "Aventail, and Cybersafe to Provide Secure Authentication For Internet and Intranet Communication," Press Release, Apr. 3, 1997. (Secure Authentication, Aventail).
- D. Wagner, et al. "Analysis of the SSL 3.0 Protocol," (Apr. 15, 1997). (Analysis, Underlying Security Technologies).
- Automotive Industry Action Group, "ANXO Certification Authority Service and Directory Service Definition for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (May 9, 1997). (AIAG Definition, ANX).
- Automotive Industry Action Group, "ANXO Certification Process and ANX Registration Process Definition for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (May 9, 1997). (AIAG Certification, ANX).
- Aventail Corp., "Aventail Announces the First VPN Solution to Assure Interoperability Across Emerging Security Protocols," Jun. 2, 1997. (First VPN, Aventail).
- Syverson, et al. "Private Web Browsing," Naval Research Laboratory, Center for High Assurance Computer Systems (Jun. 2, 1997). (Syverson, Onion Routing).
- Bellcore, "Metrics, Criteria, and Measurement Technique Requirements for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (Jun. 16, 1997). (AIAG Requirements, ANX).
- R. Atkinson, "Key Exchange Delegation Record for the DNS," Network Working Group, RFC 2230 (Nov. 1997). (RFC 2230, KX Records).
- 1998 Microsoft Professional Developers Conference DVD ("1998 PDC DVD-ROM") (including screenshots captured therefrom and produced as MSFTVX 00018827-00018832). (Conference, Microsoft Prior Art VPN Technology).
- Microsoft Corp., *Virtual Private Networking An Overview* (1998) (printed from 1998 PDC DVD-ROM) (Overview, Microsoft Prior Art VPN Technology).
- Microsoft Corp., *Windows NT 5.0 Beta Has Public Premiere at Seattle Mini-Camp Seminar attendees get first look at the performance and capabilities of Windows NT 5.0* (1998) (available at <http://www.microsoft.com/presspass/features/1998/10-19nt5.mspxpfrue>). (NT Beta, Microsoft Prior Art VPN Technology).
- "What ports does SSL use" available at [stason.org/TULARC/security/ssl-talk/3-4-What-ports-does-ssl-use.html](http://stason.org/TULARC/security/ssl-talk/3-4-What-ports-does-ssl-use.html) (1998). (Ports, DNS SRV).
- Aventail Corp., "Aventail VPN V2.6 Includes Support for More Than Ten Authentication Methods Making Extranet VPN Development Secure and Simple," Press Release, Jan. 19, 1998. (VPN V2.6, Aventail).
- R. G. Moskowitz, "Network Address Translation Issues with IPsec," Internet Draft, Internet Engineering Task Force, Feb. 6, 1998. (Moskowitz).

## US 7,921,211 B2

Page 4

- H. Schulzrinne, et al., "Internet Telephony Gateway Location," Proceedings of IEEE Infocom '98, The Conference on Computer Communications, vol. 2 (Mar. 29-Apr. 2, 1998). (Gateway, Schulzrinne).
- C. Huitema, 45 al. "Simple Gateway Control Protocol," Version 1.0 (May 5, 1998). (SGCP).
- DISA "Secret Internet Protocol Router Network," SIPRNET Program Management Office (D3113) DISN Networks, DISN Transmission Services (May 8, 1998). (DISA, SIPRNET).
- D. McDonald, et al. "PF\_KEY Key Management API, Version 2," Network Working Group, RFC 2367 (Jul. 1998). (RFC 2367).
- Microsoft Corp., *Company Focuses on Quality and Customer Feedback* (Aug. 18, 1998). (Focus, Microsoft Prior Art VPN Technology).
- Atkinson, et al. "Security Architecture for the Internet Protocol," Network Working Group, RFC 2401 (Nov. 1998). (RFC 2401, Underlying Security Technologies).
- Donald Eastlake, *Domain Name System Security Extensions*, IETF DNS Security Working Group (Dec. 1998). (DNSSEC-7).
- Kaufman et al., "Implementing IPsec," (Copyright 1999). (Implementing IPSEC, VPN References).
- Network Solutions, Inc. "Enabling SSL," NSI Registry (1999). (Enabling SSL, Underlying Security Technologies).
- C. Scott, et al. *Virtual Private Networks*, O'Reilly and Associates, Inc.; 2nd ed. (Jan. 1999). (Scott VPNs).
- Goldschlag, et al., "Onion Routing for Anonymous and Private Internet Connections," Naval Research Laboratory, Center for High Assurance Computer Systems (Jan. 28, 1999). (Goldschlag III, Onion Routing).
- H. Schulzrinne, "Internet Telephony: architecture and protocols—an IETF perspective," *Computer Networks*, vol. 31, No. 3 (Feb. 1999). (Telephony, Schulzrinne).
- M. Handley, et al. "SIP: Session Initiation Protocol," Network Working Group, RFC 2543 and Internet Drafts (Dec. 1996-Mar. 1999). (Handley, RFC 2543).
- FreeS/WAN Project, *Linux FreeS/WAN Compatibility Guide* (Mar. 4, 1999). (FreeS/WAN Compatibility Guide, FreeS/WAN).
- Telcordia Technologies, "ANX Release 1 Document Corrections," AIAG (May 11, 1999). (Telcordia, ANX).
- Ken Hornstein & Jeffrey Altman, *Distributing Kerberos KDC and Realm Information with DNS* <draft-ietf-cat-krb-dns-locate-oo.txt> (Jun. 21, 1999). (Hornstein, DNS SRV).
- Bhattacharya et al. "An LDAP Schema for Configuration and Administration of IPsec Based Virtual Private Networks (VPNs)," IETF Internet Draft (Oct. 1999). (Bhattacharya LDAP VPN).
- B. Patel, et al. "DHCP Configuration of IPSEC Tunnel Mode," IPSEC Working Group, Internet Draft 02 (Oct. 15, 1999). (Patel).
- "Building a Microsoft VPN: A Comprehensive Collection of Microsoft Resources," FirstVPN, (Jan. 2000). (FirstVPN Microsoft).
- Gulbrandsen, Vixie, & Esibov, *A DNS RR for specifying the location of services (DNS SRV)*, IETF RFC 2782 (Feb. 2000). (RFC 2782, DNS SRV).
- Mitre Organization, "Technical Description," Collaborative Operations in Joint Expeditionary Force Experiment (JEFX) 99 (Feb. 2000). (MITRE, SIPRNET).
- H. Schulzrinne, et al. "Application-Layer Mobility Using SIP," *Mobile Computing and Communications Review*, vol. 4, No. 3. pp. 47-57 (Jul. 2000). (Application, SIP).
- Kindred et al., "Dynamic VPN Communities: Implementation and Experience," DARPA Information Survivability Conference and Exposition II (Jun. 2001). (DARPA, VPN Systems).
- ANX 101: Basic ANX Service Outline. (Outline, ANX).
- ANX 201: Advanced ANX Service. (Advanced, ANX).
- Appendix A: Certificate Profile for ANX IPsec Certificates. (Appendix, ANX).
- Aventail Corp., "Aventail AutoSOCKS the Client Key to Network Security," Aventail Corporation White Paper. (Network Security, Aventail).
- Cindy Moran, "DISN Data Networks: Secret Internet Protocol Router Network (SIPRNet)." (Moran, SIPRNet).
- Data Fellows F-Secure VPN+ (F-Secure VPN+).
- Interim Operational Systems Doctrine for the Remote Access Security Program (RASP) Secret Dial-In Solution. (RASP, SIPRNET).
- Onion Routing*, "Investigation of Route Selection Algorithms," available at <http://www.onion-router.net/Archives/Route/index.html>. (Route Selection, Onion Routing).
- Secure Computing, "Bullet-Proofing an Army Net," Washington Technology. (Secure, SIPRNET).
- Sparta "Dynamic Virtual Private Network." (Sparta, VPN Systems).
- Standard Operation Procedure for Using the 1910 Secure Modems. (Standard, SIPRNET).
- Publicly available emails relating to FreeS/WAN (MSFTVX00018833-MSFTVX00019206). (FreeS/WAN emails, FreeS/WAN).
- Kaufman et al., "Implementing IPsec," (Copyright 1999) (Implementing IPsec).
- Network Associates *Gauntlet Firewall For Unix User's Guide Version 5.0* (1999). (Gauntlet User's Guide—Unix, Firewall Products).
- Network Associates *Gauntlet Firewall for Windows NT Getting Started Guide Version 5.0* (1999) (Gauntlet Getting Started Guide—NT, Firewall Products).
- Network Associates *Gauntlet Firewall for Unix Getting Started Guide Version 5.0* (1999) (Gauntlet Unix Getting Started Guide, Firewall Products).
- Network Associates *Release Notes Gauntlet Firewall for Unix 5.0* (Mar. 19, 1999) (Gauntlet Unix Release Notes, Firewall Products).
- Network Associates *Gauntlet Firewall For Windows NT Administrator's Guide Version 5.0* (1999) (Gauntlet NT Administrator's Guide, Firewall Products).
- Trusted Information Systems, Inc. *Gauntlet Internet Firewall Firewall-to-Firewall Encryption Guide Version 3.1* (1996) (Gauntlet Firewall-to-Firewall, Firewall Products).
- Network Associates *Gauntlet Firewall Global Virtual Private Network User's Guide for Windows NT Version 5.0* (1999) (Gauntlet NT GVPN, GVPN).
- Network Associates *Gauntlet Firewall For UNIX Global Virtual Private Network User's Guide Version 5.0* (1999) (Gauntlet Unix GVPN, GVPN).
- Dan Sterne *Dynamic Virtual Private Networks* (May 23, 2000) (Sterne DVPN, DVPN).
- Darrell Kindred *Dynamic Virtual Private Networks (DVPN)* (Dec. 21, 1999) (Kindred DVPN, DVPN).
- Dan Sterne et al. *TIS Dynamic Security Perimeter Research Project Demonstration* (Mar. 9, 1998) (Dynamic Security Perimeter, DVPN).
- Darrell Kindred *Dynamic Virtual Private Networks Capability Description* (Jan. 5, 2000) (Kindred DVPN Capability, DVPN) 11.
- Oct. 7, and 28, 1997 email from Domenic J. Turchi Jr. (SPARTA00001712-1714, 1808-1811) (Turchi DVPN email, DVPN).
- James Just & Dan Sterne *Security Quickstart Task Update* (Feb. 5, 1997) (Security Quickstart, DVPN).
- Virtual Private Network Demonstration dated Mar. 21, 1998 (SPARTA00001844-54) (DVPN Demonstration, DVPN).
- GTE Internetworking & BBN Technologies *DARPA Information Assurance Program Integrated Feasibility Demonstration (IFD) 1.1 Plan* (Mar. 10, 1998) (IFD 1.1, DVPN).
- Microsoft Corp. Windows NT Server Product Documentation: Administration Guide—Connection Point Services, available at <http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/cpsops.mspx> (Connection Point Services) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).
- Microsoft Corp. Windows NT Server Product Documentation: Administration Kit Guide—Connection Manager, available at <http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/cmak.mspx> (Connection Manager) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).
- Microsoft Corp. Autodial Heuristics, available at <http://support.microsoft.com/kb/164249> (Autodial Heuristics) (Although undated, this reference refers to the operation of prior art versions of Microsoft

## US 7,921,211 B2

Page 5

Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

Microsoft Corp., Cariplo: Distributed Component Object Model, (1996) available at [http://msdn2.microsoft.com/en-us/library/ms809332\(printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms809332(printer).aspx) (Cariplo I).

Marc Levy, COM Internet Services (Apr. 23, 1999), available at [http://msdn2.microsoft.com/en-us/library/ms809302\(printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms809302(printer).aspx) (Levy).

Markus Horstmann and Mary Kirtland, DCOM Architecture (Jul. 23, 1997), available at [http://msdn2.microsoft.com/en-us/library/ms809311\(printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms809311(printer).aspx) (Horstmann).

Microsoft Corp., DCOM: A Business Overview (Apr. 1997), available at [http://msdn2.microsoft.com/en-us/library/ms809320\(printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms809320(printer).aspx) (DCOM Business Overview I).

Microsoft Corp., DCOM Technical Overview (Nov. 1996), available at [http://msdn2.microsoft.com/en-us/library/ms809340\(printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms809340(printer).aspx) (DCOM Technical Overview I).

Microsoft Corp., DCOM Architecture White Paper (1998) available in PDC DVD-ROM (DCOM Architecture).

Microsoft Corp., DCOM—The Distributed Component Object Model, A Business Overview White Paper (Microsoft 1997) available in PDC DVD-ROM (DCOM Business Overview II).

Microsoft Corp., DCOM—Cariplo Home Banking Over The Internet White Paper (Microsoft 1996) available in PDC DVD-ROM (Cariplo II).

Microsoft Corp., DCOM Solutions in Action White Paper (Microsoft 1996) available in PDC DVD-ROM (DCOM Solutions in Action).

Microsoft Corp., DCOM Technical Overview White Paper (Microsoft 1996) available in PDC DVD-ROM (DCOM Technical Overview II).

Scott Suhay & Glenn Wood, DNS and Microsoft Windows NT 4.0, (1996) available at [http://msdn2.microsoft.com/en-us/library/ms810277\(printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms810277(printer).aspx) (Suhay).

Aaron Skonnard, *Essential WinNet* 313-423 (Addison Wesley Longman 1998) (Essential WinNet).

Microsoft Corp. Installing, Configuring, and Using PPTP with Microsoft Clients and Servers, (1998) available at [http://msdn2.microsoft.com/en-us/library/ms811078\(printer\).aspx](http://msdn2.microsoft.com/en-us/library/ms811078(printer).aspx) (Using PPTP).

Microsoft Corp., Internet Connection Services for MS RAS, Standard Edition, <http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/bcgstart.mspix> (Internet Connection Services I).

Microsoft Corp., Internet Connection Services for RAS, Commercial Edition, available at <http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/bcgstrtc.mspix> (Internet Connection Services II).

Microsoft Corp., Internet Explorer 5 Corporate Deployment Guide—Appendix B: Enabling Connections with the Connection Manager Administration Kit, available at <http://www.microsoft.com/technet/prodtechnol/ie/deploy/deploy5/appendb.mspix> (IE5 Corporate Development).

Mark Minasi, *Mastering Windows NT Server 4* 1359-1442 (6th ed., Jan. 15, 1999) (Mastering Windows NT Server).

*Hands On, Self-Paced Training for Supporting Version 4.0* 371-473 (Microsoft Press 1998) (Hands On).

Microsoft Corp., MS Point-to-Point Tunneling Protocol (Windows NT 4.0), available at <http://www.microsoft.com/technet/archive/winntas/maintain/feasability/pptpwp3.mspix> (MS PPTP).

Kenneth Gregg, et al., *Microsoft Windows NT Server Administrator's Bible* 173-206, 883-911, 974-1076 (IDG Books Worldwide 1999) (Gregg).

Microsoft Corp., Remote Access (Windows), available at [http://msdn2.microsoft.com/en-us/library/bb545687\(VS.85.printer\).aspx](http://msdn2.microsoft.com/en-us/library/bb545687(VS.85.printer).aspx) (Remote Access).

Microsoft Corp., Understanding PPTP (Windows NT 4.0), available at <http://www.microsoft.com/technet/archive/winntas/plan/pptpudst.mspix> (Understanding PPTP NT 4) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

Microsoft Corp., Windows NT 4.0: Virtual Private Networking, available at <http://www.microsoft.com/technet/archive/winntas/deploy/confeat/vpntwk.mspix> (NT4 VPN) (Although undated, this reference

refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

Anthony Northrup, *NT Network Plumbing: Routers, Proxies, and Web Services* 299-399 (IDG Books Worldwide 1998) (Network Plumbing).

Microsoft Corp., Chapter 1—Introduction to Windows NT Routing with Routing and Remote Access Service, Available at <http://www.microsoft.com/technet/archive/winntas/proddocs/ras40/rrasch01.mspix> (Intro to RRAS) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

Microsoft Corp., Windows NT Server Product Documentation: Chapter 5—Planning for Large-Scale Configurations, available at <http://www.microsoft.com/technet/archive/winntas/proddocs/ras40/rrasch05.mspix> (Large-Scale Configurations) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).

F-Secure, *F-Secure NameSurfer* (May 1999) (from FSECURE 00000003) (NameSurfer 3).

F-Secure, *F-Secure VPN Administrator's Guide* (May 1999) (from FSECURE 00000003) (F-Secure VPN 3).

F-Secure, *F-Secure SSH User's & Administrator's Guide* (May 1999) (from FSECURE 00000003) (SSH Guide 3).

F-Secure, *F-Secure SSH 2.0 for Windows NT and 95* (May 1999) (from FSECURE 00000003) (SSH 2.0 Guide 3).

F-Secure, *F-Secure VPN+ Administrator's Guide* (May 1999) (from Fsecure 00000003) (VPN+ Guide 3).

F-Secure, *F-Secure VPN+ 4.1* (1999) (from Fsecure 00000006) (VPN+ 4.1 Guide 6).

F-Secure, *F-Secure SSH* (1996) (from Fsecure 00000006) (F-Secure SSH 6).

F-Secure, *F-Secure SSH 2.0 for Windows NT and 95* (1998) (from Fsecure 00000006) (F-Secure SSH 2.0 Guide 6).

F-Secure, *F-Secure SSH User's & Administrator's Guide* (Sep. 1998) (from Fsecure 00000009) (SSH Guide 9).

F-Secure, *F-Secure SSH 2.0 for Windows NT and 95* (Sep. 1998) (from Fsecure 00000009) (F-Secure SSH 2.0 Guide 9).

F-Secure, *F-Secure VPN+* (Sep. 1998) (from Fsecure 00000009) (VPN+ Guide 9).

F-Secure, *F-Secure Management Tools, Administrator's Guide* (1999) (from Fsecure 00000003) (F-Secure Management Tools).

F-Secure, *F-Secure Desktop, User's Guide* (1997) (from Fsecure 00000009) (F-Secure Desktop User's Guide).

SafeNet, Inc., *VPN Policy Manager* (Jan. 2000) (VPN Policy Manager).

F-Secure, F-Secure VPN+ for Windows NT 4.0 (1998) (from Fsecure 00000009) (F-Secure VPN+).

IRE, Inc., *SafeNet / Security Center Technical Reference Addendum* (Jun. 22, 1999) (Safenet Addendum).

IRE, Inc., *System Description for VPN Policy Manager and SafeNet/SoftPK* (Mar. 30, 2000) (VPN Policy Manager System Description).

IRE, Inc., *About SafeNet / VPN Policy Manager* (1999) (About Safenet VPN Policy Manager).

Trusted Information Systems, Inc., *Gauntlet Internet Firewall, Firewall Product Functional Summary* (Jul. 22, 1996) (Gauntlet Functional Summary).

Trusted Information Systems, Inc., *Running the Gauntlet Internet Firewall, An Administrator's Guide to Gauntlet Version 3.0* (May 31, 1995) (Running the Gauntlet Internet Firewall).

Ted Harwood, *Windows NT Terminal Server and Citrix Metaframe* (New Riders 1999) (Windows NT Harwood) 79.

Todd W. Mathers and Shawn P. Genoway, *Windows NT Thing Client Solutions: Implementing Terminal Server and Citrix MetaFrame* (Macmillan Technial Publishing 1999) (Windows NT Mathers).

Bernard Aboba et al., *Securing L2TP using IPSEC* (Feb. 2, 1999).

*Finding Your Way Through the VPN Maze* (1999) ("PGP").

Linux FreeS/WAN Overview (1999) (Linux FreeS/WAN) Overview).

TimeStep, *The Business Case for Secure VPNs* (1998) ("TimeStep").

## US 7,921,211 B2

Page 6

- WatchGuard Technologies, Inc., *WatchGuard Firebox System Powerpoint* (2000).
- WatchGuard Technologies, Inc., *MSS Firewall Specifications* (1999).
- WatchGuard Technologies, Inc., *Request for Information, Security Services* (2000).
- WatchGuard Technologies, Inc., *Protecting the Internet Distributed Enterprise, White Paper* (Feb. 2000).
- WatchGuard Technologies, Inc., *WatchGuard LiveSecurity for MSS Powerpoint* (Feb. 14, 2000).
- WatchGuard Technologies, Inc., *MSS Version 2.5, Add-On for WatchGuard SOHO Release Notes* (Jul. 21, 2000).
- Air Force Research Laboratory, *Statement of Work for Information Assurance System Architecture and Integration*, PR No. N-8-6106 (Contract No. F30602-98-C-0012) (Jan. 29, 1998).
- GTE Internetworking & BBN Technologies *DARPA Information Assurance Program Integrated Feasibility Demonstration (IFD) 1.2 Report, Rev. 1.0* (Sep. 21, 1998).
- BBN Information Assurance Contract, *TIS Labs Monthly Status Report* (Mar. 16-Apr. 30, 1998).
- DARPA, *Dynamic Virtual Private Network (VPN) Powerpoint*.
- GTE Internetworking, *Contractor's Program Progress Report* (Mar. 16-Apr. 30, 1998).
- Darrell Kindred, *Dynamic Virtual Private Networks (DVPN) Countermeasure Characterization* (Jan. 30, 2001).
- Virtual Private Networking Countermeasure Characterization* (Mar. 30, 2000).
- Virtual Private Network Demonstration* (Mar. 21, 1998).
- Information Assurance/NAI Labs, *Dynamic Virtual Private Networks (VPNs) and Integrated Security Management* (2000).
- Information Assurance/NAI Labs, *Create/Add DVPN Enclave* (2000).
- NAI Labs, *IFE 3.1 Integration Demo* (2000).
- Information Assurance, *Science Fair Agenda* (2000).
- Darrell Kindred et al., *Proposed Threads for IFE 3.1* (Jan. 13, 2000).
- IFE 3.1 Technology Dependencies* (2000).
- IFE 3.1 Topology* (Feb. 9, 2000).
- Information Assurance, *Information Assurance Integration: IFE 3.1, Hypothesis & Thread Development* (Jan. 10-11, 2000).
- Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation* (2000).
- Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation v.2* (2000).
- Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation v.3* (2000).
- T. Braun et al., *Virtual Private Network Architecture*, Charging and Accounting Technology for the Internet (Aug. 1, 1999) (VPNA).
- Network Associates Products—*PGP Total Network Security Suite, Dynamic Virtual Private Networks* (1999).
- Microsoft Corporation, *Microsoft Proxy Server 2.0* (1997) (Proxy Server 2.0, Microsoft Prior Art VPN Technology).
- David Johnson et. al., *A Guide To Microsoft Proxy Server 2.0* (1999) (Johnson, Microsoft Prior Art VPN Technology).
- Microsoft Corporation, *Setting Server Parameters* (1997 (copied from Proxy Server 2.0 CD labeled MSFTVX00157288) (Setting Server Parameters, Microsoft Prior Art VPN Technology).
- Kevin Schuler, *Microsoft Proxy Server 2* (1998) (Schuler, Microsoft Prior Art VPN Technology).
- Erik Rozell et. al., *MCSE Proxy Server 2 Study Guide* (1998) (Rozell, Microsoft Prior 15 Art VPN Technology).
- M. Shane Stigler & Mark A Linsenhardt, *IIS 4 and Proxy Server 2* (1999) (Stigler, Microsoft Prior Art VPN Technology).
- David G. Schaer, *MCSE Test Success: Proxy Server 2* (1998) (Schaer, Microsoft Prior Art VPN Technology).
- John Savill, *The Windows NT and Windows 2000 Answer Book* (1999) (Savill, Microsoft Prior Art VPN Technology).
- Network Associates *Gauntlet Firewall Global Virtual Private Network User's Guide for Windows NT Version 5.0* (1999) (Gauntlet NT GVPN, GVPN).
- File History for U.S. Appl. No. 09/653,201, Applicant(s): Whittle Bryan, et al., filed Aug. 31, 2000.
- AutoSOCKS v2.1*, Datasheet, <http://web.archive.org/web/19970212013409/www.aventail.com/prod/autoskds.html>.
- Ran Atkinson, *Use of DNS to Distribute Keys*, Sep. 7, 1993, <http://ops.ietf.org/lists/namedroppers/namedroppers.199x/msg00945.html>.
- FirstVPN Enterprise Networks, Overview.
- Chapter 1: Introduction to Firewall Technology, Administration Guide; Dec. 19, 2007, [http://www.books24x7.com/book/id\\_762/viewer\\_r.asp?bookid=762&chunked=41065062](http://www.books24x7.com/book/id_762/viewer_r.asp?bookid=762&chunked=41065062).
- The TLS Protocol Version 1.0; Jan. 1999; p. 65 of 71.
- Elizabeth D. Zwicky, et al., *Building Internet Firewalls*, 2nd Ed.
- Virtual Private Networks—Assured Digital Incorporated—ADI 4500; <http://web.archive.org/web/19990224050035/www.assured-digital.com/products/prodvpn/adia4500.htm>.
- Accessware—The Third Wave in Network Security, Conclave from Internet Dynamics; <http://web.archive.org/web/11980210013830/interdyn.com/Accessware.html>.
- Extended System Press Release, Sep. 2, 1997; *Extended VPN Uses The Internet to Create Virtual Private Networks*, [www.extendedsystems.com](http://www.extendedsystems.com).
- Socks Version 5; Executive Summary; <http://web.archive.org/web/199970620031945/www.aventail.com/educate/whitepaper/sockswp.html>.
- Internet Dynamics First to Ship Integrated Security Solutions for Enterprise Intranets and Extranets; Sep. 15, 1997; <http://web.archive.org/web/19980210014150/interdyn.com>.
- Emails from various individuals to Linux IPsec re: DNS-LDAP Splicing.
- Microsoft Corporation's Fifth Amended Invalidity Contentions dated Sep. 18, 2009, *VirnetX Inc. and Science Applications International Corp. v. Microsoft Corporation* and invalidity claim charts for U.S. Patent Nos. 7,188,180 and 6,839,759.
- The IPSEC Protocol as described in Atkinson, et al., "Security Architecture for the Internet Protocol," Network Working Group, RFC 2401 (Nov. 1998) ("RFC 2401"); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).
- S. Kent and R. Atkinson, "IP Authentication Header," RFC 2402 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).
- C. Madson and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH," RFC 2403 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).
- C. Madson and R. Glenn, "The Use HMAC-SHA-1-96 within ESP and AH," RFC 2404 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).
- C. Madson and N. Doraswamy, "The ESP DES-CBC Cipher Algorithm With Explicit IV," RFC 2405 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).
- S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," RFC 2406 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).
- Derrell Piper, "The Internet IP Security Domain of Interpretation for ISAKMP," RFC 2407 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).
- Douglas Maughan, et al., "Internet Security Association and Key Management Protocol (ISAKMP)," RFC 2408 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).
- D. Harkins and D. Carrell, "The Internet Key Exchange (IKE)," RFC 2409 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).
- R. Glenn and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec," RFC 2410 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).

**US 7,921,211 B2**

Page 7

---

R. Thayer, et al., "IP Security Document Roadmap," RFC 2411 (Nov. 1998); [http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu\\_eng.html](http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html).

Hilarie K. Orman, "The OAKLEY Key Determination Protocol," RFC 2412 (Nov. 1998) in combination with J.M. Galvin, "Public Key Distribution with Secure DNS," Proceedings of the Sixth USENIX UNIX Security Symposium, San Jose California (Jul. 1996) ("Galvin").

WatchGuard Technologies, Inc., *WatchGuard Firebox System Powerpoint* (2000).

WatchGuard Technologies, Inc., *Request for Information, Security Services* (2000).

WatchGuard Technologies, Inc., *Protecting the Internet Distributed Enterprise, White Paper* (Feb. 2000).

WatchGuard Technologies, Inc., *MSS Version 2.5, Add-On for WatchGuard SOHO Release Notes* (Jul. 21, 2000).

Yuan Dong Feng, "A novel scheme combining interleaving technique with cipher in Rayleigh fading channels," Proceedings of the International Conference on Communication technology, 2:S47-02-1-S47-02-4 (1998).

D.W. Davies and W.L. Price, edited by Tadahiro Uezono, "Network Security", Japan, Nikkei McGraw-Hill, Dec. 5, 1958, First Edition, first copy, p. 102-108.

David Kosiur, "Building and Managing Virtual Private Networks" (1998).

P. Mockapetris, "Domain Names—Implementation and Specification," Network Working Group, RFC 1035 (Nov. 1987).

Request for *Inter Partes* Reexamination of Patent No. 6,502,135, dated Nov. 25, 2009.

Request for *Inter Partes* Reexamination of Patent No. 7,188,180, dated Nov. 25, 2009.

