identified that a router 22 in the embodiments of Figures 1, 4 and 7 of the '497 patent is disclosed as performing the function of *associating a maximum acceptable processing rate with each class of data packet received at said computer*, as recited in limitation [13.2].

233. As described above with respect to limitation [7.2], *Maher* discloses that the network apparatus 100, which performs the above-described classification operation, may be a router or switch in the network, or a "bump-in-the-line" type device that accepts and processes the data, and then places the processed data back on the line. EX1006, 6:63-67, 5:62-67. Therefore, *Maher* discloses or renders obvious structure corresponding to the alleged structure in the '497 patent that Patent Owner identified as performing the function of limitation [1.2].

[1.3.1] means for said computer to find information for packets it receives regarding the path by which said packets came to said computer via packet marks provided by routers leading to said host computer;

234. Limitation [1.3.1] is substantially similar to limitation [7.1.1], except that limitation [1.3.1] recites "*means for said computer to find information for packets it receives regarding the path*," whereas limitation [7.1.1] recites "*determining a path*." This difference in wording is immaterial with respect to patentability, because *Savage* discloses that a victim computer finds information for packets it receives regarding the path by which the packets came to the victim computer.

235. *Savage* discloses that each router along the path "'marks' one or more packets by augmenting them with additional information about the path they are traveling." 298, 299 (describing "Node append" algorithm in Figure 2 below as "appending each node's [i.e., router's] address to the end of the packet as it travels through the network from attacker to victim."). *Savage* discloses that "[c]onsequently, every packet received by the victim arrives with **a complete ordered list of the routers it traversed**—a built-in attack path." EX1005, 299.

Marking procedure at router R:
for each packet w, append R to w

Path reconstruction procedure at victim v:
for any packet w from attacker
extract path $(R_i..R_j)$ from the suffix of w

Figure 2: Node append algorithm.

EX1005, Fig. 2

236. *Savage* teaches that the victim V then implements a "path reconstruction procedure" to "reconstruct the attack path using only the information in these marked packets." EX1005, 298.

237. Therefore, *Savage* discloses that each router along the attack path adds a packet mark identifying that router's address to the data packet as the data packets travel along the attack path, to enable a path reconstruction procedure to be performed to determine the "complete ordered list of the routers [the data packet]

traversed." EX1005, 295, 298-299. Thus, *Savage* discloses the victim computer V, which performs the path reconstruction procedure, *find[s] information* (i.e., packet marks) *for packets it receives regarding the path by which said packets came to said computer via packet marks provided by routers leading to said host computer*, as recited in limitation [1.3.1].

238. I have been informed that Patent Owner has identified that a computer 18 in the embodiments of Figures 3, 6, and 9 of the '497 patent is disclosed as performing the function of *find[ing] information for packets it receives regarding the path by which said packets came to said computer via packet marks provided by routers leading to said host computer*, as recited in limitation [1.3.1].

239. As described above, *Savage* discloses that the victim computer V *find[s] information* (i.e., packet marks) *for packets it receives regarding the path by which said packets came to said computer via packet marks provided by routers leading to said host computer*, as recited in limitation [1.3.1]. Therefore, *Savage* discloses or renders obvious structure corresponding to the alleged structure in the '497 patent that Patent Owner identified as performing the function of limitation [1.3.1].

[1.3.2] said path comprising all routers in said network via which said packets are routed to said computer; and

240. Limitation [1.2.2] is identical to limitation [7.1.2], and is disclosed or rendered obvious by *Savage*, as described above with respect to limitation [7.1.2].

[1.4] means in said computer for using said information to allocate the processing rate available for unwanted data packets to be less than or equal to said maximum acceptable processing rate.

241. Limitation [1.4] is substantially similar to limitation [7.3], except that limitation [1.4] recites "means in said computer for using said information to allocate the processing rate…," whereas limitation [7.3] recites "allocating a processing rate…."

242. Patent Owner has identified that a computer 18 in the embodiments of Figures 1, 4 and 7 of the '497 patent is disclosed as performing the function of *using said information to allocate the processing rate available for unwanted data packets to be less than or equal to said maximum acceptable processing rate*, as recited in limitation [1.4].

243. As described above with respect to limitation [7.4], a POSITA would have been motivated to modify *Savage* to include a "bump-in-the-line" type device in the network of the host computer, as taught by both *Schuba* and *Maher*, to implement the packet flood preventive measures as taught by *Maher* and *Schuba* in the "bump-in-the-line" device. Therefore, Savage as modified by Maher or Schuba

renders obvious the structure corresponding to the alleged structure in the '497 patent that Patent Owner identified as performing the function of limitation [1.4].

D. Proposed Rejection #2 (Claims 16 and 4): Claims 16 and 4 would have been obvious over *Maher* in view of *Savage* and *Schuba*

1. Overview of Claim 16

244. Claim 16 is substantially similar to claim 10, but differs in two respects. First, claim 10 is a "method" claim, whereas claim 16 is a "system" claim. Second, claim 16 recites several "means for" limitations whose functions are substantially identical to the method steps of claim 10.

245. The following table illustrates the differences between claims 10 and 16, with underlined text denoting the differences between the claims.

Claim 10		Claim 16	
[10.0]	A method of providing packet flooding defense for a network comprising a plurality of host computers, routers, communication lines and transmitted data packets, said method comprising the steps of:	[16.0]	A packet flooding defense system for a network comprising a plurality of host computers, routers, communication lines and transmitted data packets, said system comprising:
[10.1.1]	determining a path by which data packets arrive at said router via packet marks provided by routers leading to said host computer;	[16.1.1]	means for a router to determine a path by which said packets came to said router via packet marks provided by routers leading to said router;

[10.1.2]	said path comprising all routers in said network via which said packets are routed to said computer;	[16.1.2]	said path comprising all routers in said network via which said packets are routed to said computer;
[10.2]	classifying data packets received at said router via packet marks provided by routers leading to said host computer by path;	[16.2]	means for classifying data packets received at said router into wanted data packets and unwanted data packets by path;
[10.3]	associating a maximum acceptable transmission rate with each class of data packet received at said router; and	[16.3]	means for associating a maximum acceptable transmission rate with each class of data packet received at said router; and
[10.4]	allocating a transmission rate equal to or less than said maximum acceptable transmission rate for unwanted data packets.	[16.4]	means for said router to allocate the transmission rate for unwanted data packets to be less than equal to said maximum acceptable processing rate.

[16.0] A packet flooding defense system for a network comprising a plurality of host computers, routers, communication lines and transmitted data packets, said system comprising:

246. As described above with respect to limitation [13.0], *Maher* discloses *a packet flooding defense system for a network comprising a plurality of host computers, routers, communication lines and transmitted data packets*. Therefore, *Maher* discloses or renders obvious limitation [16.0].

[16.1.1] means for a router to determine a path by which said packets came to said router via packet marks provided by routers leading to said router;

247. Limitation [16.1.1] is substantially similar to limitation [10.1.1], except that limitation [16.1.1] additionally recites "means for a router to determine." Limitation [16.1.1] expressly recites that a router performs the functional language. As described above with respect to limitation [10.1.1], a POSITA would have been motivated to perform the path reconstruction procedure disclosed by *Savage* in a router, in view of *Maher*'s disclosure that routers are configured to perform packet analysis procedures in the event of a QoS attack. Therefore, *Maher* in view of *Savage* renders obvious limitation [16.1.1].

[16.1.2] said path comprising all routers in said network via which said packets are routed to said computer;

248. Limitation [16.1.2] is identical to limitations [7.1.2] and [10.1.2], and is disclosed or rendered obvious by *Savage*, as described above with respect to limitations [7.1.2] and [10.1.2].

[16.2] means for classifying data packets received at said router into wanted data packets and unwanted data packets by path;

249. Limitation [16.2] is substantially similar to limitation [10.2], except that limitation [16.2] additionally recites "means for." Further, limitation [16.2] additionally recites that the data packets are classified into wanted data packets and

unwanted data packets by path. Both of these differences do not make claim 16 patentable, in my opinion.

250. As described above with respect to limitation [7.2], *Maher* and *Schuba* render obvious classifying data packets into wanted data packets and unwanted data packets by path, as recited in limitation [16.2].

251. I have been informed that Patent Owner has identified that a router 22 in the embodiments of Figures 1, 4 and 7 of the '497 patent is disclosed as performing the function of *classifying data packets received at said router into wanted data packets and unwanted data packets by path*, as recited in limitation [13.2].

252. As described above with respect to limitation [7.2], *Maher* discloses that the network apparatus 100, which performs the above-described classification operation, may be a router or switch in the network, or a "bump-in-the-line" type device that accepts and processes the data, and then places the processed data back on the line. EX1006, 6:63-67, 5:62-67. Therefore, *Maher* discloses or renders obvious structure corresponding to the alleged structure in the '497 patent that Patent Owner identified as performing the function of limitation [13.2].

253. Further, as described above with respect to limitation [7.2], a POSITA would have been motivated to include a bump-in-the-line type network device that receives data packets transmitted to *Savage*'s victim host computer V, as taught by

Schuba and *Maher*, to *classify[] data packets received at said host computer into wanted data packets and unwanted data packets by path*, as taught by *Schuba* and *Maher*.

[16.3] means for associating a maximum acceptable transmission rate with each class of data packet received at said router; and

254. Limitation [16.3] is substantially similar to limitation [10.3], except that limitation [16.3] additionally recites "means for."

255. I have been informed that Patent Owner identified that a router 22 in the embodiments of Figures 2, 5 and 8 of the '497 patent is disclosed as performing the function of *associating a maximum acceptable transmission rate with each class of data packet received at said router*, as recited in limitation [16.3].

256. As described above with respect to limitation [7.2], *Maher* discloses that the network apparatus 100, which performs the above-described classification operation, may be a router or switch in the network, or a "bump-in-the-line" type device that accepts and processes the data, and then places the processed data back on the line. EX1006, 6:63-67, 5:62-67. Therefore, *Maher* discloses or renders obvious structure corresponding to the alleged structure in the '497 patent that Patent Owner identified as performing the function of limitation [16.3].

[16.4] means for said router to allocate the transmission rate for unwanted data packets to be less than equal to said maximum acceptable processing rate.

257. Limitation [16.4] is substantially similar to limitation [10.4], except that limitation [16.4] additionally recites "*means for said router to allocate....*" Limitation [16.4] expressly recites that a router performs the functional language. As described above with respect to limitation [10.4], *Maher* discloses that its network apparatus 100 may be a router or switch in the network, or a "bump-in-the-line" type device that accepts and processes the data, and then places the processed data back on the line. EX1006, 6:63-67, 5:62-67. Accordingly, *Maher*'s functions of allocating a processing rate are disclosed as being performed in a router that transmits data to a downstream device. Therefore, *Maher* also teaches allocate the transmission rate for unwanted data packets to be less than equal to said maximum acceptable processing rate, as recited in limitation [16.4].

2. Overview of Claim 4

258. Claim 4 is substantially similar to claim 10, but differs in three respects. First, claim 10 is a "method" claim, whereas claim 4 is a "system" claim. Second, claim 4 recites several "means for" limitations whose functions are substantially identical to the method steps of claim 10. Third, claim 4 contains a non-substantive variation from limitation [7.1.1], of *finding information regarding a path* (limitation [4.3.1] below), rather than determining a path, as recited in limitation [10.1.1].

259. The following table illustrates the differences between claims 10 and 4, with underlined text denoting the differences between the claims.

Claim 10		Claim 4	
[10.0]	A method of providing packet flooding defense for a network comprising a plurality of host computers, routers, communication lines and transmitted data packets, said method comprising the steps of:	[4.0]	A packet flooding defense system for a network comprising a plurality of host computers, routers, communication lines and transmitted data packets, said system comprising:
[10.1.1]	determining a path by which data packets arrive at said router via packet marks provided by routers leading to said host computer;	[4.3.1]	means for said router to find information for packets it receives regarding the path by which said packets came to said router via packet marks provided by routers leading to said host computer;
[10.1.2]	said path comprising all routers in said network via which said packets are routed to said computer;	[4.3.2]	said path comprising all routers in said network via which said packets are routed to said computer;
[10.2]	classifying data packets received at said router via packet marks provided by routers leading to said host computer by path;	[4.2]	means for classifying data packets received at a router into wanted data packets and unwanted data packets;
[10.3]	associating a maximum acceptable transmission rate with each class of	[4.3]	means for associating a maximum acceptable transmission rate with

	data packet received at said router; and		each class of data packet received at said router;
[10.4]	allocating a transmission rate equal to or less than said maximum acceptable transmission rate for unwanted data packets.	[4.4]	means for said router to allocate the transmission rate for unwanted data packets to be less than equal to said maximum acceptable processing rate.

[4.0] A packet flooding defense system for a network comprising a plurality of host computers, routers, communication lines and transmitted data packets, said system comprising:

260. As described above with respect to limitation [13.0], *Maher* discloses *a packet flooding defense system for a network comprising a plurality of host computers, routers, communication lines and transmitted data packets*. Therefore, *Maher* discloses or renders obvious limitation [4.0].

[4.1] means for classifying data packets received at a router into wanted data packets and unwanted data packets;

261. Limitation [4.1] is substantially similar to limitation [16.2], except that it is broader because it does not require classifying data packets "by path," as recited in limitation [16.2]. Limitation [4.1] is rendered obvious for the reasons presented above with respect to limitation [16.2], which also shows that the prior art renders obvious the alleged structure that Patent Owner identified as corresponding to the "means for" features of limitation [4.1].

[4.2] means for associating a maximum acceptable transmission rate with each class of data packet received at said router;

262. Limitation [4.2] is identical to limitation [16.3], and is rendered obvious for the same reasons presented above with respect to limitation [16.3].

[4.3.1] means for said router to find information for packets it receives regarding the path by which said packets came to said router via packet marks provided by routers leading to said host computer;

263. Limitation [4.3.1] is substantially similar to limitation [10.1.1], except that limitation [4.3.1] recites "*means for said router computer to find information for packets it receives regarding the path*," whereas limitation [10.1.1] recites "*determining a path*." This difference in wording is immaterial with respect to patentability, because *Savage* discloses a network device receiving packets finds information for packets it receives regarding the path by which the packets came to that network device.

264. As described above with respect to limitation [10.1.1], *Savage* discloses that the victim V in Figure 1 below performs a "path reconstruction procedure" to determine the "attack path from [attacker] A_i [that] is the **unique ordered list of routers between A_i and V**." EX1005, 298, 295.

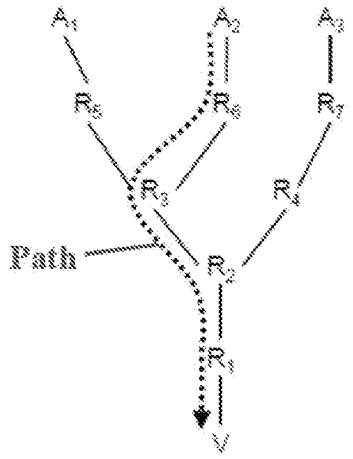


Figure 1: Network as seen from the victim of an attack, V. Routers are represented by R_i, and potential attackers by A_i. The dotted line represents a particular *attack path* between an attacker and the victim.

EX1005, Fig. 1 (annotated)

265. *Savage* discloses that the victim V in Figure 1 above "may be a single host under attack, or a **network border device** such as a firewall or intrusion detection system that represents many such hosts." EX1005, 297. *Maher* discloses that its network apparatus 100 may be a router or switch in the network, or a "bump-in-the-line" type device that accepts and processes the data, and then places the processed data back on the line. EX1006, 6:63-67, 5:62-67. A POSITA would therefore have been motivated to perform the path reconstruction procedure disclosed by *Savage* in a router, in view of *Maher*'s disclosure that routers are configured to perform packet analysis procedures in the event of a QoS attack.

266. *Savage* discloses that each router along the path "'marks' one or more packets by augmenting them with additional information about the path they are traveling." 298, 299 (describing "Node append" algorithm in Figure 2 below as "appending each node's [i.e., router's] address to the end of the packet as it travels through the network from attacker to victim."). *Savage* discloses that "[c]onsequently, every packet received by the victim arrives with **a complete ordered list of the routers it traversed**—a built-in attack path." EX1005, 299.

Marking procedure at router R:
for each packet w, append R to w

Path reconstruction procedure at victim v:
for any packet w from attacker
extract path $(R_i..R_j)$ from the suffix of w

Figure 2: Node append algorithm.

EX1005, Fig. 2

267. *Savage* teaches that the victim V then implements a "path reconstruction procedure" to "reconstruct the attack path using only the information in these marked packets." EX1005, 298.

268. Therefore, *Savage* discloses that each router along the attack path adds a packet mark identifying that router's address to the data packet as the data packets travel along the attack path, to enable a path reconstruction procedure to be performed to determine the "complete ordered list of the routers [the data packet]

traversed." EX1005, 295, 298-299. Thus, *Savage* discloses the victim computer V, which may be a router, as taught by *Maher*, performs the path reconstruction procedure, *find[s] information* (i.e., packet marks) *for packets it receives regarding the path by which said packets came to said computer via packet marks provided by routers leading to said host computer*, as recited in limitation [4.3.1].

269. I have been informed that Patent Owner has identified that a router 22 in the embodiments of Figures 3, 6, and 9 of the '497 patent is disclosed as performing the function of *find[ing] information for packets it receives regarding the path by which said packets came to said router via packet marks provided by routers leading to said host computer*, as recited in limitation [4.3.1].

270. As described above with respect to limitation [7.2], *Maher* discloses that the network apparatus 100 may be a router or switch in the network, or a "bump-in-the-line" type device that accepts and processes the data, and then places the processed data back on the line. EX1006, 6:63-67, 5:62-67. Therefore, *Maher* discloses or renders obvious structure corresponding to the alleged structure in the '497 patent that Patent Owner identified as performing the function of limitation [4.3.1].

[4.3.2] said path comprising all routers in said network via which said packets are routed to said computer; and

271. Limitation [4.3.2] is identical to limitations [7.1.2] and [10.1.2], and is disclosed or rendered obvious by *Savage*, as described above with respect to limitations [7.1.2] and [10.1.2].

[4.4] means in said router for said router to use said information to allocate the transmission rate for unwanted data packets to be less than equal to said maximum acceptable transmission rate.

272. Limitation [4.4] is substantially identical to limitation [16.4], except that limitation [4.4] recites "means in said router," whereas limitation [16.4] recites "means for said router." Limitation [4.4] is rendered obvious for the same reasons as presented above with respect to limitation [16.4].

XI. AVAILABILITY FOR CROSS-EXAMINATION

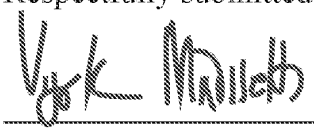
273. In signing this declaration, I recognize that the declaration will be filed as evidence in a case before the Central Reexamination Unit of the United States Patent and Trademark Office. If cross examination is required of me, I will appear for cross examination within the United States during the time allotted for cross examination.

XIII. CONCLUSION

274. I hereby declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct, and that all statements made of my own knowledge are true and that all statements made on information and belief are believed to be true. I understand that willful false statements are punishable by fine or imprisonment or both. *See* 18 U.S.C. § 1001.

Date: May 1, 2021

Respectfully submitted,



Dr. Vijay K. Madisetti

Exhibit 1004

Curriculum Vitae of Dr. Vijay Madisetti

Customer No.: 000027683

Haynes and Boone, LLP
IP Section
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Telephone [214] 651.5000
Fax [214] 200.0853

Dr. Vijay K. Madisetti
Fellow, IEEE
vkm@madisetti.com
Cell: 770-527-0177
Address:
56 Creekside Park Drive
Johns Creek, GA 30022

Employment:

- 1984-1989: Post Graduate Researcher (UC Berkeley),
- 1989-present: Full Professor of Electrical & Computer Engineering (**Georgia Tech, Atlanta, GA 30332**).

Areas of Technical Interest – Wireless & Mobile Communications, Computer Engineering, Circuit Design (Analog/Digital), Biomedical Instrumentation Hardware/Software, Software Engineering, Digital Signal Processing, Wireline & Wireless Computer Networks, Software Systems, Control Systems, Cloud Computing.

Education: PhD (EECS, University of California at Berkeley, 1989), **B.Tech** (Hons) in Electronics and Electrical Communications Engineering (Indian Institute of Technology, Kharagpur, 1984).

Startup Companies:

Director, **VP Technologies, Inc**. (1995-): A startup commercialized through Georgia Tech's Advanced Technology Development Corporation (ATDC) focusing on digital software and hardware design services for military market. http://www.vptinc.net

Director, **Soft Networks, LLC** (2001-2007): A startup commercialized through Georgia Tech support focusing on software development tools and compilers for Cellular/WiFi/VOIP/telecommunication products. http://www.soft-networks.com

Director, **Elastic Video Inc.** (2007- 2009): A startup commercialized through Georgia Tech's VentureLab (http://venturelab.gatech.edu) development image and video processing software for wireless & IP networking.

Cloudflare, Inc. Exhibit 1003 Page 824

Litigation Experience (2015-2021) With Testimony

(Note: There may be multiple cases between the parties, e.g., District Court v. ITC, US versus Foreign Cases)

Chrimar v. Adtran, Alcatel, et al.
Technology: Power over Ethernet (2015-2017)
Submitted reports & Deposition & Trial

Chamberlain v. Ryobi/TTI
Case No: 1:16-cv-06097 (ND Illinois)
Expert for Ryobi
Technology: Wireless/IoT/Barrier Movement (2016 – 2019)
Submitted reports & deposition & trial testimony

IOEngine v. IMC/Imation
Case No: cv-14-1572-GMS (US Delaware)
Expert for IMC/Imation
Technology: Networked Storage Device (2016-2017)
Submitted reports & deposition & trial testimony

Huawei v. Samsung
Case No: 3:16-cv-2787-WHO (ND Cal)
Expert for Samsung
Technology: 4G/LTE Random Access Protocols (2016-2019)
Submitted reports & deposition

Hitachi Maxell v. ZTE/Huawei
Case No: 5:16-cv-00178-RWS (ED Texas)
Expert for Hitachi Maxell
Technology: Digital Cameras
Submitted reports & deposition (2017 – 2018)

Hitachi Maxell v. Apple
Case No: 5:19-cv-00036-RWS
Expert for Hitachi Maxell
Technology: Digital Cameras
Submitted Expert Reports (2020-)

Qualcomm v. Apple
Case No: 17-ccv-0108-GPC-MDD (SD Cal) Also, Related ITC/FTC Matters
Expert for Qualcomm
Technology: 4G/Wireless Communications/Smartphones (2017-2019)

Submitted Reports and Deposition

Qualcomm v. Apple
Case No: 3:17-cv-01375-DMS-MDD (SD Cal)
Expert for Qualcomm
Technology: 4G/Wireless Communications/Smartphones (2017-2019)
Submitted Reports and Deposition

Optis v. Huawei
Case No: 2:17-cv-123 (E.D. Texas)
Expert for Optis Wireless
Technology: 4G/Video (2017-2019)
Submitted reports & deposition

Beckman Coulter v. Sysmex
Case No: 1:17-cv-24049-DPG (ND Illinois)
Expert for Sysmex
Technology: Medical Instrumentation Automation (2017-present)
Testifying Expert

TQ Delta v. Xyxel/Adtran
Expert for TQ Delta
Technology: DSL Technologies (2018-present)
Testifying Expert

3GL/KPN v. LG, Blackberry, HTC
Expert for 3GL/KPN
Technology: 4G/LTE Protocols (2018-present)
Reports and Deposition / Testifying Expert

Rovi v. Comcast
ITC No. 337-TA-1103 (ITC)
Expert for Rovi
Technology: Digital Video & interactive GUI (Jan 2018-May 2018)
Reports and Deposition

Cirba/Densify v. VMWare
Case No: 1:19-cv-00742-LPS
Expert for Cirba/Densify
Technology: Virtualization (2019-present)
Reports Deposition/PI Hearing / Testifying Expert

Power Integrations v. On Semiconductor
Case No: 17-cv-03189-BLF

Expert for On Semiconductor
Technology: Power Electronics (2018-2019)
Reports & Deposition/Testifying Expert.