\* cited by examiner

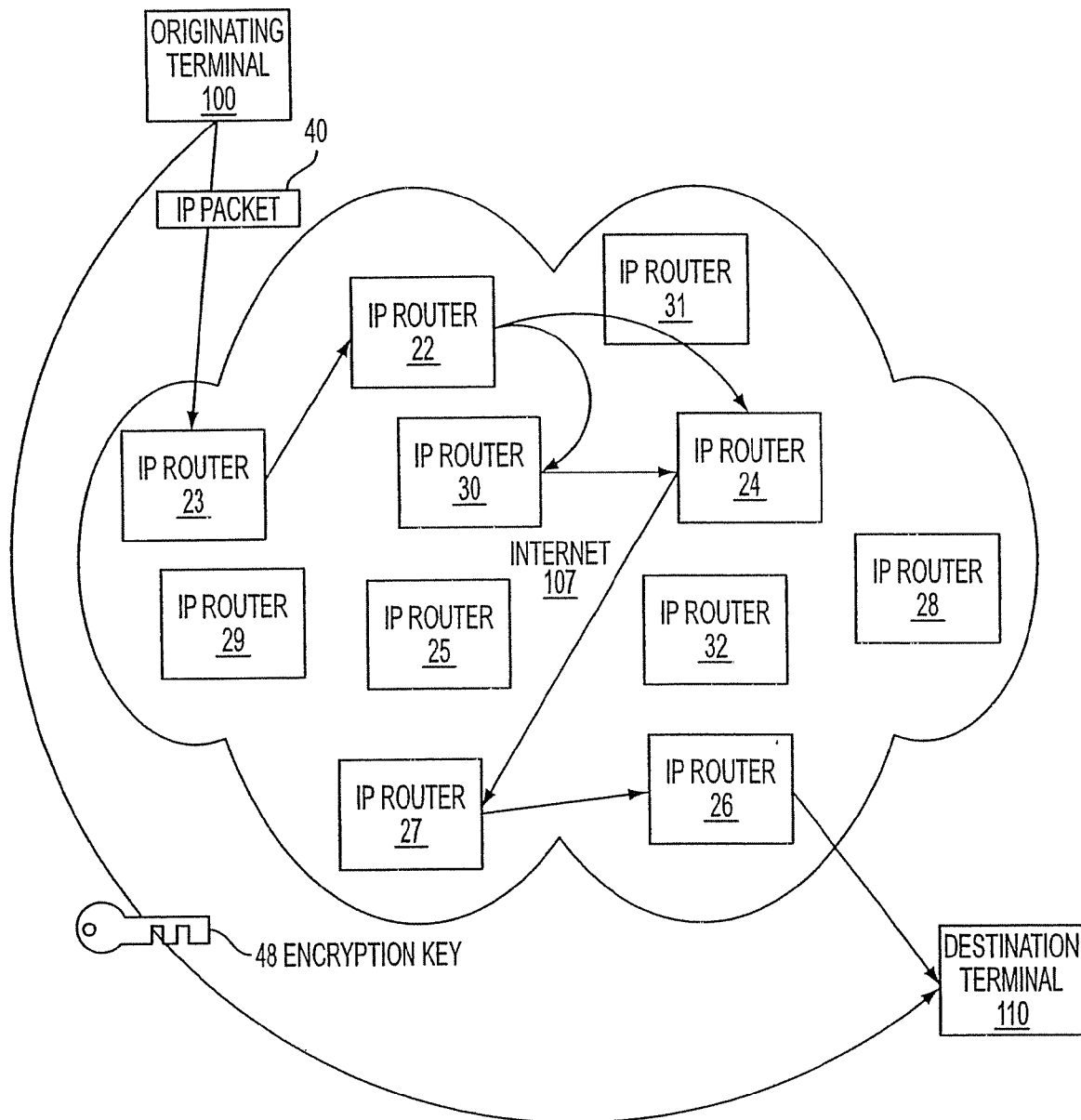


FIG. 1

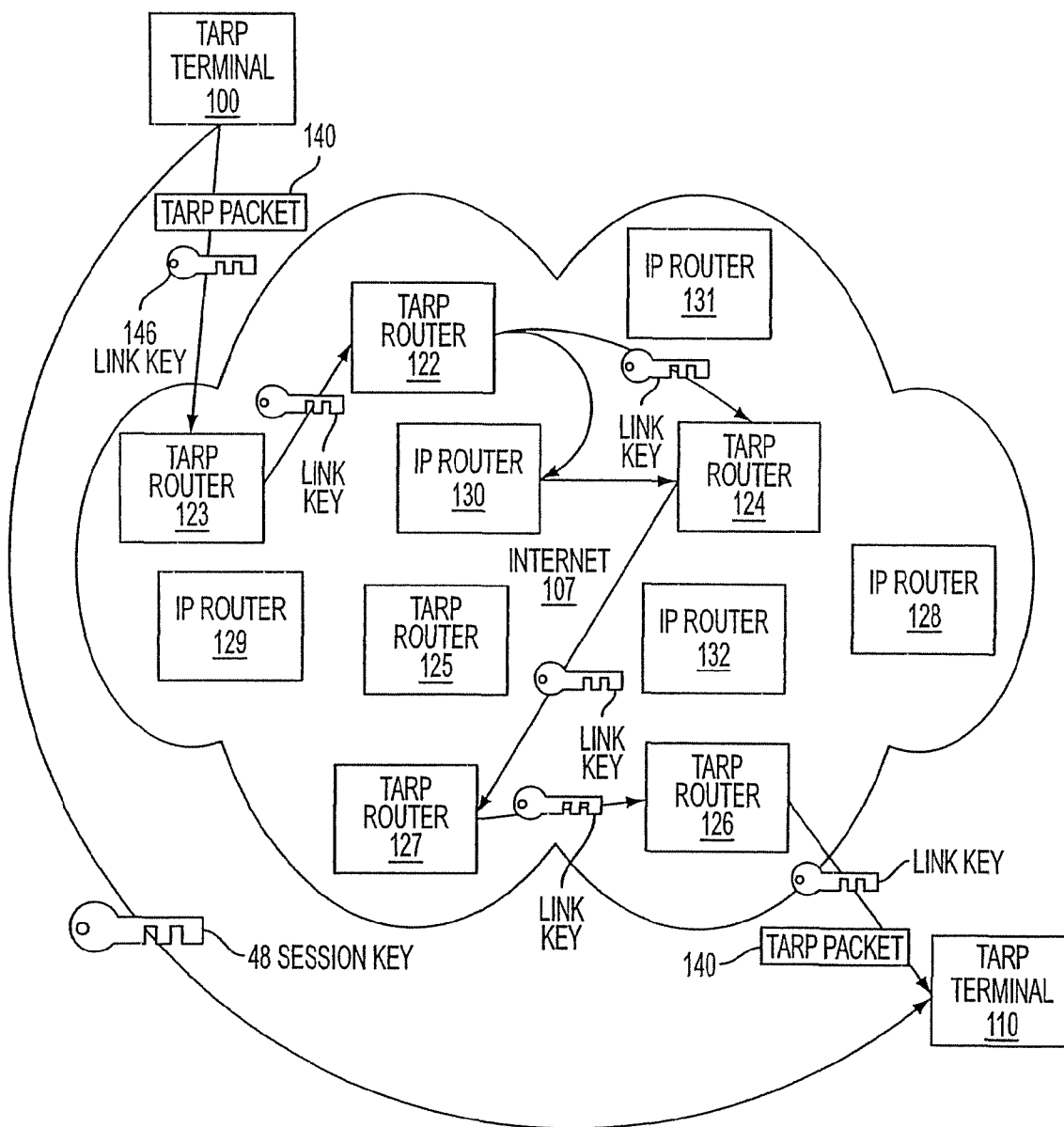


FIG. 2



U.S. Patent

Apr. 5, 2011

Sheet 3 of 40

US 7,921,211 B2

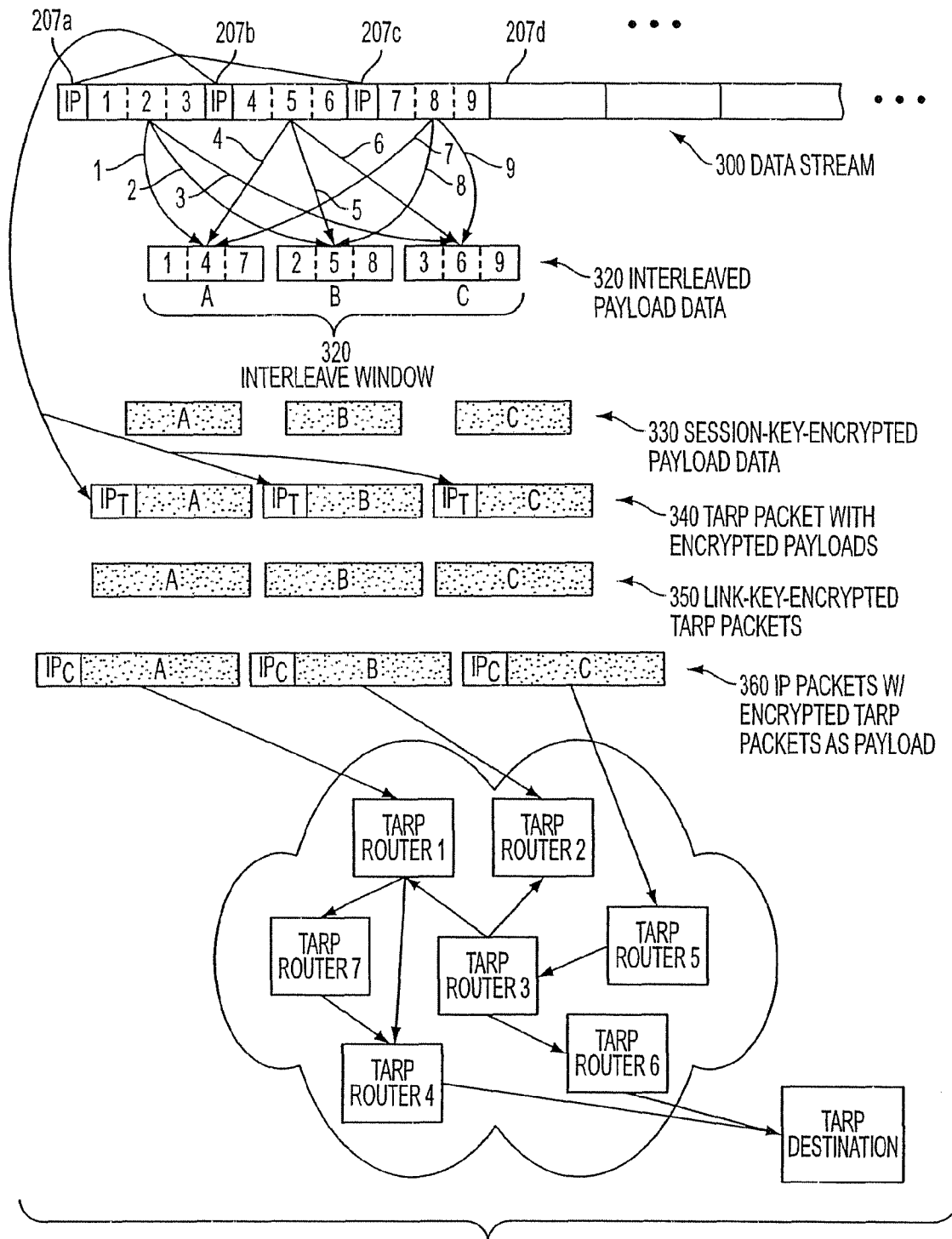


FIG. 3A

Appx336

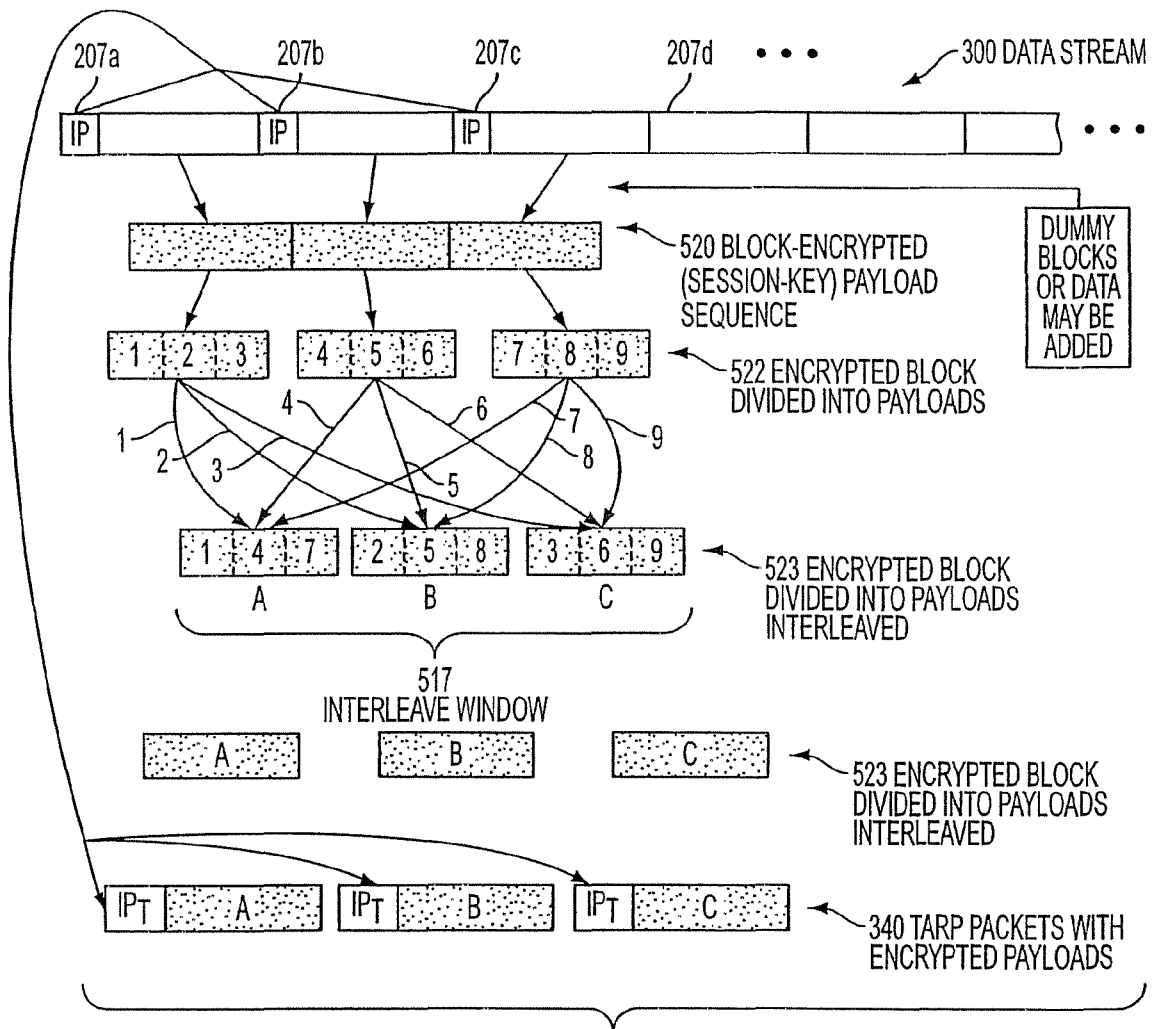


FIG. 3B

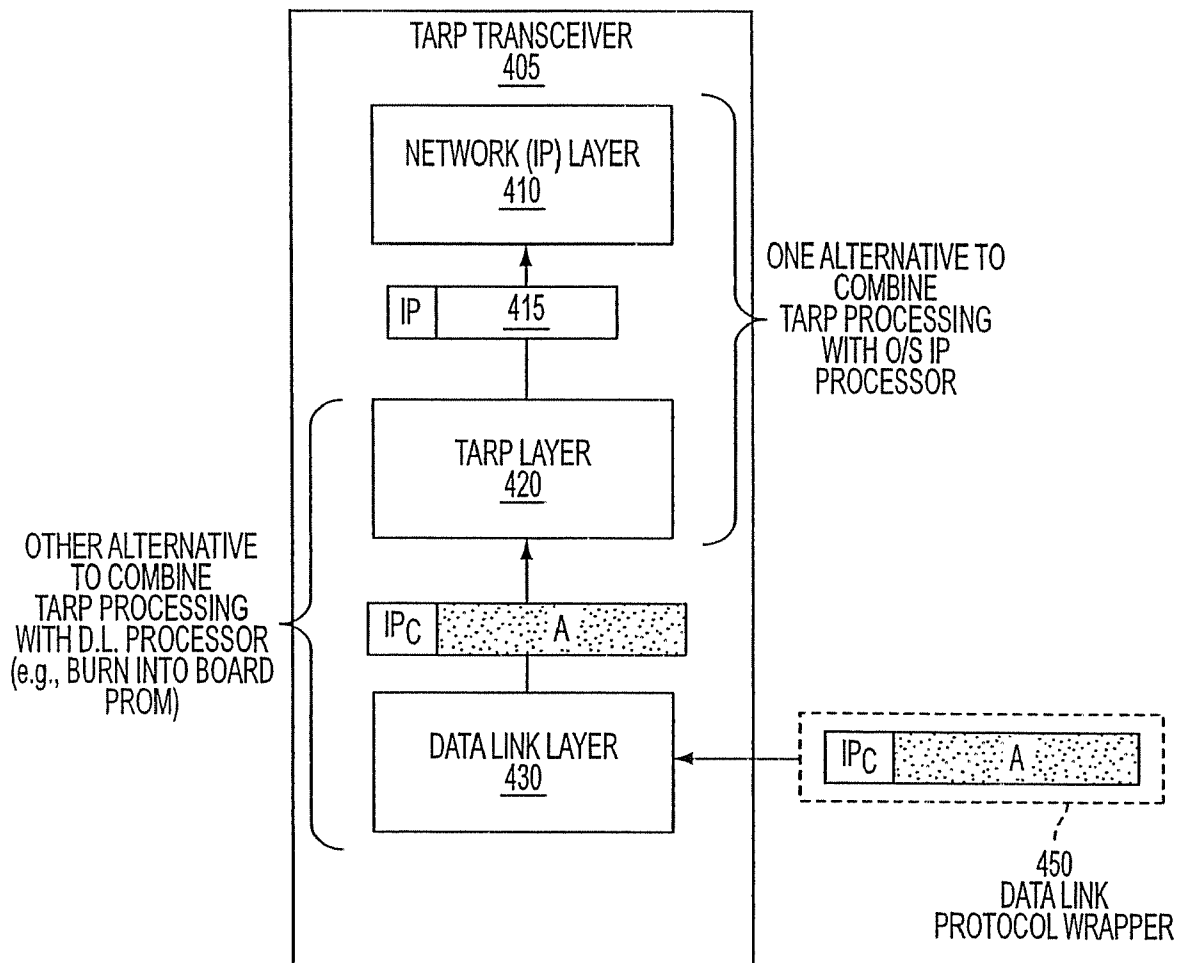


FIG. 4

U.S. Patent

Apr. 5, 2011

Sheet 6 of 40

US 7,921,211 B2

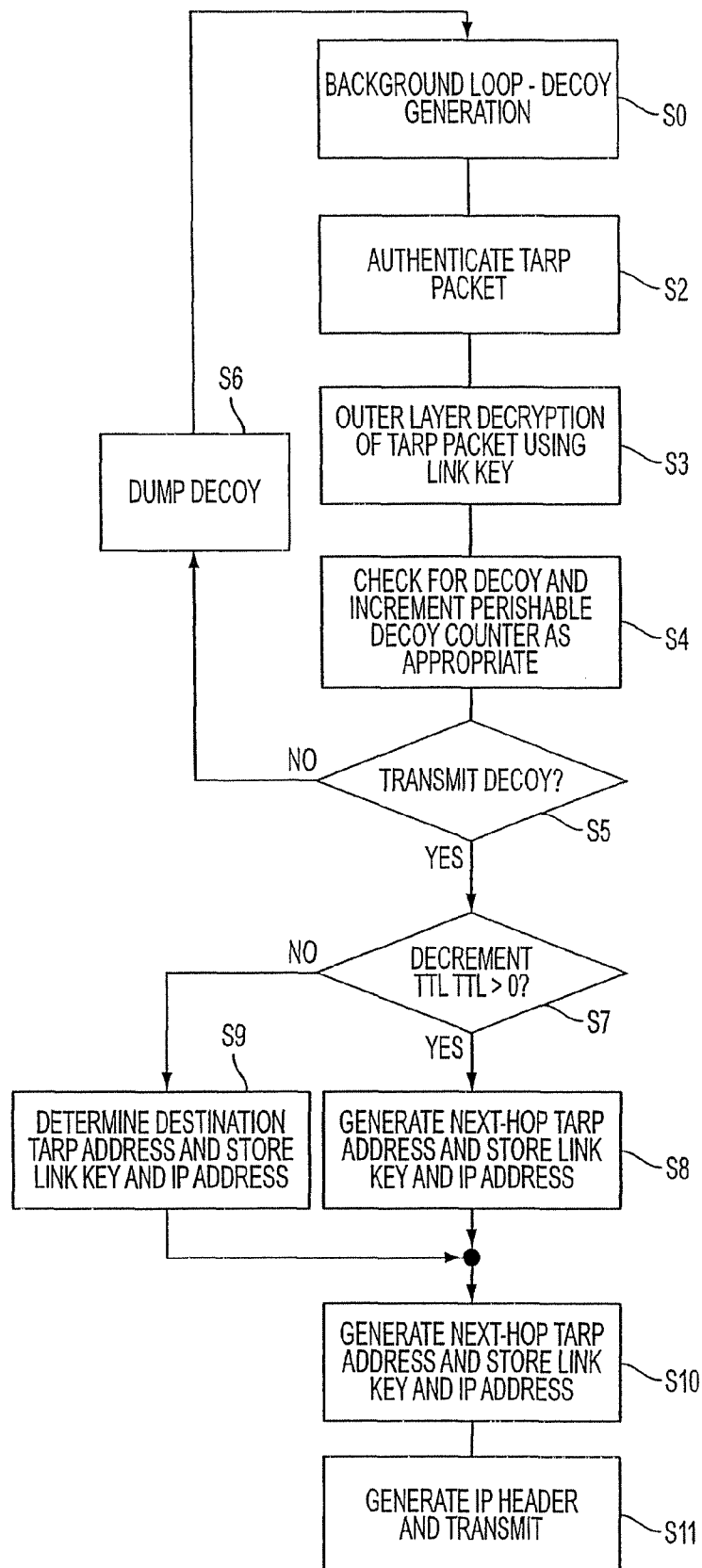


FIG. 5

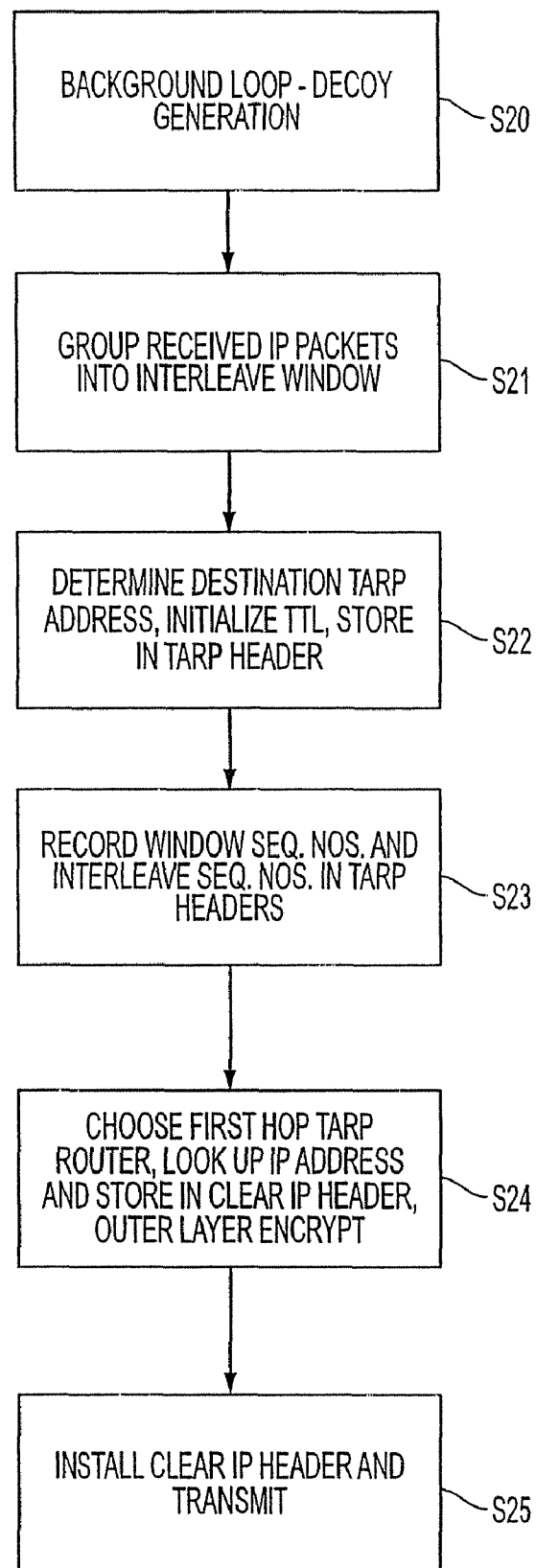
Appx339

**U.S. Patent**

**Apr. 5, 2011**

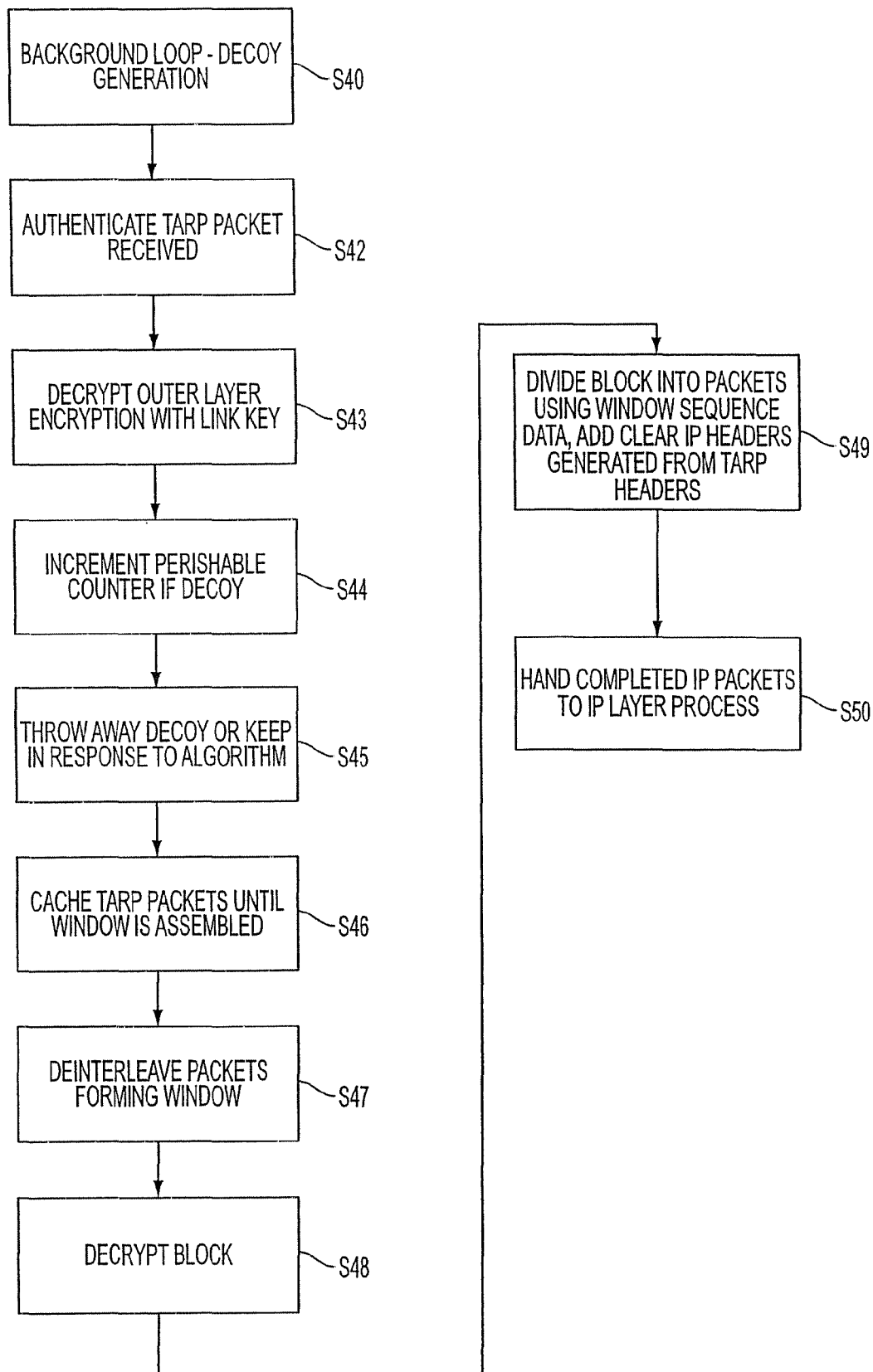
**Sheet 7 of 40**

**US 7,921,211 B2**



**FIG. 6**

**Appx340**

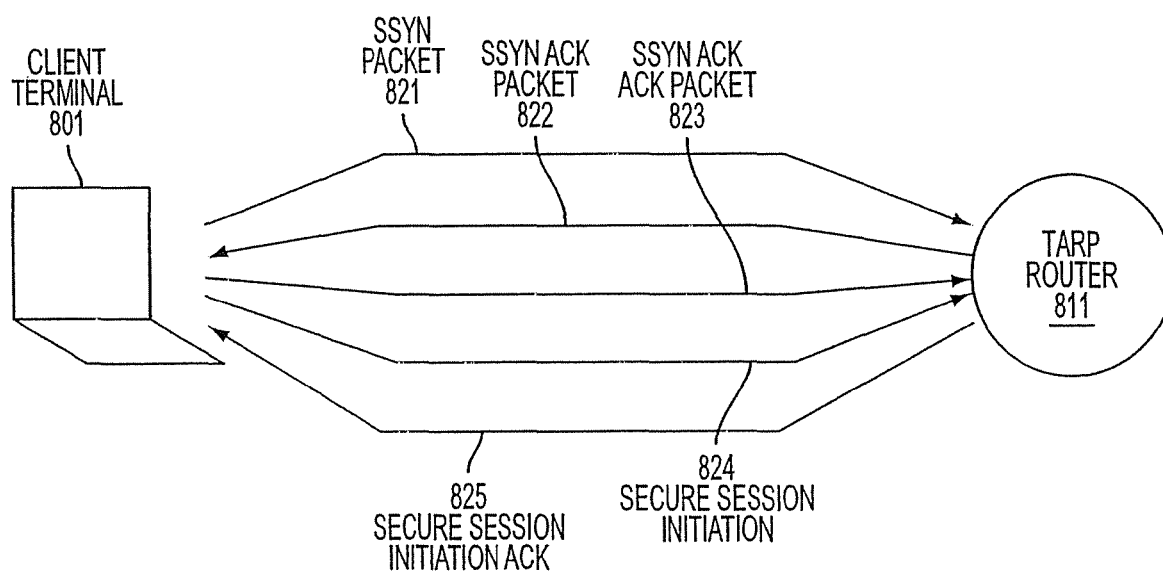
**U.S. Patent****Apr. 5, 2011****Sheet 8 of 40****US 7,921,211 B2****FIG. 7****Appx341**

**U.S. Patent**

**Apr. 5, 2011**

**Sheet 9 of 40**

**US 7,921,211 B2**

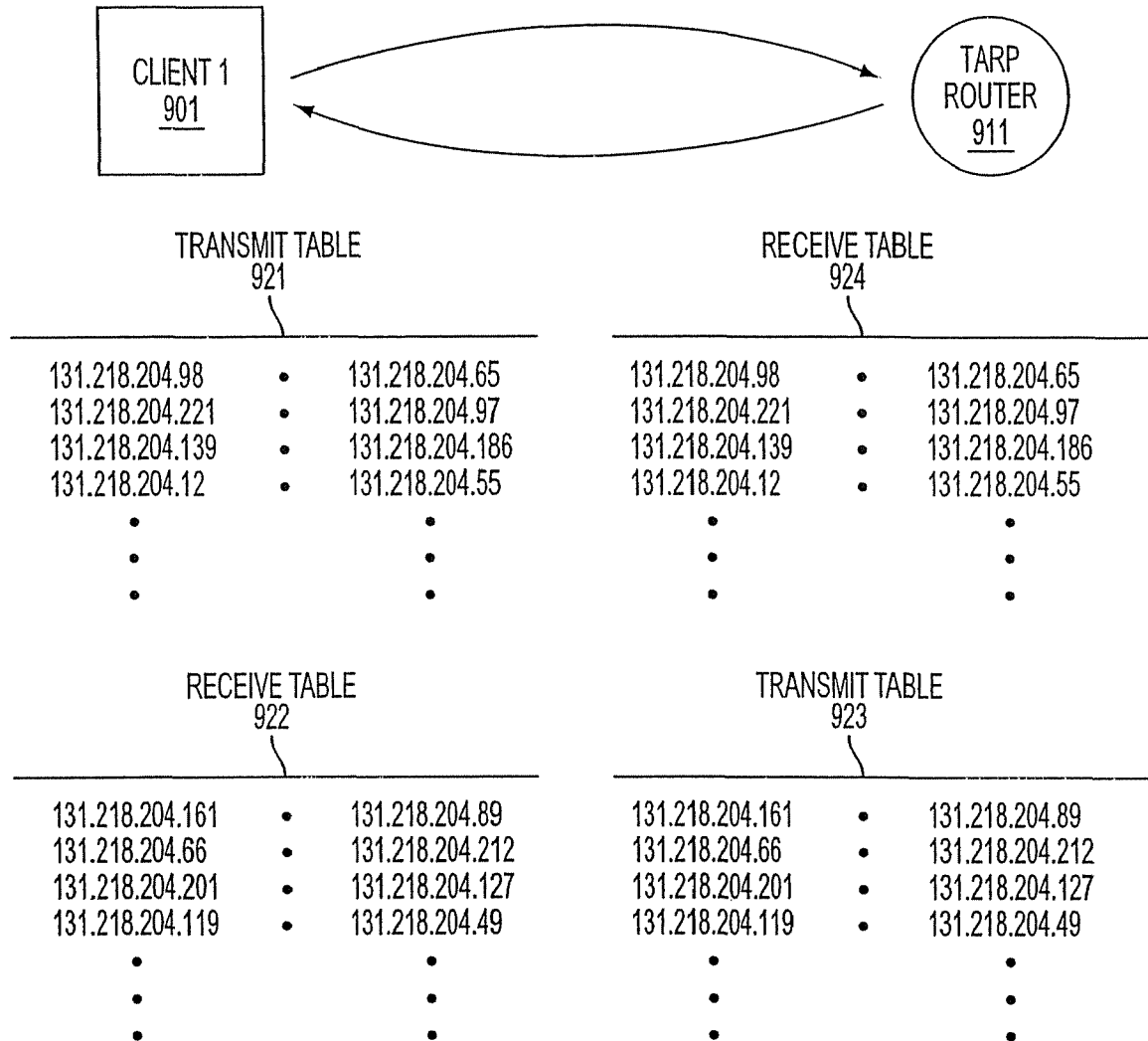


**FIG. 8**

**U.S. Patent**

Apr. 5, 2011

Sheet 10 of 40

**US 7,921,211 B2****FIG. 9**

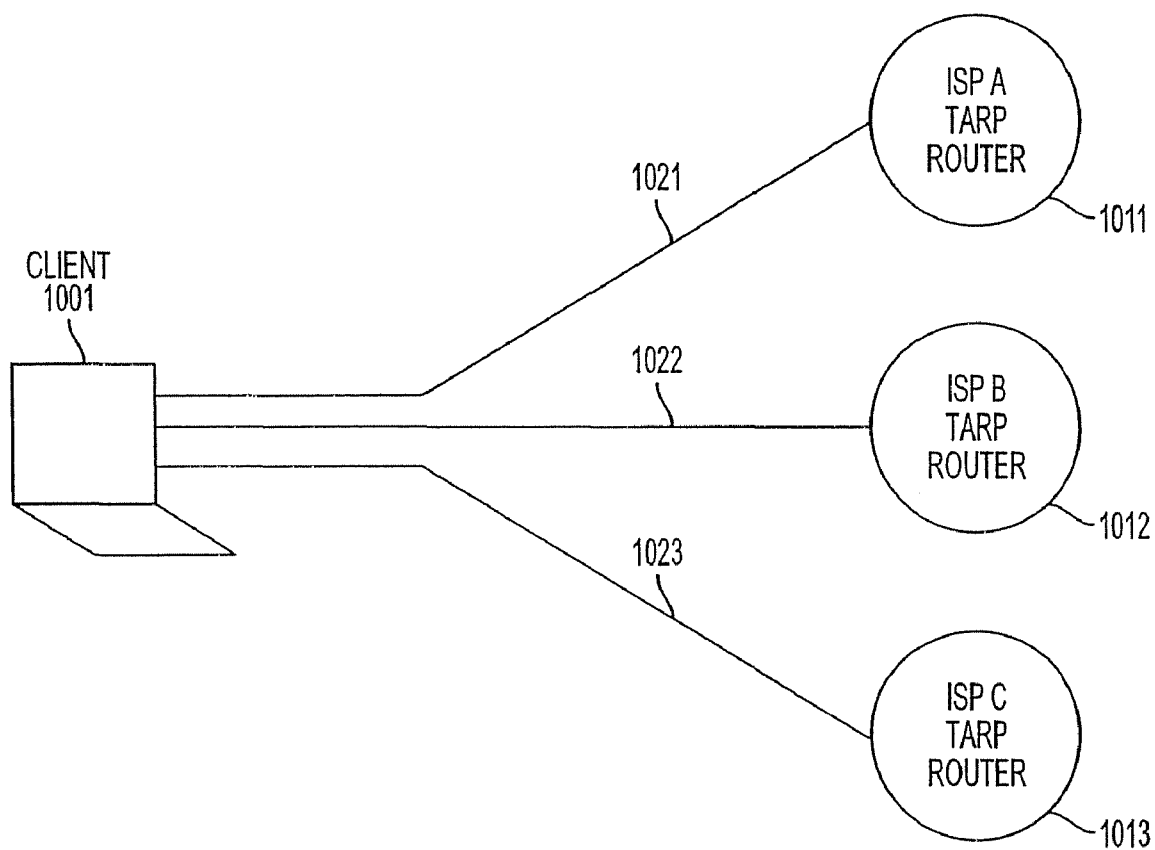


**U.S. Patent**

**Apr. 5, 2011**

**Sheet 11 of 40**

**US 7,921,211 B2**



**FIG. 10**

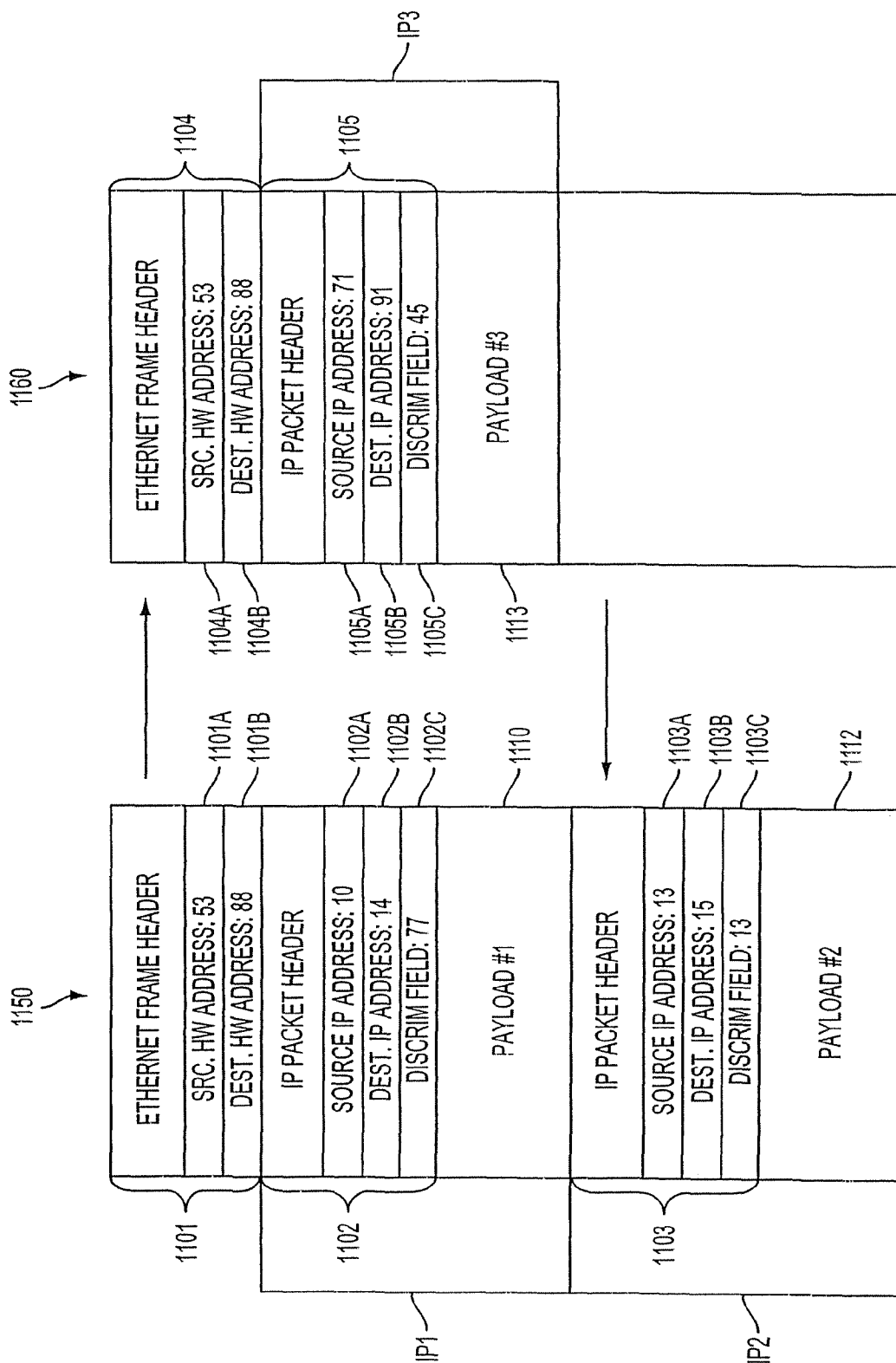


FIG. 11

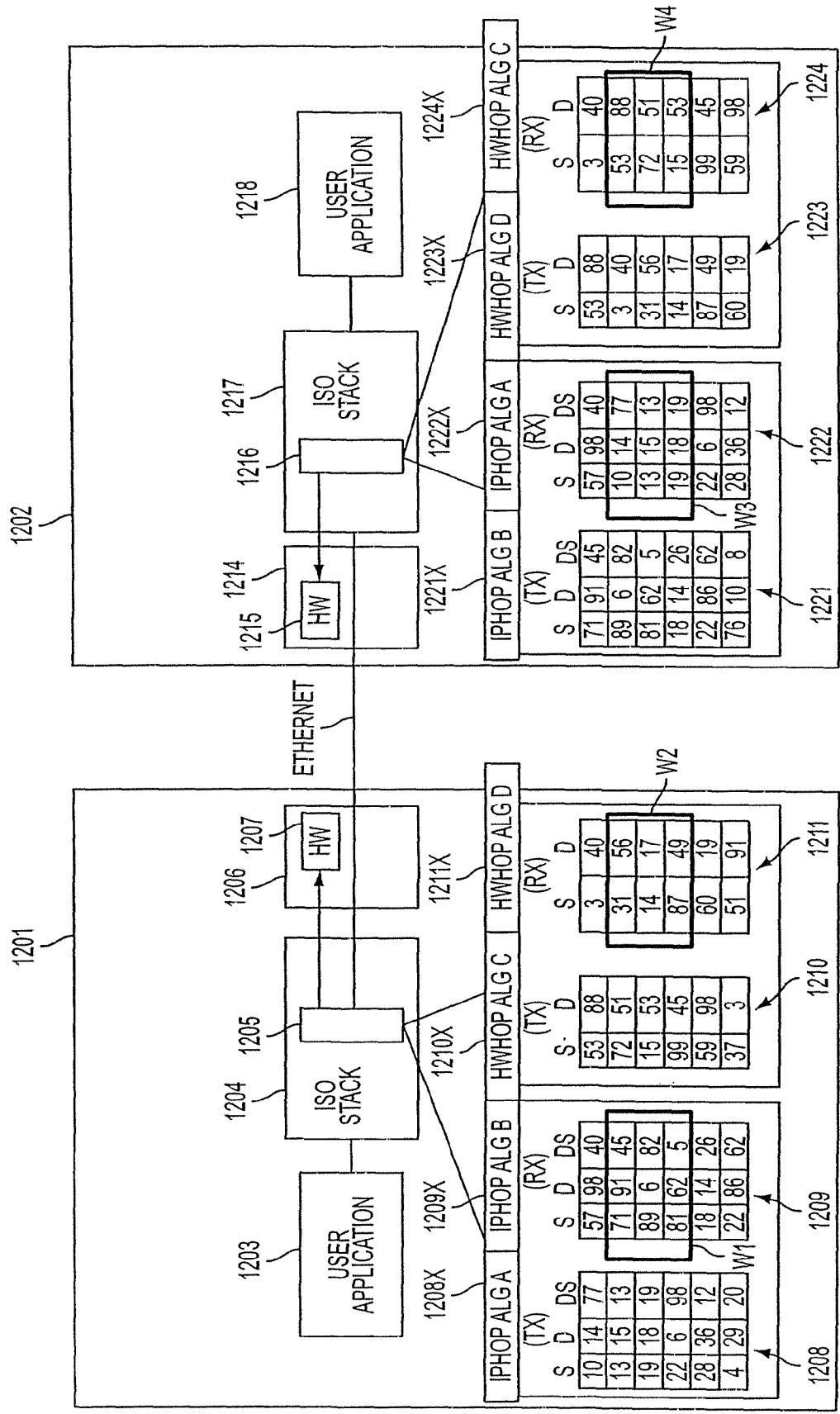


FIG. 12A

**U.S. Patent****Apr. 5, 2011****Sheet 14 of 40****US 7,921,211 B2**

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

**FIG. 12B**

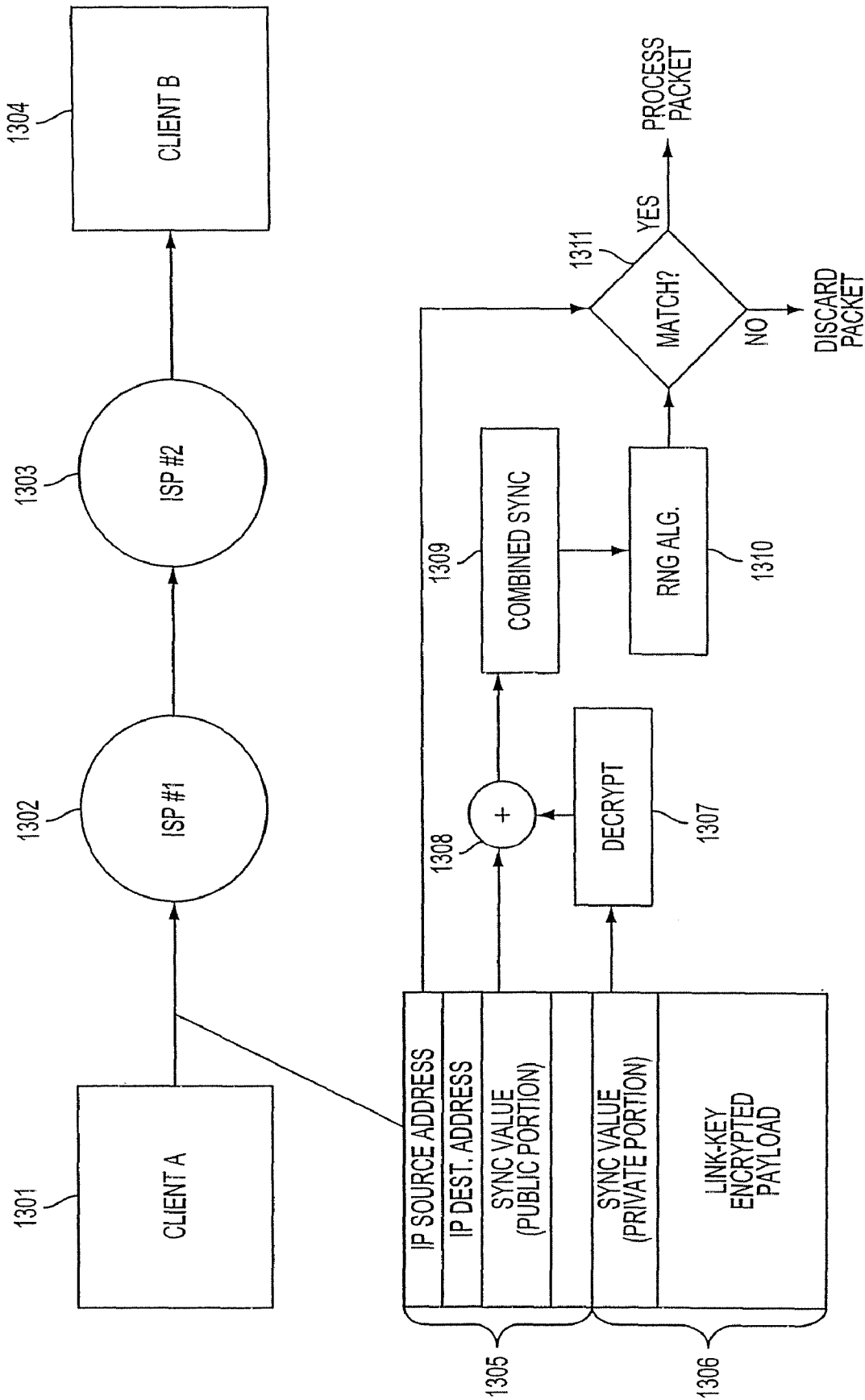


FIG. 13

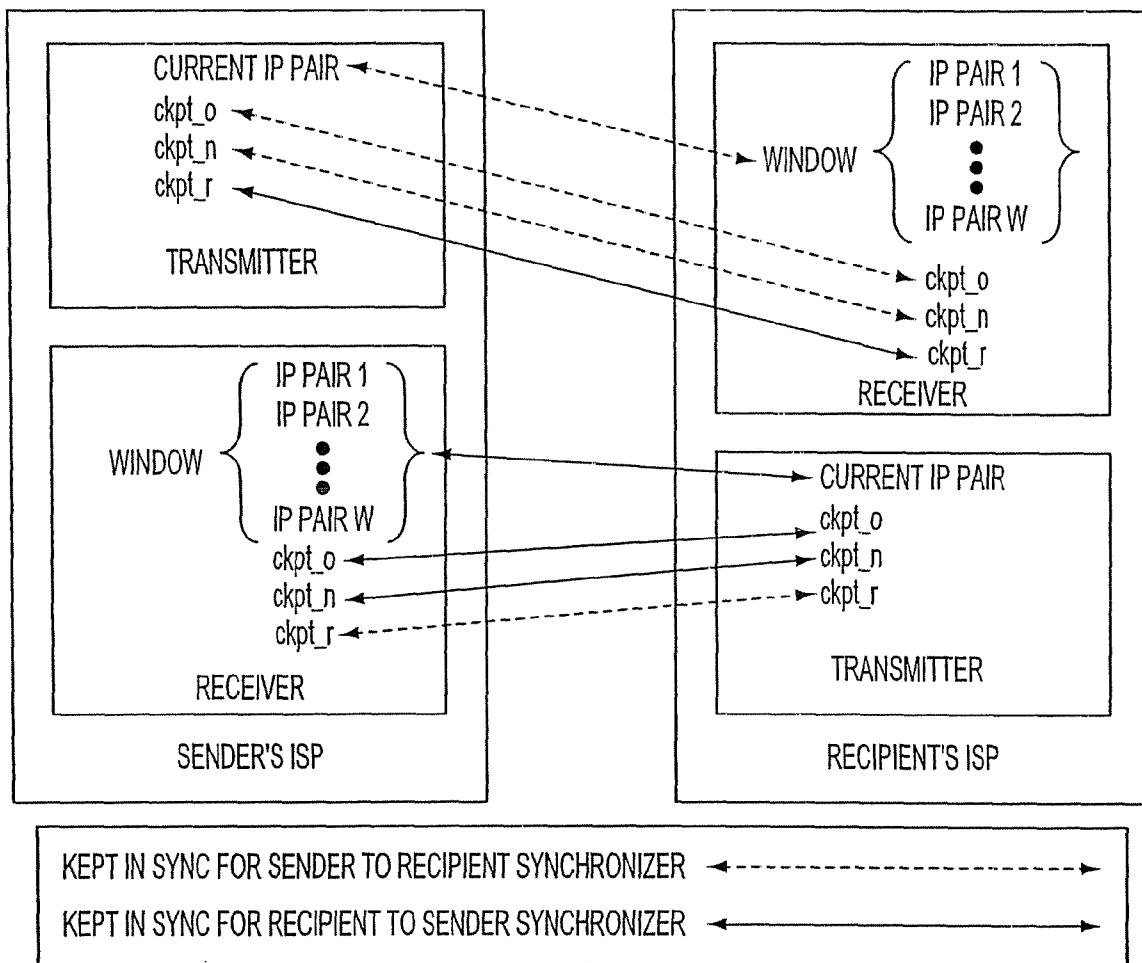


FIG. 14

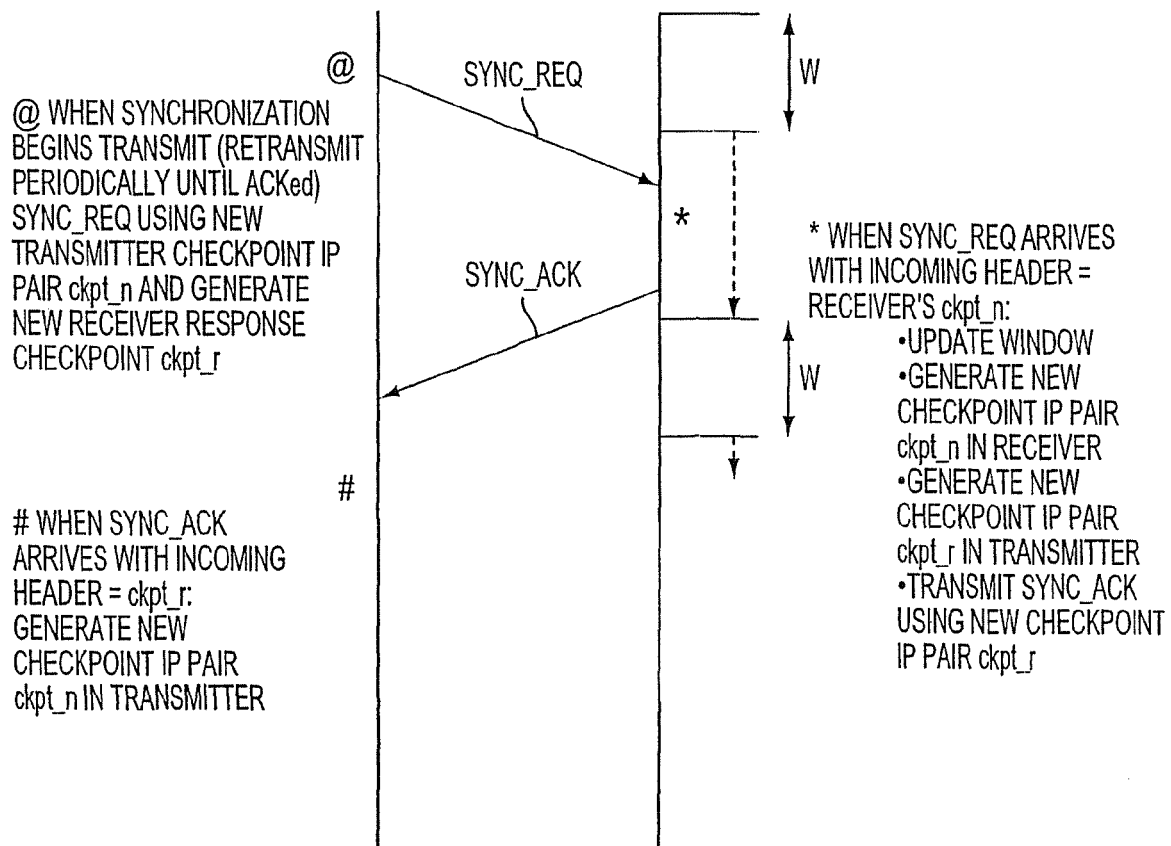


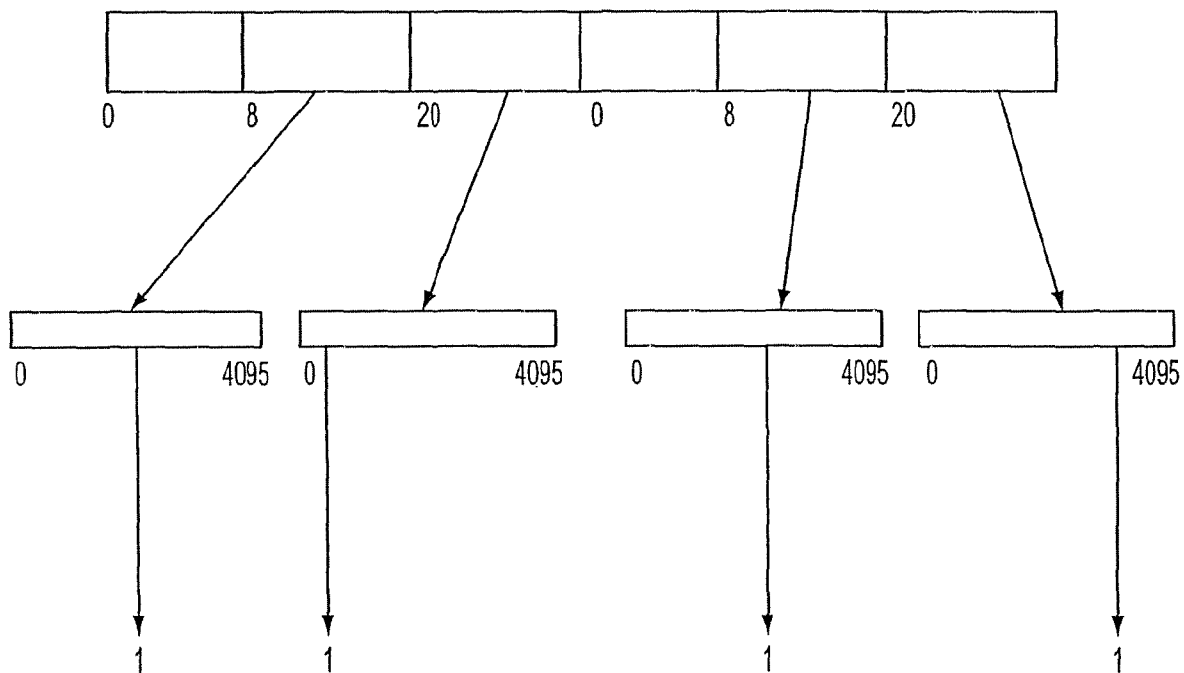
FIG. 15

**U.S. Patent**

**Apr. 5, 2011**

**Sheet 18 of 40**

**US 7,921,211 B2**



**FIG. 16**

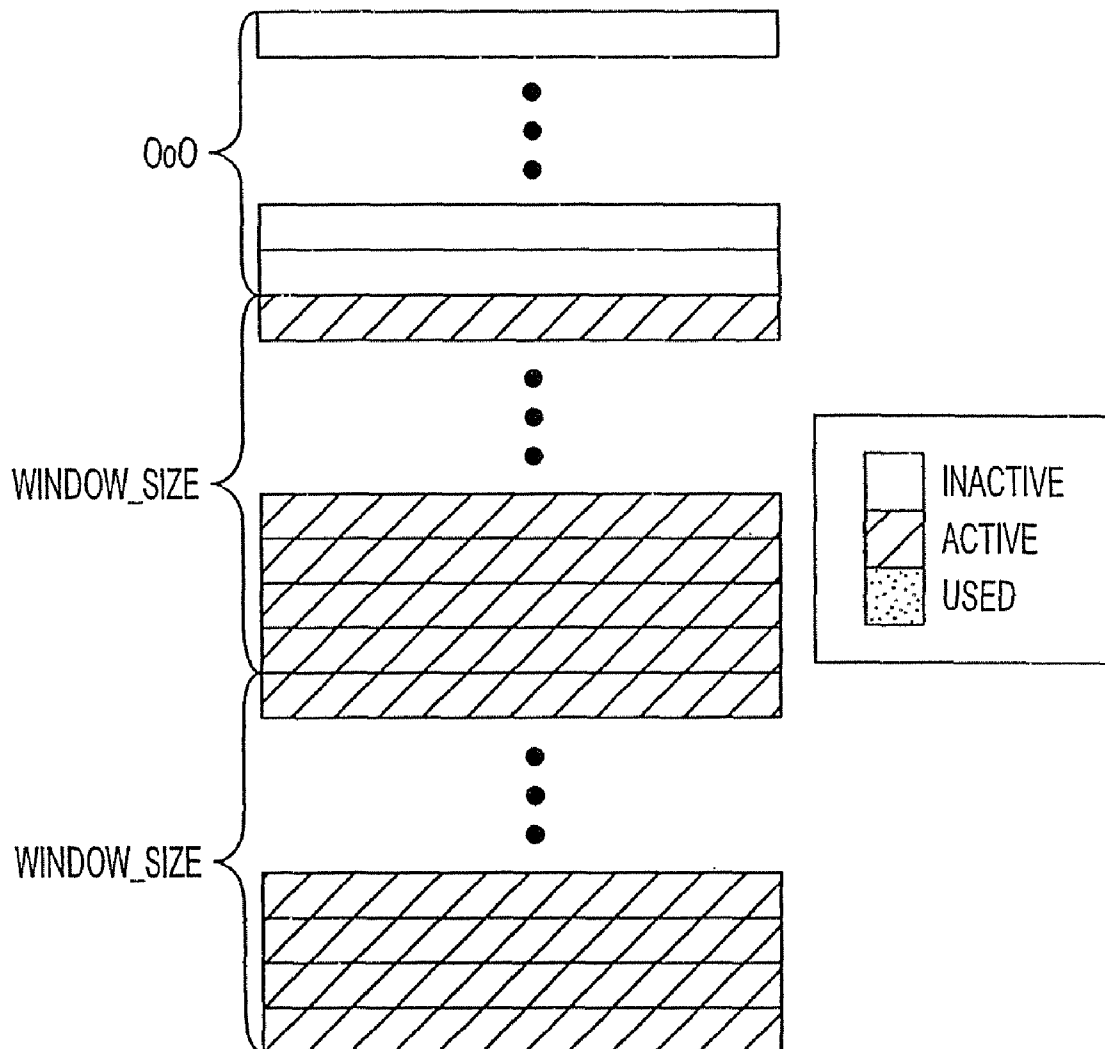


**U.S. Patent**

**Apr. 5, 2011**

**Sheet 19 of 40**

**US 7,921,211 B2**



**FIG. 17**

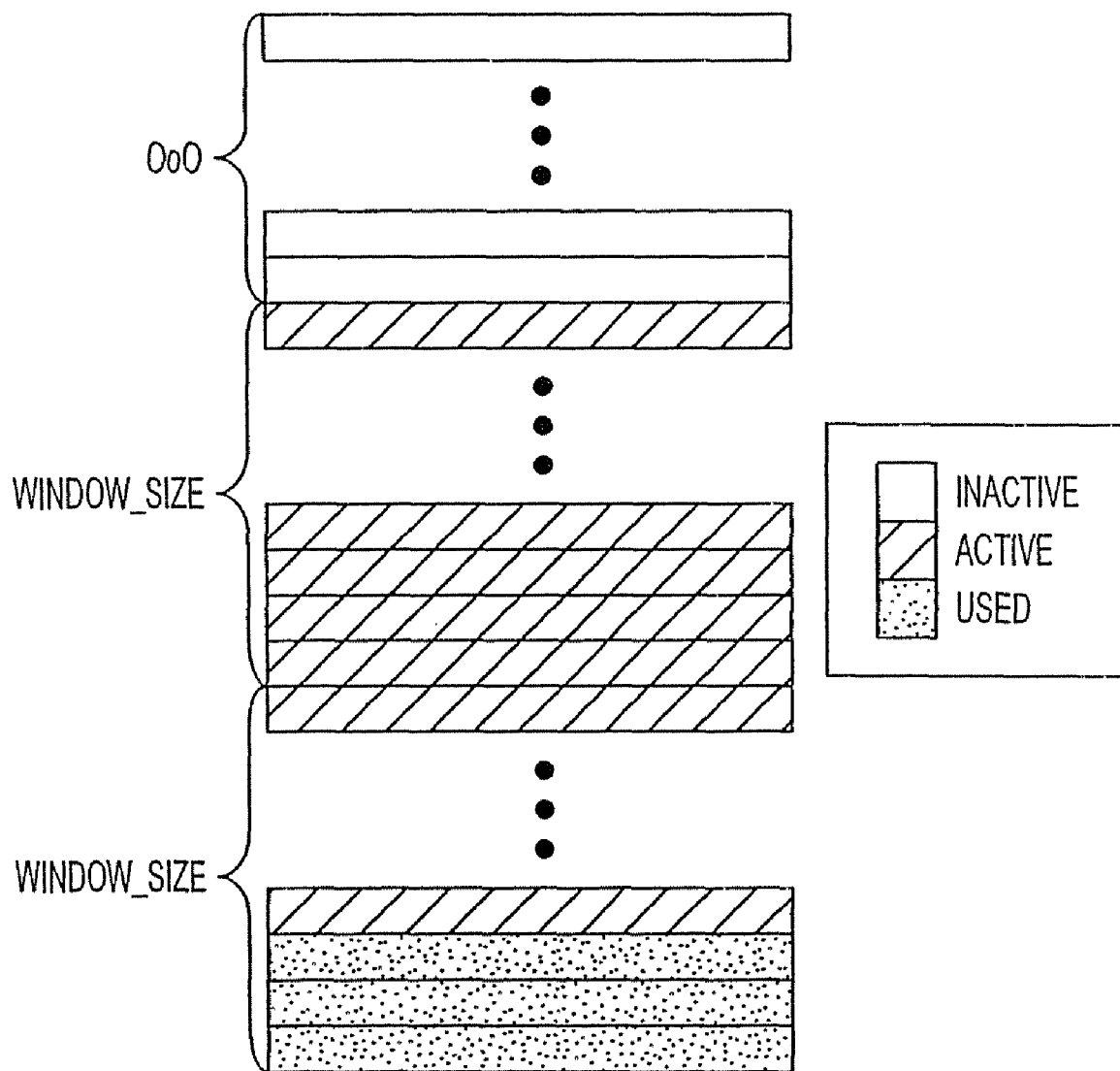
**Appx352**

**U.S. Patent**

Apr. 5, 2011

Sheet 20 of 40

**US 7,921,211 B2**



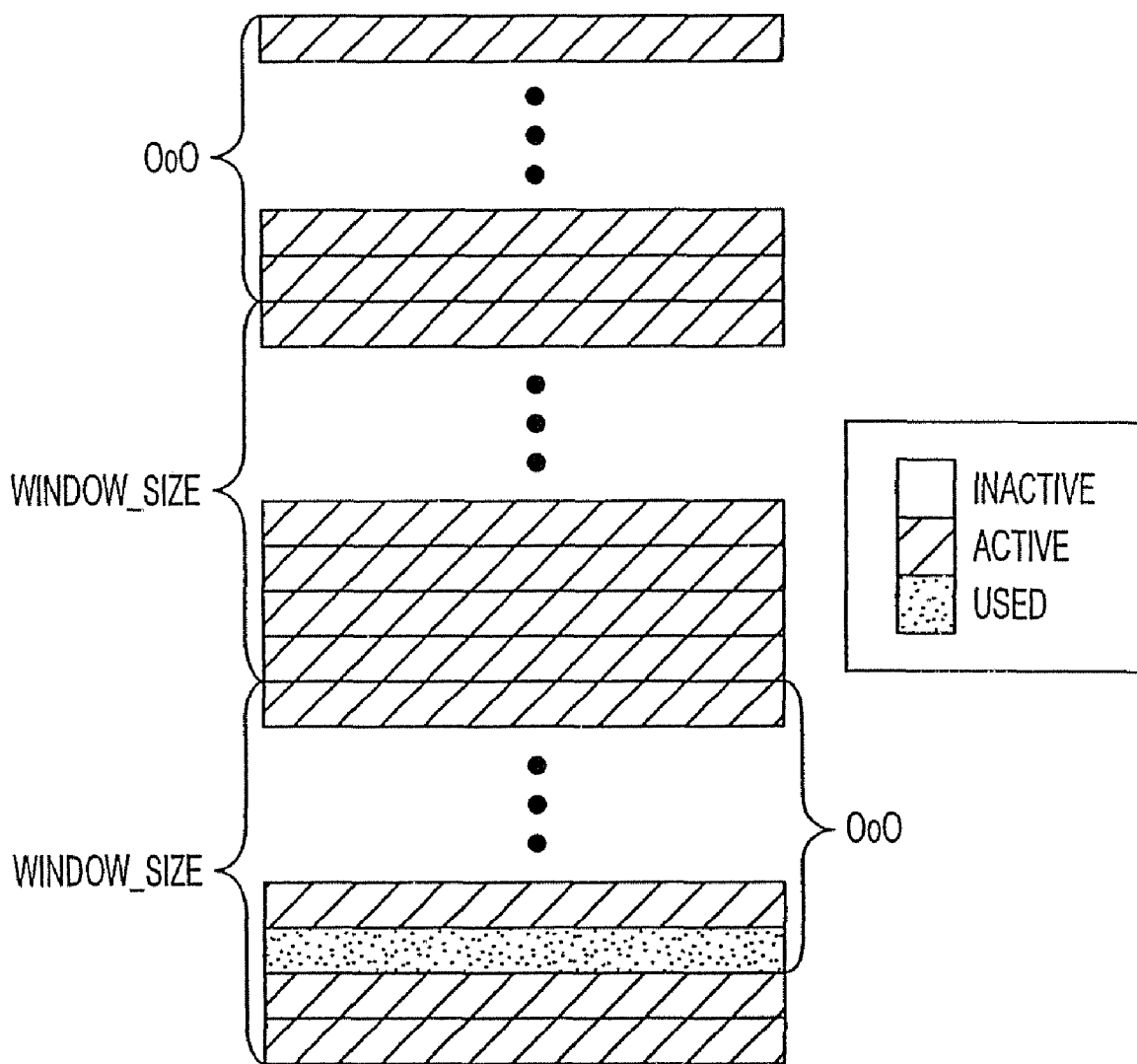
**FIG. 18**

**U.S. Patent**

**Apr. 5, 2011**

**Sheet 21 of 40**

**US 7,921,211 B2**



**FIG. 19**

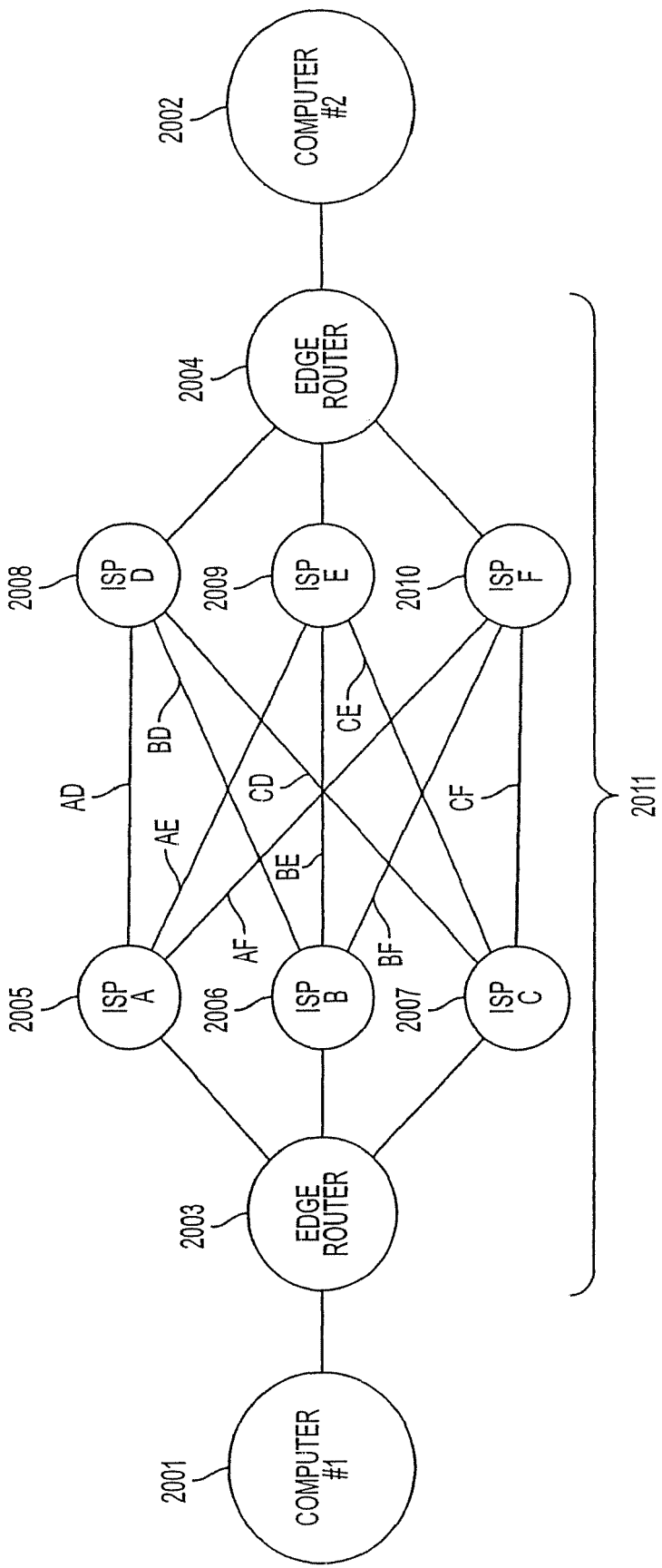
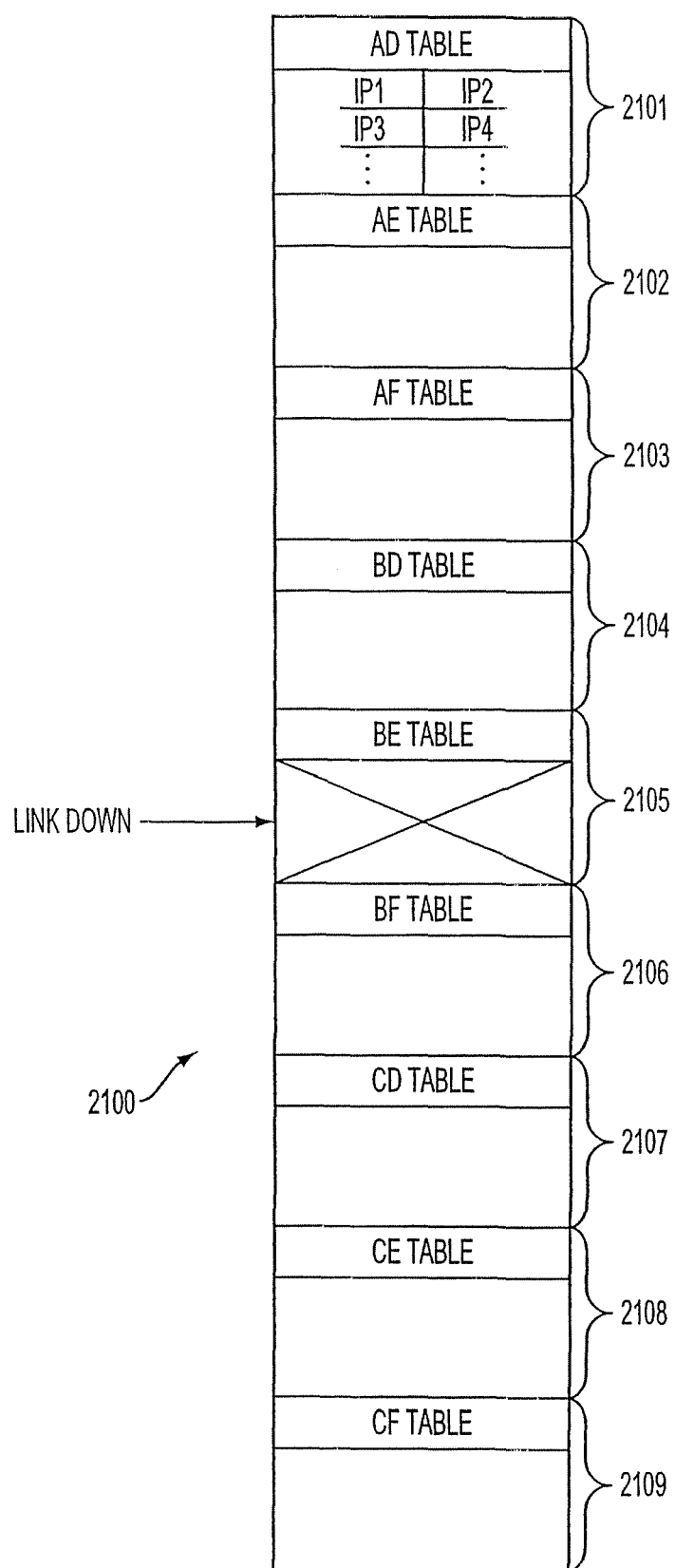


FIG. 20

**U.S. Patent****Apr. 5, 2011****Sheet 23 of 40****US 7,921,211 B2****FIG. 21****Appx356**

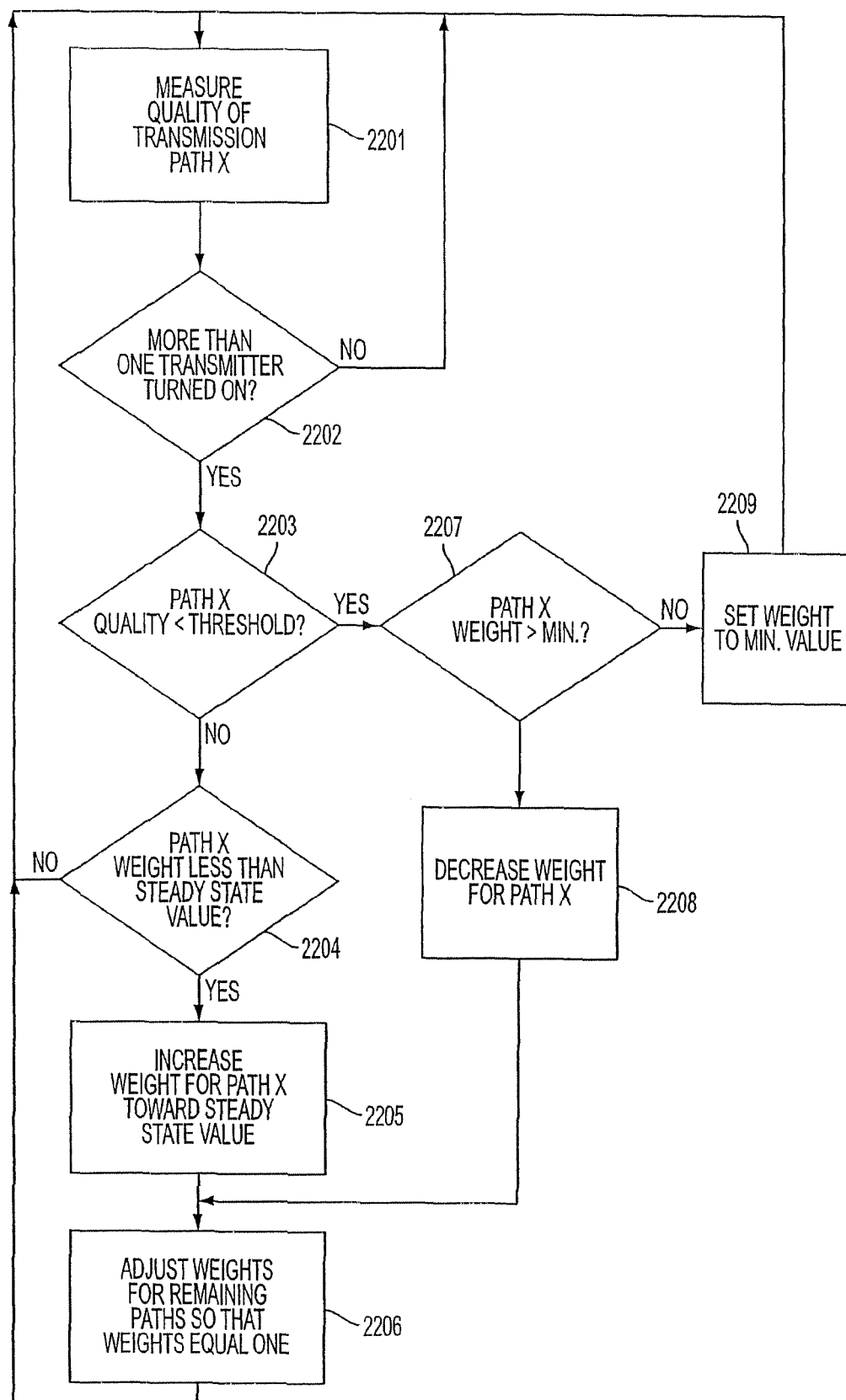


FIG. 22A

U.S. Patent

Apr. 5, 2011

Sheet 25 of 40

US 7,921,211 B2

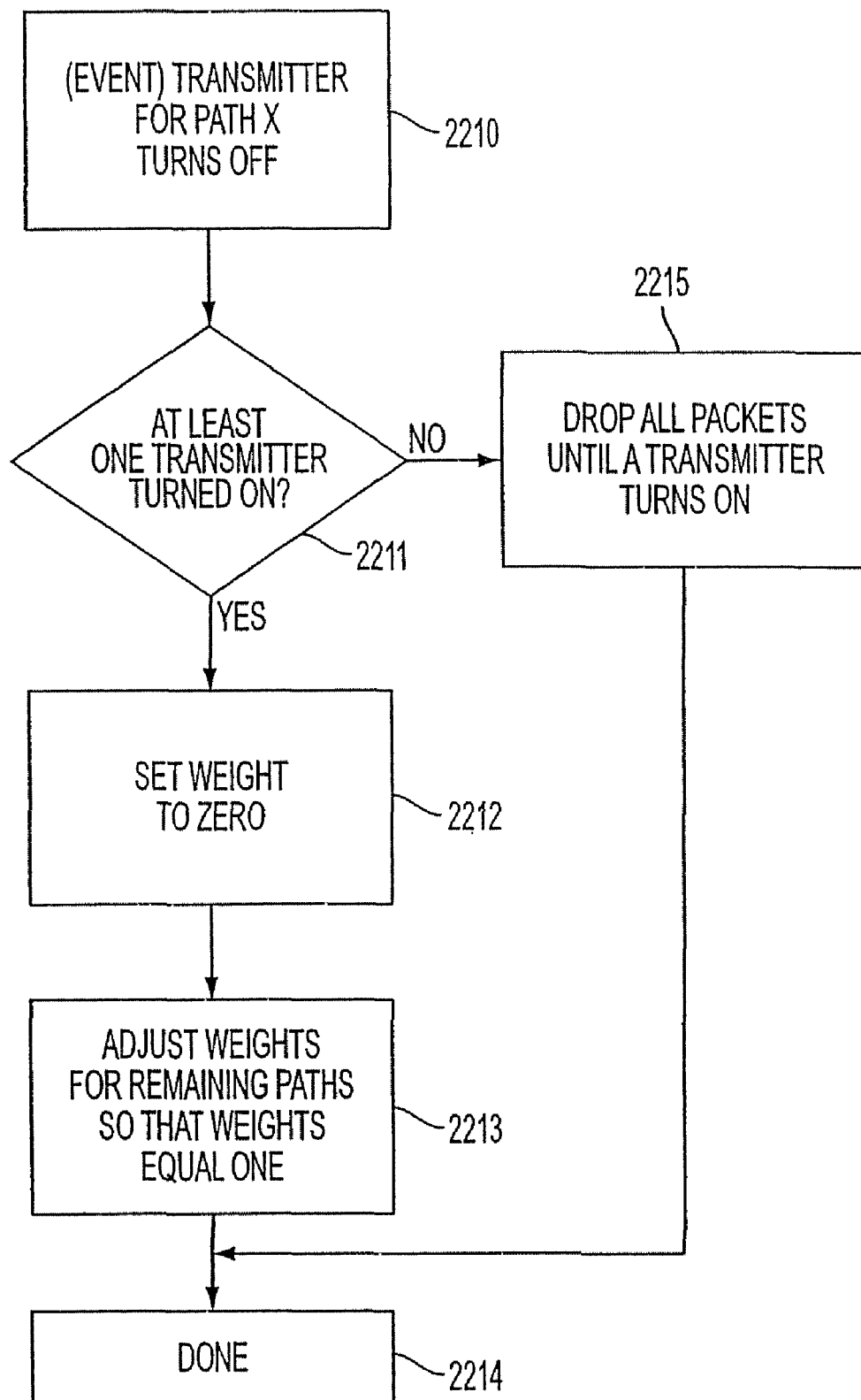


FIG. 22B

Appx358

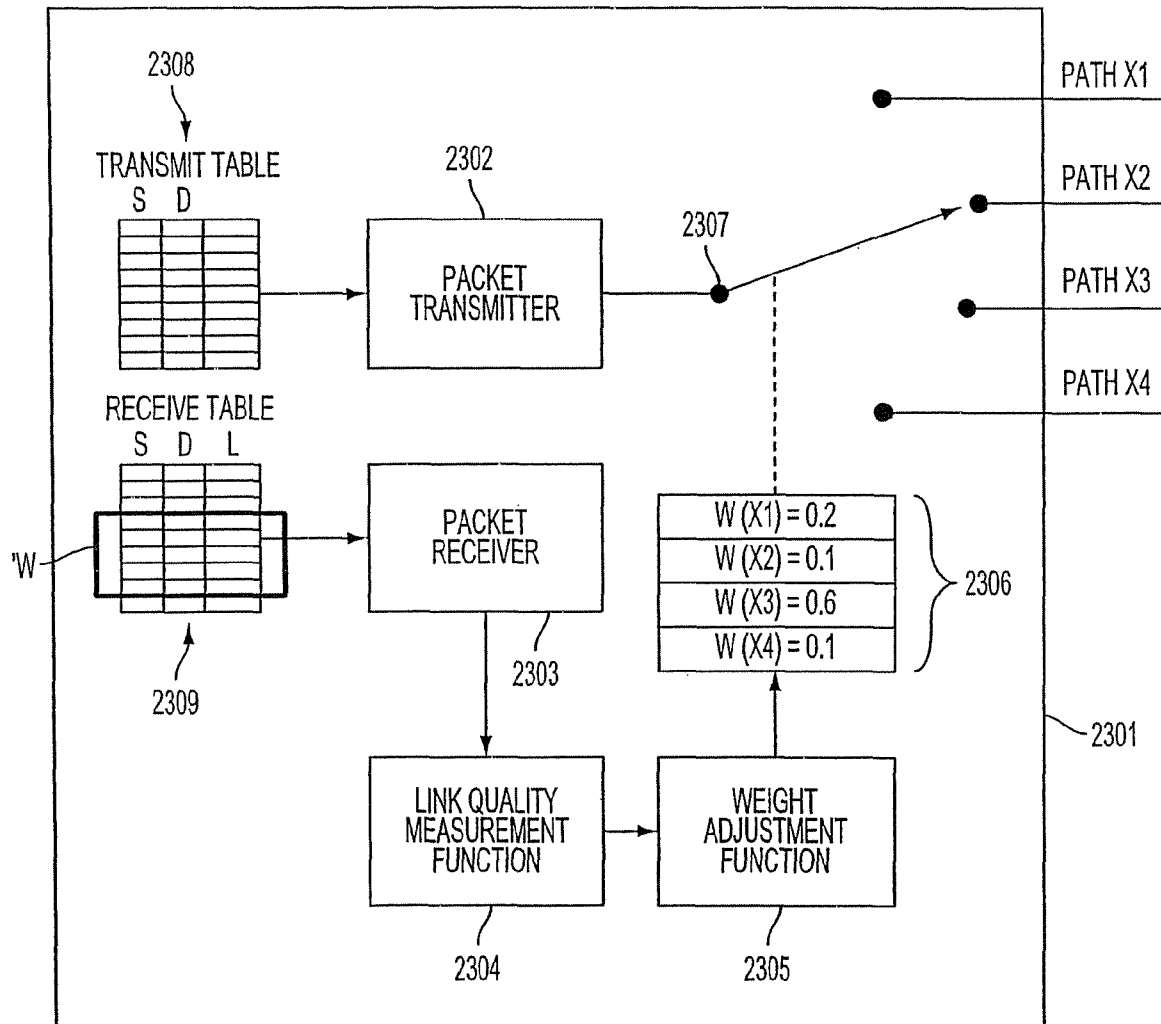


FIG. 23

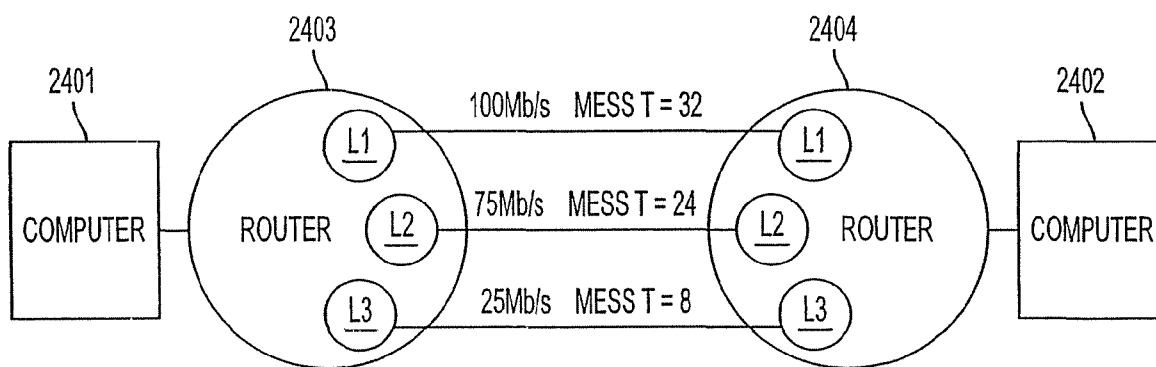


**U.S. Patent**

**Apr. 5, 2011**

**Sheet 27 of 40**

**US 7,921,211 B2**



**FIG. 24**

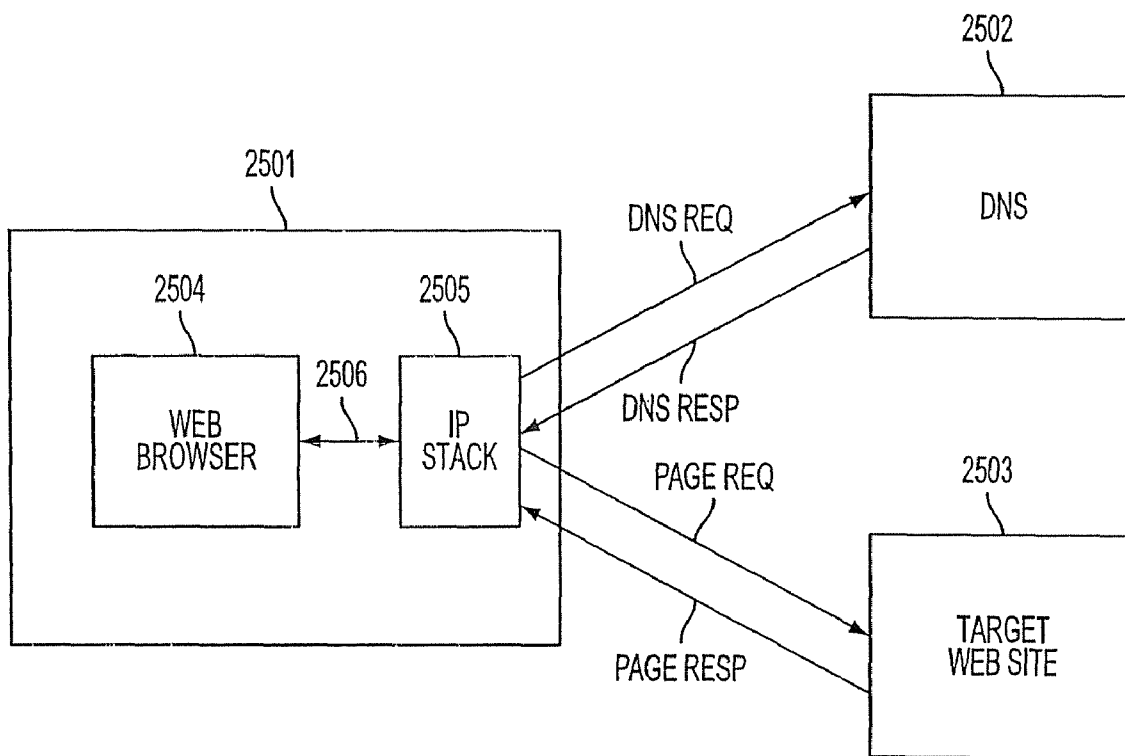


FIG. 25  
(PRIOR ART)