Pathway v. Lumens, IPEVO, Aver JDA
Case No: ITC 337-TA-1045
Expert for Lumens JDA
Technology: Document Cameras (2016-2019)
Reports & Deposition (for IPR/ITC, no trial)

Wilan v. Apple
Case No: 3:14-cv-2235
Expert for WiLan
Technology: Voice over LTE/LTE Protocols (2017-2020)
Reports and Testified through Deposition/Trials

St. Lawrence Comm. v. Amazon
Case No. 2:19-CV-00027-JRG
Expert for Amazon
Technology: Multimedia Codecs
Submitted Expert Reports/Deposition (2020-2020)

PanOptis v. Apple
Case No. 2:19-CV-00066-JRG
Expert for PanOptis
Technology: LTE
Submitted Expert Reports/Deposition/Trial (2019-2020)

GAS v. Sprint Communications
Case No. 2:20-cv-00007-RWS
Expert for General Access (GAS)
Technology: LTE
Submitted Expert Reports/Deposition (9/21-)

Additional matters include declarations supporting IPRs at the PTAB for Google (US Patent 8,601,154), On Semiconductor (US Patent 6,212,079), Ubisoft (US Patent 5,490,216), Lumens JDA (2017-2019), Broadcom (WiFi), Sony (US Patent 6,101,534), Kia, (US Patent 5,530,431), Qualcomm, Ericsson, Amazon, Ring, Digital Ally, On Semiconductor, United Patents, Lenovo, BMC Software, BMW, Daimler and Dolby.

Earned Degrees

1. **B. Tech (Hons), Electronics & Electrical Comm. Engineering**
   Indian Institute of Technology (IIT), Kharagpur, India
   1984.

2. **Ph.D., Electrical Engineering & Computer Sciences (EECS)**
   University of California (UC), Berkeley. CA
   1989.

Books

1. **VLSI Digital Signal Processors**
   Madisetti, V.K.;
   Boston: MA, IEEE Press: Butterworth Heinemann, 1995, 525 pp.

2. **Quick-Turnaround ASIC Design in VHDL**
   Romdhane, M., Madisetti, V.K., Hines, J.
   Boston: MA, Kluwer Academic Press, 1996, 190 pp.

3. **The Digital Signal Processing Handbook (First Edition)**
   Madisetti, V. K., Williams, D. (Editors)
   CRC Press, Boca Raton, Fla, 1998, 2500 pp.

4. **VHDL: Electronics Systems Design Methodologies**.
   Madisetti, V. K. (Editor)
   Boston: MA, IEEE Standards Press, 2000, ISBN 0-7381-1878-8.

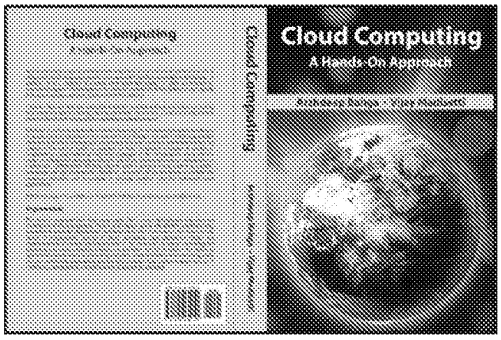
5. **Platform-Centric Approach to System-on-Chip (SoC) Design.**
   Madisetti, V. K., Arpnikanondt, A.
   Springer, Boston: MA, Springer, 2004, 280 pp.

6. **The Digital Signal Processing Handbook – Second Edition.**
   Madisetti, V. K. (2009)
   CRC Press, Boca Raton, Fla.

7. **Cloud Computing: A Hands-On Approach**
   A Bahga, V. Madisetti (2013)
   Amazon CreateSpace Publishing, 2013, 454 pp.

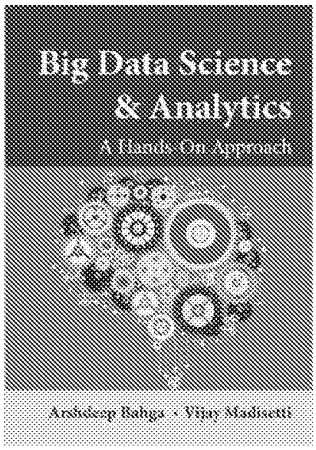
8. Internet of Things: A Hands-On Approach
A Bahga, V. Madisetti (2014)
Amazon CreateSpace Publishing, 2014, 450 pp.



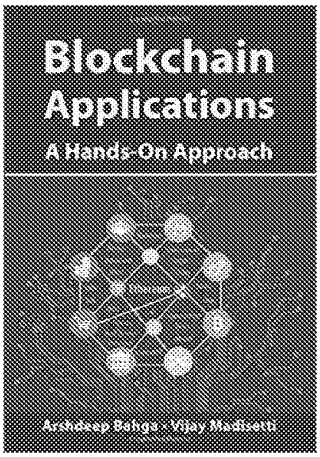


9. Big Data Science & Analytics: A Hands-On Approach
A Bahga, V. Madisetti (2016)
Amazon CreateSpace Publishing, 2016, 542 pp.

Cloudflare, Inc. Exhibit 1003 Page 829



10.Blockchain Applications: A Hands-On Approach
A Bahga, V. Madisetti (2017)
Amazon CreateSpace Publishing, 2017, 380 pp.



Edited Books & Collection of Papers

1. **Advances in Parallel & Distributed Simulation**
   Madisetti, V.K.;Nicol, D., Fujimoto, R. (Editors)
   San Diego, CA: SCS Press, 1991, 200 pp.

2. **Modeling, Analysis, Simulation of Computer & Telecommunications Systems**
   Madisetti, V., Gelenbe, E., Walrand, J. W. (Editors)
   Los Alamitos: CA, IEEE Computer Society Press, 1994, 425 pp.

3. **Modeling & Simulations on Microcomputers**
   Madisetti, V.K. (Editor)
   San Diego, CA: SCS Press, 138 pp. 1990.

Editorship of Journals & Transactions

1. **IEEE Design & Test of Computers**
   Special Issue: Reengineering Digital Systems
   April – June 1999 (Vol 16, No 2)
   Madisetti, V.K (Editor)
   Los Alamitos: CA, IEEE Computer Society Press, 1999.

2. **IEEE Design & Test of Computers**
   Special Issue: Rapid Prototyping of Digital Systems
   Fall 1996 (Vol 13, No 3)
   Madisetti, V., Richards, M. (Editors)
   Los Alamitos: CA, IEEE Computer Society Press, 1994, 425 pp.

3. **IEEE Transactions on Circuits & Systems II**
   Associate Editor: 1993-1995.

4. **International Journal in Computer Simulation**
   Associate Editor: 1990-1993

5. **International Journal in VLSI Signal Processing**
   Editorial Board: 1995 - Present

Issued US Patents

[1] 10,503,927 Method and system for securing cloud storage and databases from insider threats and optimizing performance, Issued Dec 10, 2019

[2] 10,460,283 Smart contract optimization for multiparty service or product ordering system, Issued Oct 29, 2019

[3] 10,459,946 Method and system for tuning blockchain scalability, decentralization, and security for fast and low-cost payment and transaction processing, Issued Oct 29, 2019

[4]. 10,402,589 Method and system for securing cloud storage and databases from insider threats and optimizing performance, Issued Sep 3, 2019

[5]. 10,394,845 Method and system for tuning blockchain scalability for fast and low-cost payment and transaction processing, Issued Aug 27, 2019

[6]. 10,289,631 Method and system for tuning blockchain scalability for fast and low-cost payment and transaction processing, Issued May 24, 2019

[7]. 10,255,342 Method and system for tuning blockchain scalability, decentralization, and security for fast and low-cost payment and transaction processing, Issued April 9, 2019

[8]. 10,243,743 Tokens or crypto currency using smart contracts and blockchains, Issued March 26, 2019

[9]. 10,204,339 Method and system for blockchain-based combined identity, ownership, integrity and custody management, Issued February 12, 2019

[10]. 10,204,148 Method and system for tuning blockchain scalability, decentralization, and security for fast and low-cost payment and transaction processing, Issued Feb 12, 2019

[11]. 10,121,143 Method and system for blockchain-based combined identity, ownership, integrity and custody management, Issued Nov 6, 2018

[12]. 10,102,526 Method and system for blockchain-based combined identity, ownership, integrity and custody management, October 16, 2018

[13]. 10,102,265 Method and system for tuning blockchain scalability for fast and low-cost payment and transaction processing, October 16, 2018

[14]. 9,935,772 Methods and systems for operating secure digital management aware applications, April 3, 2018

[15]. 9,769,213 Method and system for secure digital object management, Issued Sep 19, 2017

Refereed Journal Publications

1. **Trends in the Electronic Control of Mine Hoists**
   Madisetti, V. and Ramlu, M.,
   IEEE Transactions on Industry Applications, Vol IA-22, No. 6, November/December 1986. Pages 1105-1112

2. **Multilevel range/NEXT performance in digital subscriber loops**
   Brand, G.; Madisetti, V.; Messerschmitt, D.G.;
   Communications, Speech and Vision, IEE Proceedings I [see also IEE

Proceedings-Communications] ,Volume: 136 , Issue: 2 , April 1989
Pages:169 – 174

3. **Seismic migration algorithms on parallel computers**
   Madisetti, V.K.; Messerschmitt, D.G.;
   Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on] ,Volume: 39 , Issue: 7 , July 1991
   Pages:1642 – 1654

4. **Asynchronous algorithms for the parallel simulation of event-driven dynamical systems**
   Madisetti, V.K.; Walrand, J.C.; Messerschmitt, D.G.:
   ACM Transactions on Modeling and Computer Simulation, v 1, n 3, July 1991,
   Pages: 244-74

5. **Synchronization mechanisms for distributed event-driven computation**
   Madisetti, V.K.; Hardaker, D.:
   ACM Transactions on Modeling and Computer Simulation, v 2, n 1, Jan. 1992,
   Pages: 12-51

6. **Efficient VLSI Architectures for the Arithmetic Fourier Transform (AFT)**
   Kelley, B.T.; Madisetti, V.K.;
   Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on] ,Volume: 41 , Issue: 1 , January 1993
   Pages:365-378

7. **The fast discrete Radon transform. I. Theory**
   Kelley, B.T.; Madisetti, V.K.;
   Image Processing, IEEE Transactions on ,Volume: 2 , Issue: 3 , July 1993
   Pages:382 – 400

8. **The Georgia Tech digital signal multiprocessor**
   Barnwell, T.P., III; Madisetti, V.K.; McGrath, S.J.A.;
   Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on] ,Volume: 41 , Issue: 7 , July 1993
   Pages:2471 – 2487

9. **The MIMDIX Environment for Parallel Simulation**
   Madisetti, V.K.; Hardaker, D.; Fujimoto, R.M.:
   Journal of Parallel and Distributed Computing, v18, no. 4, August 1993,
   Pages: 473-83.

Cloudflare, Inc. Exhibit 1003 Page 833

10.LMSGEN: a prototyping environment for programmable adaptive digital filters in VLSI
Romdhane, M.S.B.; Madisetti, V.K.;
Chapter in VLSI Signal Processing, VII, 1994.,
Pages:33 – 42

11. Fixed-point co-design in DSP
Egolf, T.W.; Famorzadeh, S.; Madisetti, V.K.;
Chapter in VLSI Signal Processing, VII, 1994.,
Pages:113 - 126

12. A fast spotlight-mode synthetic aperture radar imaging system
Madisetti, V.K.;
Communications, IEEE Transactions on ,Volume: 42 , Issue: 234 , February-April 1994
Pages:873 – 876

13. Rapid prototyping on the Georgia Tech digital signal multiprocessor
Curtis, B.A.; Madisetti, V.K.;
Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on] ,Volume: 42 , Issue: 3 , March 1994
Pages:649 – 662

14.Low-power signaling in asymmetric noisy channels via spectral shaping
Sipitca, M.; Madisetti, V.K.;
Signal Processing Letters, IEEE, Volume: 1 , Issue: 8 , Aug 1994
Pages:117 – 118

15. A quantitative methodology for rapid prototyping and high-level synthesis of signal processing algorithms
Madisetti, V.K.; Curtis, B.A.;
Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on] ,Volume: 42 , Issue: 11 , Nov. 1994
Pages:3188 – 3208

16. Computer Simulation of Application-Specific Signal Processing Systems
Casinovi, G.; Madisetti, V.K.;
International Journal in Computer Simulation, Vol. 4, No. 4, Nov 1994.

17. System partitioning of MCMs for low power
Khan, S.A.; Madisetti, V.K.;
Design & Test of Computers, IEEE ,Volume: 12 , Issue: 1 , Spring 1995
Pages:41 – 52

18. Error correcting run-length limited codes for magnetic recording
Jaejin Lee; Madisetti, V.K.;
Magnetics, IEEE Transactions on ,Volume: 31 , Issue: 6 , Nov. 1995
Pages:3084 – 3086

19. Virtual prototyping of embedded microcontroller-based DSP systems
Madisetti, V.K.; Egolf, T.W.;
Micro, IEEE ,Volume: 15 , Issue: 5 , Oct. 1995
Pages:9 – 21

20. Constrained multitrack RLL codes for the storage channel
Lee, J.; Madisetti, V.K.;
Magnetics, IEEE Transactions on ,Volume: 31 , Issue: 3 , May 1995
Pages:2355 – 2364

21. Rapid digital system prototyping: current practice, future challenges
Madisetti, V.K.;
Design & Test of Computers, IEEE ,Volume: 13 , Issue: 3 , Fall 1996
Pages:12 – 22

22. Conceptual prototyping of scalable embedded DSP systems
Dung, L.-R.; Madisetti, V.K.;
Design & Test of Computers, IEEE ,Volume: 13 , Issue: 3 , Fall 1996
Pages:54 – 65

23. Advances in rapid prototyping of digital systems
Madisetti, V.K.; Richards, M.A.;
Design & Test of Computers, IEEE ,Volume: 13 , Issue: 3 , Fall 1996
Pages:9

24. Combined modulation and error correction codes for storage channels
Jaejin Lee; Madisetti, V.K.;
Magnetics, IEEE Transactions on ,Volume: 32 , Issue: 2 , March 1996
Pages:509 – 514

25. Model-based architectural design and verification of scalable embedded DSP systems-a RASSP approach
Dung, L.-R.; Madisetti, V.K.; Hines, J.W.;
Chapter in VLSI Signal Processing, IX, 1996.
Pages:147 – 156

26. Low-power digital filter implementations using ternary coefficients
Hezar, R.; Madisetti, V.K.;
Chapter in VLSI Signal Processing, IX, 1996.,
Pages:179 – 188

27. All-digital oversampled front-end sensors
Romdhane, M.S.B.; Madisetti, V.K.;
Signal Processing Letters, IEEE, Volume: 3 , Issue: 2 , Feb. 1996
Pages:38 – 39

28. Modeling COTS components in VHDL
Calhoun, S., Reese, R; Egolf, T., Madisetti, V.K.;
Journal of VLSI Signal Processing, Volume: 14 , Issue: 2 , Nov 1996
Pages: 24 – 31

29. VHDL-Based Rapid Systems Prototyping
Egolf, T.; Madisetti, V.K.;
Journal of VLSI Signal Processing, Volume: 14 , Issue: 2 , Nov 1996
Pages: 40-52

30. Interface design for core-based systems
Madisetti, V.K.; Lan Shen;
Design & Test of Computers, IEEE ,Volume: 14 , Issue: 4 , Oct.-Dec. 1997
Pages:42 - 51

31. Incorporating cost modeling in embedded-system design
Debardelaben, J.A.; Madisetti, V.K.; Gadient, A.J.;
Design & Test of Computers, IEEE ,Volume: 14 , Issue: 3 , July-Sept. 1997
Pages:24 – 35

32. On homomorphic deconvolution of bandpass signals
Marenco, A.L.; Madisetti, V.K.;
Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on] ,Volume: 45 , Issue: 10 , Oct. 1997
Pages:2499 – 2514

33. A case study in the development of multi-media educational material: the VHDL interactive tutorial
Gadient, A.J.; Stinson, J.A., Jr.; Taylor, T.C.; Aylor, J.H.; Klenke, R.H.; Salinas, M.H.; Madisetti, V.K.; Egolf, T.; Famorzadeh, S.; Karns, L.N.; Carter, H.W.;
Education, IEEE Transactions on ,Volume: 40 , Issue: 4 , Nov. 1997
Pages:17 pp.

Cloudflare, Inc. Exhibit 1003 Page 836

34. Adaptive mobility management in wireless networks
Jeongwook Kim; Madisetti, V.K.;
Electronics Letters ,Volume: 34 , Issue: 15 , 23 July 1998
Pages:1453 – 1455

35. Efficient implementation of two-band PR-QMF filterbanks
Hezar, R.; Madisetti, V.K.;
Signal Processing Letters, IEEE ,Volume: 5 , Issue: 4 , April 1998
Pages:92 – 94

36. On fast algorithms for computing the inverse modified discrete cosine transform
Yun-Hui Fan; Madisetti, V.K.; Mersereau, R.M.;
Signal Processing Letters, IEEE ,Volume: 6 , Issue: 3 , March 1999
Pages:61 – 64

37. System on chip or system on package?
Tummala, R.R.; Madisetti, V.K.;
Design & Test of Computers, IEEE ,Volume: 16 , Issue: 2 , April-June 1999
Pages:48 – 56

38. Reengineering legacy embedded systems
Madisetti, V.K.; Jung, Y.-K.; Khan, M.H.; Kim, J.; Finnessy, T.;
Design & Test of Computers, IEEE ,Volume: 16 , Issue: 2 , April-June 1999
Pages:38 – 47

39. Reengineering digital systems
Madisetti, V.K.;
Design & Test of Computers, IEEE ,Volume: 16 , Issue: 2 , April-June 1999
Pages:15 – 16

40. Parameter optimization of robust low-bit-rate video coders
Sangyoun Lee; Madisetti, V.K.;
Circuits and Systems for Video Technology, IEEE Transactions on, Volume: 9
Issue: 6 , Sept. 1999
Pages:849 – 855

41. Closed-form for infinite sum in bandlimited CDMA
Jatunov, L.A.; Madisetti, V.K.;
Communications Letters, IEEE ,Volume: 8 , Issue: 3 , March 2004
Pages:138 – 140

42. A new protocol to enhance path reliability and load balancing in mobile ad hoc networks
Argyriou, A.; Madisetti, V.K.;

Journal of Ad Hoc Networks, Elsevier Press, 2004

43. Closed-form analysis of CDMA systems using Nyquist pulse
Jatunov, L.A.; Madisetti, V. K.;
Communications Letters, IEEE (Under Revision), 2005.

44. Systematic Design of End-to-End Wireless Mobility Management Prototocols,
Argyriou, A.; Madisetti, V. K.;
ACM/Springer Wireless Networks (WINET), Accepted 2005.

45. A Novel End-to-End Approach for Video Streaming Over the Internet,
Argyriou, A.; Madisetti, V. K.;
Kluwer Telecommunications Systems, Vol. 28, No. 2, Pages 133-150, Jan 2005. *Special Issue on Multimedia Streaming.*

46. An Analytical Framework of RD Optimized Video Streaming with TCP,
Argyriou, A.; Madisetti, V. K.;
IEEE Transactions on Multimedia, Submitted for review in March 2005.

47. Modeling the Effect of Handoffs on Transport Protocol Performance,
Argyriou, A.; Madisetti, V. K.;
IEEE Transactions on Mobile Computing, Submitted for review in March 2005

48. Throughput Models for Transport Protocols with CBR and VBR Traffic Workloads",
Argyriou, A.; Madisetti, V. K.;
ACM Transactions on Multimedia Computing, Communications & Applications, Submitted for review in April 2005.

49. "Electronic System, Platform & Package Codesign",
Madisetti, V. K.
IEEE Design & Test of Computers, Volume 23, Issue 3, June 2006. pages 220-233.

50. "The Design of an End-to-End Handoff Management Protocol",
A. Argyriou, Madisetti, V. K.
Wireless Networks, Springer, May 2006.

51."A Soft-Handoff Transport Protocol for Media Flows in Heterogeneous Mobile Networks ",
A. Argyriou, Madisetti, V. K.
Computer Networks, Vol 50, Issue 11, Pages 1860-1871, August 2006.

52. "Computationally Efficient SNR Estimation for Bandlimited WCDMA Systems"
L. Jatunov, Madisetti, V. K.
IEEE Transactions on Wireless Communications, Volume 5, Issue 13, December 2006, Pages 3480-3491.

53."Space-Time Codes for Wireless & Mobile Applications",
M. Sinnokrot, Madisetti, V.K.
DSP Handbook, Second Edition, 2009 (to be published)

54.A. Bahga, V. Madisetti, "**Rapid Prototyping of Advanced Cloud-Based Systems**", *IEEE Computer*, vol. 46, no. 11, Nov 2013, pp 76-83, 2013

55.A. Bahga, V. Madisetti, "**Cloud-Based Information Integration & Informatics Framework for Healthcare Applications**", *IEEE Computer*, February 2015.

56.A. Bahga, V. Madisetti, "**A Cloud-based Approach for Interoperable EHRs**", *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 5, Sep 2013, pp. 894-906, 2013

57.A. Bahga, V. Madisetti, "**Cloud-Based Information Technology Framework for Data Driven Intelligent Transportation Systems**", *Journal of Transportation Technologies*, vol.3 no.2, April 2013

58.A. Bahga, V. Madisetti, **"Performance Evaluation Approach for Multi-tier Cloud Applications"**, *Journal of Software Engineering and Applications*, vol. 6, no. 2, pp. 74-83, Mar 2013.

59. Yusuf, A., V. Madisetti, **"Configuration for Predicting Travel Time Using Wavelet Packets and Support Vector Regression"**, ***Journal of Transportation Technologies***, vol 3, no. 3, June 2013.

60.A. Bahga, V. Madisetti, **"Analyzing Massive Machine Maintenance Data in a Computing Cloud"**, *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 10, pp. 1831 - 1843, 2012.

61.N. Radia. Y. Zhang, M. Tatimapula, V. Madisetti, **"Next Generation Applications on Cellular Networks: Trends, Challenges, and Solutions**," *Proceedings of the IEEE*, Vol 100, Issue 4, pp. 841-854, 2012.

62.A. Bahga, V. Madisetti, "**Rapid Prototyping of Advanced Cloud-Based Systems**", *IEEE Computer*, vol. 46, no. 11, Nov 2013, pp 76-83, 2013

63. A. Bahga, V. Madisetti, "**A Cloud-based Approach for Interoperable EHRs**", *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 5, Sep 2013, pp. 894-906, 2013

64. A. Bahga, V. Madisetti, "**Cloud-Based Information Integration & Informatics Framework for Healthcare Applications**", *IEEE Computer*, February 2015.