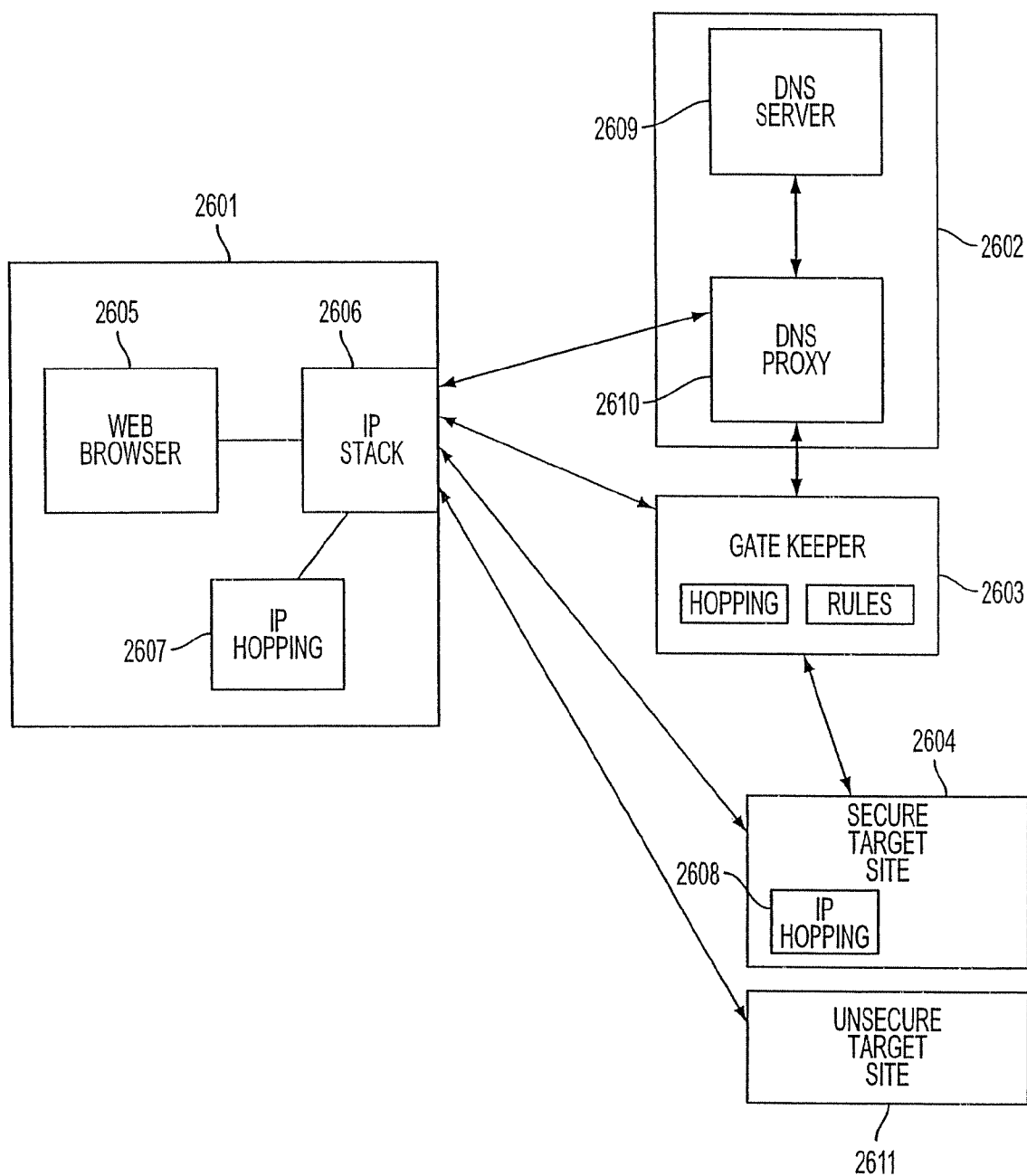


FIG. 26

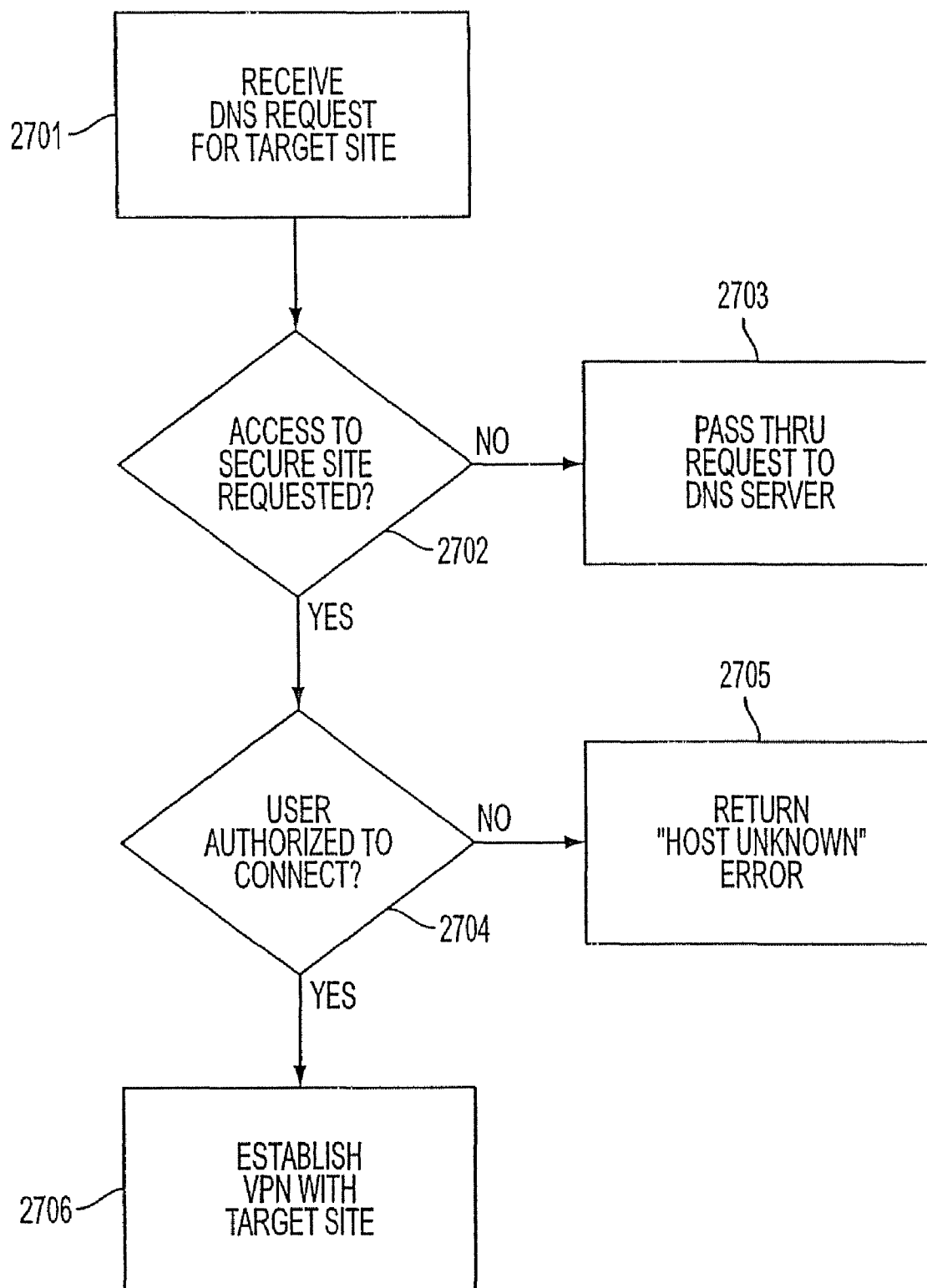


FIG. 27

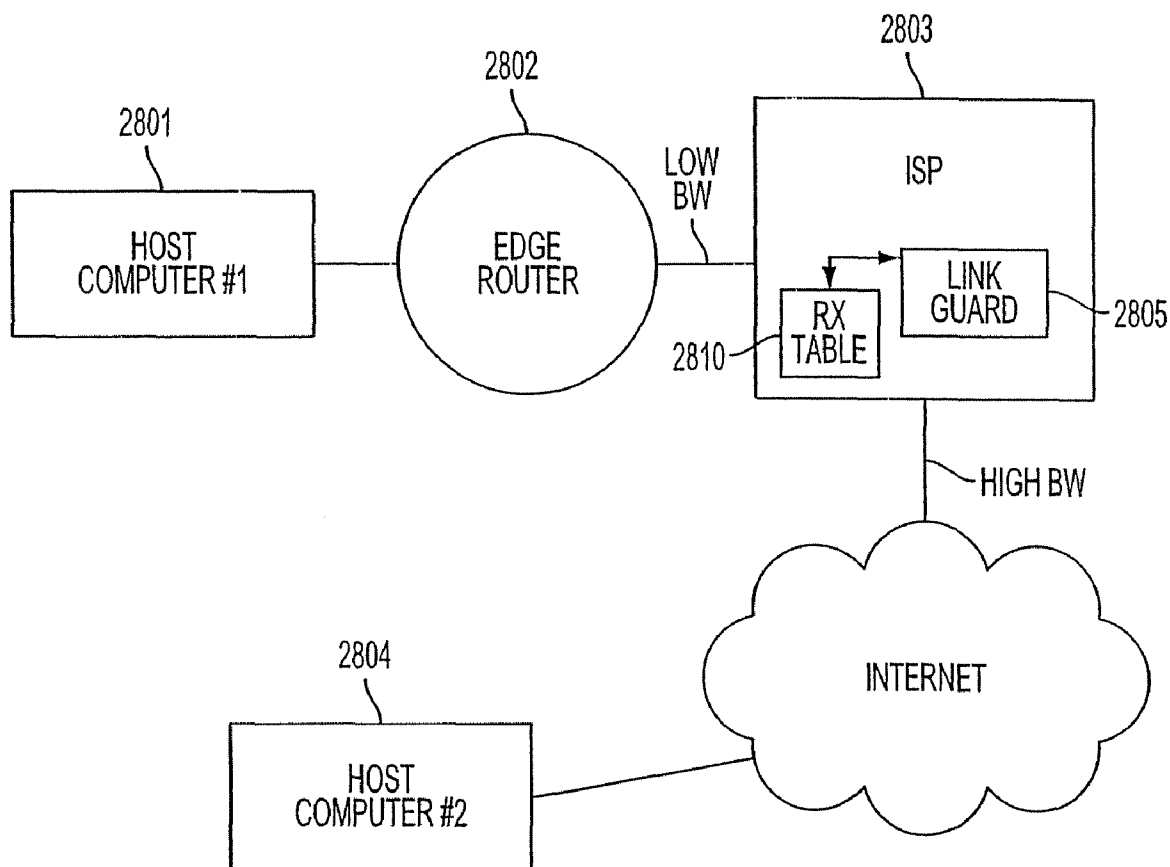


FIG. 28

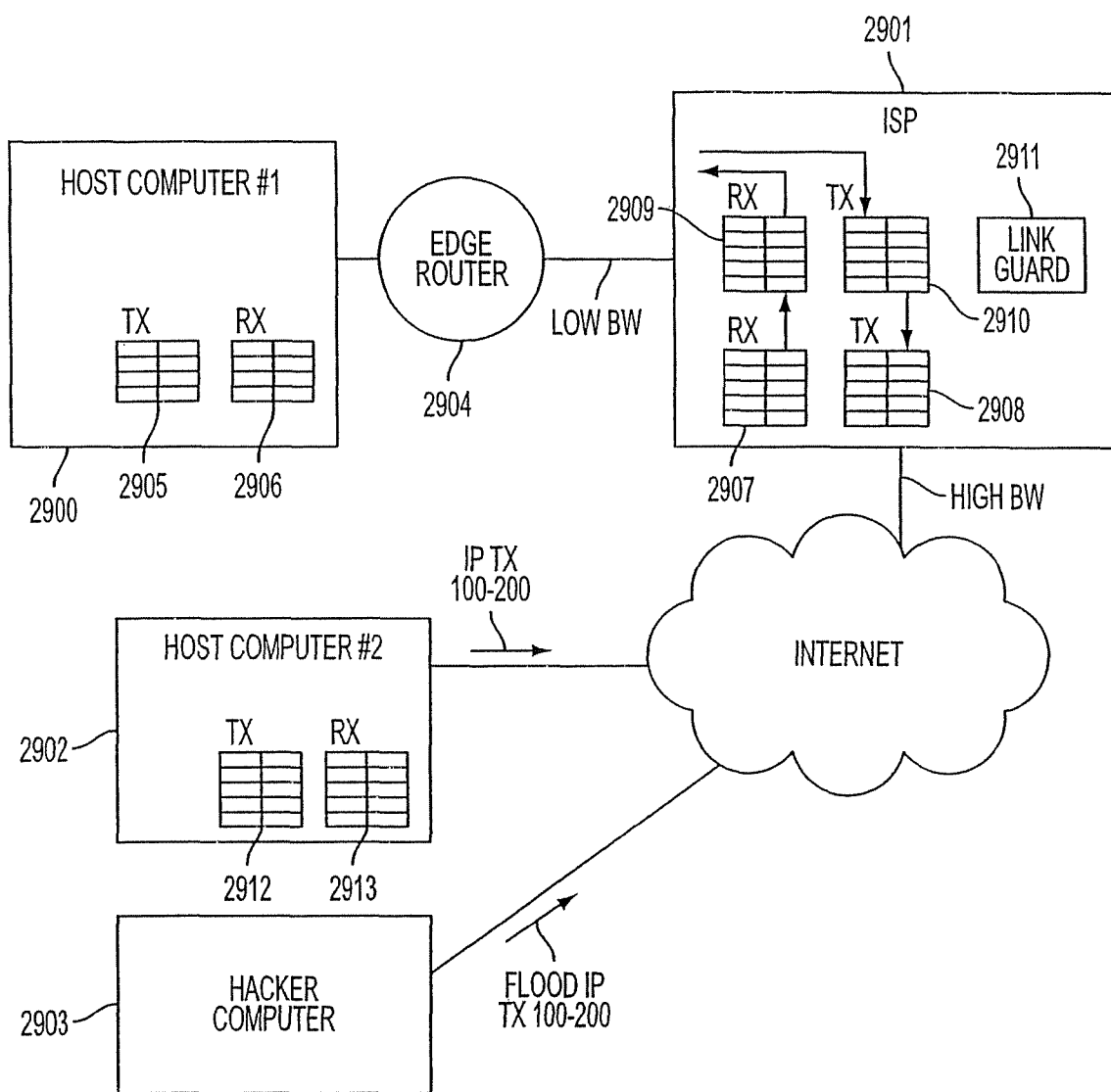


FIG. 29

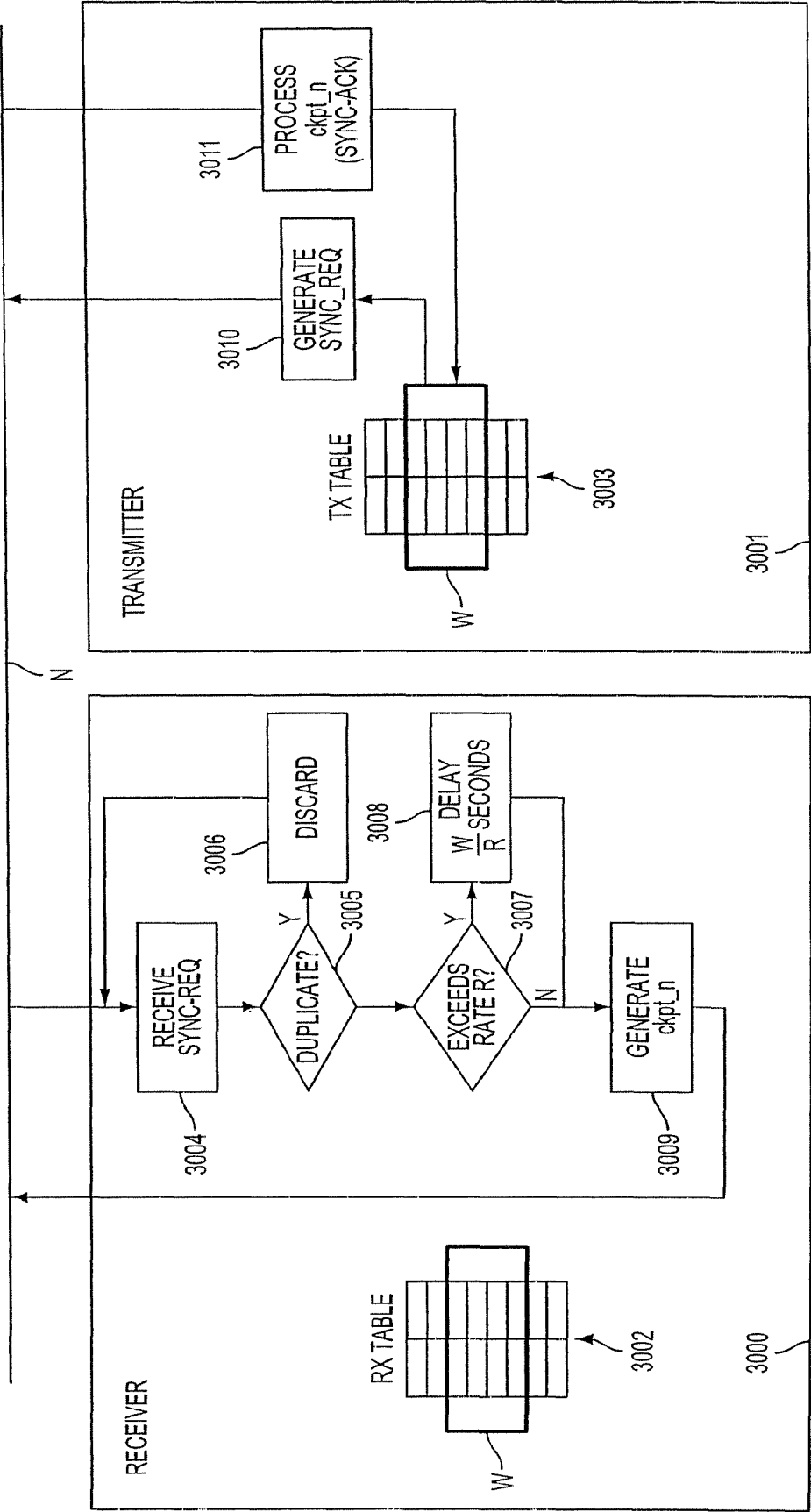


FIG. 30

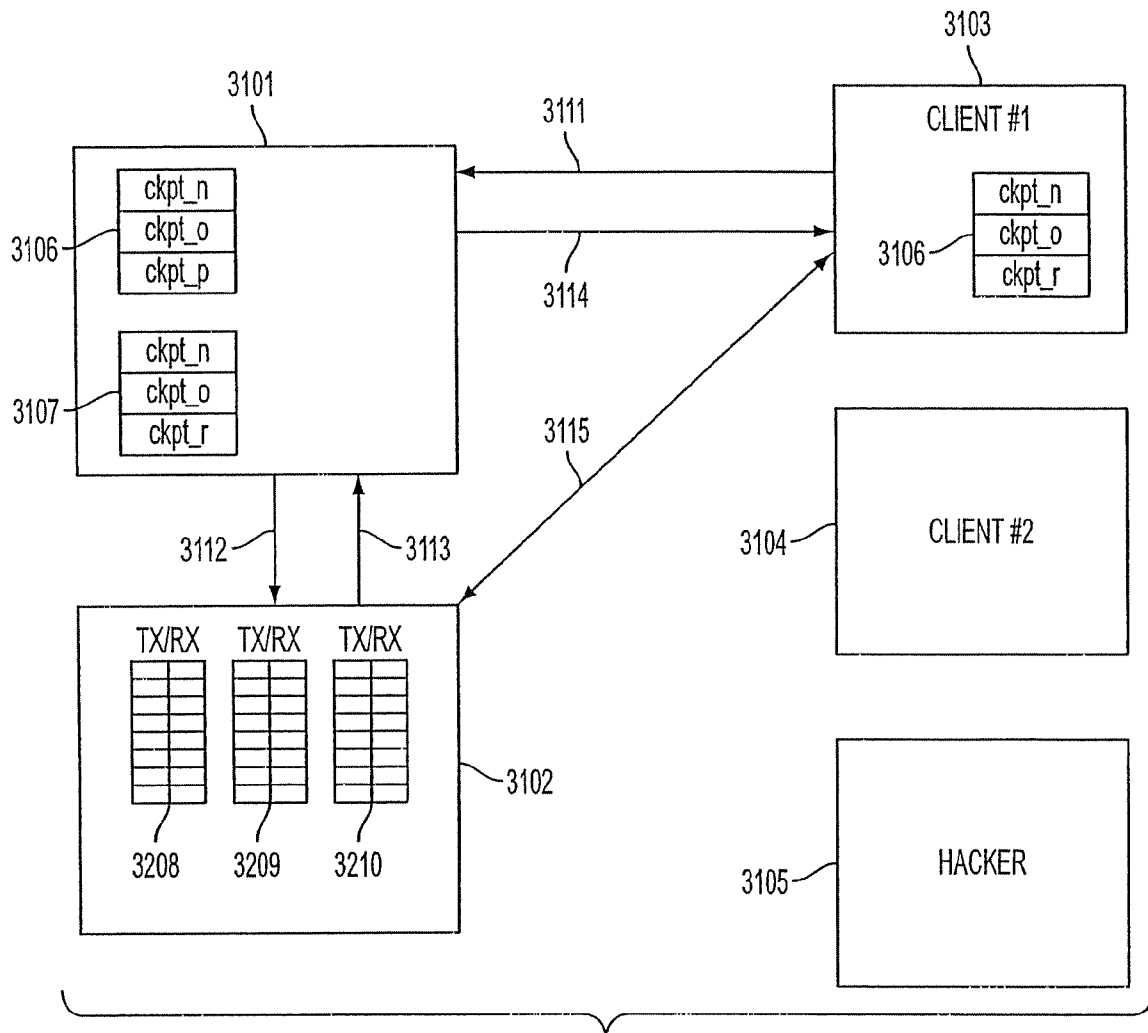


FIG. 31



**U.S. Patent**

Apr. 5, 2011

Sheet 35 of 40

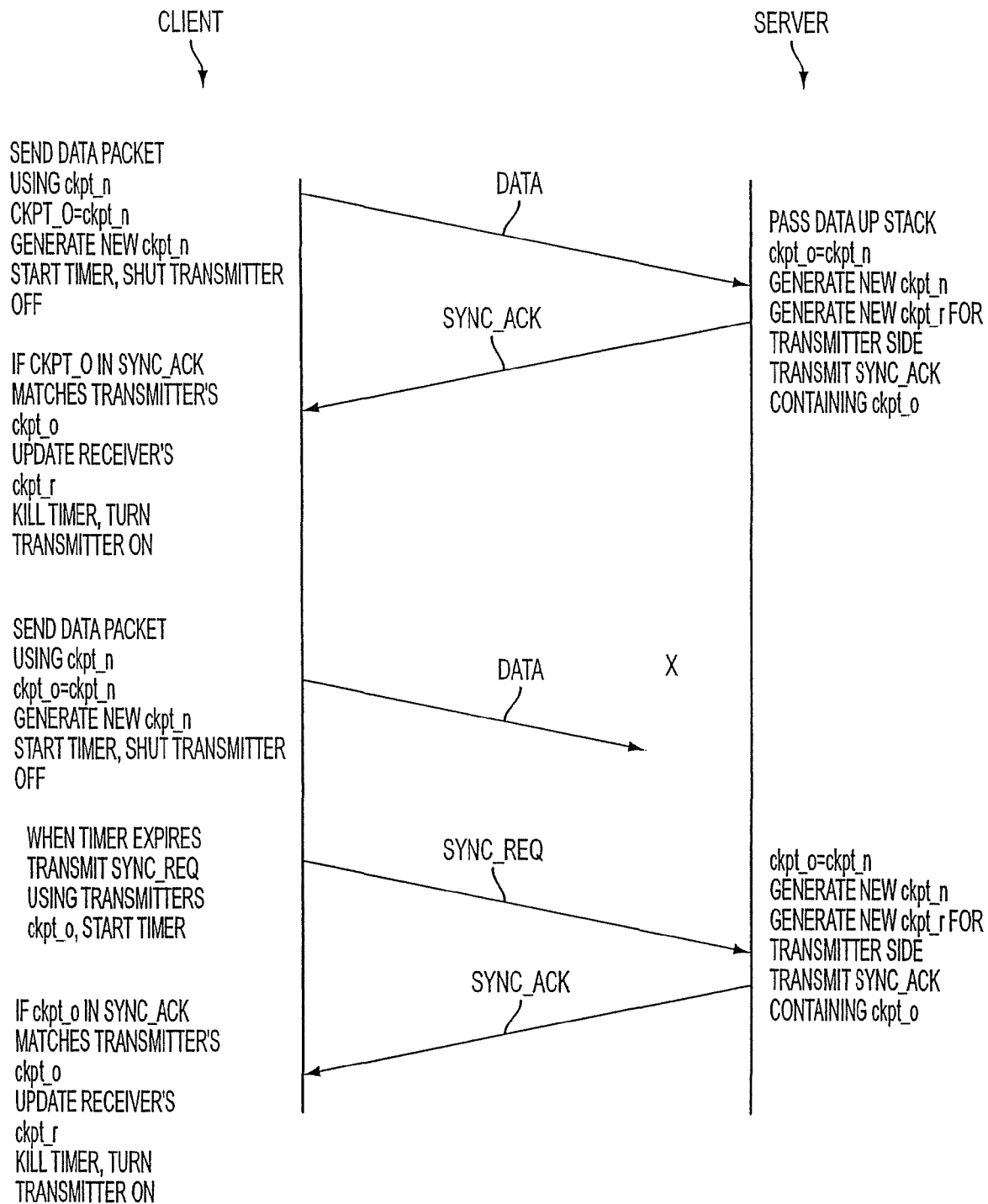
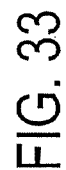
**US 7,921,211 B2**

FIG. 32



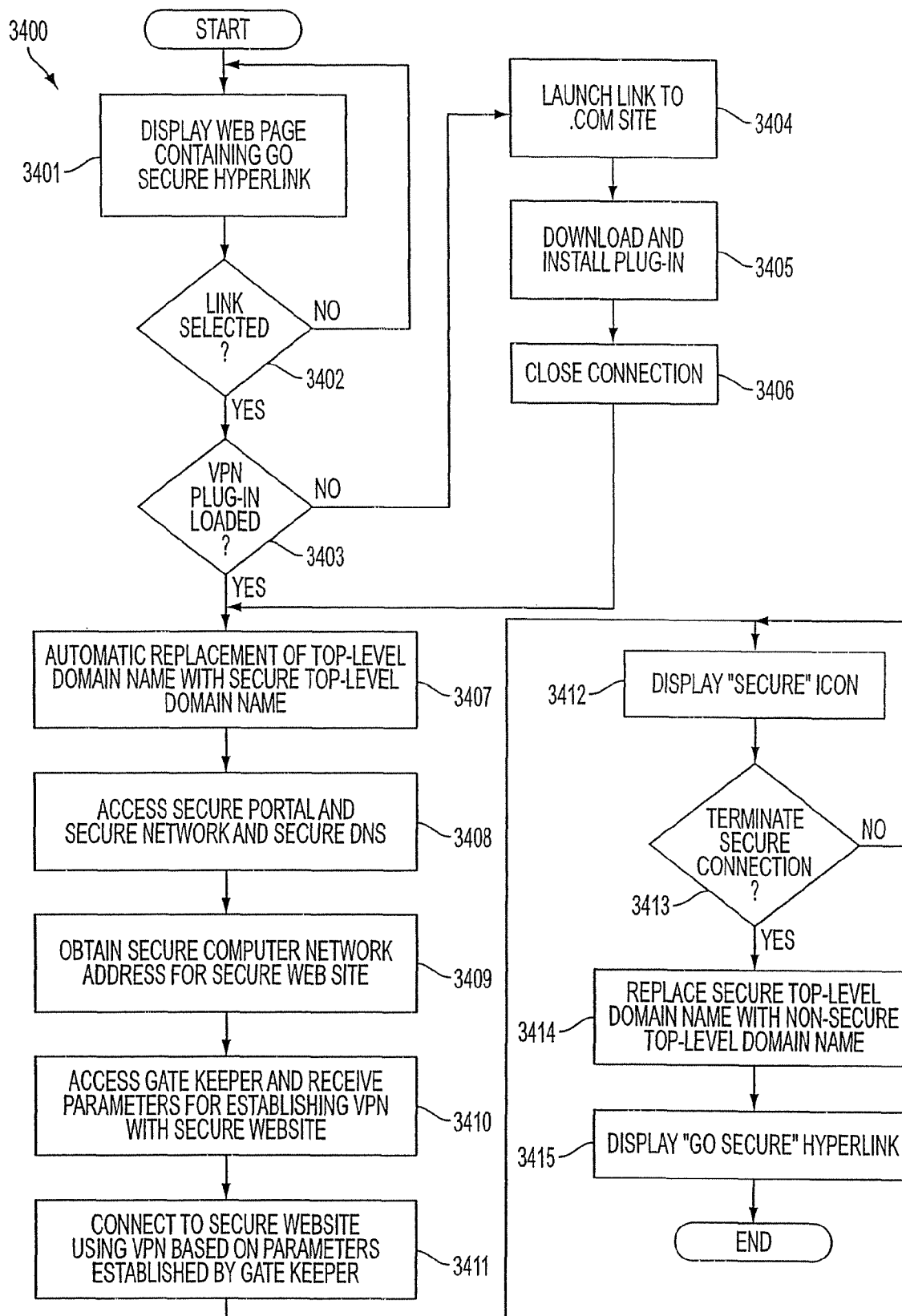
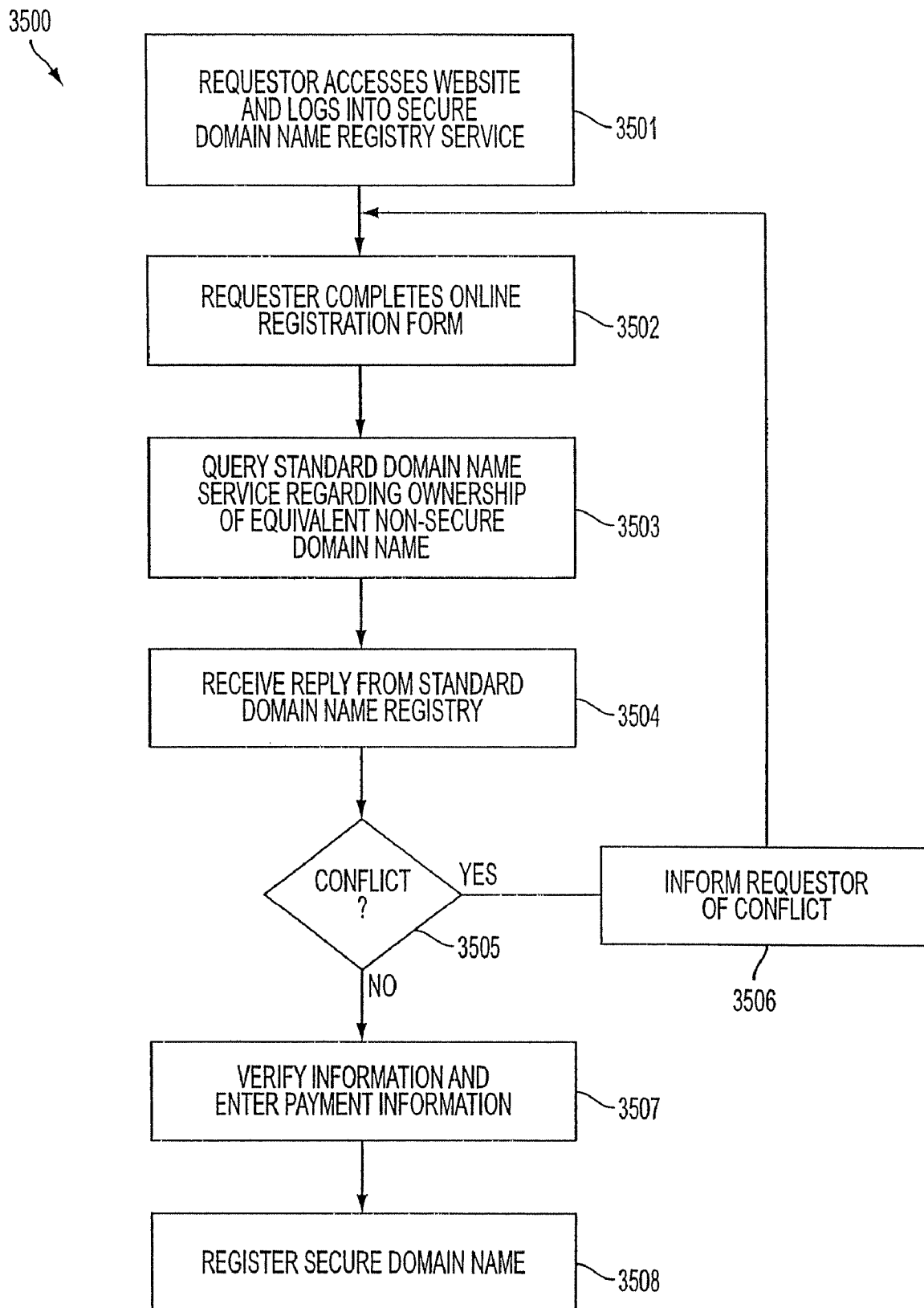


FIG. 34

**U.S. Patent**

Apr. 5, 2011

Sheet 38 of 40

**US 7,921,211 B2****FIG. 35****Appx371**

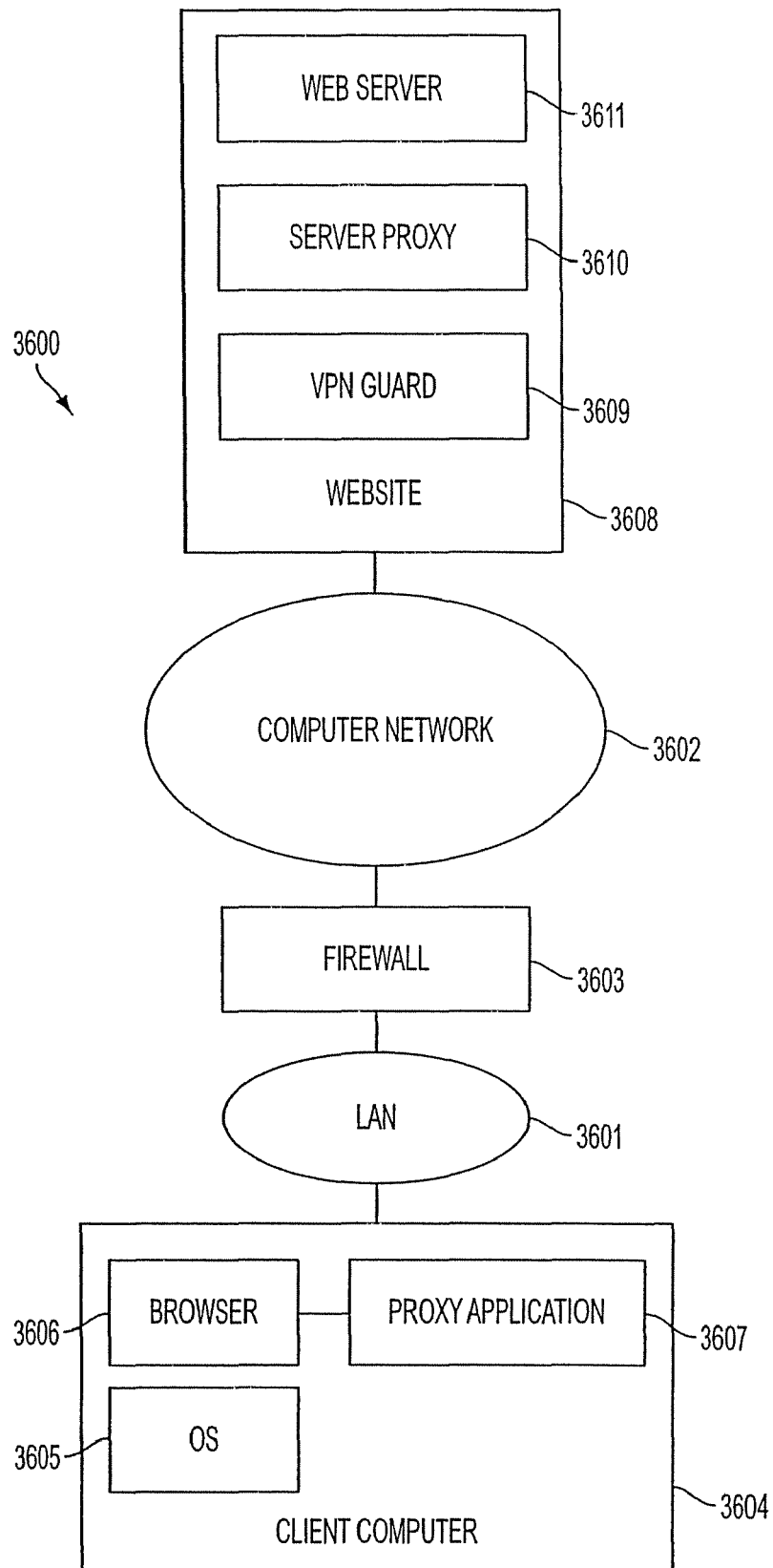


FIG. 36

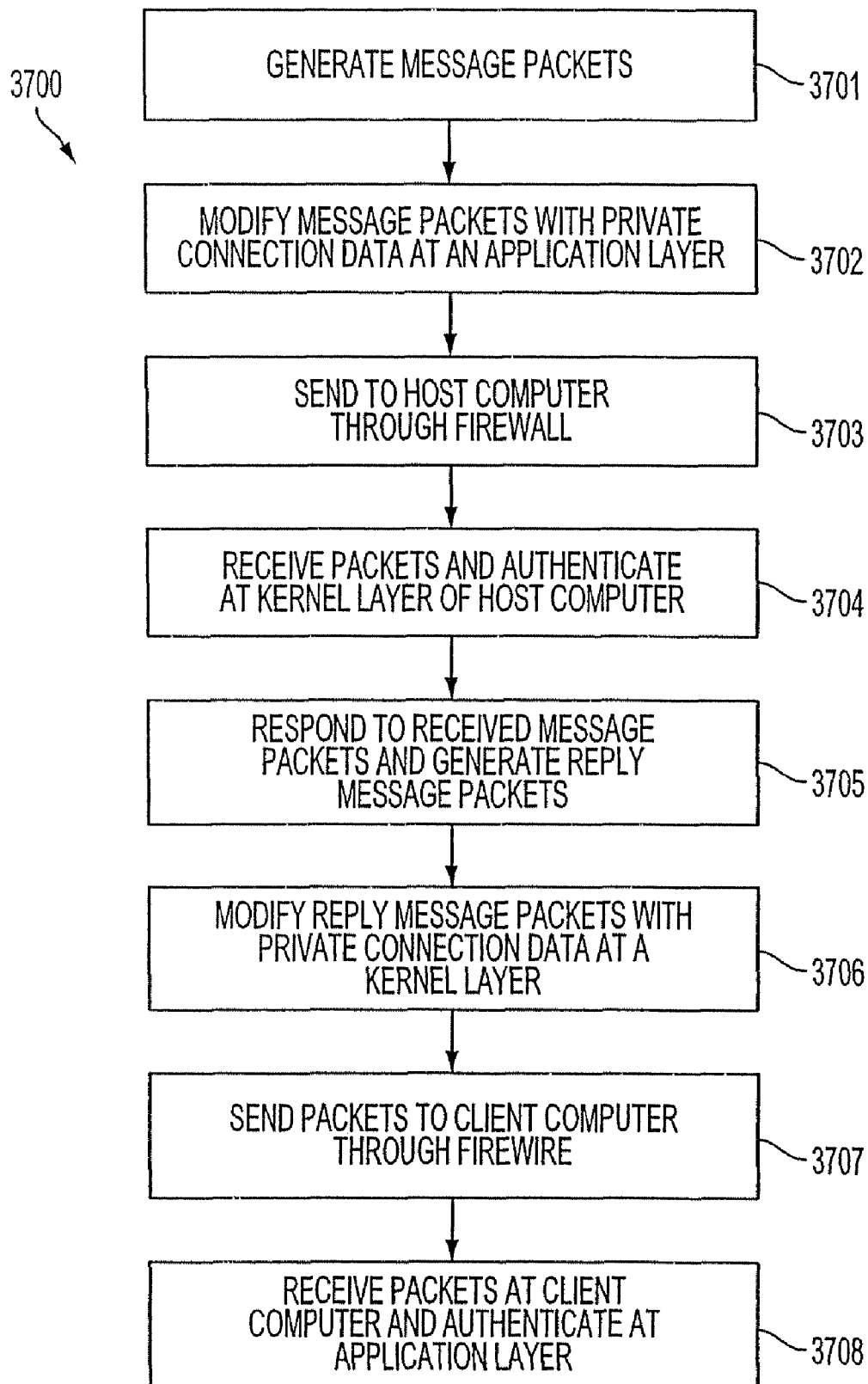
Appx372

**U.S. Patent**

**Apr. 5, 2011**

**Sheet 40 of 40**

**US 7,921,211 B2**



**FIG. 37**

**Appx373**

US 7,921,211 B2

1

# AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES

## CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority from and is a continuation of a U.S. application Ser. No. 10/714,849, filed Nov. 18, 2003, now U.S. Pat. No. 7,418,504, which is a continuation of U.S. application Ser. No. 09/558,210, filed Apr. 26, 2000, now abandoned, which in turn is a continuation-in-part of previously-filed U.S. application Ser. No. 09/504,783, filed on Feb. 15, 2000, now U.S. Pat. No. 6,502,135, issued Dec. 31, 2002, which in turn claims priority from and is a continuation-in-part patent application of previously-filed U.S. application Ser. No. 09/429,643, filed on Oct. 29, 1999, now U.S. Pat. No. 7,010,604, issued Mar. 07, 2006. The subject matter of U.S. application Ser. No. 09/429,643, now U.S. Pat. No. 7,010,604, which is bodily incorporated herein, derives from provisional U.S. application Nos. 60/106,261 (filed Oct. 30, 1998) and 60/137,704 (filed Jun. 7, 1999). The present application is also related to U.S. application Ser. No. 09/558,209, filed Apr. 26, 2000, now abandoned, and which is incorporated by reference herein.

## BACKGROUND OF THE INVENTION

A tremendous variety of methods have been proposed and implemented to provide security and anonymity for communications over the Internet. The variety stems, in part, from the different needs of different Internet users. A basic heuristic framework to aid in discussing these different security techniques is illustrated in FIG. 1. Two terminals, an originating terminal **100** and a destination terminal **110** are in communication over the Internet. It is desired for the communications to be secure, that is, immune to eavesdropping. For example, terminal **100** may transmit secret information to terminal **110** over the Internet **107**. Also, it may be desired to prevent an eavesdropper from discovering that terminal **100** is in communication with terminal **110**. For example, if terminal **100** is a user and terminal **110** hosts a web site, terminal **100**'s user may not want anyone in the intervening networks to know what web sites he is "visiting." Anonymity would thus be an issue, for example, for companies that want to keep their market research interests private and thus would prefer to prevent outsiders from knowing which web-sites or other Internet resources they are "visiting." These two security issues may be called data security and anonymity, respectively.

Data security is usually tackled using some form of data encryption. An encryption key **48** is known at both the originating and terminating terminals **100** and **110**. The keys may be private and public at the originating and destination terminals **100** and **110**, respectively or they may be symmetrical keys (the same key is used by both parties to encrypt and decrypt). Many encryption methods are known and usable in this context.

To hide traffic from a local administrator or ISP, a user can employ a local proxy server in communicating over an encrypted channel with an outside proxy such that the local administrator or ISP only sees the encrypted traffic. Proxy servers prevent destination servers from determining the identities of the originating clients. This system employs an intermediate server interposed between client and destination server. The destination server sees only the Internet Protocol (IP) address of the proxy server and not the originating client.

2

The target server only sees the address of the outside proxy. This scheme relies on a trusted outside proxy server. Also, proxy schemes are vulnerable to traffic analysis methods of determining identities of transmitters and receivers. Another important limitation of proxy servers is that the server knows the identities of both calling and called parties. In many instances, an originating terminal, such as terminal **A**, would prefer to keep its identity concealed from the proxy, for example, if the proxy server is provided by an Internet service provider (ISP).

To defeat traffic analysis, a scheme called Chaum's mixes employs a proxy server that transmits and receives fixed length messages, including dummy messages. Multiple originating terminals are connected through a mix (a server) to multiple target servers. It is difficult to tell which of the originating terminals are communicating to which of the connected target servers, and the dummy messages confuse eavesdroppers' efforts to detect communicating pairs by analyzing traffic. A drawback is that there is a risk that the mix server could be compromised. One way to deal with this risk is to spread the trust among multiple mixes. If one mix is compromised, the identities of the originating and target terminals may remain concealed. This strategy requires a number of alternative mixes so that the intermediate servers interposed between the originating and target terminals are not determinable except by compromising more than one mix. The strategy wraps the message with multiple layers of encrypted addresses. The first mix in a sequence can decrypt only the outer layer of the message to reveal the next destination mix in sequence. The second mix can decrypt the message to reveal the next mix and so on. The target server receives the message and, optionally, a multi-layer encrypted payload containing return information to send data back in the same fashion. The only way to defeat such a mix scheme is to collude among mixes. If the packets are all fixed-length and intermixed with dummy packets, there is no way to do any kind of traffic analysis.

Still another anonymity technique, called 'crowds,' protects the identity of the originating terminal from the intermediate proxies by providing that originating terminals belong to groups of proxies called crowds. The crowd proxies are interposed between originating and target terminals. Each proxy through which the message is sent is randomly chosen by an upstream proxy. Each intermediate proxy can send the message either to another randomly chosen proxy in the "crowd" or to the destination. Thus, even crowd members cannot determine if a preceding proxy is the originator of the message or if it was simply passed from another proxy.

ZKS (Zero-Knowledge Systems) Anonymous IP Protocol allows users to select up to any of five different pseudonyms, while desktop software encrypts outgoing traffic and wraps it in User Datagram Protocol (UDP) packets. The first server in a 2+-hop system gets the UDP packets, strips off one layer of encryption to add another, then sends the traffic to the next server, which strips off yet another layer of encryption and adds a new one. The user is permitted to control the number of hops. At the final server, traffic is decrypted with an untraceable IP address. The technique is called onion-routing. This method can be defeated using traffic analysis. For a simple example, bursts of packets from a user during low-duty periods can reveal the identities of sender and receiver.

Firewalls attempt to protect LANs from unauthorized access and hostile exploitation or damage to computers connected to the LAN. Firewalls provide a server through which all access to the LAN must pass. Firewalls are centralized systems that require administrative overhead to maintain. They can be compromised by virtual-machine applications

US 7,921,211 B2

3

("applets"). They instill a false sense of security that leads to security breaches for example by users sending sensitive information to servers outside the firewall or encouraging use of modems to sidestep the firewall security. Firewalls are not useful for distributed systems such as business travelers, extranets, small teams, etc.

#### SUMMARY OF THE INVENTION

A secure mechanism for communicating over the internet, including a protocol referred to as the Tunneled Agile Routing Protocol (TARP), uses a unique two-layer encryption format and special TARP routers. TARP routers are similar in function to regular IP routers. Each TARP router has one or more IP addresses and uses normal IP protocol to send IP packet messages ("packets" or "datagrams"). The IP packets exchanged between TARP terminals via TARP routers are actually encrypted packets whose true destination address is concealed except to TARP routers and servers. The normal or "clear" or "outside" IP header attached to TARP IP packets contains only the address of a next hop router or destination server. That is, instead of indicating a final destination in the destination field of the IP header, the TARP packet's IP header always points to a next-hop in a series of TARP router hops, or to the final destination. This means there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet since the destination could always be next-hop TARP router as well as the final destination.

Each TARP packet's true destination is concealed behind a layer of encryption generated using a link key. The link key is the encryption key used for encrypted communication between the hops intervening between an originating TARP terminal and a destination TARP terminal. Each TARP router can remove the outer layer of encryption to reveal the destination router for each TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal by the sender/receiver IP numbers in the cleartext IP header.

Once the outer layer of encryption is removed, the TARP router determines the final destination. Each TARP packet undergoes a minimum number of hops to help foil traffic analysis. The hops may be chosen at random or by a fixed value. As a result, each TARP packet may make random trips among a number of geographically disparate routers before reaching its destination. Each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined. This feature is called agile routing. The fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. The associated advantages have to do with the inner layer of encryption discussed below. Agile routing is combined with another feature that furthers this purpose; a feature that ensures that any message is broken into multiple packets.

The IP address of a TARP router can be changed, a feature called IP agility. Each TARP router, independently or under direction from another TARP terminal or router, can change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs.

4

The message payload is hidden behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the intervening TARP routers. The session key is used to decrypt the payloads of the TARP packets permitting the data stream to be reconstructed.

Communication may be made private using link and session keys, which in turn may be shared and used according to any desired method. For example, public/private keys or symmetric keys may be used.

To transmit a data stream, a TARP originating terminal constructs a series of TARP packets from a series of IP packets generated by a network (IP) layer process. (Note that the terms "network layer," "data link layer," "application layer," etc. used in this specification correspond to the Open Systems Interconnection (OSI) network terminology.) The payloads of these packets are assembled into a block and chain-block encrypted using the session key. This assumes, of course, that all the IP packets are destined for the same TARP terminal. The block is then interleaved and the interleaved encrypted block is broken into a series of payloads, one for each TARP packet to be generated. Special TARP headers  $IP_T$  are then added to each payload using the IP headers from the data stream packets. The TARP headers can be identical to normal IP headers or customized in some way. They should contain a formula or data for deinterleaving the data at the destination TARP terminal, a time-to-live (TTL) parameter to indicate the number of hops still to be executed, a data type identifier which indicates whether the payload contains, for example, TCP or UDP data, the sender's TARP address, the destination TARP address, and an indicator as to whether the packet contains real or decoy data or a formula for filtering out decoy data if decoy data is spread in some way through the TARP payload data.

Note that although chain-block encryption is discussed here with reference to the session key, any encryption method may be used. Preferably, as in chain block encryption, a method should be used that makes unauthorized decryption difficult without an entire result of the encryption process. Thus, by separating the encrypted block among multiple packets and making it difficult for an interloper to obtain access to all of such packets, the contents of the communications are provided an extra layer of security.

Decoy or dummy data can be added to a stream to help foil traffic analysis by reducing the peak-to-average network load. It may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or entirety, of a message, and that portion or entirety then interleaved into a number of separate packets. Considering the agile IP routing of the packets, and the attendant difficulty of reconstructing an entire sequence of packets to form a single block-encrypted message element, decoy packets can significantly increase the difficulty of reconstructing an entire data stream.



US 7,921,211 B2

5

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Because the encryption system described above is insertable between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the Network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicating that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. In addition, it may create a subprocess that maintains the original IP address and continues interacting with the attacker in some manner.

Decoy packets may be generated by each TARP terminal on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis.

In various other embodiments of the invention, a scalable version of the system may be constructed in which a plurality of IP addresses are preassigned to each pair of communicating nodes in the network. Each pair of nodes agrees upon an algorithm for "hopping" between IP addresses (both sending and receiving), such that an eavesdropper sees apparently continuously random IP address pairs (source and destination) for packets transmitted between the pair. Overlapping or "reusable" IP addresses may be allocated to different users on the same subnet, since each node merely verifies that a particular packet includes a valid source/destination pair from the agreed-upon algorithm. Source/destination pairs are pref-

6

erably not reused between any two nodes during any given end-to-end session, though limited IP block sizes or lengthy sessions might require it.

Further improvements described in this continuation-in-part application include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities.

The present invention provides key technologies for implementing a secure virtual Internet by using a new agile network protocol that is built on top of the existing Internet protocol (IP). The secure virtual Internet works over the existing Internet infrastructure, and interfaces with client applications the same way as the existing Internet. The key technologies provided by the present invention that support the secure virtual Internet include a "one-click" and "no-click" technique to become part of the secure virtual Internet, a secure domain name service (SDNS) for the secure virtual Internet, and a new approach for interfacing specific client applications onto the secure virtual Internet. According to the invention, the secure domain name service interfaces with existing applications, in addition to providing a way to register and serve domain names and addresses.

According to one aspect of the present invention, a user can conveniently establish a VPN using a "one-click" or a "no-click" technique without being required to enter user identification information, a password and/or an encryption key for establishing a VPN. The advantages of the present invention are provided by a method for establishing a secure communication link between a first computer and a second computer over a computer network, such as the Internet. In one embodiment, a secure communication mode is enabled at a first computer without a user entering any cryptographic information for establishing the secure communication mode of communication, preferably by merely selecting an icon displayed on the first computer. Alternatively, the secure communication mode of communication can be enabled by entering a command into the first computer. Then, a secure communication link is established between the first computer and a second computer over a computer network based on the enabled secure communication mode of communication. According to the invention, it is determined whether a secure communication software module is stored on the first computer in response to the step of enabling the secure communication mode of communication. A predetermined computer network address is then accessed for loading the secure communication software module when the software module is not stored on the first computer. Subsequently, the proxy software module is stored in the first computer. The secure communication link is a virtual private network communication link over the computer network. Preferably, the virtual private network can be based on inserting into each data packet one or more data values that vary according to a pseudo-random sequence. Alternatively, the virtual private network can be based on a computer network address hopping regime that is used to pseudorandomly change computer network addresses or other data values in packets transmitted between the first computer and the second computer, such that the second computer compares the data values in each data packet trans-

## US 7,921,211 B2

7

mitted between the first computer and the second computer to a moving window of valid values. Yet another alternative provides that the virtual private network can be based on a comparison between a discriminator field in each data packet to a table of valid discriminator fields maintained for the first computer.

According to another aspect of the invention, a command is entered to define a setup parameter associated with the secure communication link mode of communication. Consequently, the secure communication mode is automatically established when a communication link is established over the computer network.

The present invention also provides a computer system having a communication link to a computer network, and a display showing a hyperlink for establishing a virtual private network through the computer network. When the hyperlink for establishing the virtual private network is selected, a virtual private network is established over the computer network. A non-standard top-level domain name is then sent over the virtual private network communication to a predetermined computer network address, such as a computer network address for a secure domain name service (SDNS).

The present invention provides a domain name service that provides secure computer network addresses for secure, non-standard top-level domain names. The advantages of the present invention are provided by a secure domain name service for a computer network that includes a portal connected to a computer network, such as the Internet, and a domain name database connected to the computer network through the portal. According to the invention, the portal authenticates a query for a secure computer network address, and the domain name database stores secure computer network addresses for the computer network. Each secure computer network address is based on a non-standard top-level domain name, such as .scom, .sorg, .snet, .snet, .sedu, .smil and .sint.

The present invention provides a way to encapsulate existing application network traffic at the application layer of a client computer so that the client application can securely communicate with a server protected by an agile network protocol. The advantages of the present invention are provided by a method for communicating using a private communication link between a client computer and a server computer over a computer network, such as the Internet. According to the invention, an information packet is sent from the client computer to the server computer over the computer network. The information packet contains data that is inserted into the payload portion of the packet at the application layer of the client computer and is used for forming a virtual private connection between the client computer and the server computer. The modified information packet can be sent through a firewall before being sent over the computer network to the server computer and by working on top of existing protocols (i.e., UDP, ICMP and TCP), the present invention more easily penetrates the firewall. The information packet is received at a kernel layer of an operating system on the server side. It is then determined at the kernel layer of the operating system on the host computer whether the information packet contains the data that is used for forming the virtual private connection. The server side replies by sending an information packet to the client computer that has been modified at the kernel layer to containing virtual private connection information in the payload portion of the reply information packet. Preferably, the information packet from the client computer and the reply information packet from the server side are each a UDP protocol information packet.

8

Alternative, both information packets could be a TCP/IP protocol information packet, or an ICMP protocol information packet.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is an illustration of secure communications over the Internet according to a prior art embodiment.

FIG. 2 is an illustration of secure communications over the Internet according to an embodiment of the invention.

FIG. 3a is an illustration of a process of forming a tunneled IP packet according to an embodiment of the invention.

FIG. 3b is an illustration of a process of forming a tunneled IP packet according to another embodiment of the invention.

FIG. 4 is an illustration of an OSI layer location of processes that may be used to implement the invention.

FIG. 5 is a flow chart illustrating a process for routing a tunneled packet according to an embodiment of the invention.

FIG. 6 is a flow chart illustrating a process for forming a tunneled packet according to an embodiment of the invention.

FIG. 7 is a flow chart illustrating a process for receiving a tunneled packet according to an embodiment of the invention.

FIG. 8 shows how a secure session is established and synchronized between a client and a TARP router.

FIG. 9 shows an IP address hopping scheme between a client computer and TARP router using transmit and receive tables in each computer.

FIG. 10 shows physical link redundancy among three Internet Service Providers (ISPs) and a client computer.

FIG. 11 shows how multiple IP packets can be embedded into a single "frame" such as an Ethernet frame, and further shows the use of a discriminator field to camouflage true packet recipients.

FIG. 12A shows a system that employs hopped hardware addresses, hopped IP addresses, and hopped discriminator fields.

FIG. 12B shows several different approaches for hopping hardware addresses, IP addresses, and discriminator fields in combination.

FIG. 13 shows a technique for automatically re-establishing synchronization between sender and receiver through the use of a partially public sync value.

FIG. 14 shows a "checkpoint" scheme for regaining synchronization between a sender and recipient.

FIG. 15 shows further details of the checkpoint scheme of FIG. 14.

FIG. 16 shows how two addresses can be decomposed into a plurality of segments for comparison with presence vectors.

FIG. 17 shows a storage array for a receiver's active addresses.

FIG. 18 shows the receiver's storage array after receiving a sync request.

FIG. 19 shows the receiver's storage array after new addresses have been generated.

FIG. 20 shows a system employing distributed transmission paths.

FIG. 21 shows a plurality of link transmission tables that can be used to route packets in the system of FIG. 20.

FIG. 22A shows a flowchart for adjusting weight value distributions associated with a plurality of transmission links.

FIG. 22B shows a flowchart for setting a weight value to zero if a transmitter turns off.

FIG. 23 shows a system employing distributed transmission paths with adjusted weight value distributions for each path.

FIG. 24 shows an example using the system of FIG. 23.

## US 7,921,211 B2

9

FIG. 25 shows a conventional domain-name look-up service.

FIG. 26 shows a system employing a DNS proxy server with transparent VPN creation.

FIG. 27 shows steps that can be carried out to implement transparent VPN creation based on a DNS look-up function.

FIG. 28 shows a system including a link guard function that prevents packet overloading on a low-bandwidth link LOW BW.

FIG. 29 shows one embodiment of a system employing the principles of FIG. 28.

FIG. 30 shows a system that regulates packet transmission rates by throttling the rate at which synchronizations are performed.

FIG. 31 shows a signaling server 3101 and a transport server 3102 used to establish a VPN with a client computer.

FIG. 32 shows message flows relating to synchronization protocols of FIG. 31.

FIG. 33 shows a system block diagram of a computer network in which the "one-click" secure communication link of the present invention is suitable for use.

FIG. 34 shows a flow diagram for installing and establishing a "one-click" secure communication link over a computer network according to the present invention.

FIG. 35 shows a flow diagram for registering a secure domain name according to the present invention.

FIG. 36 shows a system block diagram of a computer network in which a private connection according to the present invention can be configured to more easily traverse a firewall between two computer networks.

FIG. 37 shows a flow diagram for establishing a virtual private connection that is encapsulated using an existing network protocol.

## DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. 2, a secure mechanism for communicating over the internet employs a number of special routers or servers, called TARP routers 122-127 that are similar to regular IP routers 128-132 in that each has one or more IP addresses and uses normal IP protocol to send normal-looking IP packet messages, called TARP packets 140. TARP packets 140 are identical to normal IP packet messages that are routed by regular IP routers 128-132 because each TARP packet 140 contains a destination address as in a normal IP packet. However, instead of indicating a final destination in the destination field of the IP header, the TARP packet's 140 IP header always points to a next-hop in a series of TARP router hops, or the final destination, TARP terminal 110. Because the header of the TARP packet contains only the next-hop destination, there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet 140 since the destination could always be the next-hop TARP router as well as the final destination, TARP terminal 110.

Each TARP packet's true destination is concealed behind an outer layer of encryption generated using a link key 146. The link key 146 is the encryption key used for encrypted communication between the end points (TARP terminals or TARP routers) of a single link in the chain of hops connecting the originating TARP terminal 100 and the destination TARP terminal 110. Each TARP router 122-127, using the link key 146 it uses to communicate with the previous hop in a chain, can use the link key to reveal the true destination of a TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal

10

(which may indicate the link key used) by the sender field of the clear IP header. Alternatively, this identity may be hidden behind another layer of encryption in available bits in the clear IP header. Each TARP router, upon receiving a TARP message, determines if the message is a TARP message by using authentication data in the TARP packet. This could be recorded in available bytes in the TARP packet's IP header. Alternatively, TARP packets could be authenticated by attempting to decrypt using the link key 146 and determining if the results are as expected. The former may have computational advantages because it does not involve a decryption process.

Once the outer layer of decryption is completed by a TARP router 122-127, the TARP router determines the final destination. The system is preferably designed to cause each TARP packet 140 to undergo a minimum number of hops to help foil traffic analysis. The time to live counter in the IP header of the TARP message may be used to indicate a number of TARP router hops yet to be completed. Each TARP router then would decrement the counter and determine from that whether it should forward the TARP packet 140 to another TARP router 122-127 or to the destination TARP terminal 110. If the time to live counter is zero or below zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to the destination TARP terminal 110. If the time to live counter is above zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to a TARP router 122-127 that the current TARP terminal chooses at random. As a result, each TARP packet 140 is routed through some minimum number of hops of TARP routers 122-127 which are chosen at random.

Thus, each TARP packet, irrespective of the traditional factors determining traffic in the Internet, makes random trips among a number of geographically disparate routers before reaching its destination and each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined as described above. This feature is called agile routing. For reasons that will become clear shortly, the fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. Agile routing is combined with another feature that furthers this purpose, a feature that ensures that any message is broken into multiple packets.

A TARP router receives a TARP packet when an IP address used by the TARP router coincides with the IP address in the TARP packet's IP header IPc. The IP address of a TARP router, however, may not remain constant. To avoid and manage attacks, each TARP router, independently or under direction from another TARP terminal or router, may change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs. In reality, whenever a TARP router looks up the address of a destination in the encrypted header, it must convert a TARP address to a real IP address using its LUT.

While every TARP router receiving a TARP packet has the ability to determine the packet's final destination, the message payload is embedded behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the TARP

## US 7,921,211 B2

11

routers **122-127** intervening between the originating **100** and destination **110** TARP terminals. The session key is used to decrypt the payloads of the TARP packets **140** permitting an entire message to be reconstructed.

In one embodiment, communication may be made private using link and session keys, which in turn may be shared and used according any desired method. For example, a public key or symmetric keys may be communicated between link or session endpoints using a public key method. Any of a variety of other mechanisms for securing data to ensure that only authorized computers can have access to the private information in the TARP packets **140** may be used as desired.

Referring to FIG. **3a**, to construct a series of TARP packets, a data stream **300** of IP packets **207a**, **207b**, **207c**, etc., such series of packets being formed by a network (IP) layer process, is broken into a series of small sized segments. In the present example, equal-sized segments **1-9** are defined and used to construct a set of interleaved data packets A, B, and C. Here it is assumed that the number of interleaved packets A, B, and C formed is three and that the number of IP packets **207a-207c** used to form the three interleaved packets A, B, and C is exactly three. Of course, the number of IP packets spread over a group of interleaved packets may be any convenient number as may be the number of interleaved packets over which the incoming data stream is spread. The latter, the number of interleaved packets over which the data stream is spread, is called the interleave window.

To create a packet, the transmitting software interleaves the normal IP packets **207a** et. seq. to form a new set of interleaved payload data **320**. This payload data **320** is then encrypted using a session key to form a set of session-key-encrypted payload data **330**, each of which, A, B, and C, will form the payload of a TARP packet. Using the IP header data, from the original packets **207a-207c**, new TARP headers IPT are formed. The TARP headers IPT can be identical to normal IP headers or customized in some way. In a preferred embodiment, the TARP headers IPT are IP headers with added data providing the following information required for routing and reconstruction of messages, some of which data is ordinarily, or capable of being, contained in normal IP headers:

1. A window sequence number—an identifier that indicates where the packet belongs in the original message sequence.
2. An interleave sequence number—an identifier that indicates the interleaving sequence used to form the packet so that the packet can be deinterleaved along with other packets in the interleave window.
3. A time-to-live (TTL) datum—indicates the number of TARP-router-hops to be executed before the packet reaches its destination. Note that the TTL parameter may provide a datum to be used in a probabilistic formula for determining whether to route the packet to the destination or to another hop.
4. Data type identifier—indicates whether the payload contains, for example, TCP or UDP data.
5. Sender's address—indicates the sender's address in the TARP network.
6. Destination address—indicates the destination terminal's address in the TARP network.
7. Decoy/Real—an indicator of whether the packet contains real message data or dummy decoy data or a combination.

Obviously, the packets going into a single interleave window must include only packets with a common destination. Thus, it is assumed in the depicted example that the IP headers of IP packets **207a-207c** all contain the same destination address or at least will be received by the same terminal so

12

that they can be deinterleaved. Note that dummy or decoy data or packets can be added to form a larger interleave window than would otherwise be required by the size of a given message. Decoy or dummy data can be added to a stream to help foil traffic analysis by leveling the load on the network. Thus, it may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or the entirety, of a message, and that portion or entirety then interleaved into a number of separate packets.

Referring to FIG. **3b**, in an alternative mode of TARP packet construction, a series of IP packets are accumulated to make up a predefined interleave window. The payloads of the packets are used to construct a single block **520** for chain block encryption using the session key. The payloads used to form the block are presumed to be destined for the same terminal. The block size may coincide with the interleave window as depicted in the example embodiment of FIG. **3b**. After encryption, the encrypted block is broken into separate payloads and segments which are interleaved as in the embodiment of FIG. **3a**. The resulting interleaved packets A, B, and C, are then packaged as TARP packets with TARP headers as in the Example of FIG. **3a**. The remaining process is as shown in, and discussed with reference to, FIG. **3a**.

Once the TARP packets **340** are formed, each entire TARP packet **340**, including the TARP header IPT, is encrypted using the link key for communication with the first-hop TARP router. The first hop TARP router is randomly chosen. A final unencrypted IP header IPc is added to each encrypted TARP packet **340** to form a normal IP packet **360** that can be transmitted to a TARP router. Note that the process of constructing the TARP packet **360** does not have to be done in stages as described. The above description is just a useful heuristic for describing the final product, namely, the TARP packet.

Note that, TARP header IP<sub>T</sub> could be a completely custom header configuration with no similarity to a normal IP header except that it contain the information identified above. This is so since this header is interpreted by only TARP routers.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Referring to FIG. **4**, a TARP transceiver **405** can be an originating terminal **100**, a destination terminal **110**, or a TARP router **122-127**. In each TARP Transceiver **405**, a transmitting process is generated to receive normal packets from the Network (IP) layer and generate TARP packets for communication over the network. A receiving process is generated to receive normal IP packets containing TARP packets and generate from these normal IP packets which are "passed up" to the Network (IP) layer. Note that where the TARP Transceiver **405** is a router, the received TARP packets **140** are not processed into a stream of IP packets **415** because they need only be authenticated as proper TARP packets and then passed to another TARP router or a TARP destination termi-

## US 7,921,211 B2

13

nal 110. The intervening process, a "TARP Layer" 420, could be combined with either the data link layer 430 or the Network layer 410. In either case, it would intervene between the data link layer 430 so that the process would receive regular IP packets containing embedded TARP packets and "hand up" a series of reassembled IP packets to the Network layer 410. As an example of combining the TARP layer 420 with the data link layer 430, a program may augment the normal processes running a communications card, for example, an Ethernet card. Alternatively, the TARP layer processes may form part of a dynamically loadable module that is loaded and executed to support communications between the network and data link layers.

Because the encryption system described above can be inserted between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

Note that IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicates that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. To accomplish this, the TARP process will construct a TARP-formatted message, in the style of Internet Control Message Protocol (ICMP) datagrams as an example; this message will contain the machine's TARP address, its previous IP address, and its new IP address. The TARP layer will transmit this packet to at least one known TARP router; then upon receipt and validation of the message, the TARP router will update its LUT with the new IP address for the stated TARP address. The TARP router will then format a similar message, and broadcast it to the other TARP routers so that they may update their LUTs. Since the total number of TARP routers on any given subnet is expected to be relatively small, this process of updating the LUTs should be relatively fast. It may not, however, work as well when there is a relatively large number of TARP routers and/or a relatively large number of clients; this has motivated a refinement of this architecture to provide scalability; this refinement has led to a second embodiment, which is discussed below.

Upon detection of an attack, the TARP process may also create a subprocess that maintains the original IP address and continues interacting with the attacker. The latter may provide an opportunity to trace the attacker or study the attacker's methods (called "fishbowling" drawing upon the analogy of a small fish in a fish bowl that "thinks" it is in the ocean but

14

is actually under captive observation). A history of the communication between the attacker and the abandoned (fish-bowled) IP address can be recorded or transmitted for human analysis or further synthesized for purposes of responding in some way.

As mentioned above, decoy or dummy data or packets can be added to outgoing data streams by TARP terminals or routers. In addition to making it convenient to spread data over a larger number of separate packets, such decoy packets can also help to level the load on inactive portions of the Internet to help foil traffic analysis efforts.

Decoy packets may be generated by each TARP terminal 100, 110 or each router 122-127 on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. That is, when a series of messages are received, the decoy packet generation rate may be increased. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis. The rate of reception of packets, decoy or otherwise, may be indicated to the decoy packet dropping and generating processes through perishable decoy and regular packet counters. (A perishable counter is one that resets or decrements its value in response to time so that it contains a high value when it is incremented in rapid succession and a small value when incremented either slowly or a small number of times in rapid succession.) Note that destination TARP terminal 110 may generate decoy packets equal in number and size to those TARP packets received to make it appear it is merely routing packets and is therefore not the destination terminal.

Referring to FIG. 5, the following particular steps may be employed in the above-described method for routing TARP packets.

- S0. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.
- S2. The TARP packet may be probed in some way to authenticate the packet before attempting to decrypt it using the link key. That is, the router may determine that the packet is an authentic TARP packet by performing a selected operation on some data included with the clear IP header attached to the encrypted TARP packet contained in the payload. This makes it possible to avoid performing decryption on packets that are not authentic TARP packets.
- S3. The TARP packet is decrypted to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.
- S4. If the packet is a decoy packet, the perishable decoy counter is incremented.
- S5. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the router may choose to throw it away. If

## US 7,921,211 B2

15

the received packet is a decoy packet and it is determined that it should be thrown away (S6), control returns to step S0.

S7. The TTL parameter of the TARP header is decremented and it is determined if the TTL parameter is greater than zero.

S8. If the TTL parameter is greater than zero, a TARP address is randomly chosen from a list of TARP addresses maintained by the router and the link key and IP address corresponding to that TARP address memorized for use in creating a new IP packet containing the TARP packet.

S9. If the TTL parameter is zero or less, the link key and IP address corresponding to the TARP address of the destination are memorized for use in creating the new IP packet containing the TARP packet.

S10. The TARP packet is encrypted using the memorized link key.

S11. An IP header is added to the packet that contains the stored IP address, the encrypted TARP packet wrapped with an IP header, and the completed packet transmitted to the next hop or destination.

Referring to FIG. 6, the following particular steps may be employed in the above-described method for generating TARP packets.

S20. A background loop operation applies an algorithm that determines the generation of decoy IP packets. The loop is interrupted when a data stream containing IP packets is received for transmission.

S21. The received IP packets are grouped into a set consisting of messages with a constant IP destination address. The set is further broken down to coincide with a maximum size of an interleave window. The set is encrypted, and interleaved into a set of payloads destined to become TARP packets.

S22. The TARP address corresponding to the IP address is determined from a lookup table and stored to generate the TARP header. An initial TTL count is generated and stored in the header. The TTL count may be random with minimum and maximum values or it may be fixed or determined by some other parameter.

S23. The window sequence numbers and interleave sequence numbers are recorded in the TARP headers of each packet.

S24. One TARP router address is randomly chosen for each TARP packet and the IP address corresponding to it stored for use in the clear IP header. The link key corresponding to this router is identified and used to encrypt TARP packets containing interleaved and encrypted data and TARP headers.

S25. A clear IP header with the first hop router's real IP address is generated and added to each of the encrypted TARP packets and the resulting packets.

Referring to FIG. 7, the following particular steps may be employed in the above-described method for receiving TARP packets.

S40. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S42. The TARP packet may be probed to authenticate the packet before attempting to decrypt it using the link key.

S43. The TARP packet is decrypted with the appropriate link key to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

16

S44. If the packet is a decoy packet, the perishable decoy counter is incremented.

S45. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the receiver may choose to throw it away.

S46. The TARP packets are cached until all packets forming an interleave window are received.

S47. Once all packets of an interleave window are received, the packets are deinterleaved.

S48. The packets block of combined packets defining the interleave window is then decrypted using the session key.

S49. The decrypted block is then divided using the window sequence data and the IP<sub>T</sub> headers are converted into normal IP<sub>C</sub> headers. The window sequence numbers are integrated in the IP<sub>C</sub> headers.

S50. The packets are then handed up to the IP layer processes.

### I. Scalability Enhancements

The IP agility feature described above relies on the ability to transmit IP address changes to all TARP routers. The embodiments including this feature will be referred to as "boutique" embodiments due to potential limitations in scaling these features up for a large network, such as the Internet. (The "boutique" embodiments would, however, be robust for use in smaller networks, such as small virtual private networks, for example). One problem with the boutique embodiments is that if IP address changes are to occur frequently, the message traffic required to update all routers sufficiently quickly creates a serious burden on the Internet when the TARP router and/or client population gets large. The bandwidth burden added to the networks, for example in ICMP packets, that would be used to update all the TARP routers could overwhelm the Internet for a large scale implementation that approached the scale of the Internet. In other words, the boutique system's scalability is limited.

A system can be constructed which trades some of the features of the above embodiments to provide the benefits of IP agility without the additional messaging burden. This is accomplished by IP address-hopping according to shared algorithms that govern IP addresses used between links participating in communications sessions between nodes such as TARP nodes. (Note that the IP hopping technique is also applicable to the boutique embodiment.) The IP agility feature discussed with respect to the boutique system can be modified so that it becomes decentralized under this scalable regime and governed by the above-described shared algorithm. Other features of the boutique system may be combined with this new type of IP-agility.