Peer Reviewed Conference Publications

1. **Dynamically-reduced complexity implementation of echo cancelers**
   Madisetti, V.; Messerschmitt, D.; Nordstrom, N.;
   Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86. ,Volume: 11 , Apr 1986

2. **Seismic migration algorithms using the FFT approach on the NCUBE multiprocessor**
   Madisetti, V.K.; Messerschmitt, D.G.;
   Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on , 11-14 April 1988

3. **Seismic migration algorithms on multiprocessors**
   Madisetti, V.K.; Messerschmitt, D.G.;
   Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on , 11-14 April 1988
   Pages:2124 - 2127 vol.4

4. **WOLF: A rollback algorithm for optimistic distributed simulation systems**
   Madisetti, V.; Walrand, J.; Messerschmitt, D.;
   Simulation Conference Proceedings, 1988 Winter , December 12-14, 1988
   Pages:296 – 305

5. **Efficient distributed simulation**
   Madisetti, V.; Walrand, J.; Messerschmitt, D.;
   Simulation Symposium, 1989. The 22nd Annual , March 28-31, 1989
   Pages:5 - 6

6. **High speed migration of multidimensional seismic data**
   Kelley, B.; Madisetti, V.;

Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on , 14-17 April 1991
Pages:1117 - 1120 vol.2

7. **Performance of a fast analog VLSI implementation of the DFT**
Buchanan, B.; Madisetti, V.; Brooke, M.;
Circuits and Systems, 1992., Proceedings of the 35th Midwest Symposium on , 9-12 Aug. 1992
Pages:1353 - 1356 vol.2

8. **Task scheduling in the Georgia Tech digital signal multiprocessor**
Curtis, B.A.; Madisetti, V.K.;
Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on ,Volume: 5 , 23-26 March 1992
Pages:589 - 592 vol.5

9. **The fast discrete Radon transform**
Kelley, B.T.; Madisetti, V.K.;
Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on ,Volume: 3 , 23-26 March 1992
Pages:409 - 412 vol.3

10. **Yield-based system partitioning strategies for MCM and ASEM design**
Khan, S.; Madisetti, V.;
Multi-Chip Module Conference, 1994. MCMC-94, Proceedings., 1994 IEEE,15-17 March 1994
Pages:144 – 149

11. **Multitrack RLL codes for the storage channel with immunity to intertrack interference**
Lee, J.; Madisetti, V.K.;
Global Telecommunications Conference, 1994. GLOBECOM '94. 'Communications: The Global Bridge'., IEEE,Volume: 3 , 28 Nov.-2 Dec. 1994
Pages:1477 - 1481 vol.3

12. **A parallel mapping of backpropagation algorithm for mesh signal processor**
Khan, S.A.; Madisetti, V.K.;
Neural Networks for Signal Processing [1995] V. Proceedings of the 1995 IEEE Workshop , 31 Aug.-2 Sept. 1995
Pages:561 – 570

13. **Virtual prototyping of embedded DSP systems**
Madisetti, V.K.; Egolf, T.; Famorzadeh, S.; Dung, L.-R.;
Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995

International Conference on ,Volume: 4 , 9-12 May 1995
Pages:2711 - 2714 vol.4

14. Assessing and improving current practice in the design of application-specific signal processors
Shaw, G.A.; Anderson, J.C.; Madisetti, V.K.;
Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on ,Volume: 4 , 9-12 May 1995
Pages:2707 - 2710 vol.4

15. Introduction to ARPA's RASSP initiative and education/facilitation program
Corley, J.H.; Madisetti, V.K.; Richards, M.A.;
Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on ,Volume: 4 , 9-12 May 1995
Pages:2695 - 2698 vol.4

16. DSP design education at Georgia Tech
Madisetti, V.K.; McClellan, J.H.; Barnwell, T.P., III;
Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on ,Volume: 5 , 9-12 May 1995
Pages:2869 - 2872 vol.5

17. Rapid prototyping of DSP systems via system interface module generation
Famorzadeh, S.; Madisetti, V.K.;
Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on ,Volume: 2 , 7-10 May 1996
Pages:1256 - 1259 vol. 2

18. Rapid prototyping of DSP chip-sets via functional reuse
Romdhane, M.S.B.; Madisetti, V.K.;
Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on ,Volume: 2 , 7-10 May 1996
Pages:1236 - 1239 vol. 2

19. A constructive deconvolution procedure of bandpass signals by homomorphic analysis
Marenco, A.L.; Madisetti, V.K.;
Geoscience and Remote Sensing Symposium, 1996. IGARSS '96. 'Remote Sensing for a Sustainable Future.', International ,Volume: 3 , 27-31 May 1996
Pages:1592 - 1596 vol.3

20. BEEHIVE: an adaptive, distributed, embedded signal processing environment
Famorzadeh, S.; Madisetti, V.; Egolf, T.; Nguyen, T.;
Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on ,Volume: 1 , 21-24 April 1997
Pages:663 - 666 vol.1

21. Target detection from coregistered visual-thermal-range images
Perez-Jacome, J.E.; Madisetti, V.K.;
Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on ,Volume: 4 , 21-24 April 1997
Pages:2741 - 2744 vol.4

22. Variable block size adaptive lapped transform-based image coding
Klausutis, T.J.; Madisetti, V.K.;
Image Processing, 1997. Proceedings., International Conference on ,Volume: 3 , 26-29 Oct. 1997
Pages:686 - 689 vol.3

23. A Rate 8/10 (0, 6) MTR Code And Its Encoder/decoder
Jaejin Lee; Madisetti, V.K.;
Magnetics Conference, 1997. Digests of INTERMAG '97., 1997 IEEE International , 1-4 April 1997
Pages:BS-15 - BS-15

24. VHDL models supporting a system-level design process: a RASSP approach
DeBardelaben, J.A.; Madisetti, V.K.; Gadient, A.J.;
VHDL International Users' Forum, 1997. Proceedings , 19-22 Oct. 1997
Pages:183 – 188

25. A performance modeling framework applied to real time infrared search and track processing
Pauer, E.K.; Pettigrew, M.N.; Myers, C.S.; Madisetti, V.K.;
VHDL International Users' Forum, 1997. Proceedings , 19-22 Oct. 1997
Pages:33 – 42

26. System design and re-engineering through virtual prototyping: a temporal model-based approach
Khan, M.H.; Madisetti, V.K.;
Signals, Systems & Computers, 1998. Conference Record of the Thirty-Second Asilomar Conference on ,Volume: 2 , 1-4 Nov. 1998
Pages:1720 - 1724 vol.2

Cloudflare, Inc. Exhibit 1003 Page 843

27. A debugger RTOS for embedded systems
Akgul, T.; Kuacharoen, P.; Mooney, V.J.; Madisetti, V.K.;
Euromicro Conference, 2001. Proceedings. 27th , 4-6 Sept. 2001
Pages:264 - 269

28. Adaptability, extensibility and flexibility in real-time operating systems
Kuacharoen, P.; Akgul, T.; Mooney, V.J.; Madisetti, V.K.;
Digital Systems, Design, 2001. Proceedings. Euromicro Symposium on , 4-6 Sept. 2001
Pages:400 – 405

29. **Effect of handoff delay on the system performance of TDMA cellular systems**
Turkboylari, M.; Madisetti, V.K.;
Mobile and Wireless Communications Network, 2002. 4th International Workshop on , 9-11 Sept. 2002
Pages:411 – 415

30. Enforcing interdependencies and executing transactions atomically over autonomous mobile data stores using SyD link technology
Prasad, S.K.; Bourgeois, A.G.; Dogdu, E.; Sunderraman, R.; Yi Pan; Navathe, S.; Madisetti, V.;
Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on , 19-22 May 2003
Pages:803 – 809

31. Performance evaluation and optimization of SCTP in wireless ad-hoc networks
Argyriou, A.; Madisetti, V.;
Local Computer Networks, 2003. LCN '03. Proceedings. 28th Annual IEEE International Conference on , 20-24 Oct. 2003
Pages:317 - 318

32. Implementation of a calendar application based on SyD coordination links
Prasad, S.K.; Bourgeois, A.G.; Dogdu, E.; Sunderraman, R.; Yi Pan; Navathe, S.; Madisetti, V.;
Parallel and Distributed Processing Symposium, 2003. Proceedings. International , 22-26 April 2003
Pages:8 pp.

33. Bandwidth aggregation with SCTP
Argyriou, A.; Madisetti, V.;
Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE Volume: 7 , 1-5 Dec. 2003
Pages:3716 - 3721 vol.7

Cloudflare, Inc. Exhibit 1003 Page 844

34. Software streaming via block streaming
Kuacharoen, P.; Mooney, V.J.; Madisetti, V.K.;
Design, Automation and Test in Europe Conference and Exhibition, 2003 , 2003
Pages:912 – 917

35. Frequency-dependent space-interleaving for MIMO OFDM systems
Mohajerani, P.; Madisetti, V.K.;
Radio and Wireless Conference, 2003. RAWCON '03. Proceedings , Aug. 10-13, 2003
Pages:79 - 82

36. A media streaming protocol for heterogeneous wireless networks
Argyriou, A.; Madisetti, V.;
Computer Communications, 2003. CCW 2003. Proceedings. 2003 IEEE 18th Annual Workshop on , 20-21 Oct. 2003
Pages:30 – 33

37. Realizing load-balancing in ad-hoc networks with a transport layer protocol
Argyriou, A.; Madisetti, V.;
Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE ,Volume: 3 , 21-25 March 2004
Pages:1897 - 1902 Vol.3

38.Streaming H.264/AVC video over the Internet
Argyriou, A.; Madisetti, V.;
Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE , 5-8 Jan. 2004
Pages:169 – 174

Cloudflare, Inc. Exhibit 1003 Page 845

Other Publications

1. **A Transport Layer Technology for Improving the QoS of Networked Multimedia Applications <draft-madisetti-arguriou-qos-sctp-00.txt).**
   Madisetti, V., Argyriou, A.
   IETF Internet-Draft, Jul 25, 2002.

2. **Voice & Video over Mobile IP Networks <draft-madisetti-argyriou-voice-video-mip-00.txt>**
   Madisetti, V., Argyriou, A.
   IETF Internet-Draft, Nov 20, 2002.

3. **Enhancements to ECRTP with Applications to Robust Header Compression for Wireless Applications. <draft-madisetti-rao-suresh-rohc-00.txt>**
   Madisetti, V.; Rao, S., Suresh, N.
   IETF Internet-Draft, June 30, 2003.

Ph.D. Students Graduated

1. **Brian T. Kelley, 1992**
   VLSI Computing Architectures for High Speed Signal Processing
   Member of Technical Staff, Motorola.

   Winner of Dr. Thurgood Marshall Dissertation Fellowship Award

2. **Bryce A. Curtis, 1992**
   Special Instruction Set Multiple Chip Computer for DSP
   Member of Technical Staff, IBM

3. **Jaejin Lee, 1994**
   Robust Multitrack Codes for the Magnetic Channel
   Professor, Yonsei University, Korea

4. **Mohamed S. Ben Romdhane, 1995**
   Design Synthesis of Application-Specific IC for DSP
   Director of IP, Rockwell.

5. **Shoab A. Khan, 1995**
   Logic and Algorithm Partitioning on MCMs
   Professor, National University of Science & Technology, Pakistan

6. **Lan-Rong Dung, 1997**
   VHDL-based Conceptual Prototyping of Embedded DSP Architectures
   Professor, National Chaio Tung University, Taiwan.

Cloudflare, Inc. Exhibit 1003 Page 846

Winner of VHDL International Best PhD Thesis Award, 1997

7. **Thomas W. Egolf, 1997**
   Virtual Prototyping of Embedded DSP Systems
   Distinguished Member of Technical Staff, Agere

8. **Alvaro Marenco, 1997**
   On Homomorphic Deconvolution of Bandpass Signals
   Professor, Texas A&M University.

   Winner of GIT ECE Outstanding Teaching Assistant Award

9. **Shahram Famorzadeh, 1997**
   BEEHIVE: A Distributed Environment for Adaptive Signal Processing
   Member of Technical Staff, Rockwell.

10. **Timothy J. Klausutis, 1997**
    Adaptive Lapped Transforms with Applications to Image Coding.
    US Air Force/Univ. of Florida.

11. **Lan Shen, 1998**
    Temporal Design of Core-Based Systems
    Member of Technical Staff, IBM

12. **James DeBardelaben, 1998**
    Optimization Based Approach to Cost Effective DSP Design
    Research Scientist, Johns Hopkins University

    Georgia Tech ECE Faculty Award

13. **Sangyoun Lee, 1999**
    Design of Robust Video Signal Processors
    Professor, Yonsei University

    US Army Sensors Lab Research Excellence Award, 1999

14. **Rahmi Hezar, 2000**
    Oversampled Digital Filters
    Member of Technical Staff, Texas Instruments

15. **Yong-kyu Jung, 2001**
    Model-Based Processor Synthesis
    Professor, Texas A&M University

16. **Mustafa Turkboylari, 2002**
    Handoff Algorithms for Wireless Applications
    Member of Technical Staff, Texas Instruments

17. **Yun-Hui Fan, 2002**
    A Stereo Audio Coder with Nearly Constant Signal to Noise Ratio
    Post-Doctoral Research Associate, Northeastern University

18. **Subrato K. De, 2002**

Cloudflare, Inc. Exhibit 1003 Page 847

Design of a Retargetable Compiler for DSP
Member of Technical Staff, Qualcomm

US Army Sensors Lab Research Excellence Award, 1999

19. Chonlameth Aripnikanondt, 2004
System-on-Chip Design with UML
Professor, King Mongkut's University, Thailand.

US Army Sensors Lab Research Excellence Award, 1999

20. Loran Jatunov, 2004
Performance Analysis of 3G CDMA Systems
Senior Research Scientist, Soft Networks, LLC.

21. Antonios Argyriou, 2005, Serving in Hellenic Army.

22. Pilho Kim, 2009, Scientist, VP Technologies, Inc.

23. M. Sinnokrot, 2009, Staff Engineer, Qualcomm.

Awards & Honors

1. **Jagasdis Bose National Science Talent Fellowship**, Indian Institute of Technology, Kharagpur, 1980-1984.

2. **General Proficiency Prize**, Indian Institute of Technology, Kharagpur, 1984.

3. **Demetri Angelakos Outstanding Graduate Student Award**, Univ. of California, Berkeley, 1989

4. **Ira Kay IEEE/ACM Best Paper Award** for Best Paper presented at IEEE Annual Simulation Symposium, 1989

5. **IBM Faculty Development Award** 1990

6. **Technical Program Chair**, IEEE Workshop on Parallel and Distributed Simulation. 1990.

7. **Technical Program Chair**, IEEE MASCOTS'94

8. **NSF RI Award**, 1990

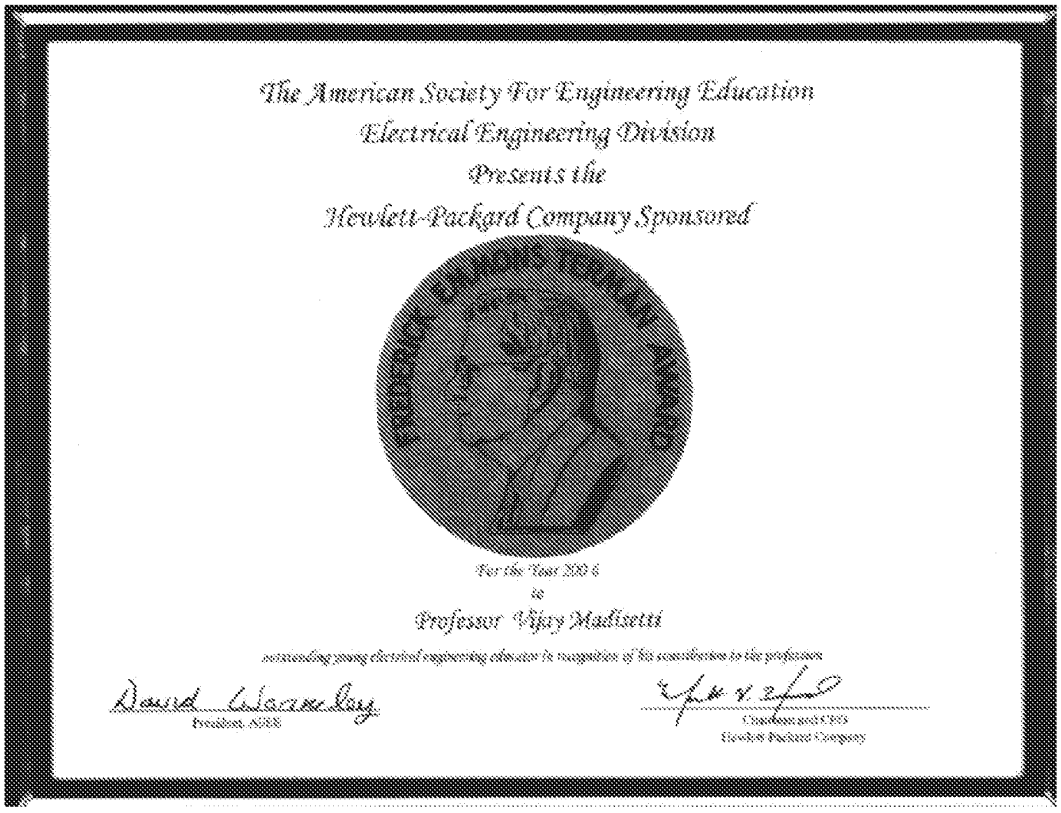
9. **VHDL International Best PhD Dissertation Advisor Award**, 1997

Cloudflare, Inc. Exhibit 1003 Page 848

10. **Georgia Tech Outstanding Doctoral Dissertation Advisor Award**, 2001.

11. **ASEE 2006 Frederick Emmons Terman Medal, 2006.**

12. **Fellow of IEEE**



Intellectual Property Disclosures (Georgia Tech)

Patent	Date	Description
2843	2004	Method and Apparatus for Improving the Performance of Wireless LANs
2825	2003	Method and Apparatus for Optimal Partitioning and Ordering of Antennas for Layered Space-Time Block Codes in MIMO Communications Systems
2815	2003	How to Rapidly Develop a SyD Application
GSU-023	2003	Rapid Development of SyD Applications

2810	2003	System on Mobile Devices Middleware Design
2718	2003	A Transport Layer Algorithm for Improved Anycast Communication
2717	2003	A Novel Transport Layer Load-Balancing Algorithm
2716	2003	A Transport Layer QoS Algorithm
2715	2003	A Novel Transport Layer Algorithm for MPLS Performance
2659	2002	A New Algorithm and Technology for Implementing Mobile IP with Applications to Voice and Video over Mobile IP
2656	2002	Debugging with Instruction-Level Reverse Execution
2655	2002	Embedded Software Streaming
2539		System of Databases: An Enabling Technology for Programming
2517	2002	A Dynamic Instantiated Real-Time Operating System Debugger
2516	2002	A Dynamic Real-Time Operating System
GSU-009	2001	System of Databases: Architecture,, Global Queries, Triggers and Constraints
2480	2001	Mobile Fleet Application based on SyD Technology
2479	2001	System of Databases: A model with coordination links and a calendar application
1893	1999	Beehive
1726	1995	Very High Scale Integrated Circuit Hardware Description Language Models (VHDL Models)
1401	1995	Self-Compensation Receiver (SCR)

Issued Patents: 9,935,772, 9,769,213

Cloudflare, Inc. Exhibit 1003 Page 850

Exhibit 1006

U.S. Patent 7,058,974 by Robert D. Maher, III ("Maher")

Customer No.: 000027683

Haynes and Boone, LLP
IP Section
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Telephone [214] 651.5000
Fax [214] 200.0853

US007058974B1

(12) **United States Patent**
Maher, III et al.

(10) **Patent No.: US 7,058,974 B1**
(45) **Date of Patent: Jun. 6, 2006**

(54) **METHOD AND APPARATUS FOR PREVENTING DENIAL OF SERVICE ATTACKS**

(75) Inventors: **Robert Daniel Maher, III**, Plano, TX (US); **Victor A. Bennett**, Rockwall, TX (US)

(73) Assignee: **Netrake Corporation**, Plano, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1030 days.

(21) Appl. No.: **09/598,631**

(22) Filed: **Jun. 21, 2000**

(51) **Int. Cl.**
G06F 11/00 (2006.01)
G06F 11/22 (2006.01)
G06F 11/30 (2006.01)
G06F 11/32 (2006.01)

(52) **U.S. Cl.** **726/13**; 726/23; 713/189; 370/229

(58) **Field of Classification Search** 713/201, 713/200, 154; 706/47; 370/474; 726/22, 726/23–26

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,813,001 A 9/1998 Bennett
6,477,669 B1 * 11/2002 Agarwal et al. 714/708
6,598,034 B1 * 7/2003 Kloth 706/47
6,636,512 B1 * 10/2003 Lorrain et al. 370/392
6,654,373 B1 * 11/2003 Maher et al. 370/392
6,735,219 B1 * 5/2004 Clauberg 370/474

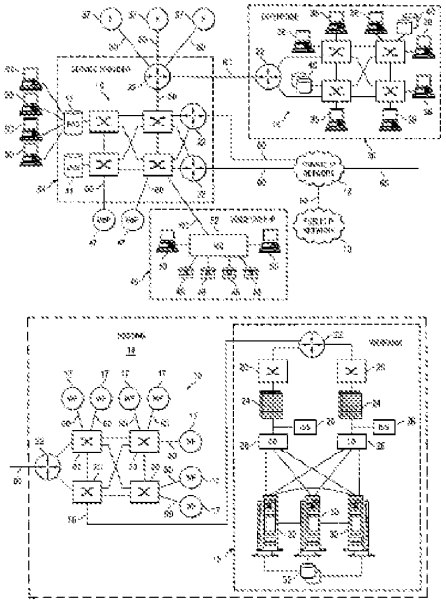
* cited by examiner

Primary Examiner—Gilberto Barron, Jr.
Assistant Examiner—Samson Lemma
(74) *Attorney, Agent, or Firm*—Haynes and Boone, LLP

(57) **ABSTRACT**

A method and apparatus for preventing denial of service type attacks on data networks is described. The method involves scanning the contents of the data packets flowing over the data network using a traffic flow scanning engine. The data packets are reordered and reassembled and then the payload contents are scanned to determine whether they conform to predetermined requirements. Data packets which do not reorder or reassemble correctly or which do not conform to the predetermined requirements may be dropped. Dropping packets which do not reorder or reassemble correctly or which do not conform to the predetermined requirements prevent denial of service attack which exploit bugs in the TCP/IP implementation or shortcomings in the TCP/IP specification. The traffic flow scanning engine is further operable to determine whether the data packets are associated with validated traffic flows. Those data packets associated with validated traffic flows are assigned to a higher priority while those not associated with a validated traffic flow are assigned to a low priority, which may occupy no more that a predetermined maximum of the available bandwidth. Assigning data packets associated with a non-validated traffic flow to a low priority prevent brute force type denial of service attacks designed to clog networks.