The new embodiment has the advantage of providing IP agility governed by a local algorithm and set of IP addresses exchanged by each communicating pair of nodes. This local governance is session-independent in that it may govern communications between a pair of nodes, irrespective of the session or end points being transferred between the directly communicating pair of nodes.

In the scalable embodiments, blocks of IP addresses are allocated to each node in the network. (This scalability will increase in the future, when Internet Protocol addresses are increased to 128-bit fields, vastly increasing the number of distinctly addressable nodes). Each node can thus use any of the IP addresses assigned to that node to communicate with other nodes in the network. Indeed, each pair of communicating nodes can use a plurality of source IP addresses and destination IP addresses for communicating with each other.

## US 7,921,211 B2

17

Each communicating pair of nodes in a chain participating in any session stores two blocks of IP addresses, called netblocks, and an algorithm and randomization seed for selecting, from each netblock, the next pair of source/destination IP addresses that will be used to transmit the next message. In other words, the algorithm governs the sequential selection of IP-address pairs, one sender and one receiver IP address, from each netblock. The combination of algorithm, seed, and netblock (IP address block) will be called a "hopblock." A router issues separate transmit and receive hopblocks to its clients. The send address and the receive address of the IP header of each outgoing packet sent by the client are filled with the send and receive IP addresses generated by the algorithm. The algorithm is "clocked" (indexed) by a counter so that each time a pair is used, the algorithm turns out a new transmit pair for the next packet to be sent.

The router's receive hopblock is identical to the client's transmit hopblock. The router uses the receive hopblock to predict what the send and receive IP address pair for the next expected packet from that client will be. Since packets can be received out of order, it is not possible for the router to predict with certainty what IP address pair will be on the next sequential packet. To account for this problem, the router generates a range of predictions encompassing the number of possible transmitted packet send/receive addresses, of which the next packet received could leap ahead. Thus, if there is a vanishingly small probability that a given packet will arrive at the router ahead of 5 packets transmitted by the client before the given packet, then the router can generate a series of 6 send/receive IP address pairs (or "hop window") to compare with the next received packet. When a packet is received, it is marked in the hop window as such, so that a second packet with the same IP address pair will be discarded. If an out-of-sequence packet does not arrive within a predetermined timeout period, it can be requested for retransmission or simply discarded from the receive table, depending upon the protocol in use for that communications session, or possibly by convention.

When the router receives the client's packet, it compares the send and receive IP addresses of the packet with the next N predicted send and receive IP address pairs and rejects the packet if it is not a member of this set. Received packets that do not have the predicted source/destination IP addresses falling within the window are rejected, thus thwarting possible hackers. (With the number of possible combinations, even a fairly large window would be hard to fall into at random.) If it is a member of this set, the router accepts the packet and processes it further. This link-based IP-hopping strategy, referred to as "IHOP," is a network element that stands on its own and is not necessarily accompanied by elements of the boutique system described above. If the routing agility feature described in connection with the boutique embodiment is combined with this link-based IP-hopping strategy, the router's next step would be to decrypt the TARP header to determine the destination TARP router for the packet and determine what should be the next hop for the packet. The TARP router would then forward the packet to a random TARP router or the destination TARP router with which the source TARP router has a link-based IP hopping communication established.

FIG. 8 shows how a client computer **801** and a TARP router **811** can establish a secure session. When client **801** seeks to establish an IHOP session with TARP router **811**, the client **801** sends "secure synchronization" request ("SSYN") packet **821** to the TARP router **811**. This SYN packet **821** contains the client's **801** authentication token, and may be sent to the router **811** in an encrypted format. The source and

18

destination IP numbers on the packet **821** are the client's **801** current fixed IP address, and a "known" fixed IP address for the router **811**. (For security purposes, it may be desirable to reject any packets from outside of the local network that are destined for the router's known fixed IP address.) Upon receipt and validation of the client's **801** SSYN packet **821**, the router **811** responds by sending an encrypted "secure synchronization acknowledgment" ("SSYN ACK") **822** to the client **801**. This SSYN ACK **822** will contain the transmit and receive hopblocks that the client **801** will use when communicating with the TARP router **811**. The client **801** will acknowledge the TARP router's **811** response packet **822** by generating an encrypted SSYN ACK ACK packet **823** which will be sent from the client's **801** fixed IP address and to the TARP router's **811** known fixed IP address. The client **801** will simultaneously generate a SSYN ACK ACK packet; this SSYN ACK packet, referred to as the Secure Session Initiation (SSI) packet **824**, will be sent with the first {sender, receiver} IP pair in the client's transmit table **921** (FIG. 9), as specified in the transmit hopblock provided by the TARP router **811** in the SSYN ACK packet **822**. The TARP router **811** will respond to the SSI packet **824** with an SSI ACK packet **825**, which will be sent with the first {sender, receiver} IP pair in the TARP router's transmit table **923**. Once these packets have been successfully exchanged, the secure communications session is established, and all further secure communications between the client **801** and the TARP router **811** will be conducted via this secure session, as long as synchronization is maintained. If synchronization is lost, then the client **801** and TARP router **802** may re-establish the secure session by the procedure outlined in FIG. 8 and described above.

While the secure session is active, both the client **901** and TARP router **911** (FIG. 9) will maintain their respective transmit tables **921**, **923** and receive tables **922**, **924**, as provided by the TARP router during session synchronization **822**. It is important that the sequence of IP pairs in the client's transmit table **921** be identical to those in the TARP router's receive table **924**; similarly, the sequence of IP pairs in the client's receive table **922** must be identical to those in the router's transmit table **923**. This is required for the session synchronization to be maintained. The client **901** need maintain only one transmit table **921** and one receive table **922** during the course of the secure session. Each sequential packet sent by the client **901** will employ the next {send, receive} IP address pair in the transmit table, regardless of TCP or UDP session. The TARP router **911** will expect each packet arriving from the client **901** to bear the next IP address pair shown in its receive table.

Since packets can arrive out of order, however, the router **911** can maintain a "look ahead" buffer in its receive table, and will mark previously-received IP pairs as invalid for future packets; any future packet containing an IP pair that is in the look-ahead buffer but is marked as previously received will be discarded. Communications from the TARP router **911** to the client **901** are maintained in an identical manner; in particular, the router **911** will select the next IP address pair from its transmit table **923** when constructing a packet to send to the client **901**, and the client **901** will maintain a look-ahead buffer of expected IP pairs on packets that it is receiving. Each TARP router will maintain separate pairs of transmit and receive tables for each client that is currently engaged in a secure session with or through that TARP router.

While clients receive their hopblocks from the first server linking them to the Internet, routers exchange hopblocks. When a router establishes a link-based IP-hopping communication regime with another router, each router of the pair



## US 7,921,211 B2

19

exchanges its transmit hopblock. The transmit hopblock of each router becomes the receive hopblock of the other router. The communication between routers is governed as described by the example of a client sending a packet to the first router.

While the above strategy works fine in the IP milieu, many local networks that are connected to the Internet are Ethernet systems. In Ethernet, the IP addresses of the destination devices must be translated into hardware addresses, and vice versa, using known processes ("address resolution protocol," and "reverse address resolution protocol"). However, if the link-based IP-hopping strategy is employed, the correlation process would become explosive and burdensome. An alternative to the link-based IP hopping strategy may be employed within an Ethernet network. The solution is to provide that the node linking the Internet to the Ethernet (call it the border node) use the link-based IP-hopping communication regime to communicate with nodes outside the Ethernet LAN. Within the Ethernet LAN, each TARP node would have a single IP address which would be addressed in the conventional way. Instead of comparing the {sender, receiver} IP address pairs to authenticate a packet, the intra-LAN TARP node would use one of the IP header extension fields to do so. Thus, the border node uses an algorithm shared by the intra-LAN TARP node to generate a symbol that is stored in the free field in the IP header, and the intra-LAN TARP node generates a range of symbols based on its prediction of the next expected packet to be received from that particular source IP address. The packet is rejected if it does not fall into the set of predicted symbols (for example, numerical values) or is accepted if it does. Communications from the intra-LAN TARP node to the border node are accomplished in the same manner, though the algorithm will necessarily be different for security reasons. Thus, each of the communicating nodes will generate transmit and receive tables in a similar manner to that of FIG. 9; the intra-LAN TARP nodes transmit table will be identical to the border node's receive table, and the intra-LAN TARP node's receive table will be identical to the border node's transmit table.

The algorithm used for IP address-hopping can be any desired algorithm. For example, the algorithm can be a given pseudo-random number generator that generates numbers of the range covering the allowed IP addresses with a given seed. Alternatively, the session participants can assume a certain type of algorithm and specify simply a parameter for applying the algorithm. For example the assumed algorithm could be a particular pseudo-random number generator and the session participants could simply exchange seed values.

Note that there is no permanent physical distinction between the originating and destination terminal nodes. Either device at either end point can initiate a synchronization of the pair. Note also that the authentication/synchronization-request (and acknowledgment) and hopblock-exchange may all be served by a single message so that separate message exchanges may not be required.

As another extension to the stated architecture, multiple physical paths can be used by a client, in order to provide link redundancy and further thwart attempts at denial of service and traffic monitoring. As shown in FIG. 10, for example, client 1001 can establish three simultaneous sessions with each of three TARP routers provided by different ISPs 1011, 1012, 1013. As an example, the client 1001 can use three different telephone lines 1021, 1022, 1023 to connect to the ISPs, or two telephone lines and a cable modem, etc. In this scheme, transmitted packets will be sent in a random fashion among the different physical paths. This architecture pro-

20

vides a high degree of communications redundancy, with improved immunity from denial-of-service attacks and traffic monitoring.

## 2. Further Extensions

The following describes various extensions to the techniques, systems, and methods described above. As described above, the security of communications occurring between computers in a computer network (such as the Internet, an Ethernet, or others) can be enhanced by using seemingly random source and destination Internet Protocol (IP) addresses for data packets transmitted over the network. This feature prevents eavesdroppers from determining which computers in the network are communicating with each other while permitting the two communicating computers to easily recognize whether a given received data packet is legitimate or not. In one embodiment of the above-described systems, an IP header extension field is used to authenticate incoming packets on an Ethernet.

Various extensions to the previously described techniques described herein include: (1) use of hopped hardware or "MAC" addresses in broadcast type network; (2) a self synchronization technique that permits a computer to automatically regain synchronization with a sender; (3) synchronization algorithms that allow transmitting and receiving computers to quickly re-establish synchronization in the event of lost packets or other events; and (4) a fast-packet rejection mechanism for rejecting invalid packets. Any or all of these extensions can be combined with the features described above in any of various ways.

### A. Hardware Address Hopping

Internet protocol-based communications techniques on a LAN—or across any dedicated physical medium—typically embed the IP packets within lower-level packets, often referred to as "frames." As shown in FIG. 11, for example, a first Ethernet frame 1150 comprises a frame header 1101 and two embedded IP packets IP1 and IP2, while a second Ethernet frame 1160 comprises a different frame header 1104 and a single IP packet IP3. Each frame header generally includes a source hardware address 1101 A and a destination hardware address 1101 B; other well-known fields in frame headers are omitted from FIG. 11 for clarity. Two hardware nodes communicating over a physical communication channel insert appropriate source and destination hardware addresses to indicate which nodes on the channel or network should receive the frame.

It may be possible for a nefarious listener to acquire information about the contents of a frame and/or its communicants by examining frames on a local network rather than (or in addition to) the IP packets themselves. This is especially true in broadcast media, such as Ethernet, where it is necessary to insert into the frame header the hardware address of the machine that generated the frame and the hardware address of the machine to which frame is being sent. All nodes on the network can potentially "see" all packets transmitted across the network. This can be a problem for secure communications, especially in cases where the communicants do not want for any third party to be able to identify who is engaging in the information exchange. One way to address this problem is to push the address-hopping scheme down to the hardware layer. In accordance with various embodiments of the invention, hardware addresses are "hopped" in a manner similar to that used to change IP addresses, such that a listener cannot



## US 7,921,211 B2

21

determine which hardware node generated a particular message nor which node is the intended recipient.

FIG. 12A shows a system in which Media Access Control ("MAC") hardware addresses are "hopped" in order to increase security over a network such as an Ethernet. While the description refers to the exemplary case of an Ethernet environment, the inventive principles are equally applicable to other types of communications media. In the Ethernet case, the MAC address of the sender and receiver are inserted into the Ethernet frame and can be observed by anyone on the LAN who is within the broadcast range for that frame. For secure communications, it becomes desirable to generate frames with MAC addresses that are not attributable to any specific sender or receiver.

As shown in FIG. 12A, two computer nodes 1201 and 1202 communicate over a communication channel such as an Ethernet. Each node executes one or more application programs 1203 and 1218 that communicate by transmitting packets through communication software 1204 and 1217, respectively. Examples of application programs include video conferencing, e-mail, word processing programs, telephony, and the like. Communication software 1204 and 1217 can comprise, for example, an OSI layered architecture or "stack" that standardizes various services provided at different levels of functionality.

The lowest levels of communication software 1204 and 1217 communicate with hardware components 1206 and 1214 respectively, each of which can include one or more registers 1207 and 1215 that allow the hardware to be reconfigured or controlled in accordance with various communication protocols. The hardware components (an Ethernet network interface card, for example) communicate with each other over the communication medium. Each hardware component is typically pre-assigned a fixed hardware address or MAC number that identifies the hardware component to other nodes on the network. One or more interface drivers control the operation of each card and can, for example, be configured to accept or reject packets from certain hardware addresses. As will be described in more detail below, various embodiments of the inventive principles provide for "hopping" different addresses using one or more algorithms and one or more moving windows that track a range of valid addresses to validate received packets. Packets transmitted according to one or more of the inventive principles will be generally referred to as "secure" packets or "secure communications" to differentiate them from ordinary data packets that are transmitted in the clear using ordinary, machine-correlated addresses.

One straightforward method of generating non-attributable MAC addresses is an extension of the IP hopping scheme. In this scenario, two machines on the same LAN that desire to communicate in a secure fashion exchange random-number generators and seeds, and create sequences of quasi-random MAC addresses for synchronized hopping. The implementation and synchronization issues are then similar to that of IP hopping.

This approach, however, runs the risk of using MAC addresses that are currently active on the LAN—which, in turn, could interrupt communications for those machines. Since an Ethernet MAC address is at present 48 bits in length, the chance of randomly misusing an active MAC address is actually quite small. However, if that figure is multiplied by a large number of nodes (as would be found on an extensive LAN), by a large number of frames (as might be the case with packet voice or streaming video), and by a large number of concurrent Virtual Private Networks (VPNs), then the chance that a non-secure machine's MAC address could be used in an

22

address-hopped frame can become non-trivial. In short, any scheme that runs even a small risk of interrupting communications for other machines on the LAN is bound to receive resistance from prospective system administrators. Nevertheless, it is technically feasible, and can be implemented without risk on a LAN on which there is a small number of machines, or if all of the machines on the LAN are engaging in MAC-hopped communications.

Synchronized MAC address hopping may incur some overhead in the course of session establishment, especially if there are multiple sessions or multiple nodes involved in the communications. A simpler method of randomizing MAC addresses is to allow each node to receive and process every incident frame on the network. Typically, each network interface driver will check the destination MAC address in the header of every incident frame to see if it matches that machine's MAC address; if there is no match, then the frame is discarded. In one embodiment, however, these checks can be disabled, and every incident packet is passed to the TARP stack for processing. This will be referred to as "promiscuous" mode, since every incident frame is processed. Promiscuous mode allows the sender to use completely random, unsynchronized MAC addresses, since the destination machine is guaranteed to process the frame. The decision as to whether the packet was truly intended for that machine is handled by the TARP stack, which checks the source and destination IP addresses for a match in its IP synchronization tables. If no match is found, the packet is discarded; if there is a match, the packet is unwrapped, the inner header is evaluated, and if the inner header indicates that the packet is destined for that machine then the packet is forwarded to the IP stack—otherwise it is discarded.

One disadvantage of purely-random MAC address hopping is its impact on processing overhead; that is, since every incident frame must be processed, the machine's CPU is engaged considerably more often than if the network interface driver is discriminating and rejecting packets unilaterally. A compromise approach is to select either a single fixed MAC address or a small number of MAC addresses (e.g., one for each virtual private network on an Ethernet) to use for MAC-hopped communications, regardless of the actual recipient for which the message is intended. In this mode, the network interface driver can check each incident frame against one (or a few) pre-established MAC addresses, thereby freeing the CPU from the task of physical-layer packet discrimination. This scheme does not betray any useful information to an interloper on the LAN; in particular, every secure packet can already be identified by a unique packet type in the outer header. However, since all machines engaged in secure communications would either be using the same MAC address, or be selecting from a small pool of predetermined MAC addresses, the association between a specific machine and a specific MAC address is effectively broken.

In this scheme, the CPU will be engaged more often than it would be in non-secure communications (or in synchronized MAC address hopping), since the network interface driver cannot always unilaterally discriminate between secure packets that are destined for that machine, and secure packets from other VPNs. However, the non-secure traffic is easily eliminated at the network interface, thereby reducing the amount of processing required of the CPU. There are boundary conditions where these statements would not hold, of course—e.g., if all of the traffic on the LAN is secure traffic, then the CPU would be engaged to the same degree as it is in the purely-random address hopping case; alternatively, if each VPN on the LAN uses a different MAC address, then the

## US 7,921,211 B2

23

network interface can perfectly discriminate secure frames destined for the local machine from those constituting other VPNs. These are engineering tradeoffs that might be best handled by providing administrative options for the users when installing the software and/or establishing VPNs.

Even in this scenario, however, there still remains a slight risk of selecting MAC addresses that are being used by one or more nodes on the LAN. One solution to this problem is to formally assign one address or a range of addresses for use in MAC-hopped communications. This is typically done via an assigned numbers registration authority; e.g., in the case of Ethernet, MAC address ranges are assigned to vendors by the Institute of Electrical and Electronics Engineers (IEEE). A formally-assigned range of addresses would ensure that secure frames do not conflict with any properly-configured and properly-functioning machines on the LAN.

Reference will now be made to FIGS. 12A and 12B in order to describe the many combinations and features that follow the inventive principles. As explained above, two computer nodes **1201** and **1202** are assumed to be communicating over a network or communication medium such as an Ethernet. A communication protocol in each node (**1204** and **1217**, respectively) contains a modified element **1205** and **1216** that performs certain functions that deviate from the standard communication protocols. In particular, computer node **1201** implements a first "hop" algorithm **1208X** that selects seemingly random source and destination IP addresses (and, in one embodiment, seemingly random IP header discriminator fields) in order to transmit each packet to the other computer node. For example, node **1201** maintains a transmit table **1208** containing triplets of source (S), destination (D), and discriminator fields (DS) that are inserted into outgoing IP packet headers. The table is generated through the use of an appropriate algorithm (e.g., a random number generator that is seeded with an appropriate seed) that is known to the recipient node **1202**. As each new IP packet is formed, the next sequential entry out of the sender's transmit table **1208** is used to populate the IP source, IP destination, and IP header extension field (e.g., discriminator field). It will be appreciated that the transmit table need not be created in advance but could instead be created on-the-fly by executing the algorithm when each packet is formed.

At the receiving node **1202**, the same IP hop algorithm **1222X** is maintained and used to generate a receive table **1222** that lists valid triplets of source IP address, destination IP address, and discriminator field. This is shown by virtue of the first five entries of transmit table **1208** matching the second five entries of receive table **1222**. (The tables may be slightly offset at any particular time due to lost packets, mis-ordered packets, or transmission delays). Additionally, node **1202** maintains a receive window **W3** that represents a list of valid IP source, IP destination, and discriminator fields that will be accepted when received as part of an incoming IP packet. As packets are received, window **W3** slides down the list of valid entries, such that the possible valid entries change over time. Two packets that arrive out of order but are nevertheless matched to entries within window **W3** will be accepted; those falling outside of window **W3** will be rejected as invalid. The length of window **W3** can be adjusted as necessary to reflect network delays or other factors.

Node **1202** maintains a similar transmit table **1221** for creating IP packets and frames destined for node **1201** using a potentially different hopping algorithm **1221 X**, and node **1201** maintains a matching receive table **1209** using the same algorithm **1209X**. As node **1202** transmits packets to node **1201** using seemingly random IP source, IP destination, and/or discriminator fields, node **1201** matches the incoming

24

packet values to those falling within window **W1** maintained in its receive table. In effect, transmit table **1208** of node **1201** is synchronized (i.e., entries are selected in the same order) to receive table **1222** of receiving node **1202**. Similarly, transmit table **1221** of node **1202** is synchronized to receive table **1209** of node **1201**. It will be appreciated that although a common algorithm is shown for the source, destination and discriminator fields in FIG. 12A (using, e.g., a different seed for each of the three fields), an entirely different algorithm could in fact be used to establish values for each of these fields. It will also be appreciated that one or two of the fields can be "hopped" rather than all three as illustrated.

In accordance with another aspect of the invention, hardware or "MAC" addresses are hopped instead of or in addition to IP addresses and/or the discriminator field in order to improve security in a local area or broadcast-type network. To that end, node **1201** further maintains a transmit table **1210** using a transmit algorithm **1210X** to generate source and destination hardware addresses that are inserted into frame headers (e.g., fields **1101A** and **1101 B** in FIG. 11) that are synchronized to a corresponding receive table **1224** at node **1202**. Similarly, node **1202** maintains a different transmit table **1223** containing source and destination hardware addresses that is synchronized with a corresponding receive table **1211** at node **1201**. In this manner, outgoing hardware frames appear to be originating from and going to completely random nodes on the network, even though each recipient can determine whether a given packet is intended for it or not. It will be appreciated that the hardware hopping feature can be implemented at a different level in the communications protocol than the IP hopping feature (e.g., in a card driver or in a hardware card itself to improve performance).

FIG. 12B shows three different embodiments or modes that can be employed using the aforementioned principles. In a first mode referred to as "promiscuous" mode, a common hardware address (e.g., a fixed address for source and another for destination) or else a completely random hardware address is used by all nodes on the network, such that a particular packet cannot be attributed to any one node. Each node must initially accept all packets containing the common (or random) hardware address and inspect the IP addresses or discriminator field to determine whether the packet is intended for that node. In this regard, either the IP addresses or the discriminator field or both can be varied in accordance with an algorithm as described above. As explained previously, this may increase each node's overhead since additional processing is involved to determine whether a given packet has valid source and destination hardware addresses.

In a second mode referred to as "promiscuous per VPN" mode, a small set of fixed hardware addresses are used, with a fixed source/destination hardware address used for all nodes communicating over a virtual private network. For example, if there are six nodes on an Ethernet, and the network is to be split up into two private virtual networks such that nodes on one VPN can communicate with only the other two nodes on its own VPN, then two sets of hardware addresses could be used: one set for the first VPN and a second set for the second VPN. This would reduce the amount of overhead involved in checking for valid frames since only packets arriving from the designated VPN would need to be checked. IP addresses and one or more discriminator fields could still be hopped as before for secure communication within the VPN. Of course, this solution compromises the anonymity of the VPNs (i.e., an outsider can easily tell what traffic belongs in which VPN, though he cannot correlate it to a specific machine/person). It also requires the use of a discriminator field to mitigate the vulnerability to certain types of DoS attacks, (For example,

## US 7,921,211 B2

25

without the discriminator field, an attacker on the LAN could stream frames containing the MAC addresses being used by the VPN; rejecting those frames could lead to excessive processing overhead. The discriminator field would provide a low-overhead means of rejecting the false packets.)

In a third mode referred to as “hardware hopping” mode, hardware addresses are varied as illustrated in FIG. 12A, such that hardware source and destination addresses are changed constantly in order to provide non-attributable addressing. Variations on these embodiments are of course possible, and the invention is not intended to be limited in any respect by these illustrative examples.

#### B. Extending the Address Space Address

Address hopping provides security and privacy. However, the level of protection is limited by the number of addresses in the blocks being hopped. A hopblock denotes a field or fields modulated on a packet-wise basis for the purpose of providing a VPN. For instance, if two nodes communicate with IP address hopping using hopblocks of 4 addresses (2 bits) each, there would be 16 possible address-pair combinations. A window of size 16 would result in most address pairs being accepted as valid most of the time. This limitation can be overcome by using a discriminator field in addition to or instead of the hopped address fields. The discriminator field would be hopped in exactly the same fashion as the address fields and it would be used to determine whether a packet should be processed by a receiver.

Suppose that two clients, each using four-bit hopblocks, would like the same level of protection afforded to clients communicating via IP hopping between two A blocks (24 address bits eligible for hopping). A discriminator field of 20 bits, used in conjunction with the 4 address bits eligible for hopping in the IP address field, provides this level of protection. A 24-bit discriminator field would provide a similar level of protection if the address fields were not hopped or ignored. Using a discriminator field offers the following advantages: (1) an arbitrarily high level of protection can be provided, and (2) address hopping is unnecessary to provide protection. This may be important in environments where address hopping would cause routing problems.

#### C. Synchronization Techniques

It is generally assumed that once a sending node and receiving node have exchanged algorithms and seeds (or similar information sufficient to generate quasi-random source and destination tables), subsequent communication between the two nodes will proceed smoothly. Realistically, however, two nodes may lose synchronization due to network delays or outages, or other problems. Consequently, it is desirable to provide means for re-establishing synchronization between nodes in a network that have lost synchronization.

One possible technique is to require that each node provide an acknowledgment upon successful receipt of each packet and, if no acknowledgment is received within a certain period of time, to re-send the unacknowledged packet. This approach, however, drives up overhead costs and may be prohibitive in high-throughput environments such as streaming video or audio, for example.

A different approach is to employ an automatic synchronizing technique that will be referred to herein as “self-synchronization.” In this approach, synchronization information is embedded into each packet, thereby enabling the receiver to re-synchronize itself upon receipt of a single packet if it

26

determines that it has lost synchronization with the sender. (If communications are already in progress, and the receiver determines that it is still in sync with the sender, then there is no need to re-synchronize.) A receiver could detect that it was out of synchronization by, for example, employing a “dead-man” timer that expires after a certain period of time, wherein the timer is reset with each valid packet. A time stamp could be hashed into the public sync field (see below) to preclude packet-retry attacks.

In one embodiment, a “sync field” is added to the header of each packet sent out by the sender. This sync field could appear in the clear or as part of an encrypted portion of the packet. Assuming that a sender and receiver have selected a random-number generator (RNG) and seed value, this combination of RNG and seed can be used to generate a random-number sequence (RNS). The RNS is then used to generate a sequence of source/destination IP pairs (and, if desired, discriminator fields and hardware source and destination addresses), as described above. It is not necessary, however, to generate the entire sequence (or the first N-1 values) in order to generate the Nth random number in the sequence; if the sequence index N is known, the random value corresponding to that index can be directly generated (see below). Different RNGs (and seeds) with different fundamental periods could be used to generate the source and destination IP sequences, but the basic concepts would still apply. For the sake of simplicity, the following discussion will assume that IP source and destination address pairs (only) are hopped using a single RNG sequencing mechanism.

In accordance with a “self-synchronization” feature, a sync field in each packet header provides an index (i.e., a sequence number) into the RNS that is being used to generate IP pairs. Plugging this index into the RNG that is being used to generate the RNS yields a specific random number value, which in turn yields a specific IP pair. That is, an IP pair can be generated directly from knowledge of the RNG, seed, and index number; it is not necessary, in this scheme, to generate the entire sequence of random numbers that precede the sequence value associated with the index number provided.

Since the communicants have presumably previously exchanged RNGs and seeds, the only new information that must be provided in order to generate an IP pair is the sequence number. If this number is provided by the sender in the packet header, then the receiver need only plug this number into the RNG in order to generate an IP pair—and thus verify that the IP pair appearing in the header of the packet is valid. In this scheme, if the sender and receiver lose synchronization, the receiver can immediately re-synchronize upon receipt of a single packet by simply comparing the IP pair in the packet header to the IP pair generated from the index number. Thus, synchronized communications can be resumed upon receipt of a single packet, making this scheme ideal for multicast communications. Taken to the extreme, it could obviate the need for synchronization tables entirely; that is, the sender and receiver could simply rely on the index number in the sync field to validate the IP pair on each packet, and thereby eliminate the tables entirely.

The aforementioned scheme may have some inherent security issues associated with it—namely, the placement of the sync field. If the field is placed in the outer header, then an interloper could observe the values of the field and their relationship to the IP stream. This could potentially compromise the algorithm that is being used to generate the IP-address sequence, which would compromise the security of the communications. If, however, the value is placed in the inner header, then the sender must decrypt the inner header before it can extract the sync value and validate the IP pair;

## US 7,921,211 B2

27

this opens up the receiver to certain types of denial-of-service (DoS) attacks, such as packet replay. That is, if the receiver must decrypt a packet before it can validate the IP pair, then it could potentially be forced to expend a significant amount of processing on decryption if an attacker simply retransmits previously valid packets. Other attack methodologies are possible in this scenario.

A possible compromise between algorithm security and processing speed is to split up the sync value between an inner (encrypted) and outer (unencrypted) header. That is, if the sync value is sufficiently long, it could potentially be split into a rapidly-changing part that can be viewed in the clear, and a fixed (or very slowly changing) part that must be protected. The part that can be viewed in the clear will be called the “public sync” portion and the part that must be protected will be called the “private sync” portion.

Both the public sync and private sync portions are needed to generate the complete sync value. The private portion, however, can be selected such that it is fixed or will change only occasionally. Thus, the private sync value can be stored by the recipient, thereby obviating the need to decrypt the header in order to retrieve it. If the sender and receiver have previously agreed upon the frequency with which the private part of the sync will change, then the receiver can selectively decrypt a single header in order to extract the new private sync if the communications gap that has led to lost synchronization has exceeded the lifetime of the previous private sync. This should not represent a burdensome amount of decryption, and thus should not open up the receiver to denial-of-service attack simply based on the need to occasionally decrypt a single header.

One implementation of this is to use a hashing function with a one-to-one mapping to generate the private and public sync portions from the sync value. This implementation is shown in FIG. 13, where (for example) a first ISP 1302 is the sender and a second ISP 1303 is the receiver. (Other alternatives are possible from FIG. 13.) A transmitted packet comprises a public or “outer” header 1305 that is not encrypted, and a private or “inner” header 1306 that is encrypted using for example a link key. Outer header 1305 includes a public sync portion while inner header 1306 contains the private sync portion. A receiving node decrypts the inner header using a decryption function 1307 in order to extract the private sync portion. This step is necessary only if the lifetime of the currently buffered private sync has expired. (If the currently-buffered private sync is still valid, then it is simply extracted from memory and “added” (which could be an inverse hash) to the public sync, as shown in step 1308.) The public and decrypted private sync portions are combined in function 1308 in order to generate the combined sync 1309. The combined sync (1309) is then fed into the RNG (1310) and compared to the IP address pair (1311) to validate or reject the packet.

An important consideration in this architecture is the concept of “future” and “past” where the public sync values are concerned. Though the sync values, themselves, should be random to prevent spoofing attacks, it may be important that the receiver be able to quickly identify a sync value that has already been sent—even if the packet containing that sync value was never actually received by the receiver. One solution is to hash a time stamp or sequence number into the public sync portion, which could be quickly extracted, checked, and discarded, thereby validating the public sync portion itself.

In one embodiment, packets can be checked by comparing the source/destination IP pair generated by the sync field with the pair appearing in the packet header. If (1) they match, (2)

28

the time stamp is valid, and (3) the dead-man timer has expired, then re-synchronization occurs; otherwise, the packet is rejected. If enough processing power is available, the dead-man timer and synchronization tables can be avoided altogether, and the receiver would simply resynchronize (e.g., validate) on every packet.

The foregoing scheme may require large-integer (e.g., 160-bit) math, which may affect its implementation. Without such large-integer registers, processing throughput would be affected, thus potentially affecting security from a denial-of-service standpoint. Nevertheless, as large integer math processing features become more prevalent, the costs of implementing such a feature will be reduced.

#### D. Other Synchronization Schemes

As explained above, if W or more consecutive packets are lost between a transmitter and receiver in a VPN (where W is the window size), the receiver’s window will not have been updated and the transmitter will be transmitting packets not in the receiver’s window. The sender and receiver will not recover synchronization until perhaps the random pairs in the window are repeated by chance. Therefore, there is a need to keep a transmitter and receiver in synchronization whenever possible and to re-establish synchronization whenever it is lost.

A “checkpoint” scheme can be used to regain synchronization between a sender and a receiver that have fallen out of synchronization. In this scheme, a checkpoint message comprising a random IP address pair is used for communicating synchronization information. In one embodiment, two messages are used to communicate synchronization information between a sender and a recipient:

1. SYNC\_REQ is a message used by the sender to indicate that it wants to synchronize; and
2. SYNC\_ACK is a message used by the receiver to inform the transmitter that it has been synchronized.

According to one variation of this approach, both the transmitter and receiver maintain three checkpoints (see FIG. 14):

1. In the transmitter, ckpt\_o (“checkpoint old”) is the IP pair that was used to re-send the last SYNC\_REQ packet to the receiver. In the receiver, ckpt\_o (“checkpoint old”) is the IP pair that receives repeated SYNC\_REQ packets from the transmitter.
2. In the transmitter, ckpt\_n (“checkpoint new”) is the IP pair that will be used to send the next SYNC\_REQ packet to the receiver. In the receiver, ckpt\_n (“checkpoint new”) is the IP pair that receives a new SYNC\_REQ packet from the transmitter and which causes the receiver’s window to be re-aligned, ckpt\_o set to ckpt\_n, a new ckpt\_n to be generated and a new ckpt\_r to be generated.
3. In the transmitter, ckpt\_r is the IP pair that will be used to send the next SYNC\_ACK packet to the receiver. In the receiver, ckpt\_r is the IP pair that receives a new SYNC\_ACK packet from the transmitter and which causes a new ckpt\_n to be generated. Since SYNC\_ACK is transmitted from the receiver ISP to the sender ISP, the transmitter ckpt\_r refers to the ckpt\_r of the receiver and the receiver ckpt\_r refers to the ckpt\_r of the transmitter (see FIG. 14).

When a transmitter initiates synchronization, the IP pair it will use to transmit the next data packet is set to a predetermined value and when a receiver first receives a SYNC\_REQ, the receiver window is updated to be centered on the transmitter’s next IP pair. This is the primary mechanism for checkpoint synchronization.

## US 7,921,211 B2

29

Synchronization can be initiated by a packet counter (e.g., after every N packets transmitted, initiate a synchronization) or by a timer (every S seconds, initiate a synchronization) or a combination of both. See FIG. 15. From the transmitter's perspective, this technique operates as follows: (1) Each transmitter periodically transmits a "sync request" message to the receiver to make sure that it is in sync. (2) If the receiver is still in sync, it sends back a "sync ack" message. (If this works, no further action is necessary). (3) If no "sync ack" has been received within a period of time, the transmitter retransmits the sync request again. If the transmitter reaches the next checkpoint without receiving a "sync ack" response, then synchronization is broken, and the transmitter should stop transmitting. The transmitter will continue to send sync\_reqs until it receives a sync\_ack, at which point transmission is reestablished.

From the receiver's perspective, the scheme operates as follows: (1) when it receives a "sync request" request from the transmitter, it advances its window to the next checkpoint position (even skipping pairs if necessary), and sends a "sync ack" message to the transmitter. If sync was never lost, then the "jump ahead" really just advances to the next available pair of addresses in the table (i.e., normal advancement).

If an interloper intercepts the "sync request" messages and tries to interfere with communication by sending new ones, it will be ignored if the synchronization has been established or it will actually help to re-establish synchronization.

A window is realigned whenever a re-synchronization occurs. This realignment entails updating the receiver's window to straddle the address pairs used by the packet transmitted immediately after the transmission of the SYNC\_REQ packet. Normally, the transmitter and receiver are in synchronization with one another. However, when network events occur, the receiver's window may have to be advanced by many steps during resynchronization. In this case, it is desirable to move the window ahead without having to step through the intervening random numbers sequentially. (This feature is also desirable for the auto-sync approach discussed above).

#### E. Random Number Generator with a Jump-Ahead Capability

An attractive method for generating randomly hopped addresses is to use identical random number generators in the transmitter and receiver and advance them as packets are transmitted and received. There are many random number generation algorithms that could be used. Each one has strengths and weaknesses for address hopping applications.

Linear congruential random number generators (LCRs) are fast, simple and well characterized random number generators that can be made to jump ahead n steps efficiently. An LCR generates random numbers  $X_1, X_2, X_3 \dots X_k$  starting with seed  $X_0$  using a recurrence

$$X_i = (aX_{i-1} + b) \bmod c, \quad (1)$$

where a, b and c define a particular LCR. Another expression for  $X_i$ ,

$$X_i = ((a^i(X_0 + b) - b) / (a - 1)) \bmod c \quad (2)$$

enables the jump-ahead capability. The factor  $a^i$  can grow very large even for modest i if left unfettered. Therefore some special properties of the modulo operation can be used to control the size and processing time required to compute (2). (2) can be rewritten as:

$$X_i = (a^i(X_0(a-1)+b)-b)/(a-1) \bmod c. \quad (3)$$

30

It can be shown that:

$$(a^i(X_0(a-1)+b)-b)/(a-1) \bmod c = ((a^i \bmod ((a-1)c)(X_0(a-1)+b)-b)/(a-1)) \bmod c \quad (4)$$

$(X_0(a-1)+b)$  can be stored as  $(X_0(a-1)+b) \bmod c$ , b as  $b \bmod c$  and compute  $a^i \bmod ((a-1)c)$  (this requires  $O(\log(i))$  steps).

A practical implementation of this algorithm would jump a fixed distance, n, between synchronizations; this is tantamount to synchronizing every n packets. The window would commence n IP pairs from the start of the previous window. Using  $X_j^w$ , the random number at the  $j^{\text{th}}$  checkpoint, as  $X_0$  and n as i, a node can store  $a^n \bmod ((a-1)c)$  once per LCR and set

$$X_{j+1}^w = X_{n(j+1)} = ((a^n \bmod ((a-1)c)(X_j^w(a-1)+b)-b)/(a-1)) \bmod c, \quad (5)$$

to generate the random number for the  $j+1^{\text{th}}$  synchronization. Using this construction, a node could jump ahead an arbitrary (but fixed) distance between synchronizations in a constant amount of time (independent of n).

Pseudo-random number generators, in general, and LCRs, in particular, will eventually repeat their cycles. This repetition may present vulnerability in the IP hopping scheme. An adversary would simply have to wait for a repeat to predict future sequences. One way of coping with this vulnerability is to create a random number generator with a known long cycle. A random sequence can be replaced by a new random number generator before it repeats. LCRs can be constructed with known long cycles. This is not currently true of many random number generators.

Random number generators can be cryptographically insecure. An adversary can derive the RNG parameters by examining the output or part of the output. This is true of LCGs. This vulnerability can be mitigated by incorporating an encryptor, designed to scramble the output as part of the random number generator. The random number generator prevents an adversary from mounting an attack—e.g., a known plaintext attack—against the encryptor.

#### F. Random Number Generator Example

Consider a RNG where  $a=31$ ,  $b=4$  and  $c=15$ . For this case equation (1) becomes:

$$X_i = (31X_{i-1} + 4) \bmod 15. \quad (6)$$

If one sets  $X_0=1$ , equation (6) will produce the sequence 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 0, 4, 8, 12. This sequence will repeat indefinitely. For a jump ahead of 3 numbers in this sequence  $a^n=31^3=29791$ ,  $c*(a-1)=15*30=450$  and  $a^n \bmod ((a-1)c)=31^3 \bmod (15*30)=29791 \bmod 450=91$ . Equation (5) becomes:

$$((91(X_0+4)-4)/30) \bmod 15 \quad (7)$$

Table 1 shows the jump ahead calculations from (7). The calculations start at 5 and jump ahead 3.

TABLE 1

I	$X_i$	$(X_i, 30 + 4)$	$91(X_i, 30 + 4) - 4$	$((91(X_i, 30 + 4) - 4)/30)$	$X_{i+3}$
1	5	154	14010	467	2
4	2	64	5820	194	14
7	14	424	38580	1286	11
10	11	334	30390	1013	8
13	8	244	22200	740	5

## US 7,921,211 B2

31

## G. Fast Packet Filter

Address hopping VPNs must rapidly determine whether a packet has a valid header and thus requires further processing, or has an invalid header (a hostile packet) and should be immediately rejected. Such rapid determinations will be referred to as “fast packet filtering.” This capability protects the VPN from attacks by an adversary who streams hostile packets at the receiver at a high rate of speed in the hope of saturating the receiver’s processor (a so-called “denial of service” attack). Fast packet filtering is an important feature for implementing VPNs on shared media such as Ethernet.

Assuming that all participants in a VPN share an unsigned “A” block of addresses, one possibility is to use an experimental “A” block that will never be assigned to any machine that is not address hopping on the shared medium. “A” blocks have a 24 bits of address that can be hopped as opposed to the 8 bits in “C” blocks. In this case a hopblock will be the “A” block. The use of the experimental “A” block is a likely option on an Ethernet because:

1. The addresses have no validity outside of the Ethernet and will not be routed out to a valid outside destination by a gateway.
2. There are  $2^{24}$  (–16 million) addresses that can be hopped within each “A” block. This yields >280 trillion possible address pairs making it very unlikely that an adversary would guess a valid address. It also provides acceptably low probability of collision between separate VPNs (all VPNs on a shared medium independently generate random address pairs from the same “A” block).
3. The packets will not be received by someone on the Ethernet who is not on a VPN (unless the machine is in promiscuous mode) minimizing impact on non-VPN computers.

The Ethernet example will be used to describe one implementation of fast packet filtering. The ideal algorithm would quickly examine a packet header, determine whether the packet is hostile, and reject any hostile packets or determine which active IP pair the packet header matches. The problem is a classical associative memory problem. A variety of techniques have been developed to solve this problem (hashing, B—trees etc). Each of these approaches has its strengths and weaknesses. For instance, hash tables can be made to operate quite fast in a statistical sense, but can occasionally degenerate into a much slower algorithm. This slowness can persist for a period of time. Since there is a need to discard hostile packets quickly at all times, hashing would be unacceptable.

## H. Presence Vector Algorithm

A presence vector is a bit vector of length  $2^n$  that can be indexed by n-bit numbers (each ranging from 0 to  $2^n-1$ ). One can indicate the presence of k n-bit numbers (not necessarily unique), by setting the bits in the presence vector indexed by each number to 1. Otherwise, the bits in the presence vector are 0. An n-bit number, x, is one of the k numbers if and only if the  $x^{th}$  bit of the presence vector is 1. A fast packet filter can be implemented by indexing the presence vector and looking for a 1, which will be referred to as the “test.”

For example, suppose one wanted to represent the number 135 using a presence vector. The  $135^{th}$  bit of the vector would be set. Consequently, one could very quickly determine whether an address of 135 was valid by checking only one bit: the  $135^{th}$  bit. The presence vectors could be created in advance corresponding to the table entries for the IP addresses. In effect, the incoming addresses can be used as indices into a long vector, making comparisons very fast. As

32

each RNG generates a new address, the presence vector is updated to reflect the information. As the window moves, the presence vector is updated to zero out addresses that are no longer valid.

There is a trade-off between efficiency of the test and the amount of memory required for storing the presence vector (s). For instance, if one were to use the 48 bits of hopping addresses as an index, the presence vector would have to be 35 terabytes. Clearly, this is too large for practical purposes. Instead, the 48 bits can be divided into several smaller fields. For instance, one could subdivide the 48 bits into four 12-bit fields (see FIG. 16). This reduces the storage requirement to 2048 bytes at the expense of occasionally having to process a hostile packet. In effect, instead of one long presence vector, the decomposed address portions must match all four shorter presence vectors before further processing is allowed. (If the first part of the address portion doesn’t match the first presence vector, there is no need to check the remaining three presence vectors).

A presence vector will have a 1 in the  $y^{th}$  bit if and only if one or more addresses with a corresponding field of y are active. An address is active only if each presence vector indexed by the appropriate sub-field of the address is 1.

Consider a window of 32 active addresses and 3 checkpoints. A hostile packet will be rejected by the indexing of one presence vector more than 99% of the time. A hostile packet will be rejected by the indexing of all 4 presence vectors more than 99.9999995% of the time. On average, hostile packets will be rejected in less than 1.02 presence vector index operations.

The small percentage of hostile packets that pass the fast packet filter will be rejected when matching pairs are not found in the active window or are active checkpoints. Hostile packets that serendipitously match a header will be rejected when the VPN software attempts to decrypt the header. However, these cases will be extremely rare. There are many other ways this method can be configured to arbitrate the space/speed tradeoffs.

## I. Further Synchronization Enhancements

A slightly modified form of the synchronization techniques described above can be employed. The basic principles of the previously described checkpoint synchronization scheme remain unchanged. The actions resulting from the reception of the checkpoints are, however, slightly different. In this variation, the receiver will maintain between OoO (“Out of Order”) and  $2 \times \text{WINDOW\_SIZE} + \text{OoO}$  active addresses ( $1 \leq \text{OoO} \leq \text{WINDOW\_SIZE}$  and  $\text{WINDOW\_SIZE} \geq 1$ ). OoO and WINDOW\_SIZE are engineerable parameters, where OoO is the minimum number of addresses needed to accommodate lost packets due to events in the network or out of order arrivals and WINDOW\_SIZE is the number of packets transmitted before a SYNC\_REQ is issued. FIG. 17 depicts a storage array for a receiver’s active addresses.

The receiver starts with the first  $2 \times \text{WINDOW\_SIZE}$  addresses loaded and active (ready to receive data). As packets are received, the corresponding entries are marked as “used” and are no longer eligible to receive packets. The transmitter maintains a packet counter, initially set to 0, containing the number of data packets transmitted since the last initial transmission of a SYNC\_REQ for which SYNC\_ACK has been received. When the transmitter packet counter equals WINDOW\_SIZE, the transmitter generates a SYNC\_C\_REQ and does its initial transmission. When the receiver receives a SYNC\_REQ corresponding to its current CKPT\_N, it generates the next WINDOW\_SIZE addresses

## US 7,921,211 B2

33

and starts loading them in order starting at the first location after the last active address wrapping around to the beginning of the array after the end of the array has been reached. The receiver's array might look like FIG. 18 when a SYNC\_REQ has been received. In this case a couple of packets have been either lost or will be received out of order when the SYNC\_REQ is received.

FIG. 19 shows the receiver's array after the new addresses have been generated. If the transmitter does not receive a SYNC\_ACK, it will re-issue the SYNC\_REQ at regular intervals. When the transmitter receives a SYNC\_ACK, the packet counter is decremented by WINDOW\_SIZE. If the packet counter reaches  $2 \times \text{WINDOW\_SIZE}$ —OoO then the transmitter ceases sending data packets until the appropriate SYNC\_ACK is finally received. The transmitter then resumes sending data packets. Future behavior is essentially a repetition of this initial cycle. The advantages of this approach are:

1. There is no need for an efficient jump ahead in the random number generator,
2. No packet is ever transmitted that does not have a corresponding entry in the receiver side
3. No timer based re-synchronization is necessary. This is a consequence of 2.
4. The receiver will always have the ability to accept data messages transmitted within OoO messages of the most recently transmitted message.

#### J. Distributed Transmission Path Variant

Another embodiment incorporating various inventive principles is shown in FIG. 20. In this embodiment, a message transmission system includes a first computer 2001 in communication with a second computer 2002 through a network 2011 of intermediary computers. In one variant of this embodiment, the network includes two edge routers 2003 and 2004 each of which is linked to a plurality of Internet Service Providers (ISPs) 2005 through 2010. Each ISP is coupled to a plurality of other ISPs in an arrangement as shown in FIG. 20, which is a representative configuration only and is not intended to be limiting. Each connection between ISPs is labeled in FIG. 20 to indicate a specific physical transmission path (e.g., AD is a physical path that links ISP A (element 2005) to ISP D (element 2008)). Packets arriving at each edge router are selectively transmitted to one of the ISPs to which the router is attached on the basis of a randomly or quasi-randomly selected basis.

As shown in FIG. 21, computer 2001 or edge router 2003 incorporates a plurality of link transmission tables 2100 that identify, for each potential transmission path through the network, valid sets of IP addresses that can be used to transmit the packet. For example, AD table 2101 contains a plurality of IP source/destination pairs that are randomly or quasi-randomly generated. When a packet is to be transmitted from first computer 2001 to second computer 2002, one of the link tables is randomly (or quasi-randomly) selected, and the next valid source/destination address pair from that table is used to transmit the packet through the network. If path AD is randomly selected, for example, the next source/destination IP address pair (which is predetermined to transmit between ISP A (element 2005) and ISP B (element 2008)) is used to transmit the packet. If one of the transmission paths becomes degraded or inoperative, that link table can be set to a "down" condition as shown in table 2105, thus preventing addresses from being selected from that table. Other transmission paths would be unaffected by this broken link.

#### 3. Continuation-in-Part Improvements

The following describes various improvements and features that can be applied to the embodiments described above.

34

The improvements include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities. Each is discussed separately below.

#### A. Load Balancer

Various embodiments described above include a system in which a transmitting node and a receiving node are coupled through a plurality of transmission paths, and wherein successive packets are distributed quasi-randomly over the plurality of paths. See, for example, FIGS. 20 and 21 and accompanying description. The improvement extends this basic concept to encompass distributing packets across different paths in such a manner that the loads on the paths are generally balanced according to transmission link quality.

In one embodiment, a system includes a transmitting node and a receiving node that are linked via a plurality of transmission paths having potentially varying transmission quality. Successive packets are transmitted over the paths based on a weight value distribution function for each path. The rate that packets will be transmitted over a given path can be different for each path. The relative "health" of each transmission path is monitored in order to identify paths that have become degraded. In one embodiment, the health of each path is monitored in the transmitter by comparing the number of packets transmitted to the number of packet acknowledgements received. Each transmission path may comprise a physically separate path (e.g., via dial-up phone line, computer network, router, bridge, or the like), or may comprise logically separate paths contained within a broadband communication medium (e.g., separate channels in an FDM, TDM, CDMA, or other type of modulated or unmodulated transmission link).

When the transmission quality of a path falls below a predetermined threshold and there are other paths that can transmit packets, the transmitter changes the weight value used for that path, making it less likely that a given packet will be transmitted over that path. The weight will preferably be set no lower than a minimum value that keeps nominal traffic on the path. The weights of the other available paths are altered to compensate for the change in the affected path. When the quality of a path degrades to where the transmitter is turned off by the synchronization function (i.e., no packets are arriving at the destination), the weight is set to zero. If all transmitters are turned off, no packets are sent.

Conventional TCP/IP protocols include a "throttling" feature that reduces the transmission rate of packets when it is determined that delays or errors are occurring in transmission. In this respect, timers are sometimes used to determine whether packets have been received. These conventional techniques for limiting transmission of packets, however, do not involve multiple transmission paths between two nodes wherein transmission across a particular path relative to the others is changed based on link quality.

According to certain embodiments, in order to damp oscillations that might otherwise occur if weight distributions are changed drastically (e.g., according to a step function), a



## US 7,921,211 B2

35

linear or an exponential decay formula can be applied to gradually decrease the weight value over time that a degrading path will be used. Similarly, if the health of a degraded path improves, the weight value for that path is gradually increased.

Transmission link health can be evaluated by comparing the number of packets that are acknowledged within the transmission window (see embodiments discussed above) to the number of packets transmitted within that window and by the state of the transmitter (i.e., on or off). In other words, rather than accumulating general transmission statistics over time for a path, one specific implementation uses the "windowing" concepts described above to evaluate transmission path health.

The same scheme can be used to shift virtual circuit paths from an "unhealthy" path to a "healthy" one, and to select a path for a new virtual circuit.

FIG. 22A shows a flowchart for adjusting weight values associated with a plurality of transmission links. It is assumed that software executing in one or more computer nodes executes the steps shown in FIG. 22A. It is also assumed that the software can be stored on a computer-readable medium such as a magnetic or optical disk for execution by a computer.

Beginning in step 2201, the transmission quality of a given transmission path is measured. As described above, this measurement can be based on a comparison between the number of packets transmitted over a particular link to the number of packet acknowledgements received over the link (e.g., per unit time, or in absolute terms). Alternatively, the quality can be evaluated by comparing the number of packets that are acknowledged within the transmission window to the number of packets that were transmitted within that window. In yet another variation, the number of missed synchronization messages can be used to indicate link quality. Many other variations are of course possible.

In step 2202, a check is made to determine whether more than one transmitter (e.g., transmission path) is turned on. If not, the process is terminated and resumes at step 2201.

In step 2203, the link quality is compared to a given threshold (e.g., 50%, or any arbitrary number). If the quality falls below the threshold, then in step 2207 a check is made to determine whether the weight is above a minimum level (e.g., 1%). If not, then in step 2209 the weight is set to the minimum level and processing resumes at step 2201. If the weight is above the minimum level, then in step 2208 the weight is gradually decreased for the path, then in step 2206 the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are increased).

If in step 2203 the quality of the path was greater than or equal to the threshold, then in step 2204 a check is made to determine whether the weight is less than a steady-state value for that path. If so, then in step 2205 the weight is increased toward the steady-state value, and in step 2206 the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are decreased). If in step 2204 the weight is not less than the steady-state value, then processing resumes at step 2201 without adjusting the weights.

The weights can be adjusted incrementally according to various functions, preferably by changing the value gradually. In one embodiment, a linearly decreasing function is used to adjust the weights; according to another embodiment, an exponential decay function is used. Gradually changing the weights helps to damp oscillators that might otherwise occur if the probabilities were abruptly.

Although not explicitly shown in FIG. 22A the process can be performed only periodically (e.g., according to a time

36

schedule), or it can be continuously run, such as in a background mode of operation. In one embodiment, the combined weights of all potential paths should add up to unity (e.g., when the weighting for one path is decreased, the corresponding weights that the other paths will be selected will increase).

Adjustments to weight values for other paths can be prorated. For example, a decrease of 10% in weight value for one path could result in an evenly distributed increase in the weights for the remaining paths. Alternatively, weightings could be adjusted according to a weighted formula as desired (e.g., favoring healthy paths over less healthy paths). In yet another variation, the difference in weight value can be amortized over the remaining links in a manner that is proportional to their traffic weighting.

FIG. 22B shows steps that can be executed to shut down transmission links where a transmitter turns off. In step 2210, a transmitter shut-down event occurs. In step 2211, a test is made to determine whether at least one transmitter is still turned on. If not, then in step 2215 all packets are dropped until a transmitter turns on. If in step 2211 at least one transmitter is turned on, then in step 2212 the weight for the path is set to zero, and the weights for the remaining paths are adjusted accordingly.

FIG. 23 shows a computer node 2301 employing various principles of the above-described embodiments. It is assumed that two computer nodes of the type shown in FIG. 23 communicate over a plurality of separate physical transmission paths. As shown in FIG. 23, four transmission paths X1 through X4 are defined for communicating between the two nodes. Each node includes a packet transmitter 2302 that operates in accordance with a transmit table 2308 as described above. (The packet transmitter could also operate without using the IP-hopping features described above, but the following description assumes that some form of hopping is employed in conjunction with the path selection mechanism.). The computer node also includes a packet receiver 2303 that operates in accordance with a receive table 2309, including a moving window W that moves as valid packets are received. Invalid packets having source and destination addresses that do not fall within window W are rejected.

As each packet is readied for transmission, source and destination IP addresses (or other discriminator values) are selected from transmit table 2308 according to any of the various algorithms described above, and packets containing these source/destination address pairs, which correspond to the node to which the four transmission paths are linked, are generated to a transmission path switch 2307. Switch 2307, which can comprise a software function, selects from one of the available transmission paths according to a weight distribution table 2306. For example, if the weight for path X1 is 0.2, then every fifth packet will be transmitted on path X1. A similar regime holds true for the other paths as shown. Initially, each link's weight value can be set such that it is proportional to its bandwidth, which will be referred to as its "steady-state" value.

Packet receiver 2303 generates an output to a link quality measurement function 2304 that operates as described above to determine the quality of each transmission path. (The input to packet receiver 2303 for receiving incoming packets is omitted for clarity). Link quality measurement function 2304 compares the link quality to a threshold for each transmission link and, if necessary, generates an output to weight adjustment function 2305. If a weight adjustment is required, then the weights in table 2306 are adjusted accordingly, preferably according to a gradual (e.g., linearly or exponentially declining) function. In one embodiment, the weight values for all



## US 7,921,211 B2

37

available paths are initially set to the same value, and only when paths degrade in quality are the weights changed to reflect differences.

Link quality measurement function **2304** can be made to operate as part of a synchronizer function as described above. That is, if resynchronization occurs and the receiver detects that synchronization has been lost (e.g., resulting in the synchronization window *W* being advanced out of sequence), that fact can be used to drive link quality measurement function **2304**. According to one embodiment, load balancing is performed using information garnered during the normal synchronization, augmented slightly to communicate link health from the receiver to the transmitter. The receiver maintains a count, *MESS\_R(W)*, of the messages received in synchronization window *W*. When it receives a synchronization request (*SYNC\_REQ*) corresponding to the end of window *W*, the receiver includes counter *MESS\_R* in the resulting synchronization acknowledgement (*SYNC\_ACK*) sent back to the transmitter. This allows the transmitter to compare messages sent to messages received in order to assess the health of the link.

If synchronization is completely lost, weight adjustment function **2305** decreases the weight value on the affected path to zero. When synchronization is regained, the weight value for the affected path is gradually increased to its original value. Alternatively, link quality can be measured by evaluating the length of time required for the receiver to acknowledge a synchronization request. In one embodiment, separate transmit and receive tables are used for each transmission path.

When the transmitter receives a *SYNC\_ACK*, the *MESS\_R* is compared with the number of messages transmitted in a window (*MESS\_T*). When the transmitter receives a *SYNC\_ACK*, the traffic probabilities will be examined and adjusted if necessary. *MESS\_R* is compared with the number of messages transmitted in a window (*MESS\_T*). There are two possibilities:

1. If *MESS\_R* is less than a threshold value, *THRESH*, then the link will be deemed to be unhealthy. If the transmitter was turned off, the transmitter is turned on and the weight *P* for that link will be set to a minimum value *MIN*. This will keep a trickle of traffic on the link for monitoring purposes until it recovers. If the transmitter was turned on, the weight *P* for that link will be set to:

$$P' = \alpha \times \text{MIN} + (1 - \alpha) \times P \quad (1)$$

Equation 1 will exponentially damp the traffic weight value to *MIN* during sustained periods of degraded service.

2. If *MESS\_R* for a link is greater than or equal to *THRESH*, the link will be deemed healthy. If the weight *P* for that link is greater than or equal to the steady state value *S* for that link, then *P* is left unaltered. If the weight *P* for that link is less than *THRESH* then *P* will be set to:

$$P' = \beta \times S + (1 - \beta) \times P \quad (2)$$

where  $\beta$  is a parameter such that  $0 \leq \beta \leq 1$  that determines the damping rate of *P*.

Equation 2 will increase the traffic weight to *S* during sustained periods of acceptable service in a damped exponential fashion.

A detailed example will now be provided with reference to FIG. 24. As shown in FIG. 24, a first computer **2401** communicates with a second computer **2402** through two routers **2403** and **2404**. Each router is coupled to the other router through three transmission links. As described above, these may be physically diverse links or logical links (including virtual private networks).

38

Suppose that a first link *L1* can sustain a transmission bandwidth of 100 Mb/s and has a window size of 32; link *L2* can sustain 75 Mb/s and has a window size of 24; and link *L3* can sustain 25 Mb/s and has a window size of 8. The combined links can thus sustain 200 Mb/s. The steady state traffic weights are 0.5 for link *L1*; 0.375 for link *L2*, and 0.125 for link *L3*. *MIN*=1 Mb/s, *THRESH*=0.8 *MESS\_T* for each link,  $\alpha$ =0.75 and  $\beta$ =0.5. These traffic weights will remain stable until a link stops for synchronization or reports a number of packets received less than its *THRESH*. Consider the following sequence of events:

1. Link *L1* receives a *SYNC\_ACK* containing a *MESS\_R* of 24, indicating that only 75% of the *MESS\_T* (32) messages transmitted in the last window were successfully received. Link *L1* would be below *THRESH* (0.8). Consequently, link *L1*'s traffic weight value would be reduced to 0.12825, while link *L2*'s traffic weight value would be increased to 0.65812 and link *L3*'s traffic weight value would be increased to 0.217938.
2. Link *L2* and *L3* remained healthy and link *L1* stopped to synchronize. Then link *L1*'s traffic weight value would be set to 0, link *L2*'s traffic weight value would be set to 0.75, and link *L3*'s traffic weight value would be set to 0.25.
3. Link *L1* finally received a *SYNC\_ACK* containing a *MESS\_R* of 0 indicating that none of the *MESS\_T* (32) messages transmitted in the last window were successfully received. Link *L1* would be below *THRESH*. Link *L1*'s traffic weight value would be increased to 0.005, link *L2*'s traffic weight value would be decreased to 0.74625, and link *L3*'s traffic weight value would be decreased to 0.24875.
4. Link *L1* received a *SYNC\_ACK* containing a *MESS\_R* of 32 indicating that 100% of the *MESS\_T* (32) messages transmitted in the last window were successfully received. Link *L1* would be above *THRESH*. Link *L1*'s traffic weight value would be increased to 0.2525, while link *L2*'s traffic weight value would be decreased to 0.560625 and link *L3*'s traffic weight value would be decreased to 0.186875.
5. Link *L1* received a *SYNC\_ACK* containing a *MESS\_R* of 32 indicating that 100% of the *MESS\_T* (32) messages transmitted in the last window were successfully received. Link *L1* would be above *THRESH*. Link *L1*'s traffic weight value would be increased to 0.37625; link *L2*'s traffic weight value would be decreased to 0.4678125, and link *L3*'s traffic weight value would be decreased to 0.1559375.
6. Link *L1* remains healthy and the traffic probabilities approach their steady state traffic probabilities.

#### B. Use of a DNS Proxy to Transparently Create Virtual Private Networks

A second improvement concerns the automatic creation of a virtual private network (VPN) in response to a domain-name server look-up function.

Conventional Domain Name Servers (DNSs) provide a look-up function that returns the IP address of a requested computer or host. For example, when a computer user types in the web name "Yahoo.com," the user's web browser transmits a request to a DNS, which converts the name into a four-part IP address that is returned to the user's browser and then used by the browser to contact the destination web site.

This conventional scheme is shown in FIG. 25. A user's computer **2501** includes a client application **2504** (for example, a web browser) and an IP protocol stack **2505**.

US 7,921,211 B2

39

When the user enters the name of a destination host, a request DNS REQ is made (through IP protocol stack **2505**) to a DNS **2502** to look up the IP address associated with the name. The DNS returns the IP address DNS RESP to client application **2504**, which is then able to use the IP address to communicate with the host **2503** through separate transactions such as PAGE REQ and PAGE RESP.

In the conventional architecture shown in FIG. 25, nefarious listeners on the Internet could intercept the DNS REQ and DNS RESP packets and thus learn what IP addresses the user was contacting. For example, if a user wanted to set up a secure communication path with a web site having the name "Target.com," when the user's browser contacted a DNS to find the IP address for that web site, the true IP address of that web site would be revealed over the Internet as part of the DNS inquiry. This would hamper anonymous communications on the Internet.

One conventional scheme that provides secure virtual private networks over the Internet provides the DNS server with the public keys of the machines that the DNS server has the addresses for. This allows hosts to retrieve automatically the public keys of a host that the host is to communicate with so that the host can set up a VPN without having the user enter the public key of the destination host. One implementation of this standard is presently being developed as part of the FreeS/WAN project (RFC 2535).

The conventional scheme suffers from certain drawbacks. For example, any user can perform a DNS request. Moreover, DNS requests resolve to the same value for all users.

According to certain aspects of the invention, a specialized DNS server traps DNS requests and, if the request is from a special type of user (e.g., one for which secure communication services are defined), the server does not return the true IP address of the target node, but instead automatically sets up a virtual private network between the target node and the user. The VPN is preferably implemented using the IP address "hopping" features of the basic invention described above, such that the true identity of the two nodes cannot be determined even if packets during the communication are intercepted. For DNS requests that are determined to not require secure services (e.g., an unregistered user), the DNS server transparently "passes through" the request to provide a normal look-up function and return the IP address of the target web server, provided that the requesting host has permissions to resolve unsecured sites. Different users who make an identical DNS request could be provided with different results.

FIG. 26 shows a system employing various principles summarized above. A user's computer **2601** includes a conventional client (e.g., a web browser) **2605** and an IP protocol stack **2606** that preferably operates in accordance with an IP hopping function **2607** as outlined above. A modified DNS server **2602** includes a conventional DNS server function **2609** and a DNS proxy **2610**. A gatekeeper server **2603** is interposed between the modified DNS server and a secure target site **04**. An "unsecure" target site **2611** is also accessible via conventional IP protocols.

According to one embodiment, DNS proxy **2610** intercepts all DNS lookup functions from client **2605** and determines whether access to a secure site has been requested. If access to a secure site has been requested (as determined, for example, by a domain name extension, or by reference to an internal table of such sites), DNS proxy **2610** determines whether the user has sufficient security privileges to access the site. If so, DNS proxy **2610** transmits a message to gatekeeper **2603** requesting that a virtual private network be created between user computer **2601** and secure target site **2604**. In one embodiment, gatekeeper **2603** creates "hopblocks" to be used

40

by computer **2601** and secure target site **2604** for secure communication. Then, gatekeeper **2603** communicates these to user computer **2601**. Thereafter, DNS proxy **2610** returns to user computer **2601** the resolved address passed to it by the gatekeeper (this address could be different from the actual target computer) **2604**, preferably using a secure administrative VPN. The address that is returned need not be the actual address of the destination computer.

Had the user requested lookup of a non-secure web site such as site **2611**, DNS proxy would merely pass through to conventional DNS server **2609** the look-up request, which would be handled in a conventional manner, returning the IP address of non-secure web site **2611**. If the user had requested lookup of a secure web site but lacked credentials to create such a connection, DNS proxy **2610** would return a "host unknown" error to the user. In this manner, different users requesting access to the same DNS name could be provided with different look-up results.

Gatekeeper **2603** can be implemented on a separate computer (as shown in FIG. 26) or as a function within modified DNS server **2602**. In general, it is anticipated that gatekeeper **03** facilitates the allocation and exchange of information needed to communicate securely, such as using "hopped" IP addresses. Secure hosts such as site **2604** are assumed to be equipped with a secure communication function such as an IP hopping function **2608**.

It will be appreciated that the functions of DNS proxy **2610** and DNS server **2609** can be combined into a single server for convenience. Moreover, although element **2602** is shown as combining the functions of two servers, the two servers can be made to operate independently.

FIG. 27 shows steps that can be executed by DNS proxy server **2610** to handle requests for DNS look-up for secure hosts. In step **01**, a DNS look-up request is received for a target host. In step **02**, a check is made to determine whether access to a secure host was requested. If not, then in step **03** the DNS request is passed to conventional DNS server **2609**, which looks up the IP address of the target site and returns it to the user's application for further processing.

In step **02**, if access to a secure host was requested, then in step **04** a further check is made to determine whether the user is authorized to connect to the secure host. Such a check can be made with reference to an internally stored list of authorized IP addresses, or can be made by communicating with gatekeeper **2603** (e.g., over an "administrative" VPN that is secure). It will be appreciated that different levels of security can also be provided for different categories of hosts. For example, some sites may be designated as having a certain security level, and the security level of the user requesting access must match that security level. The user's security level can also be determined by transmitting a request message back to the user's computer requiring that it prove that it has sufficient privileges.

If the user is not authorized to access the secure site, then a "host unknown" message is returned (step **05**). If the user has sufficient security privileges, then in step **06** a secure VPN is established between the user's computer and the secure target site. As described above, this is preferably done by allocating a hopping regime that will be carried out between the user's computer and the secure target site, and is preferably performed transparently to the user (i.e., the user need not be involved in creating the secure link). As described in various embodiments of this application, any of various fields can be "hopped" (e.g., IP source/destination addresses; a field in the header; etc.) in order to communicate securely.

Some or all of the security functions can be embedded in gatekeeper **2603**, such that it handles all requests to connect to

## US 7,921,211 B2

41

secure sites. In this embodiment, DNS proxy **2610** communicates with gatekeeper **2603** to determine (preferably over a secure administrative VPN) whether the user has access to a particular web site. Various scenarios for implementing these features are described by way of example below:

Scenario #1: Client has permission to access target computer, and gatekeeper has a rule to make a VPN for the client. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would establish a VPN between the client and the requested target. The gatekeeper would provide the address of the destination to the DNS proxy, which would then return the resolved name as a result. The resolved address can be transmitted back to the client in a secure administrative VPN.

Scenario #2: Client does not have permission to access target computer. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would reject the request, informing DNS proxy server **2610** that it was unable to find the target computer. The DNS proxy **2610** would then return a "host unknown" error message to the client.

Scenario #3: Client has permission to connect using a normal non-VPN link, and the gatekeeper does not have a rule to set up a VPN for the client to the target site. In this scenario, the client's DNS request is received by DNS proxy server **2610**, which would check its rules and determine that no VPN is needed. Gatekeeper **2603** would then inform the DNS proxy server to forward the request to conventional DNS server **2609**, which would resolve the request and return the result to the DNS proxy server and then back to the client.

Scenario #4: Client does not have permission to establish a normal/non-VPN link, and the gatekeeper does not have a rule to make a VPN for the client to the target site. In this scenario, the DNS proxy server would receive the client's DNS request and forward it to gatekeeper **2603**. Gatekeeper **2603** would determine that no special VPN was needed, but that the client is not authorized to communicate with non-VPN members. The gatekeeper would reject the request, causing DNS proxy server **2610** to return an error message to the client.

### C. Large Link to Small Link Bandwidth Management

One feature of the basic architecture is the ability to prevent so-called "denial of service" attacks that can occur if a computer hacker floods a known Internet node with packets, thus preventing the node from communicating with other nodes. Because IP addresses or other fields are "hopped" and packets arriving with invalid addresses are quickly discarded, Internet nodes are protected against flooding targeted at a single IP address.

In a system in which a computer is coupled through a link having a limited bandwidth (e.g., an edge router) to a node that can support a much higher-bandwidth link (e.g., an Internet Service Provider), a potential weakness could be exploited by a determined hacker. Referring to FIG. **28**, suppose that a first host computer **2801** is communicating with a second host computer **2804** using the IP address hopping principles described above. The first host computer is coupled through an edge router **2802** to an Internet Service Provider (ISP) **2803** through a low bandwidth link (LOW BW), and is in turn coupled to second host computer **2804** through parts of the Internet through a high bandwidth link (HIGH BW). In

42

this architecture, the ISP is able to support a high bandwidth to the internet, but a much lower bandwidth to the edge router **2802**.

Suppose that a computer hacker is able to transmit a large quantity of dummy packets addressed to first host computer **2801** across high bandwidth link HIGH BW. Normally, host computer **2801** would be able to quickly reject the packets since they would not fall within the acceptance window permitted by the IP address hopping scheme. However, because the packets must travel across low bandwidth link LOW BW, the packets overwhelm the lower bandwidth link before they are received by host computer **2801**. Consequently, the link to host computer **2801** is effectively flooded before the packets can be discarded.

According to one inventive improvement, a "link guard" function **2805** is inserted into the high-bandwidth node (e.g., ISP **2803**) that quickly discards packets destined for a low-bandwidth target node if they are not valid packets. Each packet destined for a low-bandwidth node is cryptographically authenticated to determine whether it belongs to a VPN. If it is not a valid VPN packet, the packet is discarded at the high-bandwidth node. If the packet is authenticated as belonging to a VPN, the packet is passed with high preference. If the packet is a valid non-VPN packet, it is passed with a lower quality of service (e.g., lower priority).

In one embodiment, the ISP distinguishes between VPN and non-VPN packets using the protocol of the packet. In the case of IPSEC [rfc **2401**], the packets have IP protocols **420** and **421**. In the case of the TARP VPN, the packets will have an IP protocol that is not yet defined. The ISP's link guard, **2805**, maintains a table of valid VPNs which it uses to validate whether VPN packets are cryptographically valid. According to one embodiment, packets that do not fall within any hop windows used by nodes on the low-bandwidth link are rejected, or are sent with a lower quality of service. One approach for doing this is to provide a copy of the IP hopping tables used by the low-bandwidth nodes to the high-bandwidth node, such that both the high-bandwidth and low-bandwidth nodes track hopped packets (e.g., the high-bandwidth node moves its hopping window as valid packets are received). In such a scenario, the high-bandwidth node discards packets that do not fall within the hopping window before they are transmitted over the low-bandwidth link. Thus, for example, ISP **2903** maintains a copy **2910** of the receive table used by host computer **2901**. Incoming packets that do not fall within this receive table are discarded. According to a different embodiment, link guard **2805** validates each VPN packet using a keyed hashed message authentication code (HMAC) [rfc **2104**].

According to another embodiment, separate VPNs (using, for example, hopblocks) can be established for communicating between the low-bandwidth node and the high-bandwidth node (i.e., packets arriving at the high-bandwidth node are converted into different packets before being transmitted to the low-bandwidth node).

As shown in FIG. **29**, for example, suppose that a first host computer **2900** is communicating with a second host computer **2902** over the Internet, and the path includes a high bandwidth link HIGH BW to an ISP **2901** and a low bandwidth link LOW BW through an edge router **2904**. In accordance with the basic architecture described above, first host computer **2900** and second host computer **2902** would exchange hopblocks (or a hopblock algorithm) and would be able to create matching transmit and receive tables **2905**, **2906**, **2912** and **2913**. Then in accordance with the basic architecture, the two computers would transmit packets having seemingly random IP source and destination addresses,

## US 7,921,211 B2

43

and each would move a corresponding hopping window in its receive table as valid packets were received.

Suppose that a nefarious computer hacker **2903** was able to deduce that packets having a certain range of IP addresses (e.g., addresses **100** to **200** for the sake of simplicity) are being transmitted to ISP **2901**, and that these packets are being forwarded over a low-bandwidth link. Hacker computer **2903** could thus “flood” packets having addresses falling into the range **100** to **200**, expecting that they would be forwarded along low bandwidth link LOW BW, thus causing the low bandwidth link to become overwhelmed. The fast packet reject mechanism in first host computer **3000** would be of little use in rejecting these packets, since the low bandwidth link was effectively jammed before the packets could be rejected. In accordance with one aspect of the improvement, however, VPN link guard **2911** would prevent the attack from impacting the performance of VPN traffic because the packets would either be rejected as invalid VPN packets or given a lower quality of service than VPN traffic over the lower bandwidth link. A denial-of-service flood attack could, however, still disrupt non-VPN traffic.

According to one embodiment of the improvement, ISP **2901** maintains a separate VPN with first host computer **2900**, and thus translates packets arriving at the ISP into packets having a different IP header before they are transmitted to host computer **2900**. The cryptographic keys used to authenticate VPN packets at the link guard **2911** and the cryptographic keys used to encrypt and decrypt the VPN packets at host **2902** and host **2901** can be different, so that link guard **2911** does not have access to the private host data; it only has the capability to authenticate those packets.

According to yet a third embodiment, the low-bandwidth node can transmit a special message to the high-bandwidth node instructing it to shut down all transmissions on a particular IP address, such that only hopped packets will pass through to the low-bandwidth node. This embodiment would prevent a hacker from flooding packets using a single IP address. According to yet a fourth embodiment, the high-bandwidth node can be configured to discard packets transmitted to the low-bandwidth node if the transmission rate exceeds a certain predetermined threshold for any given IP address; this would allow hopped packets to go through. In this respect, link guard **2911** can be used to detect that the rate of packets on a given IP address are exceeding a threshold rate; further packets addressed to that same IP address would be dropped or transmitted at a lower priority (e.g., delayed).

## D. Traffic Limiter

In a system in which multiple nodes are communicating using “hopping” technology, a treasonous insider could internally flood the system with packets. In order to prevent this possibility, one inventive improvement involves setting up “contracts” between nodes in the system, such that a receiver can impose a bandwidth limitation on each packet sender. One technique for doing this is to delay acceptance of a checkpoint synchronization request from a sender until a certain time period (e.g., one minute) has elapsed. Each receiver can effectively control the rate at which its hopping window moves by delaying “SYNC\_ACK” responses to “SYNC\_REQ” messages.

A simple modification to the checkpoint synchronizer will serve to protect a receiver from accidental or deliberate overload from an internally treasonous client. This modification is based on the observation that a receiver will not update its tables until a SYNC\_REQ is received on hopped address

44

CKPT\_N. It is a simple matter of deferring the generation of a new CKPT\_N until an appropriate interval after previous checkpoints.

Suppose a receiver wished to restrict reception from a transmitter to 100 packets a second, and that checkpoint synchronization messages were triggered every 50 packets. A compliant transmitter would not issue new SYNC\_REQ messages more often than every 0.5 seconds. The receiver could delay a non-compliant transmitter from synchronizing by delaying the issuance of CKPT\_N for 0.5 second after the last SYNC\_REQ was accepted.

In general, if M receivers need to restrict N transmitters issuing new SYNC\_REQ messages after every W messages to sending R messages a second in aggregate, each receiver could defer issuing a new CKPT\_N until  $M \times N \times W / R$  seconds have elapsed since the last SYNC\_REQ has been received and accepted. If the transmitter exceeds this rate between a pair of checkpoints, it will issue the new checkpoint before the receiver is ready to receive it, and the SYNC\_REQ will be discarded by the receiver. After this, the transmitter will re-issue the SYNC\_REQ every T1 seconds until it receives a SYNC\_ACK. The receiver will eventually update CKPT\_N and the SYNC\_REQ will be acknowledged. If the transmission rate greatly exceeds the allowed rate, the transmitter will stop until it is compliant. If the transmitter exceeds the allowed rate by a little, it will eventually stop after several rounds of delayed synchronization until it is in compliance. Hacking the transmitter’s code to not shut off only permits the transmitter to lose the acceptance window. In this case it can recover the window and proceed only after it is compliant again.

Two practical issues should be considered when implementing the above scheme:

1. The receiver rate should be slightly higher than the permitted rate in order to allow for statistical fluctuations in traffic arrival times and non-uniform load balancing.
2. Since a transmitter will rightfully continue to transmit for a period after a SYNC\_REQ is transmitted, the algorithm above can artificially reduce the transmitter’s bandwidth. If events prevent a compliant transmitter from synchronizing for a period (e.g. the network dropping a SYNC\_REQ or a SYNC\_ACK) a SYNC\_REQ will be accepted later than expected. After this, the transmitter will transmit fewer than expected messages before encountering the next checkpoint. The new checkpoint will not have been activated and the transmitter will have to retransmit the SYNC\_REQ. This will appear to the receiver as if the transmitter is not compliant. Therefore, the next checkpoint will be accepted late from the transmitter’s perspective. This has the effect of reducing the transmitter’s allowed packet rate until the transmitter transmits at a packet rate below the agreed upon rate for a period of time.