16 Claims, 6 Drawing Sheets



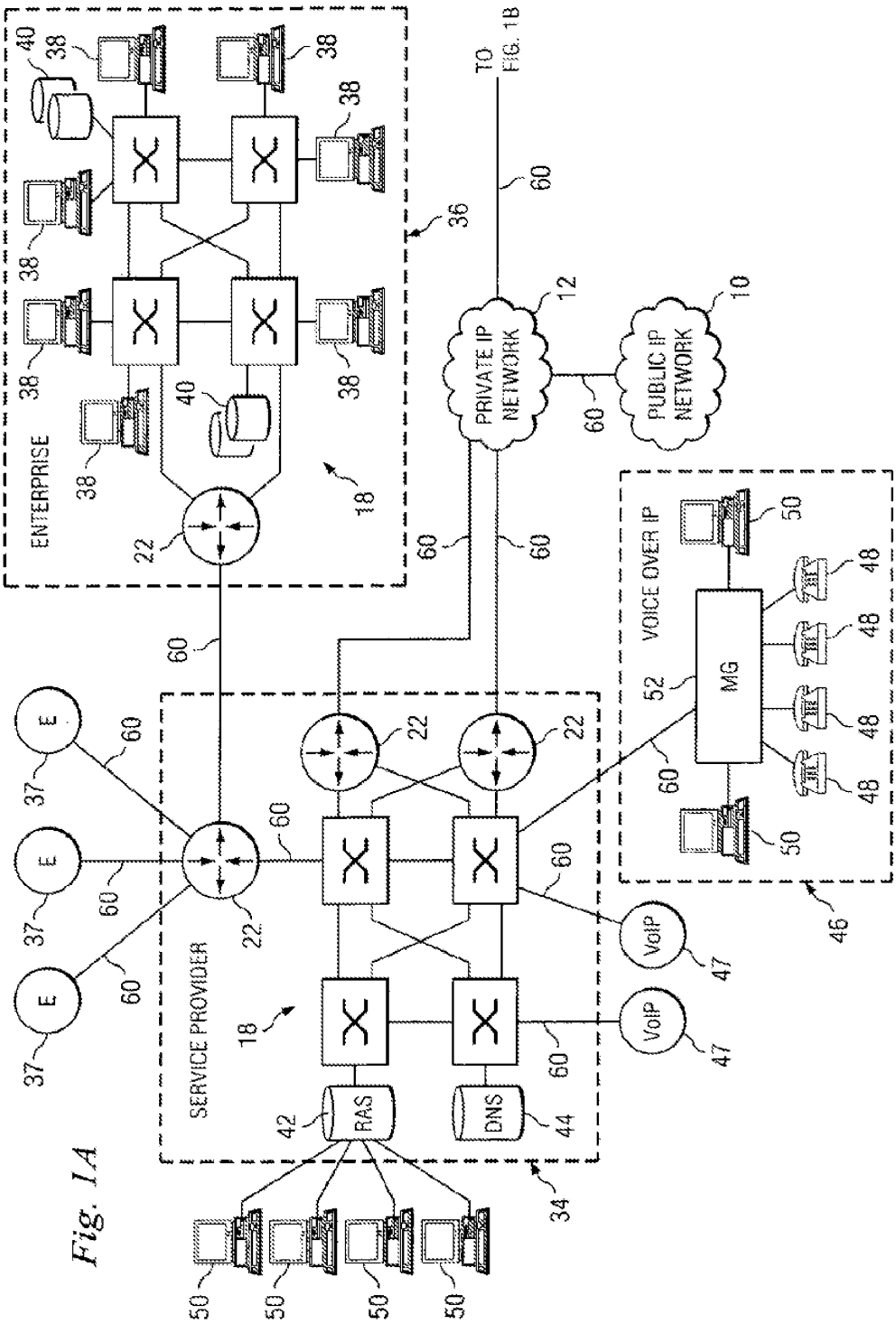

ENTERPRISE
SERVICE PROVIDER
VOICE OVER IP
PRIVATE IP NETWORK
PUBLIC IP NETWORK
TO FIG. 1B
RAS
DNS
MG
VoIP

Fig. 1A

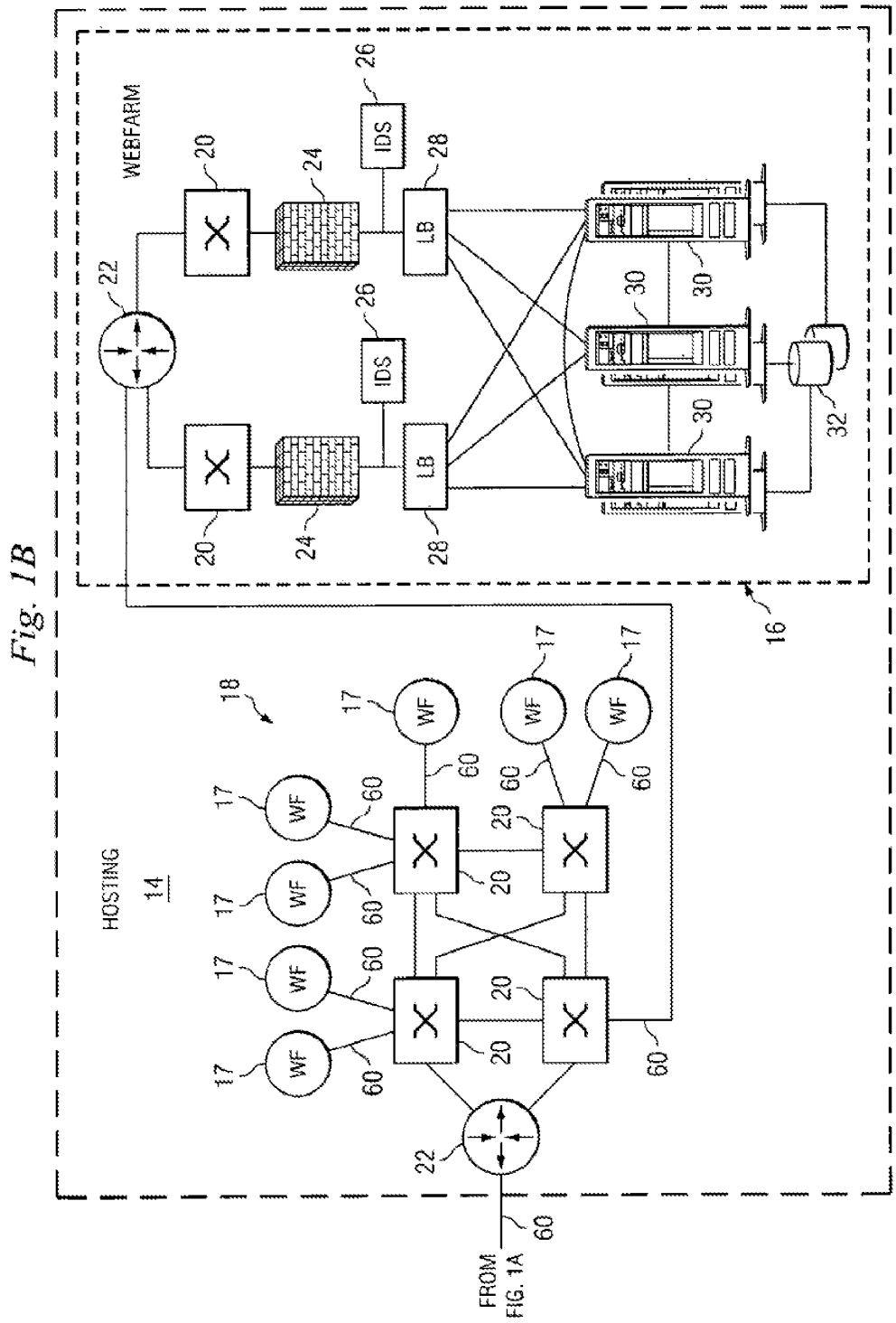

WEBFARM
HOSTING
14
16
18
20
22
24
26
28
30
32
60
IDS
LB
WF
FROM FIG. 1A

Fig. 1B

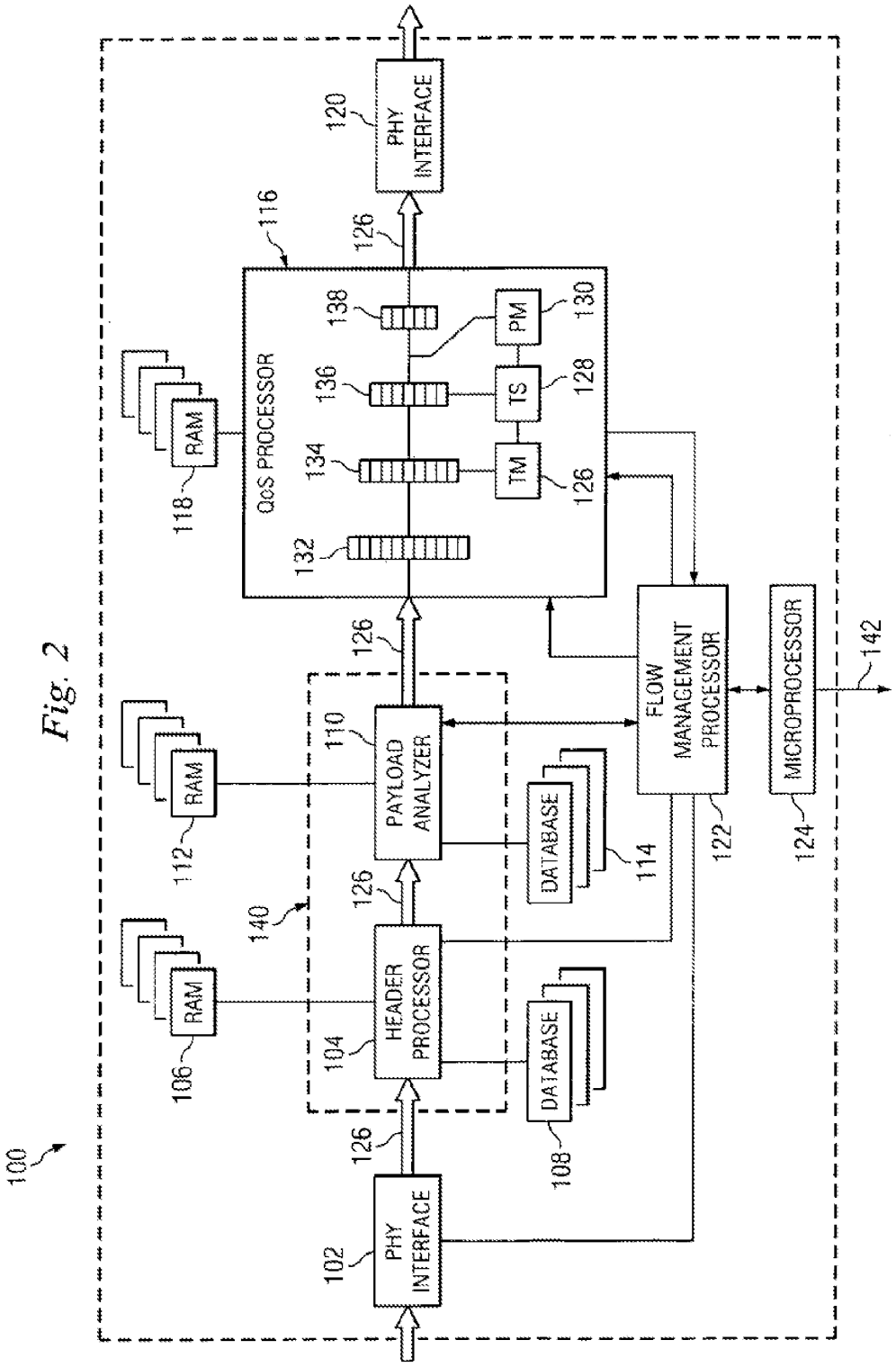

100
102
PHY INTERFACE
104
HEADER PROCESSOR
106
RAM
108
DATABASE
110
PAYLOAD ANALYZER
112
RAM
114
DATABASE
116
QoS PROCESSOR
118
RAM
120
PHY INTERFACE
122
FLOW MANAGEMENT PROCESSOR
124
MICROPROCESSOR
126
126
126
126
126
128
TS
130
PM
132
134
TM
136
138
140
142

Fig. 2

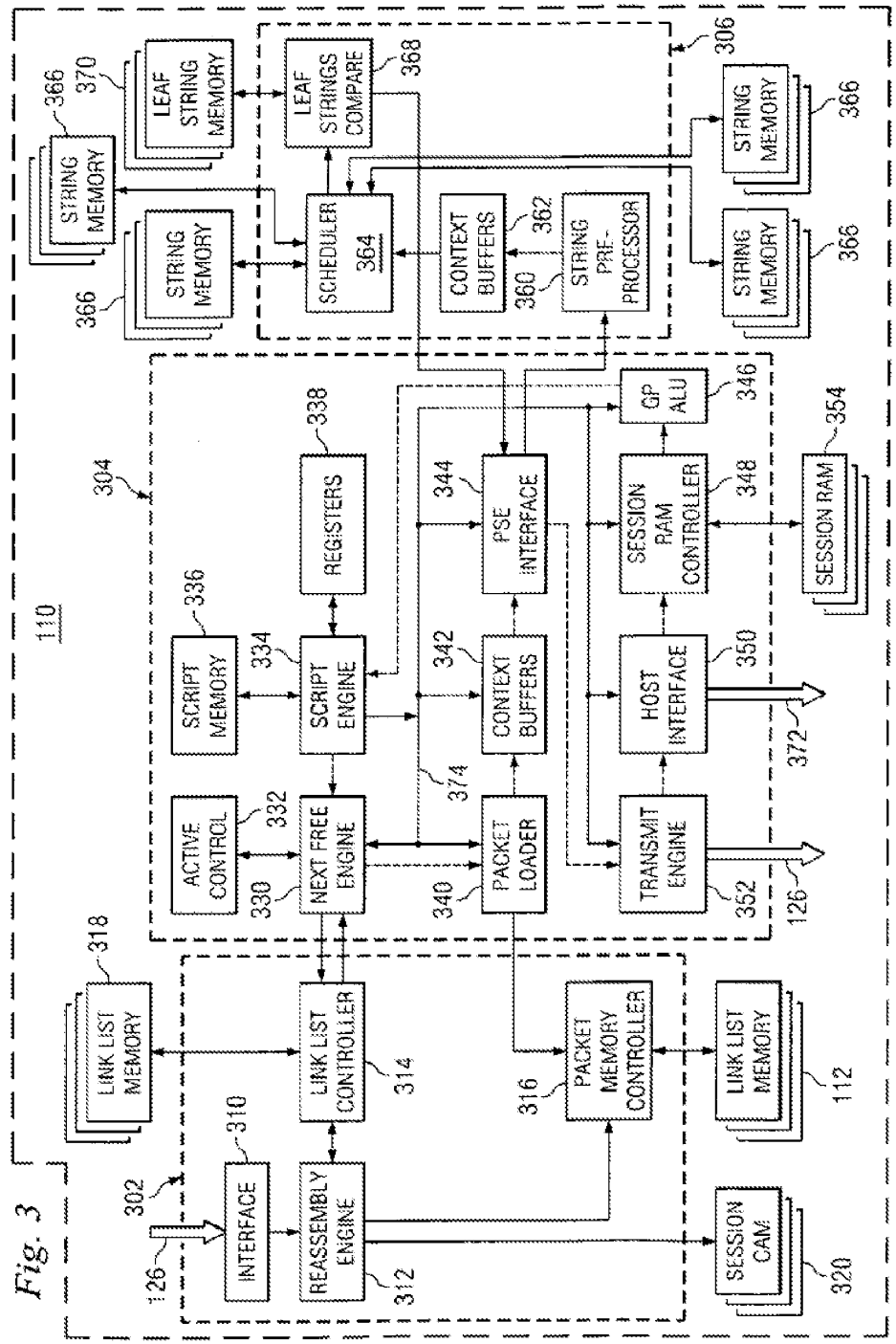

110
126
302
INTERFACE
310
REASSEMBLY ENGINE
312
LINK LIST CONTROLLER
314
LINK LIST MEMORY
318
PACKET MEMORY CONTROLLER
316
LINK LIST MEMORY
112
SESSION CAM
320
304
ACTIVE CONTROL
332
330
NEXT FREE ENGINE
SCRIPT MEMORY
336
SCRIPT ENGINE
334
REGISTERS
338
PACKET LOADER
340
CONTEXT BUFFERS
342
PSE INTERFACE
344
374
TRANSMIT ENGINE
352
HOST INTERFACE
350
372
126
SESSION RAM CONTROLLER
348
GP ALU
346
SESSION RAM
354
306
LEAF STRINGS COMPARE
368
LEAF STRING MEMORY
370
366
STRING MEMORY
SCHEDULER 364
CONTEXT BUFFERS
360
362
STRING PRE-PROCESSOR
STRING MEMORY
366
STRING MEMORY
366
STRING MEMORY
366

Fig. 3

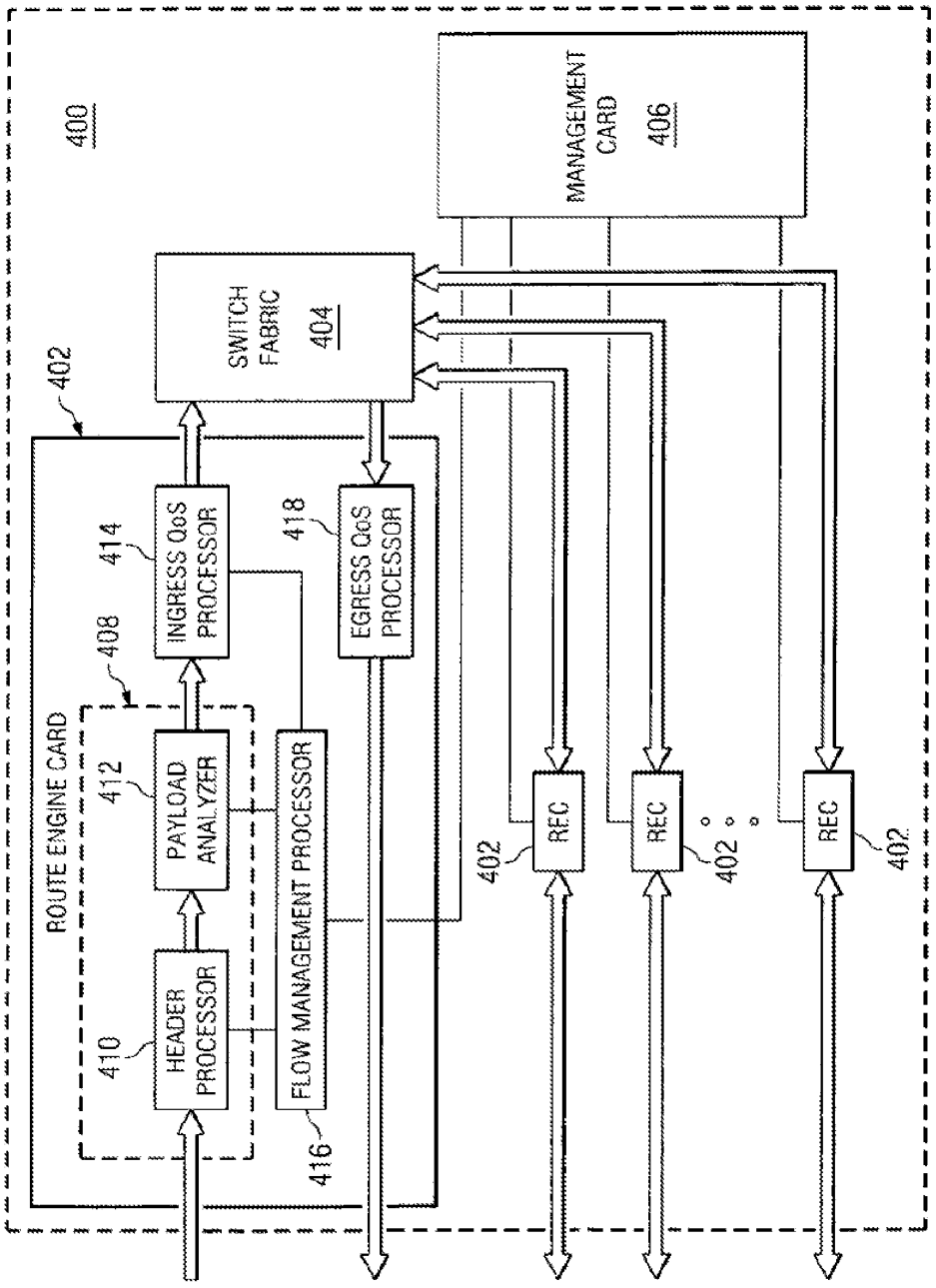
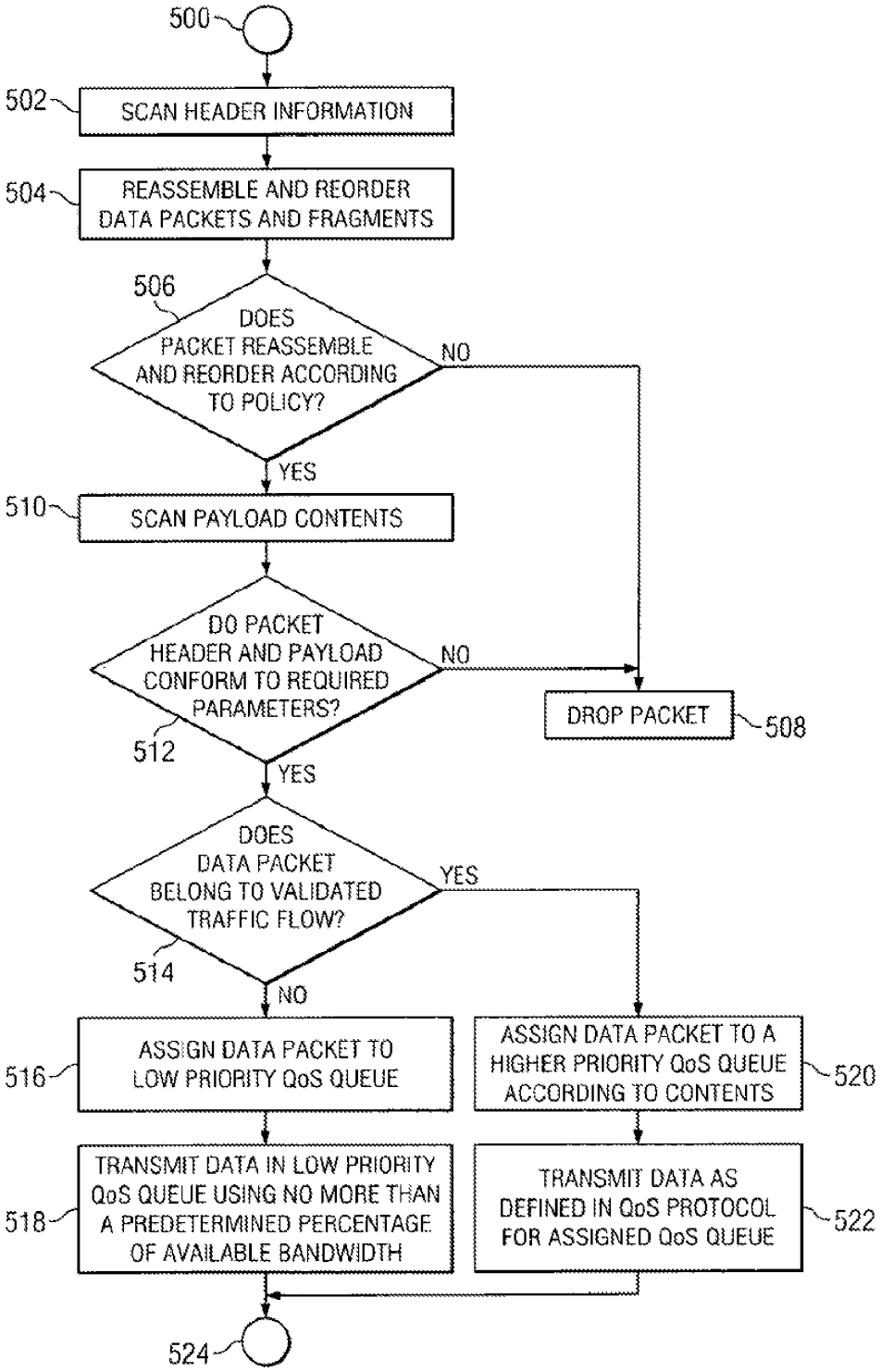

400
MANAGEMENT CARD
406
SWITCH FABRIC
404
402
ROUTE ENGINE CARD
408
INGRESS QoS PROCESSOR
414
EGRESS QoS PROCESSOR
418
PAYLOAD ANALYZER
412
HEADER PROCESSOR
410
FLOW MANAGEMENT PROCESSOR
416
REC
REC
REC
402
402
402

Fig. 4

Fig. 5


500
502
SCAN HEADER INFORMATION
504
REASSEMBLE AND REORDER DATA PACKETS AND FRAGMENTS
506
DOES PACKET REASSEMBLE AND REORDER ACCORDING TO POLICY?
NO
YES
510
SCAN PAYLOAD CONTENTS
DO PACKET HEADER AND PAYLOAD CONFORM TO REQUIRED PARAMETERS?
NO
512
DROP PACKET
508
YES
DOES DATA PACKET BELONG TO VALIDATED TRAFFIC FLOW?
YES
514
NO
516
ASSIGN DATA PACKET TO LOW PRIORITY QoS QUEUE
ASSIGN DATA PACKET TO A HIGHER PRIORITY QoS QUEUE ACCORDING TO CONTENTS
520
518
TRANSMIT DATA IN LOW PRIORITY QoS QUEUE USING NO MORE THAN A PREDETERMINED PERCENTAGE OF AVAILABLE BANDWIDTH
TRANSMIT DATA AS DEFINED IN QoS PROTOCOL FOR ASSIGNED QoS QUEUE
522
524

METHOD AND APPARATUS FOR PREVENTING DENIAL OF SERVICE ATTACKS

TECHNICAL FIELD OF THE INVENTION

The present invention relates to broadband data networking equipment. Specifically, the present invention relates to a method and device that prevents denial of service type internet attacks.

BACKGROUND OF THE INVENTION

A "denial of service" (DoS) attack is characterized by an explicit attempt by attackers to prevent legitimate users of a service from using that service. DoS attacks are aimed at devices and networks with exposure to the Internet. Their goal is to cripple a device or network so that external users no longer have access to network resources. Without hacking password files or stealing sensitive data, a denial-of-service hacker simply fires up a program that will generate enough traffic to a particular site that it denies service to the site's legitimate users.

There are three types of DoS attacks: those that exploit a bug in a TCP/IP implementation, those that exploit a shortcoming in the TCP/IP specification, and brute-force attacks that clog up the network with so much useless traffic that no other traffic can get in or out.

Two lethal attacks, the well-known Ping of Death and the newer Teardrop attack, exploit known bugs in TCP/IP implementations. The Ping of Death uses a ping system utility to create an IP packet that exceeds the maximum 65,536 bytes of data allowed by the IP specification. The oversize packet is then sent to an unsuspecting system. Systems may crash, hang, or reboot when they receive such a maliciously crafted packet.

The recently developed Teardrop attack exploits weaknesses in the reassembly of IP packet fragments. During its journey through the Internet, an IP packet may be broken up into smaller chunks. Each fragment looks like the original IP packet except that it contains an offset field that says, for instance, "This fragment is carrying bytes **600** through **800** of the original (nonfragmented) IP packet." The Teardrop program creates a series of IP fragments with overlapping offset fields. When these fragments are reassembled at the destination host, some systems will crash, hang, or reboot.

Weaknesses in the TCP/IP specification leave hosts open to SYN attacks, executed during the three-way handshake that kicks off a TCP conversation between two applications. Under normal circumstances, the application that initiates a TCP session sends a TCP SYN synchronization packet to the receiving application. The receiver sends back a TCP SYN-ACK acknowledgment packet and then the initiator responds with an ACK acknowledgment. After this handshake, the applications are set to send and receive data.

But a SYN attack floods a targeted system with a series of TCP SYN packets. Each packet causes the targeted system to issue a SYN-ACK response. While the targeted system waits for the ACK that follows the SYN-ACK, it queues up all outstanding SYN-ACK responses on what is known as a backlog queue. This backlog queue has a finite length that is usually quite small. Once the queue is full, the system will either ignore all incoming SYN requests, or more likely crash. SYN-ACKs are moved off the queue only when an ACK comes back or when an internal timer (which is set at relatively long intervals) terminates the three-way handshake.

A SYN attack creates each SYN packet in the flood with a bad source IP address, which under routine procedure identifies the original packet. All responses are sent to the source IP address. But a bad source IP address either does not actually exist or is down; therefore the ACK that should follow a SYN-ACK response will never come back. This creates a backlog queue that's always full, making it nearly impossible for legitimate TCP SYN requests to get into the system.

In a Land attack—a simple hybrid of the SYN attack—hackers flood SYN packets into the network with a spoofed source IP address of the targeted system.

A lot more dangerous than any initiative launched by their cartoon namesakes, the Smurf attack is a brute-force attack targeted at a feature in the IP specification known as direct broadcast addressing. A Smurf hacker floods the router of the victim with Internet Control Message Protocol (ICMP) echo request packets (pings). Since the destination IP address of each packet is the broadcast address of the victim's network, the victim's router will broadcast the ICMP echo request packet to all hosts on its network. If the victim has numerous hosts, this will create a large amount of ICMP echo request and response traffic.

If a hacker chooses to spoof the source IP address of the ICMP echo request packet, the resulting ICMP traffic will not only clog up the primary victim's network—the "intermediary" network—but will also congest the network of the spoofed source IP address—known as the "secondary victim" network.

The User Datagram Protocol (UDP) Flood denial-of-service attack also links two unsuspecting systems. By spoofing, the UDP Flood attack hooks up one system's UDP chargen service, which for testing purposes generates a series of characters for each packet it receives, with another system's UDP echo service, which echoes any character it receives in an attempt to test network programs. As a result, a nonstop flood of useless data passes between the two systems.

Prevention of a UDP Flood, can be accomplished by either disabling all UDP services on each host in the network or by having a firewall filter all incoming UDP service requests. However, categorically denying all UDP traffic, you will rebuff legitimate applications, such as RealAudio, that use UDP as their transport mechanism.

Accordingly, what is needed is a method of preventing DoS attacks and a network device that can perform that method in order to prevent DoS attacks from disrupting entire networks.

SUMMARY OF THE INVENTION

The present invention provides for a method of preventing DoS attacks. The method involves scanning the contents of the data packets then verifying that the contents of the data packets conform to a set of predetermined requirements, the predetermined requirement including reordering and reassembling data packets according to a defined policy and insuring that they conform to required parameters such as packet length, non-overlapping offsets, and adherence to protocol standards. Data Packets that do not verify may be dropped.

After the contents have been verified, the data packets are checked to determine if they are associated with a validated traffic flow. If the data packet is associated with a validated traffic flow it is assigned to a higher priority quality of service for transmission back onto the network. If the data packet is not associated with a validated traffic flow it is

assigned to a low priority quality of service queue, such that data packets in the low priority quality of service queue can occupy no more that a predetermined maximum of the available network bandwidth when they are transmitted back onto the network.

The present invention also includes a network device for preventing DoS attacks. The network device includes a traffic flow scanning engine and a quality of service processor. The traffic flow scanning engine is operable to scan the contents of the data packets, including the header and the payload, and to associate the data packets with particular traffic flows. The traffic flow scanning engine is also operable to determine whether each traffic flow is a validated or non-validated. Further, the traffic flow scanning engine reorders and reassembles the data packets and associated traffic flows and insures that the data packets conform to predetermined requirements. The network device may drop packets that do not reorder or reassemble correctly or which do not conform to the predetermined requirements. Once the traffic flow scanning engine has scanned the data packet it produces a conclusion associated with that data packet, both of which are passed to the quality of service processor.

The quality of service processor uses the conclusion from the traffic flow scanning engine to place the data packets in the appropriate quality of service queue. Data packets associated with validated traffic flow are placed in higher priority queues and transmitted back onto the network according to the protocol for the particular queue. Data packets not assigned to a validated traffic flow are placed in low priority QoS queue. Data packets in the low priority QoS queue are transmitted onto the network such that they occupy no more than a predetermined maximum of available bandwidth, thereby preventing flood type DoS attacks.

The foregoing has outlined, rather broadly, preferred and alternative features of the present invention so that those skilled in the art may better understand the detailed description of the invention that follows. Additional features of the invention will be described hereinafter that form the subject of the claims of the invention. Those skilled in the art will appreciate that they can readily use the disclosed conception and specific embodiment as a basis for designing or modifying other structures for carrying out the same purposes of the present invention. Those skilled in the art will also realize that such equivalent constructions do not depart from the spirit and scope of the invention in its broadest form.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, in which:

FIGS. **1***a* and **1***b* is a network topology diagram illustrating example environments in which the present invention can operate;

FIG. **2** is a block diagram of a "bump-in-the-line" network apparatus according to the present invention;

FIG. **3** is a block diagram of the payload scanning engine from FIG. **2**;

FIG. **4** is a block diagram of a routing network apparatus according to the present invention; and

FIG. **5** is a flow chart illustrating a method according to the present invention for preventing denial of service attacks.

DETAILED DESCRIPTION OF THE DRAWINGS

Referring now to FIGS. **1***a* and **1***b*, a network topology is shown which is an example of several network infrastructures that connect in some manner to a broader public IP network **10** such as the internet. FIGS. **1***a* and **1***b* is in no way meant to be a precise network architecture, but only to serve as a rough illustration of a variety of network structures which can exist on a broadband IP network. Public IP network **10** can be accessed in a variety of ways. FIGS. **1***a* and **1***b* shows the public IP network being accessed through a private IP network **12** which can be the IP network of a company such as MCI or UUNET which provide private core networks. An endless variety of network structures can be connected to private IP network **12** in order to access other networks connected to private IP network **12** or to access public IP network **10**.

One example of a network structure connecting to private IP network **12** is hosting network **14**. Hosting network **14** is an example of a network structure that provides hosting services for internet websites. These hosting services can be in the form of webfarm **16**. Webfarm **16** begins with webservers **30** and database **32** which contain the webpages, programs and databases associated with a particular website such as amazon.com or yahoo.com. Webservers **30** connect to redundant load balancers **28** which receive incoming internet traffic and assign it to a particular webserver to balance the loads across all of webservers **30**. Redundant intrusion detection systems **26** and firewalls connect to load balancers **28** and provide security for webfarm **16**. Individual webfarms **16** and **17** connect to hosting network **14**'s switched backbone **18** by means of a network of switches **20** and routers **22**. Hosting network **14**'s switched backbone **18** is itself made up of a network of switches **20** which then connect to one or more routers **22** to connect to private IP network **12**. Connections between individual webfarms **16** and **17** and the switched backbone **18** of hosting network **14** are usually made at speeds such as OC-3 or OC-12 (approx. 150 megabits/sec or 625 megabits/sec), while the connection from router **22** of hosting network **14** to private IP network **12** are on the order OC-48 speeds (approx. 2.5 gigabits/sec).

Another example of network structures connecting to private IP network are illustrated with service provider network **34**. Service provider network **34** is an example of a network structure for Internet Service Providers (ISPs) or Local Exchange Carriers (LECs) to provide both data and voice access to private IP network **12** and public IP network **10**. Service provider network **34** provides services such as internet and intranet access for enterprise networks **36** and **37**. Enterprise networks **36** and **37** are, for example, company networks such as the company network for Lucent Technologies or Merril Lynch. Each enterprise network, such as enterprise network **36**, includes a plurality of network servers and individual workstations connected to a switched backbone **18**, which can be connected by routers **22** to service provider network **34**.

In addition to internet access for enterprise networks, service provider network **34** provides dial-up internet access for individuals or small businesses. Dial-up access is provided in service provider network **34** by remote access server (RAS) **42**, which allows personal computers (PCs) to call into service provider network **34** through the public switched telephone network (PSTN), not shown. Once a connection has been made between the PC **50** and RAS **42** through the PSTN, PC **50** can then access the private or public IP networks **12** and **10**.

Service provider network **34** also provides the ability to use the internet to provide voice calls over a data network referred to as Voice over IP (VoIP). VoIP networks **46** and **47** allow IP phones **48** and PCs **50** equipped with the proper software to make telephone calls to other phones, or PCs connected to the internet or even to regular phones connected to the PSTN. VoIP networks, such as VoIP network **46**, include media gateways **52** and other equipment, not shown, to collect and concentrate the VoIP calls which are sent through service provider network **34** and private and public internet **12** and **10** as required. As mentioned, the advent of VoIP as well as other real time services such as video over the internet make quality of service a priority for service providers in order to match the traditional telephone service provided by traditional telephone companies.

Service provider network **34** includes a switched backbone **18** formed by switches **20** as well as routers **22** between it and its end users and between it and private IP network **12**. Domain name servers **44** and other networking equipment, which are not shown, are also included in service provider network **34**. Similar to hosting network **34**, connection speeds for service provider network **34** can range from speeds such as T1, T3, OC-3 and OC-12 for connecting to enterprise networks **36** and **37** as well as VoIP networks **46** and **47** all the way to OC-48 and conceivably even OC-192 for connections to the private IP network.

It can easily be seen that aggregation points **60** exist at the edges of these various network structures where data is passed from one network structure to another at speeds such as OC-3, OC-12, and OC-48. One major problem in the network structures shown in FIG. **1** is the lack on any type of intelligence at these aggregation points **60** which would allow the network to provide services such as security, metering and quality of service. The intelligence to provide these services would require that the network understand the type of data passing through the aggregation points **60** and not just the destination and/or source information which is currently all that is understood. Understanding the type of data, or its contents, including the contents of the associated payloads as well as header information, and further understanding and maintaining a state awareness across each individual traffic flow would allow the network to configure itself in real time to bandwidth requirements on the network for applications such as VoIP or video where quality of service is a fundamental requirement. An intelligent, or "content aware", network would also be able to identify and filter out security problems such as email worms, viruses, denial of service (DoS) attacks, and illegal hacking in a manner that would be transparent to end users. Further, a content aware network would provide for metering capabilities by hosting companies and service providers, allowing these companies to regulate the amount of bandwidth allotted to individual customers as well as to charge precisely for bandwidth and additional features such as security.

In accordance with the requirements set forth above, the present invention provides for a network device that is able to scan, classify, and modify network traffic including payload information at speeds of OC-3, OC-12, OC-48 and greater thereby providing a "content aware" network capable of preventing denial of service attacks on the network.

Referring now to FIG. **2**, one embodiment of a network apparatus according to the present invention is shown. Network apparatus **100**, as shown, acts as a "bump-in-the-line" type device by accepting data received from a high-speed network line, processing the data, and then placing the data back on the line. Network apparatus **100** accepts data from the line by means of input physical interface **102**. Input physical interface **102** can consist of a plurality of ports, and can accept any number of network speeds and protocols, including such high speeds as OC-3, OC-12, OC-48, and protocols including $^{10}/_{100}$ Ethernet, gigabit Ethernet, and SONET. Input physical interface **102** takes the data from the physical ports, frames the data, and then formats the data for placement on fast-path data bus **126** which is preferably an industry standard data bus such as a POS-PHY Level 3, or an ATM UTOPIA Level 3 type data bus.

Fast-path data bus **126** feeds the data to traffic flow scanning processor **140**, which includes header processor **104** and payload analyzer **110**. The data is first sent to header processor **104**, which is operable to perform several operations using information contained in the data packet headers. Header processor **104** stores the received data packets in packet storage memory **106** and scans the header information. The header information is scanned to identify the type, or protocol, of the data packet, which is used to determine routing information as well as to create a session id using predetermined attributes of the data packet.

In the preferred embodiment, a session id is created using session information consisting of the source address, destination address, source port, destination port and protocol, although one skilled in the art would understand that a session id could be created using any subset of fields listed or any additional fields in the data packet without departing from the scope of the present invention. When a data packet is received that has new session information the header processor creates a unique session id to identify that particular traffic flow. Each successive data packet with the same session information is assigned the same session id to identify each packet within that flow. Session ids are retired when the particular traffic flow is ended through an explicit action, or when the traffic flow times out, meaning that a data packet for that traffic flow has not been received within a predetermined amount of time. While the session id is discussed herein as being created by the header processor **104** the session id can be created anywhere in traffic flow scanning engine **140** including in payload analyzer **110**.

As will be discussed below, network apparatus **100** in order to function properly needs to reorder out of order data packets and reassemble data packet fragments. Header processor **104** is operable to perform the assembly of asynchronous transfer mode (ATM) cells into complete data packets (PDUs), which could include the stripping of ATM header information.

Header processor **104** is also operable to perform routing functions. Routing tables and information can be stored in database memory **108**. Routing instructions received by network apparatus **100** are identified, recorded and passed to microprocessor **124** by header processor **104** so that microprocessor **124** is able to update the routing tables in database memory **108** accordingly. While network apparatus **100** is referred to as a "bump-in-the-line" apparatus, The input and the output could be formed by multiple lines, for example four OC-12 lines could be connected to network apparatus **100** which operates at OC-48 speeds. In such a case, "bump-in-the-line" network apparatus **100** will have limited routing or switching capabilities between the multiple lines, although the switching capability will be less than in a conventional router or switch. Additionally, a network apparatus can be constructed according to the principles of the present invention, which is able to operate as a network router or switch. Such an implementation is discussed in greater detail with reference to FIG. **4**.

After data packets have been processed by header processor **104** the data packets, their associated session id and any conclusion formed by the header processor, such as routing or QoS information, are sent on fast-data path **126** to the other half of traffic flow scanning engine **140**, payload analyzer **110**. The received packets are stored in packet storage memory **112** while they are processed by payload analyzer **110**. Payload analyzer **110** is operable to scan the contents of data packets received from header processor **104**, particularly the payload contents of the data packets, although header information can also be scanned as required. The contents of any or all data packets are compared to a database of known signatures and if the contents of a data packet or packets matches a known signature, an action associated with that signature and/or session id can be taken by network apparatus **100**. Additionally, payload analyzer **110** is operable to maintain state awareness throughout each individual traffic flow. In other words, payload analyzer **110** maintains a database for each session which stores state information related to not only the current data packets from a traffic flow, but state information related to the entirety of the traffic flow. This allows network apparatus **100** to act on not only based on the content of the data packets being scanned but also based on the contents of the entire traffic flow. The specific operation of payload analyzer **110** will be described with reference to FIG. **3**.

Once the contents of the packets have been scanned and a conclusion reached by traffic flow scanning engine **140**, the packets and the associated conclusions of either or both the header processor and the payload analyzer are sent to quality of service (QoS) processor **116**. QoS processor **116** again stores the packets in its own packet storage memory **118** for forwarding. QoS processor **116** is operable to perform the traffic flow management for the stream of data packets processed by network apparatus **100**. QoS processor contains engines for traffic management **126**, traffic shaping **128** and packet modification **130**.

QoS processor **116** takes the conclusion of either or both of header processor **104** and payload analyzer **110** and assigns the data packet to one of its internal quality of service queues **132** based on the conclusion. The quality of service queues **132** can be assigned priority relative to one another or can be assigned a maximum or minimum percentage of the traffic flow through the device. This allows QoS processor to assign the necessary bandwidth to traffic flows such as VoIP, video and other flows with high quality and reliability requirements while assigning remaining bandwidth to traffic flows with low quality requirements such as email and general web surfing to low priority queues. Information in queues that do not have the available bandwidth to transmit all the data currently residing in the queue according to the QoS engine is selectively discarded thereby removing that data from the traffic flow.

The quality of service queues **132** also allow network apparatus **100** to manage network attacks such as denial of service (DoS) attacks. Network apparatus **100** can act to qualify traffic flows by scanning the contents of the packets and verifying that the contents contain valid network traffic between known sources and destinations. Traffic flows that have not been verified because they are from unknown sources or because they are new unclassified flows can be assigned to a low quality of service queue until the sources are verified or the traffic flow classified as valid traffic. Since most DoS attacks send either new session information, data from spoofed sources, or meaningless data, network apparatus **100** would assign those traffic flows to low quality traffic queues. This ensures that the DoS traffic would receive no more that a small percentage (i.e. 5%) of the available bandwidth thereby preventing the attacker from flooding downstream network equipment.

The QoS queues **132** in QoS processor **116** (there are 65 k queues in the present embodiment of the QoS processor although any number of queues could be used) feed into schedulers **134** (1024 in the present embodiment), which feed into logic ports **136** (256 in the present embodiment), which send the data to flow control port managers **138** (32 is the present embodiment) which can correspond to physical egress ports for the network device. The traffic management engine **126** and the traffic shaping engine **128** determine the operation of the schedulers and logic ports in order to maintain traffic flow in accordance with the programmed parameters.

QoS processor **116** also includes packet modification engine **130**, which is operable to modify, add, or delete bits in any of the fields of a data packet. This allows QoS processor **116** to change addresses for routing or to place the appropriate headers on the data packets for the required protocol. The packet modification engine **130** can also be used to change information within the payload itself if necessary. Data packets are then sent along fast-data path **126** to output PHY interface **120** where it is converted back into an analog or optical signal and placed on the network.

As with all network equipment, a certain amount of network traffic will not be able to be processed along fast-data path **126**. This traffic will need to be processed by on board microprocessor **124**. The fast-path traffic flow scanning engine **140** and QoS processor **116** send packets requiring additional processing to flow management processor **122**, which forwards them to microprocessor **124** for processing. The microprocessor **124** then communicates back to traffic flow scanning engine **140** and QoS processor **116** through flow management processor **122**. Flow management processor **122** is also operable to collect data and statistics on the nature of the traffic flow through network apparatus **100**. In addition to processing odd, or missing packets, microprocessor **124** also controls the user management interface **142** and recompiles databases **108** and **114** to accommodate new signatures and can be used to learn and unlearn sessions identified by the traffic flow scanning engine **140**.

The abilities of network apparatus **100** are unique in a number of respects. Network apparatus **100** has the ability to scan the contents of any data packet or packets for any information that can be represented as a signature or series of signatures. The signatures can be of any arbitrary length, can begin and end anywhere within the packets and can cross packet boundaries. Further, network apparatus **100** is able to maintain state awareness throughout all of the individual traffic flow by storing state information for each traffic flow representing any or all signatures matched during the course of that traffic flow. Existing network devices operate by looking for fixed length information at a precise point within each data packet and cannot look across packet boundaries. By only being able to look at fixed length information at precise points in a packet, existing network equipment is limited to acting on information contained at an identifiable location within some level of the packet headers and cannot look into the payload of a data packet much less make decisions on state information for the entire traffic flow or even on the contents of the data packet including the payload.

Referring now to FIG. **3**, the payload analyzer **110** of FIG. **2** is described in greater detail. As described above payload analyzer **110** is operable to scan the contents of data packets

forwarded from header processor **104** from FIG. **2**. Payload analyzer **110** includes three separate engines, queue engine **302**, context engine **304**, and payload scanning engine **306**.

Since payload analyzer **110** scans the contents of the payload, and is able to scan across packet boundaries, payload analyzer **110** must be able reassemble fragmented packets and reorder out of order packets on a per session basis. Reordering and reassembling is the function of queue engine **302**. Queue engine **302** receives data off the fast-path data bus **126** using fast-path interface **310**. Packets are then sent to packet reorder and reassembly engine **312**, which uses packet memory controller **316** to store the packets into packet memory **112**. Reordering and reassembly engine **312** also uses link list controller **314** and link list memory **318** to develop detailed link lists that are used to order the data packets for processing. Session CAM **320** can store the session id generated by queue engine **302** of payload analyzer **110**. Reordering and reassembly engine **312** uses the session id to link data packets belonging to the same data flow.

In order to obtain the high throughput speeds required, payload analyzer **110** must be able to process packets from multiple sessions simultaneously. Payload analyzer **110** processes blocks of data from multiple data packets each belonging to a unique traffic flow having an associated session id. In the preferred embodiment of the present invention, payload analyzer **110** processes 64 byte blocks of 64 different data packets from unique traffic flows simultaneously. Each of the 64 byte blocks of the 64 different data flows represents a single context for the payload analyzer. The scheduling and management of all the simultaneous contexts for payload analyzer **110** is handled by context engine **304**.

Context engine **304** works with queue engine **302** to select a new context when a context has finished processing and been transmitted out of payload analyzer **110**. Next free context/next free block engine **330** communicates with link list controller **314** to identify the next block of a data packet to process. Since payload analyzer **110** must scan data packets in order, only one data packet or traffic flow with a particular session id can be active at one time. Active control list **332** keeps a list of session ids with active contexts and checks new contexts against the active list to insure that the new context is from an inactive session id. When a new context has been identified packet loader **340** uses the link list information retrieved by the next free context/next free block engine to retrieve the required block of data from packet memory **112** using packet memory controller **316**. The new data block is then loaded into a free buffer from context buffers **342** where it waits to be retrieved by payload scanning engine interface **344**.