To guard against this, the receiver should keep track of the times that the last C SYNC\_REQs were received and accepted and use the minimum of  $M \times N \times W / R$  seconds after the last SYNC\_REQ has been received and accepted,  $2 \times M \times N \times W / R$  seconds after next to the last SYNC\_REQ has been received and accepted,  $C \times M \times N \times W / R$  seconds after  $(C-1)^{th}$  to the last SYNC\_REQ has been received, as the time to activate CKPT\_N. This prevents the receiver from inappropriately limiting the transmitter’s packet rate if at least one out of the last C SYNC\_REQs was processed on the first attempt.

FIG. 30 shows a system employing the above-described principles. In FIG. 30, two computers **3000** and **3001** are assumed to be communicating over a network N in accordance with the “hopping” principles described above (e.g.,

hopped IP addresses, discriminator values, etc.). For the sake of simplicity, computer **3000** will be referred to as the receiving computer and computer **3001** will be referred to as the transmitting computer, although full duplex operation is of course contemplated. Moreover, although only a single transmitter is shown, multiple transmitters can transmit to receiver **3000**.

As described above, receiving computer **3000** maintains a receive table **3002** including a window *W* that defines valid IP address pairs that will be accepted when appearing in incoming data packets. Transmitting computer **3001** maintains a transmit table **3003** from which the next IP address pairs will be selected when transmitting a packet to receiving computer **3000**. (For the sake of illustration, window *W* is also illustrated with reference to transmit table **3003**). As transmitting computer moves through its table, it will eventually generate a SYNC\_REQ message as illustrated in function **3010**. This is a request to receiver **3000** to synchronize the receive table **3002**, from which transmitter **3001** expects a response in the form of a CKPT\_N (included as part of a SYNC\_ACK message). If transmitting computer **3001** transmits more messages than its allotment, it will prematurely generate the SYNC\_REQ message. (If it has been altered to remove the SYNC\_REQ message generation altogether, it will fall out of synchronization since receiver **3000** will quickly reject packets that fall outside of window *W*, and the extra packets generated by transmitter **3001** will be discarded).

In accordance with the improvements described above, receiving computer **3000** performs certain steps when a SYNC\_REQ message is received, as illustrated in FIG. **30**. In step **3004**, receiving computer **3000** receives the SYNC\_REQ message. In step **3005**, a check is made to determine whether the request is a duplicate. If so, it is discarded in step **3006**. In step **3007**, a check is made to determine whether the SYNC\_REQ received from transmitter **3001** was received at a rate that exceeds the allowable rate *R* (i.e., the period between the time of the last SYNC\_REQ message). The value *R* can be a constant, or it can be made to fluctuate as desired. If the rate exceeds *R*, then in step **3008** the next activation of the next CKPT\_N hopping table entry is delayed by *W/R* seconds after the last SYNC\_REQ has been accepted.

Otherwise, if the rate has not been exceeded, then in step **3109** the next CKPT\_N value is calculated and inserted into the receiver's hopping table prior to the next SYNC\_REQ from the transmitter **3101**. Transmitter **3101** then processes the SYNC\_REQ in the normal manner.

#### E. Signaling Synchronizer

In a system in which a large number of users communicate with a central node using secure hopping technology, a large amount of memory must be set aside for hopping tables and their supporting data structures. For example, if one million subscribers to a web site occasionally communicate with the web site, the site must maintain one million hopping tables, thus using up valuable computer resources, even though only a small percentage of the users may actually be using the system at any one time. A desirable solution would be a system that permits a certain maximum number of simultaneous links to be maintained, but which would "recognize" millions of registered users at any one time. In other words, out of a population of a million registered users, a few thousand at a time could simultaneously communicate with a central server, without requiring that the server maintain one million hopping tables of appreciable size.

One solution is to partition the central node into two nodes: a signaling server that performs session initiation for user

log-on and log-off (and requires only minimally sized tables), and a transport server that contains larger hopping tables for the users. The signaling server listens for the millions of known users and performs a fast-packet reject of other (bogus) packets. When a packet is received from a known user, the signaling server activates a virtual private link (VPL) between the user and the transport server, where hopping tables are allocated and maintained. When the user logs onto the signaling server, the user's computer is provided with hopping tables for communicating with the transport server, thus activating the VPL. The VPLs can be torn down when they become inactive for a time period, or they can be torn down upon user log-out. Communication with the signaling server to allow user log-on and log-off can be accomplished using a specialized version of the checkpoint scheme described above.

FIG. **31** shows a system employing certain of the above-described principles. In FIG. **31**, a signaling server **3101** and a transport server **3102** communicate over a link. Signaling server **3101** contains a large number of small tables **3106** and **3107** that contain enough information to authenticate a communication request with one or more clients **3103** and **3104**. As described in more detail below, these small tables may advantageously be constructed as a special case of the synchronizing checkpoint tables described previously. Transport server **3102**, which is preferably a separate computer in communication with signaling server **3101**, contains a smaller number of larger hopping tables **3108**, **3109**, and **3110** that can be allocated to create a VPN with one of the client computers.

According to one embodiment, a client that has previously registered with the system (e.g., via a system administration function, a user registration procedure, or some other method) transmits a request for information from a computer (e.g., a web site). In one variation, the request is made using a "hopped" packet, such that signaling server **3101** will quickly reject invalid packets from unauthorized computers such as hacker computer **3105**. An "administrative" VPN can be established between all of the clients and the signaling server in order to ensure that a hacker cannot flood signaling server **3101** with bogus packets. Details of this scheme are provided below.

Signaling server **3101** receives the request **3111** and uses it to determine that client **3103** is a validly registered user. Next, signaling server **3101** issues a request to transport server **3102** to allocate a hopping table (or hopping algorithm or other regime) for the purpose of creating a VPN with client **3103**. The allocated hopping parameters are returned to signaling server **3101** (path **3113**), which then supplies the hopping parameters to client **3103** via path **3114**, preferably in encrypted form.

Thereafter, client **3103** communicates with transport server **3102** using the normal hopping techniques described above. It will be appreciated that although signaling server **3101** and transport server **3102** are illustrated as being two separate computers, they could of course be combined into a single computer and their functions performed on the single computer. Alternatively, it is possible to partition the functions shown in FIG. **31** differently from as shown without departing from the inventive principles.

One advantage of the above-described architecture is that signaling server **3101** need only maintain a small amount of information on a large number of potential users, yet it retains the capability of quickly rejecting packets from unauthorized users such as hacker computer **3105**. Larger data tables needed to perform the hopping and synchronization functions are instead maintained in a transport server **3102**, and a

## US 7,921,211 B2

47

smaller number of these tables are needed since they are only allocated for "active" links. After a VPN has become inactive for a certain time period (e.g., one hour), the VPN can be automatically torn down by transport server **3102** or signaling server **3101**.

A more detailed description will now be provided regarding how a special case of the checkpoint synchronization feature can be used to implement the signaling scheme described above.

The signaling synchronizer may be required to support many (millions) of standing, low bandwidth connections. It therefore should minimize per-VPL memory usage while providing the security offered by hopping technology. In order to reduce memory usage in the signaling server, the data hopping tables can be completely eliminated and data can be carried as part of the SYNC\_REQ message. The table used by the server side (receiver) and client side (transmitter) is shown schematically as element **3106** in FIG. **31**.

The meaning and behaviors of CKPT\_N, CKPT\_O and CKPT\_R remain the same from the previous description, except that CKPT\_N can receive a combined data and SYNC\_REQ message or a SYNC\_REQ message without the data.

The protocol is a straightforward extension of the earlier synchronizer. Assume that a client transmitter is on and the tables are synchronized. The initial tables can be generated "out of band." For example, a client can log into a web server to establish an account over the Internet. The client will receive keys etc encrypted over the Internet. Meanwhile, the server will set up the signaling VPN on the signaling server.

Assuming that a client application wishes to send a packet to the server on the client's standing signaling VPL:

1. The client sends the message marked as a data message on the inner header using the transmitter's CKPT\_N address. It turns the transmitter off and starts a timer T1 noting CKPT\_O. Messages can be one of three types: DATA, SYNC\_REQ and SYNC\_ACK. In the normal algorithm, some potential problems can be prevented by identifying each message type as part of the encrypted inner header field. In this algorithm, it is important to distinguish a data packet and a SYNC\_REQ in the signaling synchronizer since the data and the SYNC\_REQ come in on the same address.
2. When the server receives a data message on its CKPT\_N, it verifies the message and passes it up the stack. The message can be verified by checking message type and other information (i.e., user credentials) contained in the inner header. It replaces its CKPT\_O with CKPT\_N and generates the next CKPT\_N. It updates its transmitter side CKPT\_R to correspond to the client's receiver side CKPT\_O and transmits a SYNC\_ACK containing CKPT\_O in its payload.
3. When the client side receiver receives a SYNC\_ACK on its CKPT\_R with a payload matching its transmitter side CKPT\_O and the transmitter is off, the transmitter is turned on and the receiver side CKPT\_R is updated. If the SYNC\_ACK's payload does not match the transmitter side CKPT\_O or the transmitter is on, the SYNC\_ACK is simply discarded.
4. T1 expires: If the transmitter is off and the client's transmitter side CKPT\_O matches the CKPT\_O associated with the timer, it starts timer T1 noting CKPT\_O again, and a SYNC\_REQ is sent using the transmitter's CKPT\_O address. Otherwise, no action is taken.
5. When the server receives a SYNC\_REQ on its CKPT\_N, it replaces its CKPT\_O with CKPT\_N and generates the next CKPT\_N. It updates its transmitter side CKPT\_R to

48

correspond to the client's receiver side CKPT\_R and transmits a SYNC\_ACK containing CKPT\_O in its payload.

6. When the server receives a SYNC\_REQ on its CKPT\_O, it updates its transmitter side CKPT\_R to correspond to the client's receiver side CKPT\_R and transmits a SYNC\_ACK containing CKPT\_O in its payload.

FIG. **32** shows message flows to highlight the protocol. Reading from top to bottom, the client sends data to the server using its transmitter side CKPT\_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT\_N into CKPT\_O and updates CKPT\_N. This message is successfully received and a passed up the stack. It also synchronizes the receiver i.e., the server loads CKPT\_N into CKPT\_O and generates a new CKPT\_N, it generates a new CKPT\_R in the server side transmitter and transmits a SYNC\_ACK containing the server side receiver's CKPT\_O the server. The SYNC\_ACK is successfully received at the client. The client side receiver's CKPT\_R is updated, the transmitter is turned on and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

Next, the client sends data to the server using its transmitter side CKPT\_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT\_N into CKPT\_O and updates CKPT\_N. This message is lost. The client side timer expires and as a result a SYNC\_REQ is transmitted on the client side transmitter's CKPT\_O (this will keep happening until the SYNC\_ACK has been received at the client). The SYNC\_REQ is successfully received at the server. It synchronizes the receiver i.e., the server loads CKPT\_N into CKPT\_O and generates a new CKPT\_N, it generates a new CKPT\_R in the server side transmitter and transmits a SYNC\_ACK containing the server side receiver's CKPT\_O the server. The SYNC\_ACK is successfully received at the client. The client side receiver's CKPT\_R is updated, the transmitter is turned off and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

There are numerous other scenarios that follow this flow. For example, the SYNC\_ACK could be lost. The transmitter would continue to re-send the SYNC\_REQ until the receiver synchronizes and responds.

The above-described procedures allow a client to be authenticated at signaling server **3201** while maintaining the ability of signaling server **3201** to quickly reject invalid packets, such as might be generated by hacker computer **3205**. In various embodiments, the signaling synchronizer is really a derivative of the synchronizer. It provides the same protection as the hopping protocol, and it does so for a large number of low bandwidth connections.

#### F. One-Click Secure On-line Communications and Secure Domain Name Service

The present invention provides a technique for establishing a secure communication link between a first computer and a second computer over a computer network. Preferably, a user enables a secure communication link using a single click of a mouse, or a corresponding minimal input from another input device, such as a keystroke entered on a keyboard or a click entered through a trackball. Alternatively, the secure link is automatically established as a default setting at boot-up of the computer (i.e., no click). FIG. **33** shows a system block diagram **3300** of a computer network in which the one-click

## US 7,921,211 B2

49

secure communication method of the present invention is suitable. In FIG. 33, a computer terminal or client computer 3301, such as a personal computer (PC), is connected to a computer network 3302, such as the Internet, through an ISP 3303. Alternatively, computer 3301 can be connected to computer network 3302 through an edge router. Computer 3301 includes an input device, such as a keyboard and/or mouse, and a display device, such as a monitor. Computer 3301 can communicate conventionally with another computer 3304 connected to computer network 3302 over a communication link 3305 using a browser 3306 that is installed and operates on computer 3301 in a well-known manner.

Computer 3304 can be, for example, a server computer that is used for conducting e-commerce. In the situation when computer network 3302 is the Internet, computer 3304 typically will have a standard top-level domain name such as .com, .net, .org, .edu, .mil or .gov.

FIG. 34 shows a flow diagram 3400 for installing and establishing a "one-click" secure communication link over a computer network according to the present invention. At step 3401, computer 3301 is connected to server computer 3304 over a non-VPN communication link 3305. Web browser 3306 displays a web page associated with server 3304 in a well-known manner. According to one variation of the invention, the display of computer 3301 contains a hyperlink, or an icon representing a hyperlink, for selecting a virtual private network (VPN) communication link ("go secure" hyperlink) through computer network 3302 between terminal 3301 and server 3304. Preferably, the "go secure" hyperlink is displayed as part of the web page downloaded from server computer 3304, thereby indicating that the entity providing server 3304 also provides VPN capability.

By displaying the "go secure" hyperlink, a user at computer 3301 is informed that the current communication link between computer 3301 and server computer 3304 is a non-secure, non-VPN communication link. At step 3402, it is determined whether a user of computer 3301 has selected the "go secure" hyperlink. If not, processing resumes using a non-secure (conventional) communication method (not shown). If, at step 3402, it is determined that the user has selected the "go secure" hyperlink, flow continues to step 3403 where an object associated with the hyperlink determines whether a VPN communication software module has already been installed on computer 3301. Alternatively, a user can enter a command into computer 3301 to "go secure."

If, at step 3403, the object determines that the software module has been installed, flow continues to step 3407. If, at step 3403, the object determines that the software module has not been installed, flow continues to step 3404 where a non-VPN communication link 3307 is launched between computer 3301 and a website 3308 over computer network 3302 in a well-known manner. Website 3308 is accessible by all computer terminals connected to computer network 3302 through a non-VPN communication link. Once connected to website 3308, a software module for establishing a secure communication link over computer network 3302 can be downloaded and installed. Flow continues to step 3405 where, after computer 3301 connects to website 3308, the software module for establishing a communication link is downloaded and installed in a well-known manner on computer terminal 3301 as software module 3309. At step 3405, a user can optionally select parameters for the software module, such as enabling a secure communication link mode of communication for all communication links over computer network 3302. At step 3406, the communication link between computer 3301 and website 3308 is then terminated in a well-known manner.

50

By clicking on the "go secure" hyperlink, a user at computer 3301 has enabled a secure communication mode of communication between computer 3301 and server computer 3304. According to one variation of the invention, the user is not required to do anything more than merely click the "go secure" hyperlink. The user does not need to enter any user identification information, passwords or encryption keys for establishing a secure communication link. All procedures required for establishing a secure communication link between computer 3301 and server computer 3304 are performed transparently to a user at computer 3301.

At step 3407, a secure VPN communications mode of operation has been enabled and software module 3309 begins to establish a VPN communication link. In one embodiment, software module 3309 automatically replaces the top-level domain name for server 3304 within browser 3406 with a secure top-level domain name for server computer 3304. For example, if the top-level domain name for server 3304 is .com, software module 3309 replaces the .com top-level domain name with a .scom top-level domain name, where the "s" stands for secure. Alternatively, software module 3409 can replace the top-level domain name of server 3304 with any other non-standard top-level domain name.

Because the secure top-level domain name is a non-standard domain name, a query to a standard domain name service (DNS) will return a message indicating that the universal resource locator (URL) is unknown. According to the invention, software module 3409 contains the URL for querying a secure domain name service (SDNS) for obtaining the URL for a secure top-level domain name. In this regard, software module 3309 accesses a secure portal 3310 that interfaces a secure network 3311 to computer network 3302. Secure network 3311 includes an internal router 3312, a secure domain name service (SDNS) 3313, a VPN gatekeeper 3314 and a secure proxy 3315. The secure network can include other network services, such as e-mail 3316, a plurality of chatrooms (of which only one chatroom 3317 is shown), and a standard domain name service (STD DNS) 3318. Of course, secure network 3311 can include other resources and services that are not shown in FIG. 33.

When software module 3309 replaces the standard top-level domain name for server 3304 with the secure top-level domain name, software module 3309 sends a query to SDNS 3313 at step 3408 through secure portal 3310 preferably using an administrative VPN communication link 3319. In this configuration, secure portal 3310 can only be accessed using a VPN communication link. Preferably, such a VPN communication link can be based on a technique of inserting a source and destination IP address pair into each data packet that is selected according to a pseudo-random sequence; an IP address hopping regime that pseudorandomly changes IP addresses in packets transmitted between a client computer and a secure target computer; periodically changing at least one field in a series of data packets according to a known sequence; an Internet Protocol (IP) address in a header of each data packet that is compared to a table of valid IP addresses maintained in a table in the second computer; and/or a comparison of the IP address in the header of each data packet to a moving window of valid IP addresses, and rejecting data packets having IP addresses that do not fall within the moving window. Other types of VPNs can alternatively be used. Secure portal 3310 authenticates the query from software module 3309 based on the particular information hopping technique used for VPN communication link 3319.

SDNS 3313 contains a cross-reference database of secure domain names and corresponding secure network addresses. That is, for each secure domain name, SDNS 3313 stores a



## US 7,921,211 B2

51

computer network address corresponding to the secure domain name. An entity can register a secure domain name in SDNS 3313 so that a user who desires a secure communication link to the website of the entity can automatically obtain the secure computer network address for the secure website. Moreover, an entity can register several secure domain names, with each respective secure domain name representing a different priority level of access in a hierarchy of access levels to a secure website. For example, a securities trading website can provide users secure access so that a denial of service attack on the website will be ineffectual with respect to users subscribing to the secure website service. Different levels of subscription can be arranged based on, for example, an escalating fee, so that a user can select a desired level of guarantee for connecting to the secure securities trading website. When a user queries SDNS 3313 for the secure computer network address for the securities trading website, SDNS 3313 determines the particular secure computer network address based on the user's identity and the user's subscription level.

At step 3409, SDNS 3313 accesses VPN gatekeeper 3314 for establishing a VPN communication link between software module 3309 and secure server 3320. Server 3320 can only be accessed through a VPN communication link. VPN gatekeeper 3314 provisions computer 3301 and secure web server computer 3320, or a secure edge router for server computer 3320, thereby creating the VPN. Secure server computer 3320 can be a separate server computer from server computer 3304, or can be the same server computer having both non-VPN and VPN communication link capability, such as shown by server computer 3322. Returning to FIG. 34, in step 3410, SDNS 3313 returns a secure URL to software module 3309 for the .com server address for a secure server 3320 corresponding to server 3304.

Alternatively, SDNS 3313 can be accessed through secure portal 3310 "in the clear", that is, without using an administrative VPN communication link. In this situation, secure portal 3310 preferably authenticates the query using any well-known technique, such as a cryptographic technique, before allowing the query to proceed to SDNS 3313. Because the initial communication link in this situation is not a VPN communication link, the reply to the query can be "in the clear." The querying computer can use the clear reply for establishing a VPN link to the desired domain name. Alternatively, the query to SDNS 3313 can be in the clear, and SDNS 3313 and gatekeeper 3314 can operate to establish a VPN communication link to the querying computer for sending the reply.

At step 3411, software module 3309 accesses secure server 3320 through VPN communication link 3321 based on the VPN resources allocated by VPN gatekeeper 3314. At step 3412, web browser 3306 displays a secure icon indicating that the current communication link to server 3320 is a secure VPN communication link. Further communication between computers 3301 and 3320 occurs via the VPN, e.g., using a "hopping" regime as discussed above. When VPN link 3321 is terminated at step 3413, flow continues to step 3414 where software module 3309 automatically replaces the secure top-level domain name with the corresponding non-secure top-level domain name for server 3304. Browser 3306 accesses a standard DNS 3325 for obtaining the non-secure URL for server 3304. Browser 3306 then connects to server 3304 in a well-known manner. At step 3415, browser 3306 displays the "go secure" hyperlink or icon for selecting a VPN communication link between terminal 3301 and server 3304. By again

52

displaying the "go secure" hyperlink, a user is informed that the current communication link is a non-secure, non-VPN communication link.

When software module 3309 is being installed or when the user is off-line, the user can optionally specify that all communication links established over computer network 3302 are secure communication links. Thus, anytime that a communication link is established, the link is a VPN link. Consequently, software module 3309 transparently accesses SDNS 3313 for obtaining the URL for a selected secure website. In other words, in one embodiment, the user need not "click" on the secure option each time secure communication is to be effected.

Additionally, a user at computer 3301 can optionally select a secure communication link through proxy computer 3315. Accordingly, computer 3301 can establish a VPN communication link 3323 with secure server computer 3320 through proxy computer 3315. Alternatively, computer 3301 can establish a non-VPN communication link 3324 to a non-secure website, such as non-secure server computer 3304.

FIG. 35 shows a flow diagram 3500 for registering a secure domain name according to the present invention. At step 3501, a requester accesses website 3308 and logs into a secure domain name registry service that is available through website 3308. At step 3502, the requestor completes an online registration form for registering a secure domain name having a top-level domain name, such as .com, .net, .org, .edu, .mil or .gov. Of course, other secure top-level domain names can also be used. Preferably, the requestor must have previously registered a non-secure domain name corresponding to the equivalent secure domain name that is being requested. For example, a requestor attempting to register secure domain name "website.scom" must have previously registered the corresponding non-secure domain name "website.com".

At step 3503, the secure domain name registry service at website 3308 queries a non-secure domain name server database, such as standard DNS 3322, using, for example, a whois query, for determining ownership information relating to the non-secure domain name corresponding to the requested secure domain name. At step 3504, the secure domain name registry service at website 3308 receives a reply from standard DNS 3322 and at step 3505 determines whether there is conflicting ownership information for the corresponding non-secure domain name. If there is no conflicting ownership information, flow continues to step 3507, otherwise flow continues to step 3506 where the requestor is informed of the conflicting ownership information. Flow returns to step 3502.

When there is no conflicting ownership information at step 3505, the secure domain name registry service (website 3308) informs the requestor that there is no conflicting ownership information and prompts the requestor to verify the information entered into the online form and select an approved form of payment. After confirmation of the entered information and appropriate payment information, flow continues to step 3508 where the newly registered secure domain name sent to SDNS 3313 over communication link 3326.

If, at step 3505, the requested secure domain name does not have a corresponding equivalent non-secure domain name, the present invention informs the requestor of the situation and prompts the requestor for acquiring the corresponding equivalent non-secure domain name for an increased fee. By accepting the offer, the present invention automatically registers the corresponding equivalent non-secure domain name with standard DNS 3325 in a well-known manner. Flow then continues to step 3508.



US 7,921,211 B2

53

G. Tunneling Secure Address Hopping Protocol  
Through Existing Protocol Using Web Proxy

The present invention also provides a technique for implementing the field hopping schemes described above in an application program on the client side of a firewall between two computer networks, and in the network stack on the server side of the firewall. The present invention uses a new secure connectionless protocol that provides good denial of service rejection capabilities by layering the new protocol on top of an existing IP protocol, such as the ICMP, UDP or TCP protocols. Thus, this aspect of the present invention does not require changes in the Internet infrastructure.

According to the invention, communications are protected by a client-side proxy application program that accepts unencrypted, unprotected communication packets from a local browser application. The client-side proxy application program tunnels the unencrypted, unprotected communication packets through a new protocol, thereby protecting the communications from a denial of service at the server side. Of course, the unencrypted, unprotected communication packets can be encrypted prior to tunneling.

The client-side proxy application program is not an operating system extension and does not involve any modifications to the operating system network stack and drivers. Consequently, the client is easier to install, remove and support in comparison to a VPN. Moreover, the client-side proxy application can be allowed through a corporate firewall using a much smaller "hole" in the firewall and is less of a security risk in comparison to allowing a protocol layer VPN through a corporate firewall.

The server-side implementation of the present invention authenticates valid field-hopped packets as valid or invalid very early in the server packet processing, similar to a standard virtual private network, for greatly minimizing the impact of a denial of service attempt in comparison to normal TCP/IP and HTTP communications, thereby protecting the server from invalid communications.

FIG. 36 shows a system block diagram of a computer network 3600 in which a virtual private connection according to the present invention can be configured to more easily traverse a firewall between two computer networks. FIG. 37 shows a flow diagram 3700 for establishing a virtual private connection that is encapsulated using an existing network protocol.

In FIG. 36 a local area network (LAN) 3601 is connected to another computer network 3602, such as the Internet, through a firewall arrangement 3603. Firewall arrangement operates in a well-known manner to interface LAN 3601 to computer network 3602 and to protect LAN 3601 from attacks initiated outside of LAN 3601.

A client computer 3604 is connected to LAN 3601 in a well-known manner. Client computer 3604 includes an operating system 3605 and a web browser 3606. Operating system 3605 provides kernel mode functions for operating client computer 3604. Browser 3606 is an application program for accessing computer network resources connected to LAN 3601 and computer network 3602 in a well-known manner. According to the present invention, a proxy application 3607 is also stored on client computer 3604 and operates at an application layer in conjunction with browser 3606. Proxy application 3607 operates at the application layer within client computer 3604 and when enabled, modifies unprotected, unencrypted message packets generated by browser 3606 by inserting data into the message packets that are used for forming a virtual private connection between client computer 3604 and a server computer connected to LAN 3601 or com-

54

puter network 3602. According to the invention, a virtual private connection does not provide the same level of security to the client computer as a virtual private network. A virtual private connection can be conveniently authenticated so that, for example, a denial of service attack can be rapidly rejected, thereby providing different levels of service that can be subscribed to by a user.

Proxy application 3607 is conveniently installed and uninstalled by a user because proxy application 3607 operates at the application layer within client computer 3604. On installation, proxy application 3607 preferably configures browser 3606 to use proxy application for all web communications. That is, the payload portion of all message packets is modified with the data for forming a virtual private connection between client computer 3604 and a server computer. Preferably, the data for forming the virtual private connection contains field-hopping data, such as described above in connection with VPNs. Also, the modified message packets preferably conform to the UDP protocol. Alternatively, the modified message packets can conform to the TCP/IP protocol or the ICMP protocol. Alternatively, proxy application 3606 can be selected and enabled through, for example, an option provided by browser 3606. Additionally, proxy application 3607 can be enabled so that only the payload portion of specially designated message packets is modified with the data for forming a virtual private connection between client computer 3604 and a designated host computer. Specially designated message packets can be, for example, selected predetermined domain names.

Referring to FIG. 37, at step 3701, unprotected and unencrypted message packets are generated by browser 3606. At step 3702, proxy application 3607 modifies the payload portion of all message packets by tunneling the data for forming a virtual private connection between client computer 3604 and a destination server computer into the payload portion. At step, 3703, the modified message packets are sent from client computer 3604 to, for example, website (server computer) 3608 over computer network 3602.

Website 3608 includes a VPN guard portion 3609, a server proxy portion 3610 and a web server portion 3611. VPN guard portion 3609 is embedded within the kernel layer of the operating system of website 3608 so that large bandwidth attacks on website 3608 are rapidly rejected. When client computer 3604 initiates an authenticated connection to website 3608, VPN guard portion 3609 is keyed with the hopping sequence contained in the message packets from client computer 3604, thereby performing a strong authentication of the client packet streams entering website 3608 at step 3704. VPN guard portion 3609 can be configured for providing different levels of authentication and, hence, quality of service, depending upon a subscribed level of service. That is, VPN guard portion 3609 can be configured to let all message packets through until a denial of service attack is detected, in which case VPN guard portion 3609 would allow only client packet streams conforming to a keyed hopping sequence, such as that of the present invention.

Server proxy portion 3610 also operates at the kernel layer within website 3608 and catches incoming message packets from client computer 3604 at the VPN level. At step 3705, server proxy portion 3610 authenticates the message packets at the kernel level within host computer 3604 using the destination IP address, UDP ports and discriminator fields. The authenticated message packets are then forwarded to the authenticated message packets to web server portion 3611 as normal TCP web transactions.

At step 3705, web server portion 3611 responds to message packets received from client computer 3604 in accordance

US 7,921,211 B2

55

with the particular nature of the message packets by generating reply message packets. For example, when a client computer requests a webpage, web server portion 3611 generates message packets corresponding to the requested webpage. At step 3706, the reply message packets pass through server proxy portion 3610, which inserts data into the payload portion of the message packets that are used for forming the virtual private connection between host computer 3608 and client computer 3604 over computer network 3602. Preferably, the data for forming the virtual private connection is contains field-hopping data, such as described above in connection with VPNs. Server proxy portion 3610 operates at the kernel layer within host computer 3608 to insert the virtual private connection data into the payload portion of the reply message packets. Preferably, the modified message packets sent by host computer 3608 to client computer 3604 conform to the UDP protocol. Alternatively, the modified message packets can conform to the TCP/IP protocol or the ICMP protocol.

At step 3707, the modified packets are sent from host computer 3608 over computer network 3602 and pass through firewall 3603. Once through firewall 3603, the modified packets are directed to client computer 3604 over LAN 3601 and are received at step 3708 by proxy application 3607 at the application layer within client computer 3604. Proxy application 3607 operates to rapidly evaluate the modified message packets for determining whether the received packets should be accepted or dropped. If the virtual private connection data inserted into the received information packets conforms to expected virtual private connection data, then the received packets are accepted. Otherwise, the received packets are dropped.

While the present invention has been described in connection with the illustrated embodiments, it will be appreciated and understood that modifications may be made without departing from the true spirit and scope of the invention.

What is claimed is:

1. A system for providing a domain name service for establishing a secure communication link, the system comprising: a domain name service system configured and arranged to be connected to a communication network, store a plurality of domain names and corresponding network addresses, receive a query for a network address, and indicate in response to the query whether the domain name service system supports establishing a secure communication link.

2. The system of claim 1, wherein at least one of the plurality of domain names comprises a top-level domain name.

3. The system of claim 2, wherein the top-level domain name is a non-standard top-level domain name.

4. The system of claim 3, wherein the non-standard top-level domain name is one of .scom, .sorg, .snet, .sgov, .sedu, .smil and .sint.

5. The system of claim 2, wherein the domain name service system is configured to authenticate the query using a cryptographic technique.

6. The system of claim 1, wherein the communication network includes the Internet.

7. The system of claim 1, wherein the domain name service system comprises an edge router.

8. The system of claim 1, wherein the domain name service system is connectable to a virtual private network through the communication network.

9. The system of claim 8, wherein the virtual private network is one of a plurality of secure communication links in a hierarchy of secure communication links.

56

10. The system of claim 8, wherein the virtual private network is based on inserting into each data packet communicated over a secure communication link one or more data values that vary according to a pseudo-random sequence.

11. The system of claim 8, wherein the virtual private network is based on a network address hopping regime that is used to pseudorandomly change network addresses in packets transmitted between a first device and a second device.

12. The system of claim 8, wherein the virtual private network is based on comparing a value in each data packet transmitted between a first device and a second device to a moving window of valid values.

13. The system of claim 8, wherein the virtual private network is based on a comparison of a discriminator field in a header of each data packet to a table of valid discriminator fields maintained for a first device.

14. The system of claim 1, wherein the domain name service system is configured to respond to the query for the network address.

15. The system of claim 1, wherein the domain name service system is configured to provide, in response to the query, the network address corresponding to a domain name from the plurality of domain names and the corresponding network addresses.

16. The system of claim 1, wherein the domain name service system is configured to receive the query initiated from a first location, the query requesting the network address associated with a domain name, wherein the domain name service system is configured to provide the network address associated with a second location, and wherein the domain name service system is configured to support establishing a secure communication link between the first location and the second location.

17. The system of claim 1, wherein the domain name service system is connected to a communication network, stores a plurality of domain names and corresponding network addresses, and comprises an indication that the domain name service system supports establishing a secure communication link.

18. The system of claim 1, wherein at least one of the plurality of domain names is reserved for secure communication links.

19. The system of claim 1, wherein the domain name service system comprises a server.

20. The system of claim 19, wherein the domain name service system further comprises a domain name database, and wherein the domain name database stores the plurality of domain names and the corresponding network addresses.

21. The system of claim 1, wherein the domain name service system comprises a server, wherein the server comprises a domain name database, and wherein the domain name database stores the plurality of domain names and the corresponding network addresses.

22. The system of claim 1, wherein the domain name service system is configured to store the corresponding network addresses for use in establishing secure communication links.

23. The system of claim 1, wherein the domain name service system is configured to authenticate the query for the network address.

24. The system of claim 1, wherein at least one of the plurality of domain names comprises an indication that the domain name service system supports establishing a secure communication link.

25. The system of claim 1, wherein at least one of the plurality of domain names comprises a secure name.

## US 7,921,211 B2

57

26. The system of claim 1, wherein at least one of the plurality of domain names enables establishment of a secure communication link.

27. The system of claim 1, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location transparently to a user at the first location.

28. The system of claim 1, wherein the secure communication link uses encryption.

29. The system of claim 1, wherein the secure communication link is capable of supporting a plurality of services.

30. The system of claim 29, wherein the plurality of services comprises a plurality of communication protocols, a plurality of application programs, multiple sessions, or a combination thereof.

31. The system of claim 30, wherein the plurality of application programs comprises items selected from a group consisting of the following: video conferencing, e-mail, a word processing program, and telephony.

32. The system of claim 29, wherein the plurality of services comprises audio, video, or a combination thereof.

33. The system of claim 1, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location.

34. The system of claim 33, wherein the query is initiated from the first location, wherein the second location comprises a computer, and wherein the network address is an address associated with the computer.

35. The system of claim 1, wherein the domain name service system comprises a domain name database connected to a communication network and storing a plurality of domain names and corresponding network addresses for communication, wherein the domain name database is configured so as to provide a network address corresponding to a domain name in response to a query in order to establish a secure communication link.

36. A non-transitory machine-readable medium comprising instructions executable in a domain name service system, the instructions comprising code for: connecting the domain name service system to a communication network; storing a plurality of domain names and corresponding network addresses; receiving a query for a network address; and indicating in response to the query whether the domain name service system supports establishing a secure communication link.

37. The non-transitory machine-readable medium of claim 36, wherein the instructions comprise code for storing the plurality of domain names and corresponding network addresses including at least one top-level domain name.

38. The non-transitory machine-readable medium of claim 36, wherein the instructions comprise code for responding to the query for the network address.

39. The non-transitory machine-readable medium of claim 36, wherein the instructions comprise code for providing, in response to the query, the network address corresponding to a domain name from the plurality of domain names and the corresponding network addresses.

40. The non-transitory machine-readable medium of claim 36, wherein the instructions comprise code for receiving the query for a network address associated with a domain name and initiated from a first location, and providing a network address associated with a second location, and establishing a secure communication link between the first location and the second location.

41. The non-transitory machine-readable medium of claim 36, wherein the instructions comprise code for indicating that

58

the domain name service system supports the establishment of a secure communication link.

42. The non-transitory machine-readable medium of claim 36, wherein the instructions comprise code for reserving at least one of the plurality of domain names for secure communication links.

43. The non-transitory machine-readable medium of claim 36, wherein the code resides on a server.

44. The non-transitory machine-readable medium of claim 36, wherein the instructions comprise code for storing a plurality of domain names and corresponding network addresses so as to define a domain name database.

45. The non-transitory machine-readable medium of claim 36, wherein the code resides on a server, and the instructions comprise code for creating a domain name database configured to store the plurality of domain names and the corresponding network addresses.

46. The non-transitory machine-readable medium of claim 36, wherein the instructions comprise code for storing the corresponding network addresses for use in establishing secure communication links.

47. The non-transitory machine-readable medium of claim 36, wherein the instructions comprise code for authenticating the query for the network address.

48. The non-transitory machine-readable medium of claim 36, wherein at least one of the plurality of domain names includes an indication that the domain name service system supports the establishment of a secure communication link.

49. The non-transitory machine-readable medium of claim 36, wherein at least one of the plurality of domain names includes a secure name.

50. The non-transitory machine-readable medium of claim 36, wherein at least one of the plurality of domain names is configured so as to enable establishment of a secure communication link.

51. The non-transitory machine-readable medium of claim 36, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location transparently to a user at the first location.

52. The non-transitory machine-readable medium of claim 36, wherein the secure communication link uses encryption.

53. The non-transitory machine-readable medium of claim 36, wherein the secure communication link is capable of supporting a plurality of services.

54. The non-transitory machine-readable medium of claim 53, wherein the plurality of services comprises a plurality of communication protocols, a plurality of application programs, multiple sessions, or a combination thereof.

55. The non-transitory machine-readable medium of claim 54, wherein the plurality of application programs comprises items selected from a group consisting of the following: video conferencing, e-mail, a word processing program, and telephony.

56. The non-transitory machine-readable medium of claim 53, wherein the plurality of services comprises audio, video, or a combination thereof.

57. The non-transitory machine-readable medium of claim 36, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location.

58. The non-transitory machine-readable medium of claim 57, wherein the instructions include code for receiving a query initiated from the first location, wherein the second location comprises a computer, and wherein the network address is an address associated with the computer.

US 7,921,211 B2

59

59. The non-transitory machine-readable medium of claim 36, wherein the domain name service system comprises a domain name database connected to a communication network and storing a plurality of domain names and corresponding network addresses for communication, wherein the domain name database is configured so as to provide a network address corresponding to a domain name in response to the query in order to establish a secure communication link.

60. A method of providing a domain name service for establishing a secure communication link, the method comprising:

60

connecting a domain name service system to a communication network;  
storing a plurality of domain names and corresponding network addresses; and  
upon receiving a query for a network address for communication, indicating whether the domain name service system supports establishing a secure communication link.

\* \* \* \* \*

## **CERTIFICATE OF SERVICE**

I hereby certify that, on this 24th day of January, 2019, I filed the foregoing Non-Confidential Brief for Defendant-Appellant Apple Inc. with the Clerk of the United States Court of Appeals for the Federal Circuit via the CM/ECF system, which will send notice of such filing to all registered CM/ECF users.

/s/ William F. Lee  
WILLIAM F. LEE  
WILMER CUTLER PICKERING  
HALE AND DORR LLP  
60 State Street  
Boston, MA 02109  
(617) 526-6000

## **CERTIFICATE OF COMPLIANCE**

Pursuant to Fed. R. App. P. 32(g), the undersigned hereby certifies that this brief complies with the type-volume limitation of Federal Circuit Rule 32(a).

1. Exclusive of the exempted portions of the brief, as provided in Fed. R. App. P. 32(f) and Fed. Cir. R. 32(b), the brief contains 13,984 words.

2. The brief has been prepared in proportionally spaced typeface using Microsoft Word 2010 in 14 point Times New Roman font. As permitted by Fed. R. App. P. 32(g), the undersigned has relied upon the word count feature of this word processing system in preparing this certificate.

/s/ William F. Lee  
WILLIAM F. LEE  
WILMER CUTLER PICKERING  
HALE AND DORR LLP  
60 State Street  
Boston, MA 02109  
(617) 526-6000

January 24, 2019