Payload scanning engine interface **344** is the interface between context engine **304** and payload scanning engine **306**. When payload scanning engine **306** has room for a new context to be scanned, payload scanning engine interface **344** sends a new context to string preprocessor **360** in payload scanning engine **306**. String preprocessor **360** is operable to simplify the context by performing operations such as compressing white space (i.e. spaces, tabs, returns) into a single space to simplify scanning. Once string preprocessor **360** has finished, the context is loaded into one of the buffers in context buffers **362** until it is retrieved by scheduler **364**. Scheduler controls the input and output to signature memory **366**. While four signature memories **366**, each of which is potentially capable of handling multiple contexts, are shown any number could be used to increase or decrease the throughput through payload scanning engine **110**. In the present embodiment, each of the signature memories **366** is capable of processing four contexts at one time.

One of the signature memories **366** is assigned the context by scheduler **364** and then compares the significant bits of the context to the database of known strings that reside in signature memory **366**. The signature memory **366** determines whether there is a potential match between the context and one of the known signatures using significant bits, which are those bits that are unique to a particular signature. If there is a potential match, the context and the potentially matched string are sent to leaf string compare **368** which uses leaf string memory **370** to perform a bit to bit comparison of the context and the potentially matched string.

The conclusion of the payload scanning are then sent back to the payload scanning interface **344** along with possibly a request for new data to be scanned. The conclusion of the payload scanning can be any of a number of possible conclusions. The scanning may not have reached a conclusion yet and may need additional data from a new data packet to continue scanning in which case the state of the traffic flow and any incomplete scans are stored in session memory **354** along with other appropriate information such as sequence numbers, counters etc. The conclusion reached by signature memory **366** may also be that scanning is complete and there is or isn't a match, in which case the data packet and the conclusion are sent to transmit engine **352** for passing to QoS processor **116** from FIG. **2**. The scanning could also determine that the data packet needs to be forwarded to microprocessor **124** from FIG. **2** for further processing, so that the data packet is sent to host interface **350** and placed on host interface bus **372**. In addition to handling odd packets, host interface bus **350** allows microprocessor **124** to control any aspect of the operation of payload analyzer **110** by letting microprocessor **124** write to any buffer or register in context engine **304**.

State information is stored in session memory **354** and is updated as necessary after data associated with the particular traffic flow is scanned. The state information for each traffic flow represents the content awareness of network apparatus **100** from FIG. **2**, and allows network apparatus to act not only on the information scanned, but also on all the information that has been scanned for each traffic flow.

The operation of transmit engine **352**, host interface **350**, session memory controller **348**, which controls the use of session memory **354**, and of general-purpose arithmetic logic unit (GP ALU) **346**, which is used to increment or decrement counter, move pointers, etc., is controlled by script engine **334**. Script engine **334** operates to execute programmable scripts stored in script memory **336** using registers **338** as necessary. Script engine **334** uses control bus **374** to send instruction to any of elements in context engine **304**.

As can be seen from the description of FIG. **3**, payload analyzer **110** allows the entire contents of any or all data packets received by a network device to be scanned against a database of known signatures. The scanned contents can be any variable or arbitrary length and can even cross packet boundaries. The abilities of payload analyzer **110** allow the construction of a network device that is content aware which gives the network device the ability to operate on data packets based on the content of that data packet as has already been described herein.

Referring now to FIG. **4** an embodiment of the network apparatus of the present invention with routing capabilities is described. Routing network apparatus **400** is formed by two or more route engine cards **402** connected to switch

fabric **404**. One or more management cards **406** are also included to provide a user interface and to manage route engine cards **402**. Each of route engine cards **402** operate fundamentally as described with respect to network apparatus **100** of FIG. **2**. Traffic flow scanning engine **408**, formed by header processor **410** and payload analyzer **412**, scans the contents of the data packets and generates a conclusion based on the contents. The packets and associated conclusions are forwarded to ingress QoS processor **414**, which assigns the packets to a QoS queue. The data packets are then sent to the switch fabric, which forwards the data packets to the proper route engine card **402** for it's assigned output port. The data packet then flows through the egress QoS processor **418**, which schedules the traffic received from all the route engine cards **402** for transmission onto the network. The microprocessor **124** shown in FIG. **2** could be present on the route engine card **402** or could potentially be moved to the management card **406** to allow one microprocessor to support multiple route engine cards **402**. Each of the route engine cards **402** could even have its own microprocessor with an additional microprocessor on management card **406**.

Having multiple route engine cards with multiple ingress and egress paths allows routing network apparatus to function as a routing network device, as opposed to the single ingress and egress path of the "bump-in-the-line" device described with respect to FIG. **2**. This allows the routing functions of header processor **410** to be utilized in routing network apparatus **400**.

Referring now to FIG. **5**, a method of preventing denial of service attacks according to the present invention is shown. The method begins at start block **500** and proceeds to block **502**, which scans the header information of the data packets. The method then passes to block **504** where the data packets are reassembled and reordered to create an ordered flow of complete data packets. During reorder and reassembly, block **506** determines whether the data packets reorder and reassemble correctly. If they do not reorder or reassemble correctly, such as with a teardrop type attack, the packet or packets are dropped as shown with block **508**, thereby preventing the attack from reaching its destination. The dropping of data packets can be accomplished by flagging the data packet to be dropped immediately or at some point in the future, for example by the QoS processor **116** from FIG. **2**.

If the blocks do reorder and reassemble correctly the method passes to block **510**, which scans the payload contents of the data packets. During the scanning of the payload contents, the method determines whether the data conforms to enforced protocols such as length, structure, proper context within the data, etc. If the data does not conform to enforced protocols, such as exceeding maximum packet length with a Ping of Death type attack, the packet is also dropped as shown by block **508**.

Data packets which reorder and reassemble correctly and which contain data that conforms to enforced protocols are then checked to see if they are part of a validated traffic flow. As described above with reference to FIGS. **2** and **3**, each individual traffic flow is identified and assigned a session id. After scanning a particular traffic flow over enough data packets to determine the nature of the traffic flow the traffic flow is classified, or in other words assigned a quality of service level and either validated as an appropriate traffic flow or left non-validated. The classification of the traffic flow is dynamic and can change over the course of the traffic flow as more information about the traffic flow is learned from scanning. The classification is part of the state of the traffic flow stored in session memory **354** from FIG. **3**.

Block **514** shows the determination of whether the data packet belongs to a validated traffic flow. If the packet does belong to a validated traffic flow, it is assigned to a higher priority QoS queue based on the nature of the traffic flow, shown in block **520**. The data packet is then transmitted onto the network as defined by the QoS protocol for the QoS queue to which it was assigned, as shown by block **522**. Block **524** then represents the end of the present method.

If the packet does not belong to a validated traffic flow, such as with a Smurf or SYN flood where valid sessions are never established, it is assigned to a low priority QoS queue, as shown in block **516**. The traffic management and traffic shaping engines **126** and **126** of QoS processor **116** from FIG. **2** can be programmed to insure that the low priority QoS queues are only allowed a predetermined percentage of the available bandwidth. This insures that the non-validated data packets cannot consume more than a small portion of the overall traffic flowing through network apparatus **100** as shown in block **518**. By limiting the amount of bandwidth available to brute force attacks, such as the SYN flood attack or Smurf attacks, the attack is never allowed to consume enough network bandwidth to overwhelm the network.

FIG. **5** shows how the present invention can use its awareness of the contents of data packets and associated traffic flows, including keeping state over each traffic flow, to prevent DoS type attacks. Network apparatus **100** from FIG. **2** is able to either drop nonconforming data packets or to prevent flood type attacks from consuming all available bandwidth by understanding the nature of the data passing through it.

While the header processor, the QoS processors, and the flow management processor described with reference to FIGS. **2** and **4** can be any suitable processor capable of executing the described functions, in the preferred embodiment the header processor is the Fast Pattern Processor (FPP), the QoS processor is the Routing Switch Processor (RSP), and the flow management processor is the ASI processor, all manufactured by the Agere Division of Lucent Technologies, Austin Tex. Similarly the switch fabric may be any suitable switch fabric as is well known in the industry, including those manufactured by Power X Networks, Inc., 2833 Junction Ave., Suite 110, San Jose, Calif. The microprocessor described with reference to FIGS. **2** and **4** could be any suitable microprocessor including the PowerPC line of microprocessors from Motorola, Inc., or the ×86 or Pentium line of microprocessors available from Intel Corporation. Although particular references have been made to specific protocols, implementations and materials, those skilled in the art should understand that the network apparatus, both the "bump-in-the-line" and the routing apparatus can function independent of protocol, and in a variety of different implementations without departing from the scope of the invention.

Although the present invention has been described in detail, those skilled in the art should understand that they can make various changes, substitutions and alterations herein without departing from the spirit and scope of the invention in its broadest form.

We claim:

1. A method for preventing denial of service attacks over a data network including a plurality of traffic flows each formed by a plurality of data packets, the method comprising:

scanning the contents of the data packets;

verifying that the data packets conform to a set of predetermined requirements;

checking if the data packet is associated with a validated traffic flow; and

placing the data packet in a higher priority quality of service if the data packet is associated with a validated

traffic flow and to a low priority quality of service if it is not associated with a validated traffic flow.

2. The method of claim 1 wherein verifying includes insuring that the data packets reorder and reassemble according to a defined policy and insuring that the data packets conform to required parameters.

3. The method of claim 1 further comprising between verifying and checking:

dropping the data packet if it does not conform to the set of predetermined requirements.

4. The method of claim 3 wherein scanning includes scanning of the data packet's header information and scanning the data packet's payload contents.

5. The method of claim 1 wherein the predetermined requirements include packet length, non-overlapping offset fields, and adherence to protocol standards.

6. The method of claim 5 wherein the validated traffic flows are identified by a state associated with each traffic flow.

7. A method of preventing denial of service attacks on a data network which includes a plurality of traffic flows each formed by multiple data packets having header and payload information, the method using a network device comprising a traffic flow scanning engine and a quality of service processor having a low priority queue and higher priority queues, the method comprising:

scanning the header information using the traffic flow scanning engine;

reordering and reassembling the data packets using the traffic flow scanning engine;

flagging data packets that do not reorder or reassemble correctly to be dropped;

scanning the payload contents using the traffic flow scanning engine;

determining whether the data packets conform to a set of predetermined requirements;

flagging data packets that do not conform to be dropped;

checking if the data packets are associated with a validated traffic flow; and

assigning data packets to a higher priority quality of service if the data packet is associated with a validated traffic flow and to a low priority quality of service if the data packet is not associated with a validated traffic flow.

8. The network device of claim 7 wherein the set of predetermined requirements include packet length, non-overlapping offset fields, and adherence to protocol standards.

9. The method of claim 7 wherein flagged data packets are dropped by the traffic flow scanning engine.

10. The method of claim 7 wherein flagged data packets are dropped by the quality of service processor.

11. The method of claim 7 wherein the validated traffic flows are identified by a state associated with each traffic flow.

12. A network device for preventing denial of service attacks on a data network which includes a plurality of traffic flows each formed by multiple data packets having contents including header information and payload information, the network device comprising:

a traffic flow scanning engine operable to scan the header and payload information of the data packets, to associate each data packet with a particular traffic flow and to determine whether each traffic flow is a validated traffic flow or a non-validated traffic flow, wherein the traffic flow scanning engine is further operable to reorder and reassemble the data packets and to verify that the data packet conforms to predetermined requirements such that the traffic flow scanning engine produces a conclusion associated with each data packet; and

a quality of service processor connected to the traffic flow scanning engine and operable to place the data packets into a quality of service queue from a plurality of quality of service queues based on the conclusion from the traffic flow scanning engine, wherein data packets from non-validated traffic flows are assigned to a low priority queue and data packets from validated traffic flow are assigned to a higher priority queue based on its contents.

13. The network device of claim 12 wherein the low priority queue is assigned a maximum percentage of network bandwidth.

14. The network device of claim 12 wherein data packets that do not reorder or reassemble correctly and data packets that do not conform to the predetermined requirements are dropped by the network device.

15. The network apparatus of claim 12 wherein the traffic flows are identified by a state associated with each traffic flow, the state representing whether the traffic flow is validated or non-validated.

16. The network apparatus of claim 12 wherein the set of predetermined requirements include packet length, non-overlapping offset fields, and adherence to protocol standards.

* * * * *

Exhibit 1007

U.S. Patent 6,725,378 by Christoph L. Schuba et al. ("Schuba")

Customer No.: 000027683

Haynes and Boone, LLP
IP Section
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Telephone [214] 651.5000
Fax [214] 200.0853

US006725378B1

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

(10) **Patent No.: US 6,725,378 B1**
(45) **Date of Patent: Apr. 20, 2004**

(54) **NETWORK PROTECTION FOR DENIAL OF SERVICE ATTACKS**

(75) Inventors: **Christoph L. Schuba**, Mountain View, CA (US); **Ivan V. Krsul**, LaPaz (BO); **Diego Zamboni**, West Lafayette, IN (US); **Eugene H. Spafford**, West Lafayette, IN (US); **Aurobindo M. Sundaram**, Austin, TX (US); **Markus G. Kuhn**, Uttenreuth (DE)

(73) Assignee: **Purdue Research Foundation**, West Lafayette, IN (US)

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

(21) Appl. No.: **09/293,479**

(22) Filed: **Apr. 15, 1999**

Related U.S. Application Data

(60) Provisional application No. 60/081,829, filed on Apr. 15, 1998.

(51) **Int. Cl.**[7] **H04L 9/22**; G06F 13/36

(52) **U.S. Cl.** **713/201**; 713/201; 709/206; 709/223; 709/224; 707/10

(58) **Field of Search** 713/201; 709/206, 709/223, 224; 707/10

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,481,611	A		1/1996	Owens et al. 380/25
5,511,122	A		4/1996	Atkinson 380/25
5,684,800	A		2/1997	Aziz 380/25
5,606,668	A		2/1997	Shwed 395/200.11
5,699,513	A	*	12/1997	Feigen et al.
5,724,425	A		3/1998	Chang et al. 380/25
5,751,812	A		5/1998	Anderson 380/48
5,822,434	A		10/1998	Caronni et al. 380/49
5,826,014	A		10/1998	Coley et al. 395/187.01
5,828,846	A	*	10/1998	Kirby et al.
5,842,040	A		11/1998	Hughes et al. 395/831
5,845,068	A		12/1998	Winiger 395/186
5,850,449	A		12/1998	McManis 380/25
5,884,025	A		3/1999	Baehr et al. 395/187.01
5,991,881	A	*	11/1999	Conklin et al. 713/201
5,999,932	A	*	12/1999	Paul 707/10
6,061,798	A	*	5/2000	Coley et al. 713/201
6,070,242	A	*	5/2000	Wong et al. 713/201
6,304,975	B1	*	10/2001	Shipley 713/201
6,453,345	B2	*	9/2002	Trcka et al. 709/24

OTHER PUBLICATIONS

RFC 793, Transmission Control, Protocol, Information Sciences Institute, University of Southern California, USA, 1981.*

Firewalls fend off invasions from the Net, Steven W. Lodin and Christoph L. Schuba; IEEE Spectrum, Feb. 1998, pp. 26–34.

Defining Strategies to Protect Against TCP SYN Denial of Service Attacks, Security Advisory, Cisco Systems, Inc., 1996.

Advisory CA–96.21; TCP SYN Flooding and IP Spoofing Attacks, CERT Advisory, Sep. 19, 1996.

SUN Microsystems Security Bulletin #00136, Oct. 9, 1996.

Check Point Firewall–1 White Paper, Check Point Software Technologies, Ltd., Version 3.0 –Jun. 1997.

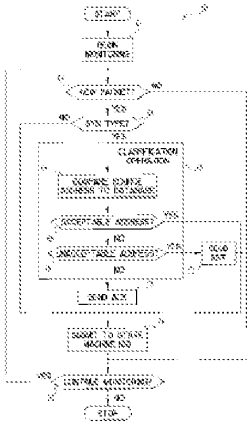
* cited by examiner

Primary Examiner—Ayaz Sheikh
Assistant Examiner—Taghi T. Arani
(74) *Attorney, Agent, or Firm*—Baniak Pine & Gannon

(57) **ABSTRACT**

An active monitor detects and classifies messages transmitted on a network. In one form, the monitor includes a routine for classifying TCP packet source addresses as being of an acceptable, unacceptable, or suspect type. Suspect source addresses may be further processed in accordance with a state machine having a number of conditionally linked states including a good address state, a new address state, and a bad address state. For this form, the monitor selectively sends signals to targeted destination hosts for addresses in the unacceptable

4 Claims, 5 Drawing Sheets



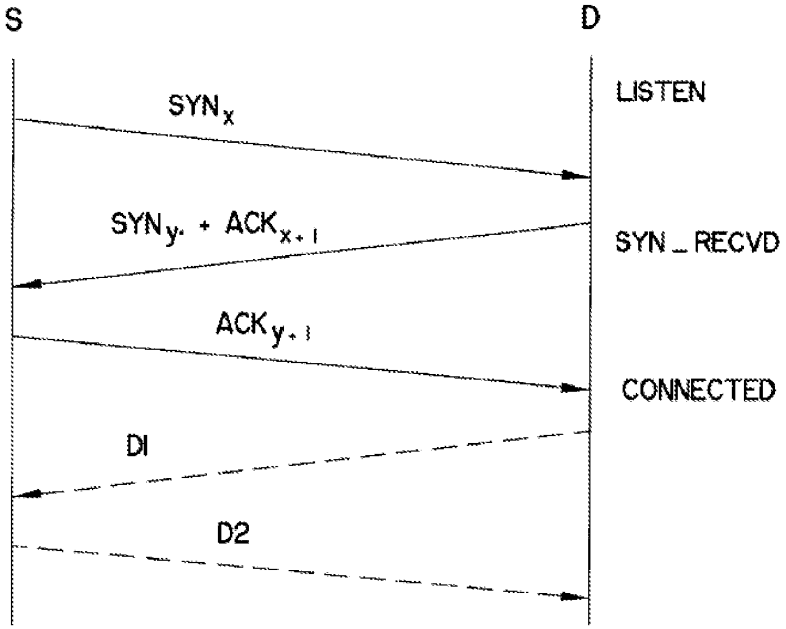

S
D
LISTEN
SYN_x
SYN_y. + ACK_x+1
SYN_RECVD
ACK_y+1
CONNECTED
DI
D2

Fig. 1

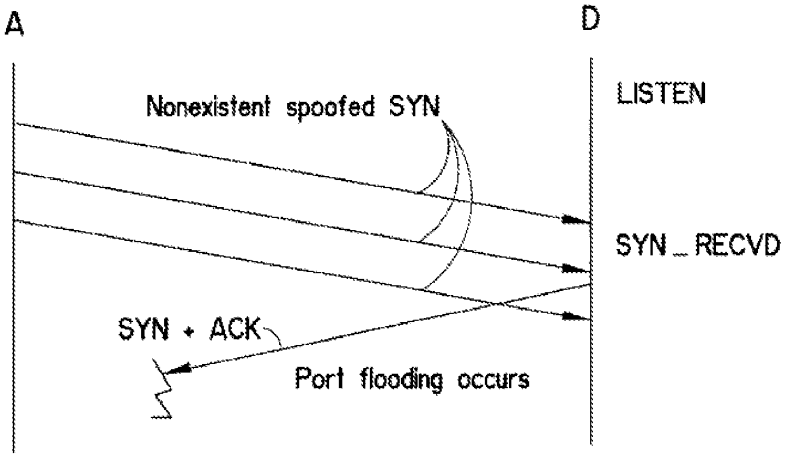

A
D
LISTEN
Nonexistent spoofed SYN
SYN_RECVD
SYN + ACK
Port flooding occurs

Fig. 2

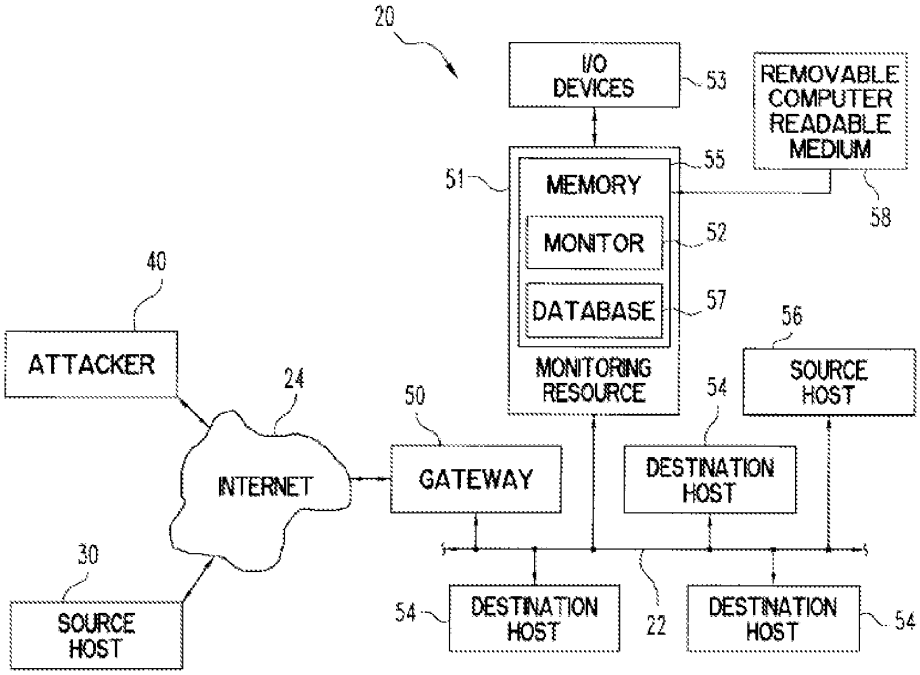

20
I/O DEVICES
53
REMOVABLE COMPUTER READABLE MEDIUM
55
51
MEMORY
58
MONITOR
52
DATABASE
57
56
40
ATTACKER
24
50
54
SOURCE HOST
MONITORING RESOURCE
INTERNET
GATEWAY
DESTINATION HOST
30
SOURCE HOST
54
DESTINATION HOST
22
DESTINATION HOST
54

Fig. 3

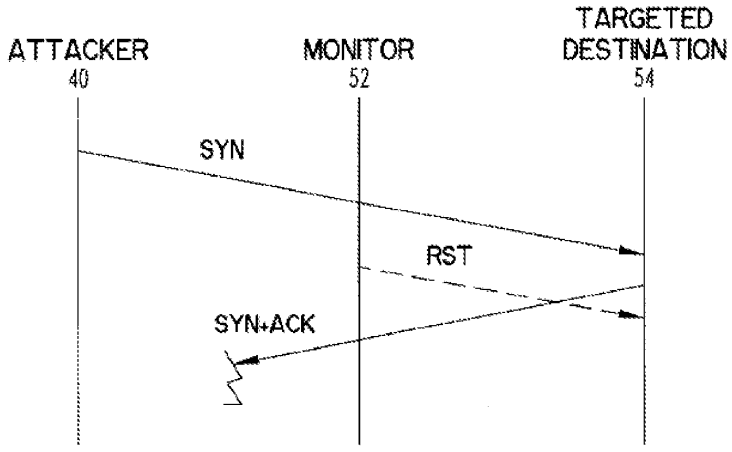

ATTACKER
40
MONITOR
52
TARGETED DESTINATION
54
SYN
RST
SYN+ACK

Fig. 5

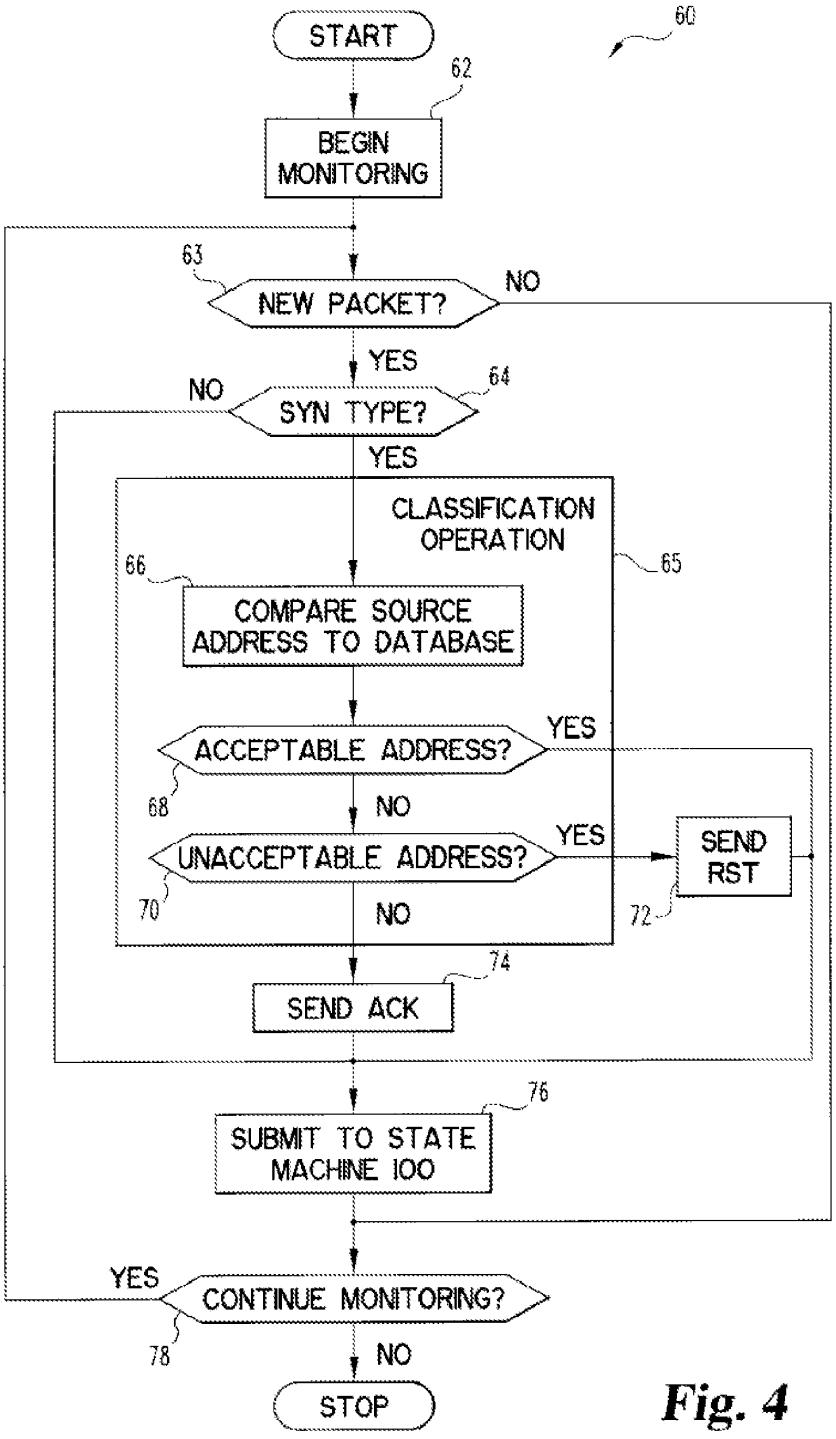

START
60
62
BEGIN MONITORING
63
NEW PACKET?
NO
YES
64
SYN TYPE?
NO
YES
CLASSIFICATION OPERATION
65
66
COMPARE SOURCE ADDRESS TO DATABASE
ACCEPTABLE ADDRESS?
YES
68
NO
UNACCEPTABLE ADDRESS?
YES
SEND RST
70
72
NO
74
SEND ACK
76
SUBMIT TO STATE MACHINE 100
YES
CONTINUE MONITORING?
78
NO
STOP

Fig. 4

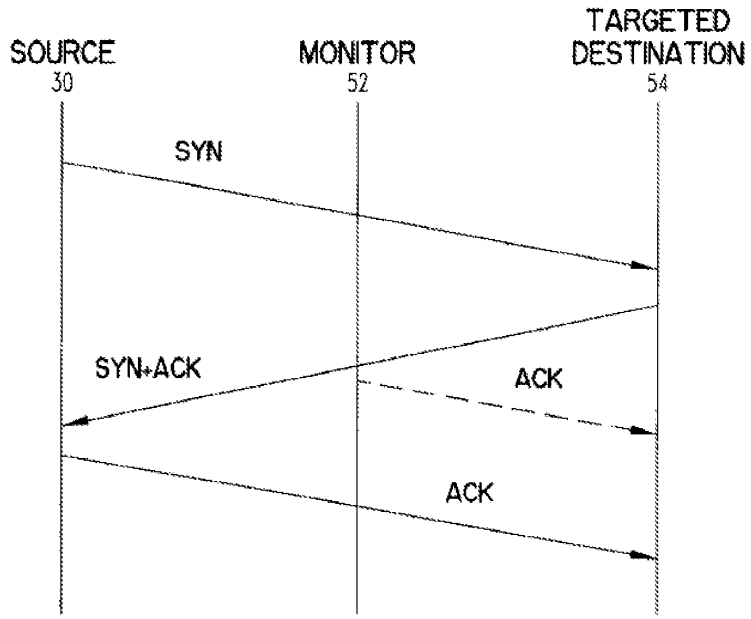

SOURCE
30
MONITOR
52
TARGETED
DESTINATION
54
SYN
SYN+ACK
ACK
ACK

Fig. 6

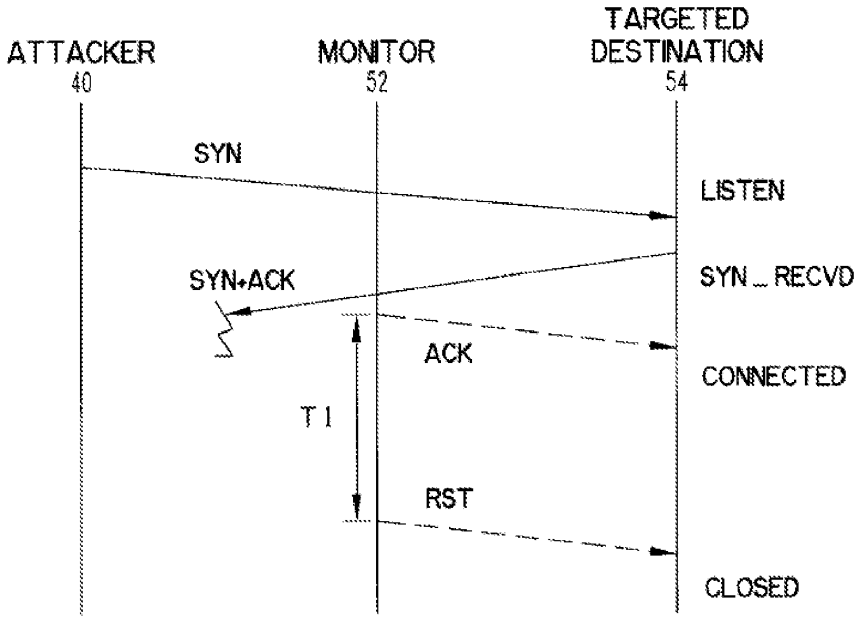

ATTACKER
40
MONITOR
52
TARGETED
DESTINATION
54
SYN
LISTEN
SYN+ACK
SYN_RECVD
ACK
CONNECTED
T1
RST
CLOSED

Fig. 8

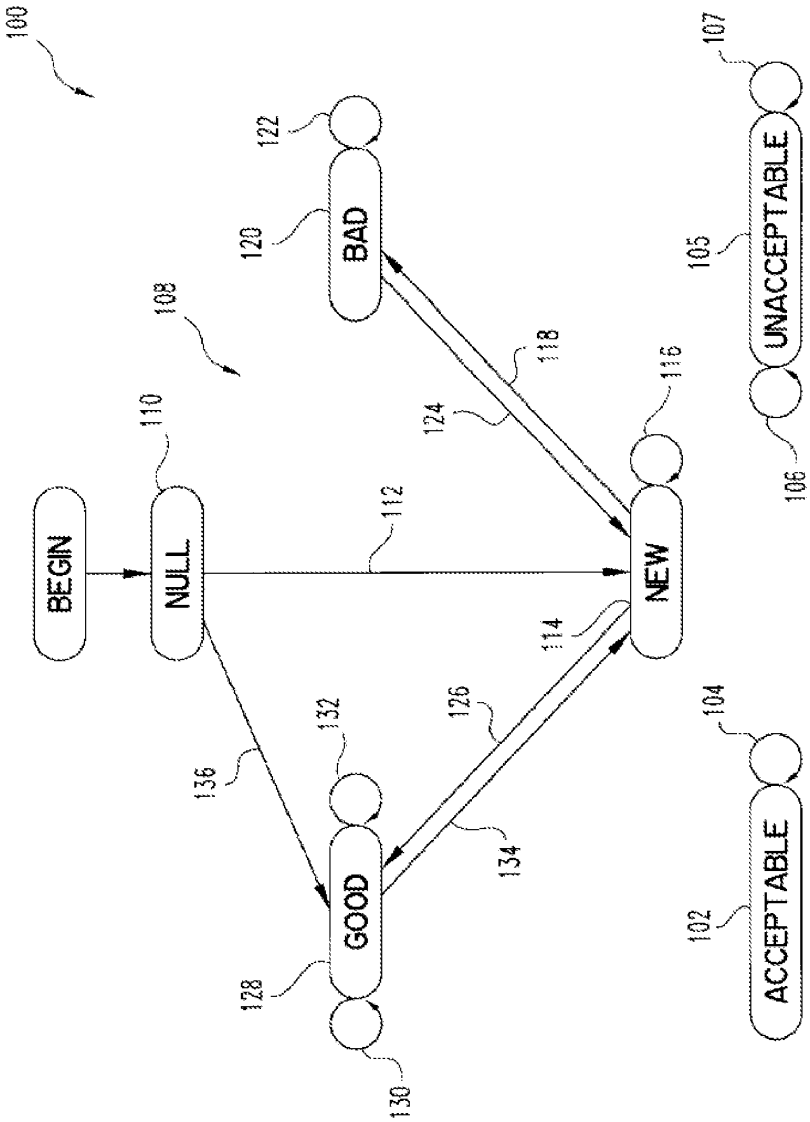

100
BEGIN
NULL
110
112
NEW
116
108
BAD
120
122
118
124
UNACCEPTABLE
105
107
106
ACCEPTABLE
102
104
GOOD
128
130
132
136
134
126
114

Fig. 7

NETWORK PROTECTION FOR DENIAL OF SERVICE ATTACKS

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application claims the benefit of U.S. Provisional Patent Application Ser. No. 60/081,829 filed Apr. 15, 1998 that is hereby incorporated by reference in its entirety herein.

BACKGROUND OF THE INVENTION

The present invention relates to computer network security, and more particularly, but not exclusively, relates to protection from denial of service attacks caused by initiating, but not completing a number of spurious connections through the Internet.

Concomitant with the development of computer networks, schemes to degrade or defeat operation of such networks have been devised. In response, a host of techniques have been proposed to improve network security. U.S. Pat. No. 5,884,025 to Baehr et al.; U.S. Pat. No. 5,842,040 to Hughes et al.; U.S. Pat. No. 5,826,014 to Coley et al.; U.S. Pat. No. 5,822,434 to Caronni et al.; U.S. Pat. No. 5,604,803 to Aziz; U.S. Pat. No. 5,511,122 to Atkinson; and U.S. Pat. No. 5,481,611 to Owens et al. are cited as sources of additional background information regarding various attempts to improve network security.

One particular type of attack monopolizes available network resources in such a manner that legitimate users are denied service. This type of "denial of service attack" has been recognized in various contexts. One especially troublesome denial of service attack for Internet Protocol (IP) based networks is called "SYN flooding." SYN flooding arises when an attacker sends many Transmission Control Protocol (TCP) connection requests, each initiated with a "SYNchronize" (also called SYN) packet, to a victim's machine. Each request causes the targeted host to instantiate data structures out of a limited pool of resources; however, because the source address for each of these SYN packets is illegitimate or "spoofed," completion of a proper connection is impossible. Consequently, the limited resources of the targeted host are quickly used up in response to the spurious SYN packets so that no more incoming TCP connections can be established—thus denying further legitimate access. The SYN flooding attack exploits weaknesses with TCP/IP (Transmission Control Protocol/Internet Protocol) that cannot be corrected without significant protocol modifications. Moreover, this type of denial of service attack can be launched with little effort, and is difficult to trace back to its originator.

As a result, there is a need for protective techniques that reduce, if not eliminate, the impact of denial of service attacks, such as SYN flooding.

SUMMARY OF THE INVENTIONS

One form of the present invention includes a unique computer network monitoring technique. A further form of the present invention includes a unique defense for denial of service attacks.

In another form of the present invention, network messages passing to one or more hosts from an untrusted network are actively monitored. Suspect messages are identified. The behavior of each suspect message is tracked in terms of a number of conditionally coupled states to determine whether any of the suspect messages present a security threat requiring action.

In yet another form, network messages may be classified into one or more other categories besides the suspect category. By way of nonlimiting example, TCP packets may be categorized as having an unacceptable source address, suspect source address, or acceptable source address.

In an additional form, host resources allocated in response to connection initiation requests from an untrusted network are released by monitoring such requests, determining which requests are unacceptable or suspect, and selectively sending a command that changes the status of such requests from the perspective of the hosts. In one nonlimiting example, host resources dedicated to incomplete TCP connections that are quickly saturated by suspect SYN packets may be released by sending ACKnowledge packets (also called ACK packets) corresponding to the SYN packets. In another nonlimiting example, host resources may be relieved by closing spurious connections through transmission of a ReSeT packet (also called a RST packet).

In still another form, a technique of the present invention includes detecting a number of SYN packets sent from an untrusted network to a destination host to regulate spurious connection attempts. The corresponding source address of each SYN packet is classified in one of a plurality of different categories, the categories including a suspect source address category and an unacceptable source address category. TCP packet behavior is monitored for each address classified in the suspect source address category with a state machine process including at least three different conditionally coupled states, and a RST packet is sent to the destination host for any address classified in the unacceptable source address category.

Accordingly, it is one object of the present invention to provide a unique computer network monitoring technique.

It is another object of the present invention to provide a unique defense for denial of service attacks.

Further objects, forms, features, advantages, aspects, and benefits of the present invention will become apparent from the description and drawings provided herein.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a diagram illustrating a Transmission Control Protocol (TCP) three way handshake packet sequence.

FIG. **2** is a diagram illustrating a SYN flood packet sequence.

FIG. **3** is a schematic view of a network system.

FIG. **4** is a flow chart illustrating a routine performed by the system of FIG. **3**.

FIG. **5** is a diagram of a first packet sequence illustrative of certain aspects of the routine of FIG. **4**.

FIG. **6** is a diagram of a second packet sequence illustrative of certain aspects of the routine of FIG. **4**.

FIG. **7** is a diagram illustrating a state machine process operated as part of the routine of FIG. **4**.

FIG. **8** is a diagram of a third packet sequence illustrative of certain aspects of the routine of FIG. **4** and the state machine process of FIG. **7**.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

For the purpose of promoting an understanding of the principles of the invention, reference will now be made to the embodiments illustrated in the drawings and specific language will be used to describe the same. It will nevertheless be understood that no limitation of the scope of the

invention is thereby intended. Any alterations and further modifications in the described embodiments, and any further applications of the principles of the invention as described herein are contemplated as would normally occur to one skilled in the art to which the invention relates.

The preferred embodiments of the present invention include an active monitor that performs a process to reduce service degradation caused by SYN flooding. In one form, the process includes classifying source addresses of SYN packets as being of an acceptable, unacceptable, or suspect type. Suspect source addresses may be further processed in accordance with a state machine having a number of conditionally linked states. The monitor selectively sends various packets to one or more hosts receiving the SYN packets to reduce or eliminate service denial.

To better explain various aspects of the preferred embodiments, certain features of the Transmission Control Protocol/Internet protocol (TCP/IP) are first described. The Internet Protocol (IP) is the standard network layer protocol of the Internet that provides a connectionless, best effort packet delivery service. IP defines the basic unit of the data transfer used throughout an IP network, called a datagram. The delivery of datagrams is not guaranteed. Datagrams may be lost, duplicated, delayed, or delivered out of order. IP is connectionless, because each packet is treated independently of the others—each may travel over different paths and some may be lost while others are delivered. IP provides best-effort delivery, because packets are not discarded unless resources are exhausted or underlying networks fail. Datagrams are routed towards their destination host. A set of rules characterizes how hosts and gateways should process packets, how and when error messages should be generated, and when packets should be discarded.

To ensure reliable communications for applications and services that need them, the Transmission Control Protocol (TCP) is available. It resides between IP and the application layer. TCP provides a reliable, connection-oriented data stream delivery service. As long as there is link layer communication between two communicating endpoints, TCP guarantees that datagrams will be delivered in order, without errors, and without duplication. It provides these services by using flow control mechanisms, such as the sliding window protocol, and adaptive retransmission techniques.

Referring to FIG. 1, a diagram is provided that illustrates the TCP packet sequence of a three-way handshake needed to establish a TCP connection. Before data can be transmitted between a source host S and a destination host D, TCP needs to establish a connection between source host S and destination host D. The connection establishment process is called the three-way handshake. The three-way handshake is established by exchanging certain TCP packet types between source host S and destination host D. The TCP packet types are distinguished by dedicated flag bits set in a TCP header code field and are listed in Table I as follows:

TABLE I

TCP Header Flag Bits	Abbreviation
SYNchronize	SYN
ACKnowledgement	ACK
ReSeT	RST

It should be appreciated that, under appropriate conditions, more than one of the flag bits may be set in the same TCP packet.

The first transmission in the three-way handshake is from source host S to destination host D in the form of a SYN packet (SYN flag bit set) while destination host D is in the LISTEN state. The second message, from destination host D to source host S, has both the SYN and ACK bit flags set (SYN+ACK) indicating that destination host D acknowledges the SYN packet and is continuing the handshake. At this point, destination host D is in the SYN_RECVD state. The third message, from source host S to destination host D has its ACK bit flag set, and is an indication to destination host D that both hosts S and D agree that a connection has been established, resulting in the CONNECTED state of destination host D. The third message may contain user payload data. Datagrams D1 and D2 represent data exchanges that take place after proper establishment of the TCP connection.

The three-way handshake also initializes the sequence numbers for a new connection between source host S and destination host D. The sequence numbers are represented in FIG. 1 by subscripts to the packet abbreviations. Sequence numbers are needed by the TCP protocol to enable reliable packet delivery and retransmission. Source host S sends an initial sequence number x with the first datagram: SYN_x. In the second message, destination host D acknowledges the first datagram with ACK_{x+1} and sends its own sequence number y: SYN_y. Source host S acknowledges the destination host D's packet in the final message of the three-way handshake with ACK_{y+1}.

For any TCP connection, there are memory structures that need to be allocated by both endpoints. For an example under the BSD style of network code, three memory structures need to be allocated at each endpoint. The socket structure (socket) holds information related to the local end of the communication link: protocol used, state information, addressing information, connection queues, buggers and flags. TCP uses the Internet protocol control block structure (inpcb) at the transport layer to hold information such as TCP state information, IP address information, port numbers, IP header prototype and options, and a pointer to a routing table entry for the destination address. The TCP control block structure (tcpcb) contains TCP specific information such as timer information, sequence number information, flow control status, and out-of-band data. The combined size of these data structures for a single TCP connection may typically exceed 280 bytes. Although the data structures and allocation schemes used to accommodate TCP connection formation differ with various types of operating systems, generally every TCP connection establishment requires an allocation of significant memory resources.

When a SYN packet arrives at a port on which a TCP server is listening, the above-mentioned data structures are allocated. There is a limit on the number of concurrent TCP connections that can be in a half-open connection state, called the SYN–RECVD state (i.e., SYN received). When the maximum number of half-open connections per port is reached, TCP discards all new incoming connection requests until it has either cleared or completed some of the half-open connections. Typically, several ports can be flooded in this manner, resulting in degraded service or worse. Moreover, it should be appreciated that without a limit on the number of half-open connections, a different denial of service attack would result in which an attacker could request so many connections that the target machine's memory is completely exhausted by allocating data structures for half-open TCP connections. Table II illustrates the half-open connection

states that may be accommodated by various operating systems as follows:

TABLE II

Operating System	Backlog	Backlog + Queue
FreeBSD 2.1.5	n.a.	128
Linux 1.2.x	10	10
Solaris 2.4	5	n.a
Solaris 2.5.1	32	n.a.
SunOS 4.x	5	8
Windows NT 3.51	6	6
Windows NTw 4.0	6	6

With reference to FIG. **1**, detailed below is one typical example of what happens from the point of view of destination host D (alternatively designated as "server"):

1. A packet arrives at the destination machine when the TCP state machine is in the LISTEN state.
2. If the datagram checksum is incorrect, the packet will be discarded, and the client is expected to retransmit it.
3. The tcpcb associated with the connection is searched for. If it is not found, the server will discard the packet and will send a RST (i.e., inform the client that it reset the connection). If the tcpcb exists, but the TCP state machine is not in the LISTEN state, the server will discard the packet, but will not send a RST (this would, for example, be the case when the server is just coming up, but has not yet started listening).
4. If the SYN packet arrives for a socket that is in the LISTEN state, the above mentioned data structures will be allocated. However, the server will also set a flag indicating that it will destroy the connection and associated memory structures if it encounters an error. If the backlog queue is full, the server will consider this an error and will terminate the connection.
5. The packet will be ignored if it contains a RST. If it contains an ACK, it will be discarded and a RST sent to the other side. The packet will be discarded if the SYN bit is not set. Otherwise, the server copies information, such as the client's address information, into a buffer, connects its tcpcb to the client, and initializes its initial send sequence (ISS) number y.
6. The server now sends the second message of the three-way handshake (SYN_{x+1} and ACK_y) to the client. The state changes to SYN-RECVD. A connection establishment timer is started for this half-open connection. The connection remains in the SYN-RECVD state until either an ACK (the third message of the handshake) is received, resulting in the CONNECTED state, or a timer expires. This timer is usually set to about 75 seconds. During this period of time retransmissions of the first and second message of the three-way handshake may occur. When the timer expires, all memory structures associated with the connection are deallocated, and the server goes back to the LISTEN state.

As previously described, TCP implementations are designed with a small limit on how many half-open connections per port are possible at any given time. With reference to FIG. **2**, a SYN flood attack by attacker A is next described. Attacker A initiates a SYN flooding attack by sending many connection requests with spoofed source addresses to the victim machine, destination host D. In response, destination host D allocates resources to each half-opened connection. Once the limit of half-open connections is reached, destination host D refuses all successive connection establishment attempts—including any legitimate attempts. This condition exists until either the timer expires, or some connections are completed or reset. If the timer expires for a particular half-open connection, the host will reset the connection and release all the resources allocated for it. It should be appreciated that neither ongoing connection attempts nor connections that are already established are affected by this type of attack.

If a spoofed SYN packet contains the source address of a reachable IP source host S, that host will receive the second message of the three-way handshake generated by destination host D. Not expecting the SYN+ACK without having requested a connection, the source host S will send a RST packet to destination host D, and consequently cause destination host D to reset the connection. It is therefore in the interest of attacker A to forge source addresses that do not belong to hosts that are reachable from the victim destination host D.

If attacker A wants the denial of service condition to last longer than the timeout period, he needs to continuously keep requesting new connections from the victim destination host D. The amount of CPU and network bandwidth required by attacker A for a sustained attack is negligible.

Typical SYN flooding attacks can vary several parameters: the number of SYN packets per source address sent in a batch (=: batch size), the delay between successive batches (=: delay), and the mode of source address allocation (=: mode). For the purposes of SYN flooding, "spoofed" source addresses of the corresponding SYN packets generally include source addresses of source hosts S that are not reachable from destination host D, because the addresses are not yet allocated, assigned, or associated with source hosts S that are very slow in response, virtually or physically disconnected, or down. Possible modes of source address allocation include: a single address, a short list of addresses, or no list at all.

The single address approach takes as a parameter a single spoofed address that is used as the source address of all SYN packets. In the absence of any defense, this mode of attack is as effective as the other three modes described. For the short list mode, attacker A can generate a small pool of addresses and use them as source addresses to generate SYN packets. In the no list mode, attacker A can use a different, randomly generated source address for each successive batch of SYN packets.

Referring to FIG. **3**, network system **20** of one embodiment of the present invention is illustrated. System **20** includes a trusted, local area network **22** operatively coupled to an untrusted network **24**, such as the internet, by gateway **50**. Untrusted network **24** includes an attacker **40** and an external source host **30**. In practice, many more hosts would be present on network **24**, but are not shown to preserve clarity.

In addition to gateway **50**, monitoring resource **51** is also operatively coupled to network **22**. A monitoring program, monitor **52**, is configured for execution by monitoring resource **51** to protect other resources on network **22** from SYN flooding by sources from untrusted network **24**, such as from attacker **40**, and still facilitate establishment of connections with legitimate external sources, such as source host **30**. Among the resources protected by monitor **52** are a number of destination hosts **54**.

Destination hosts **54** are each operatively coupled to network **22**. While only three destination hosts **54** are shown in FIG. **3**, in practice many more destinations may be coupled to network **22** and correspondingly protected by monitor **52**. Also operatively coupled to network **22** is local

source host **56**. In practice, many more source hosts may also be present on network **22**, but are not illustrated to preserve clarity. Furthermore, with regard to network **22** and network **24**, it should be understood that designations of a host as a source or destination type are for clarity of explanation, and that in practice equipment coupled to network **22** may typically operate as both a source and destination relative to another host. In one nonlimiting example, network **22** is Ethernet based.

It is preferred that monitoring resource **51** be in the form of a programmable digital computer that is operable to execute monitor **52**. Monitoring resource **51** may be in the form of a single processing unit operatively coupled to network **22** or a distributed system of different units operatively coupled to network **22**, as would occur to those skilled in the art. Resource **51** includes various hardware elements such as one or more Input/Output (I/O) devices **53** and memory **55**. It is preferred I/O devices **53** include at least one means of administrative input to facilitate selective communication and control of monitor **52**. I/O devices **53** may include a keyboard, mouse, visual display, and/or printer to name only a few possibilities.

Programming for resource **51**, including monitor **52**, is encoded on a computer readable medium, such as memory **55**. Memory **55** may be of the random access, solid-state type, the optical type (such as a Compact Disk type), the electromagnetic type (such as a floppy or hard disk), or another type as would occur to those skilled in the art. Furthermore, memory **55** may be a combination of one or more of these types. As schematically shown in FIG. **3**, at least a portion of memory **55** may be in the form of a removable computer readable medium **58**, such as a disk or cartridge.

Also schematically represented in FIG. **3** is database **57** stored in memory **55**. Database **57** may be accessed by monitor **52** and is preferably initially installed, updated, and maintained by administrative input via I/O devices **53**.

Monitor **52** is preferably arranged to capture IP/TCP datagrams passing along network **22** regardless of source or destination, including any TCP packets involved in a three-way handshake with destinations **54**. Preferably, monitor **52** does not block or interfere with the transmission of packets to destination hosts **54**, but rather examines the packets and reacts by selectively generating packets for one or more of destination hosts **54**. Further, when a destination host **54** responds with a packet, such as a SYN+ACK packet, monitor **52** permits it to proceed. Monitor **52** is also arranged to asynchronously time selected events and respond to administrative inputs.

FIG. **4** is a flow chart illustrating routine **60** of monitor **52**. Routine **60** begins with stage **62**. Stage **62** initiates the detection and examination of TCP packets sent to destination hosts **54** along network **22**. After stage **62**, conditional **63** is encountered. Conditional **63** tests whether a new TCP packet has been sent to one of destination hosts **54** along network **22**. If a new TCP packet is not detected, then conditional **78** is encountered to determine whether monitoring is to continue. Generally, monitoring continues absent an administrative input to the contrary via I/O devices **53**. If monitoring is to continue, control loops back to conditional **63** to continue packet detection. If monitoring is not continued, routine **60** halts.

When conditional **63** results in the detection of a TCP packet, conditional **64** is next encountered. Conditional **64** tests whether the newly detected packet is of the SYN variety. If the detected TCP packet is not of the SYN variety, the new packet is submitted to state machine **100** in stage **76**. The operation of state machine **100** is described in detail in connection with FIG. **7** hereinafter.

If the newly detected TCP packet is of the SYN variety, then source address classification operation **65** is initiated. Operation **65** begins with accessing database **57** of monitor **52** at stage **66**. After accessing database **57**, operation **65** continues with conditional **68**. At conditional **68**, the source address of the newly detected TCP packet is compared to a set of entries in database **57** representative of acceptable addresses. If the source address of the newly detected packet is one of the acceptable addresses, classification operation **65** is exited and the acceptable source address is submitted to state machine **100** in stage **76** for continued monitoring. Correspondingly, routine **60** takes no further action for any packet having a source address classified as acceptable or "perfect," and presumes the three way handshake illustrated in FIG. **1** will be completed in the usual manner.

If, on the other hand, the source address of a newly detected SYN packet is not one of the acceptable addresses, control flows to conditional **70** to test whether the newly detected SYN packet has a source address that is unacceptable or "evil" as represented by one of another set of entries in database **57**. If so, classification operation **65** of routine **60** is exited and a RST packet is sent in stage **72** to the targeted destination host **54** that receives the SYN packet with an unacceptable address to close the corresponding TCP connection. By closing the connection, the RST packet frees resources of the targeted destination host **54**. FIG. **5** is a diagram illustrating the packet sequence observed for processing through state **72** for a SYN packet with an unacceptable source address. Any subsequent RST packets are ignored with regard to the FIG. **5** packet sequence. From stage **72**, routine **60** continues to stage **76**.

The two sets of entries in database **57** corresponding to acceptable and unacceptable addresses, respectively, are preferably input via one or more I/O devices **53**. These addresses may be updated as needs require. Preferably, the acceptable addresses include addresses of well-known, frequently used source hosts, such as local source host **56**, and the unacceptable addresses include as a minimum those addresses (or address ranges) that are reserved, impossible (such as net 0.0.0.0 or 127.0.0.0), or confirmed spoof addresses (such as nets 10.0.0.0, 172.16.0.0, and 192.168.0.0).

Source addresses of SYN packets that are neither of the acceptable or unacceptable type, as determined by accessing database **57**, are designated "suspect" addresses. For a suspect address, an ACK packet is sent to the targeted host destination **54** that has received the detected SYN packet. This ACK packet is arranged to appear as though is originates from the same source address as the detected SYN packet. Sending the ACK packet completes the three way handshake, moving the TCP connection from a half-open status (SYN-RECVD) to a completed status (CONNECTED). Changing a connection from half-open status to completed status removes it from the limited backlog queue of the targeted destination host **54** receiving the ACK packet, freeing it to establish another half-TCP connections. The diagram of FIG. **6** illustrates the sequence of packets corresponding to processing through stage **74** for a newly detected SYN packet with a suspect address. As shown in FIG. **6**, if an ACK packet is later received by the targeted destination host **54** from a legitimate host, such as source host **30**, it is simply ignored.

After stage **74**, the suspect address is submitted to state machine **100** in stage **76**. Referring to FIG. **7**, state machine **100** is shown in greater detail. State machine **100** is a process

of monitor **52** that further evaluates and reacts to selected TCP packets in accordance with source address classification by operation **65**.

Referring to state machine **100**, any detected TCP packet that has an acceptable source address is placed in acceptable address state **102**. State **102** is not conditionally coupled to any other states of state machine **100**—in other words, source addresses classified in state **102** remain in state **102**. Condition loop **104** of state **102** is triggered when either an ACK packet, a RST packet, or a SYN packet with the acceptable address is detected with monitor **52** via routine **60**. In response, the corresponding source address remains in state **102**. Accordingly, state **102** signifies that monitoring routine **60** does not interfere with the usual handling of TCP packets with an acceptable source address.

Any detected TCP packet that has an unacceptable source address is placed in unacceptable address state **105**. State **105** is not conditionally coupled to any other states of state machine **100**—in other words, source addresses classified in state **105** remain in state **105**. Conditional loop **106** is triggered whenever an ACK packet or a RST packet is detected that has an unacceptable address. In response, the corresponding source address remains in state **105**. Conditional loop **107** is triggered whenever a SYN packet with an unacceptable address is detected. SYN packet detection causes the transmission of an appropriately configured RST packet to close the connection in stage **72** as described in connection with FIG. **4**. Like loop **106**, there is no change of state caused by loop **107**. Accordingly, any attempts to request a connection with one of destinations **54** that has an unacceptable address result in closing the connection to release destination host **54** resources.

Generally, states **102** and **105** are symbolic of the manner in which each of two categories of addresses are treated after classification by operation **65**, and do not involve any appreciable amount of processing beyond the stages shown in FIG. **4**. The third category, suspect addresses, are further classified and processed in accordance with the conditionally coupled states **108** of state machine **100**. States **108** are initially engaged at null address state **110**. All source addresses that are not in state **102** or **105**, that is not included in database **57**, are initially considered to be in null address state **110**. Unlike acceptable and unacceptable addresses, a database entry for every suspect addresses is preferably not kept to conserve storage resources.

From the null address state **110**, transition path **112** is followed if a SYN packet for the suspect source address in the null state is detected and submitted to state machine **100** from routine **60**. In response, the source address is placed in new address state **114** and a record of the corresponding datagram is established, including information such as IP addresses, ports, and sequence numbers. This information is saved in case a RST packet needs to be sent to the corresponding source address in response to a later condition. A timestamp for the given source address is also established when the transition from null address state **110** to new address state **114** occurs. This timestamp is used in connection with certain timer events described in connection with other states **108**. New address state **114** corresponds to suspect source addresses for which there is too little information to determine if a connection with a destination host **54** should be allowed.

New address state **114** is coupled to conditional loop **116**. Loop **116** is triggered by the receipt of further SYN packets, symbolizing that new address state **114** is maintained in response to multiple SYN packets. In response to the triggering of loop **116**, current datagram information is recorded for use in case a RST packet needs to be sent to a corresponding destination host **54** in a different state. Transition path **118** conditionally couples new address state **114** and bad address state **120**. If a given source address remains in state **114** for a time period T1 determined by monitor **52** relative to its timestamp, an "expiry" timer event is generated. A change of state from new address state **114** to bad address state **120** along transition path **118** is triggered by an expiry event.

Conditional loop **122** is triggered by the detection of any successive SYN packets with the suspect source address in bad address state **120**. The triggering of conditional loop **122** causes a RST packet to be sent to the targeted destination host **54** in the manner described in connection with stage **72**. Accordingly, the connection associated with the source address in state **120** is closed, freeing resources of the targeted destination host **54**. Conditional loop **122** returns to bad address state **120** with each successive SYN packet corresponding to the suspect source address. As a result, suspect addresses that are likely to be spoofed reach bad address state **120** after first being monitored at the new address state **114** for the time period T1. The diagram of FIG. **8** illustrates the packet sequence for a suspect source address in bad address state **120** for which a RST packet is transmitted to the targeted destination host by triggering conditional loop **122** with another SYN packet. Preferably, time period T1 is set to a value significantly less than the default time-out duration commonly associated with the targeted destination host **54**. However, it should be appreciated that a time period T1 that is too low may adversely impact the formation of relatively slow legitimate connections.

On the other hand, the detection of a ACK packet or a RST packet for the suspect address in the bad address state **120** triggers a change of state along transition path **124** to new address state **114**. This change of state also causes the timestamp associated with the suspect source address undergoing the transition to be updated for timing another potential expiry timer event.

Besides bad address state **120**, new address state **114** is also conditionally coupled to good address state **128**. The good address state corresponds to suspect source addresses that are most likely legitimate. For a suspect address placed in the new address state **114**, a change of state to good address state **128** via transition path **126** is triggered by detection of an ACK packet or RST packet with that suspect source address. This change of state also results in the updating of the timestamp associated with the suspect source address.

Once in the good address state **128**, subsequent ACK or RST packets trigger conditional loop **130** which does not cause further changes of state. Subsequent SYN packets trigger conditional loop **132** that also does not cause a change of state. However, unlike conditional loop **130**, conditional loop **132** updates the timestamp for the corresponding suspect source address in the good address state.

If a suspect source address remains in the good address state **128** for a predetermined time period T2 without any TCP packet traffic having this address, a "staleness" timer event is generated that triggers a change of state from good address state **128** to new address state **114** along transition path **134**. Also, this event results in a fresh datagram information record and timestamp updating to prepare for the possibility of a later transition to bad address state **120** in response to generation of an expiry timer event while in new address state **114**. Thus in accordance with state machine **100**, a given suspect address that was once "good" can become "bad" in response to suspect TCP packet behavior.

Further, it should also be appreciated that the transition of a suspect address from null address state **110** to good address state **128** may occur directly when an associated ACK packet or a RST packet is detected, indicating a legitimate connection. This change of state is represented by transition path **136** and is also accompanied by timestamp updating. The operation of state machine **100** is arranged to halt in accordance with conditional **78** of routine **60**.

It should be appreciated that routine **60** classifies source addresses into one of a number of categories. Classification operation **65** classifies the source IP addresses of TCP packets in one of three categories: (1) acceptable, (2) unacceptable, or (3) suspect. State machine **100** further classifies suspect addresses as: (1) never seen (=: null); (2) correctly behaving (=: good) hosts; (3) potentially spoofed (=: new), or (4) as most certainly spoofed (=: bad).

In accordance with these classifications, monitor **52** sends a RST packet to unacceptable or bad addresses, closing corresponding connections to free resources of destination hosts **54**. Further, an ACK packet is sent for suspect source addresses to free resources of the destination hosts **54** by removing connections from a half-open backlog queue. Further, states **108** of state machine **100**, conditionally coupled by transition paths **112**, **118**, **124**, **126**, **134**, and **136**, permit suspect address classification to change based on observed behavior of network traffic and asynchronous events, such as expiry and staleness event timers.

In other embodiments of the present invention, an ACK packet transmission in stage **74** may not be performed or may only be selectively performed for suspect addresses. Additionally or alternatively, in one further embodiment the acceptable address classification is not performed. In still another embodiment, all addresses are considered suspect and correspondingly submitted to a state machine process rather than each being initially classified into one of a plurality of different categories by operation **65**. Furthermore, it is envisioned that other embodiments of the present invention may be applied to improve security of networks having a protocol other than TCP/IP.

Yet another embodiment of the present invention includes timing the frequency of unacceptable and/or bad source address SYN packets and using a conventional statistical model to predict arrival of the next spoofed SYN packet. A RST may be sent for any packets arriving in the spoofed time interval to enhance protection of destination hosts **54**. In still another embodiment suited to SYN flooding based on pseudorandom number generated addresses, routines may be implemented that detect use of pseudorandom sequences generated by various conventional generators to identify spoofed SYN packets. In other embodiments, as trusted address ranges are identified, they may be incorporated into the acceptable address testing of classification operation **65**. An additional embodiment of the present invention includes gathering statistics concerning good and bad address states to selectively add the corresponding good or bad source addresses as acceptable or unacceptable address entries in database **57**, respectively.

Another embodiment of the present invention includes a monitoring resource responsive to TCP connection packets to detect a source address for each and determine if the source address is of a suspect type. The monitoring resource is operable to permit one or more hosts to receive the TCP connection packets and is programmed with a state machine having a number of conditionally coupled states to process an address of the suspect type. The states include a new address state, a good address state, and a bad address state.

An additional embodiment of the present invention includes a computer readable medium encoded with a monitoring program to reduce performance degradation of a network caused by a denial of service attack with SYN flooding. The program is operable to detect each of a plurality of SYN packets sent on the network to one or more hosts and classify a source address for each of the SYN packets in one of a plurality of categories. A first one of the categories corresponds to a suspect source address and a second one of the categories corresponds to an unacceptable source address. The program defines a state machine process to monitor any address in the first one of the categories in accordance with at least three conditionally coupled states. The program is operable to send a corresponding RST packet to any of the one or more hosts receiving a SYN packet with an address classified in the second one of the categories.

Other embodiments include monitoring TCP connection status from an untrusted network to a number of destinations operatively coupled to the untrusted network to reduce a performance degradation caused by SYN packet flooding from the untrusted network. A TCP connection source address is classified in a good address state when an ACK packet or a RST packet is received from the untrusted network with the source address. The source address is reclassified from the good address state to a new address state after a first predetermined time period passes without receiving the ACK packet or the RST packet from the untrusted network. Furthermore, the source address is reclassified from the new address state to a bad address state after a second predetermined time period passes without receiving the ACK packet or the RST packet from the untrusted network. A RST packet is sent to a corresponding one of the destinations to close a respective connection for a source address in the bad address state.

All publications, patents, and patent applications cited in this specification are herein incorporated by reference as if each individual publications, patent, or patent application were specifically and individually indicated to be incorporated by reference and set forth in its entirety herein. While the invention has been illustrated and described in detail in the figures and foregoing description, the same is to be considered as illustrative and not restrictive in character, it being understood that only the preferred embodiment has been shown and described and that all changes, equivalents, and modifications that come within the spirit of the invention as defined by the following claims are desired to be protected.

What is claimed is:

1. A method, comprising:

monitoring TCP connection status from an untrusted network to a number of destination hosts operatively coupled to the untrusted network to reduce a performance degradation caused by SYN flooding from the untrusted network;

classifying a TCP connection source address in a good address state when an ACK packet or a RST packet is received from the untrusted network with the source address;

reclassifying the source address from the good address state to a new address state after a first predetermined time period passes without receiving the ACK packet or the RST packet from the untrusted network;

reclassifying the source address from the new address state to a bad address state after a second predetermined time period passes without receiving the ACK packet or the RST packet from the untrusted network; and

sending a RST packet to a corresponding one of the destination hosts to close a respective connection after the source address is classified in the bad address state.

2. A method, comprising:

monitoring a number of SYN packets sent from an untrusted network to a destination host to regulate spurious TCP connection attempts, the number of SYN packets each having a corresponding source address;

classifying the corresponding source address in one of a plurality of different categories, the different categories including a suspect source address category and an unacceptable source address category, said classifying including comparing the corresponding source address to a number of entries in a database;

sending a ACK packet to the destination host for each of the number of SYN packets having the corresponding source address classified in the suspect source address category to complete formation of a corresponding TCP connection with the destination host;

sending a RST packet to the destination host for the each of the number of SYN packets classified in the unacceptable source address category to close a respective connection with the destination host;

said monitoring including examining each of the corresponding source addresses in the suspect source addresses category in accordance with a state machine including at least three conditionally coupled states; and

wherein a one state of the at least three conditionally coupled states corresponds to a good source address, a second state corresponds to a new source address, and a third state corresponds to a bad source address, and the corresponding source address classified in the suspect source address category cannot change from the one state to the third state without first changing to the second state.

3. A method, comprising:

monitoring a number of SYN packets sent from an untrusted network to a destination host to regulate spurious TCP connection attempts, the number of SYN packets each having a corresponding source address;

classifying the corresponding source address in one of a plurality of different categories, the different categories including a suspect source address category and an unacceptable source address category, said classifying including comparing the corresponding source address to a number of entries in a database;

sending a ACK packet to the destination host for each of the number of SYN packets having the corresponding source address classified in the suspect source address category to complete formation of a corresponding TCP connection with the destination host;

sending a RST packet to the destination host for the each of the number of SYN packets classified in the unacceptable source address category to close a respective connection with the destination host;

said monitoring including examining each of the corresponding source addresses in the suspect source address category in accordance with a state machine including at least three conditionally coupled states;

a one state of the at least three conditionally coupled states corresponds to a good source address, a second state corresponds to a new source address, and a third state corresponds to a bad source address, and the corresponding source address classified in the suspect source address category cannot change from the one state to the third state without first changing to the second state; and

wherein the RST packet is sent to the destination host for any of the number of SYN packets having suspect sources addresses in the third state.

4. A method, comprising:

monitoring a number of SYN packets sent from an untrusted network to a destination host to regulate spurious TCP connection attempts, the number of SYN packets each having a corresponding source address;

classifying the corresponding source address in one of a plurality of different categories, the different categories including a suspect source address category and an unacceptable source address category, said classifying including comparing the corresponding source address to a number of entries in a database;

sending a ACK packet to the destination host for each of the number of SYN packets having the corresponding source address classified in the suspect source address category to complete formation of a corresponding TCP connection with the destination host;

sending a RST packet to the destination host for the each of the number of SYN packets classified in the unacceptable source address category to close a respective connection with the destination host; and

wherein TCP packets designating the designation host are received by the destination host.

* * * * *

Exhibit 1008

U.S. Patent 5,367,523 by Rong-Feng Chang et al. ("Chang")

Customer No.: 000027683

Haynes and Boone, LLP
IP Section
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Telephone [214] 651.5000
Fax [214] 200.0853

US005367523A

United States Patent [19]

Chang et al.

[11] Patent Number: 5,367,523

[45] Date of Patent: Nov. 22, 1994

[54] **ADAPTIVE RATE-BASED CONGESTION AND FLOW CONTROL IN PACKET COMMUNICATIONS NETWORKS**

[75] Inventors: **Rong-Feng Chang**, Chai-yi, Taiwan, Prov. of China; **John E. Drake, Jr.**, Pittsboro, N.C.; **Levent Gün**, Durham, N.C.; **Lap T. Huynh**, Cary, N.C.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **112,737**

[22] Filed: **Aug. 26, 1993**

[51] **Int. Cl.**[5] **H04J 3/22**

[52] **U.S. Cl.** **370/84**; 370/94.1

[58] **Field of Search** 370/17, 60, 60.1, 79, 370/84, 94.1, 94.2, 94.3

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,042,029	8/1991	Hayakawa	370/94.1
5,063,562	11/1991	Barzilai et al.	370/79
5,280,470	1/1994	Buhrke et al.	370/94.2

OTHER PUBLICATIONS

Z. Haas, "Adaptive Admission Congestion Control," *ACM SIGCOMM Computer Communications Review*, vol. 21, No. 5, pp. 58–76, Oct. 1991.

W. Dabbous, "On High Speed Transport Protocols," *IEEE*, Mar. 1989.

Primary Examiner—Douglas W. Olms
Assistant Examiner—Hassan Kizou
Attorney, Agent, or Firm—Gerald R. Woods

[57] **ABSTRACT**

An end-to-end, closed loop flow and congestion control system for packet communications networks exchanges rate request and rate response messages between data senders and receivers to allow the sender to adjust the data rate to avoid congestion and to control the data flow. Requests and responses are piggy-backed on data packets and result in changes in the input data rate in a direction to optimize data throughput. GREEN, YELLOW and RED operating modes are defined to increase data input, reduce data input and reduce data input drastically, respectively. Incremental changes in data input are altered non-linearly to change more quickly when further away from the optimum operating point than when closer to the optimum operating point.

22 Claims, 6 Drawing Sheets

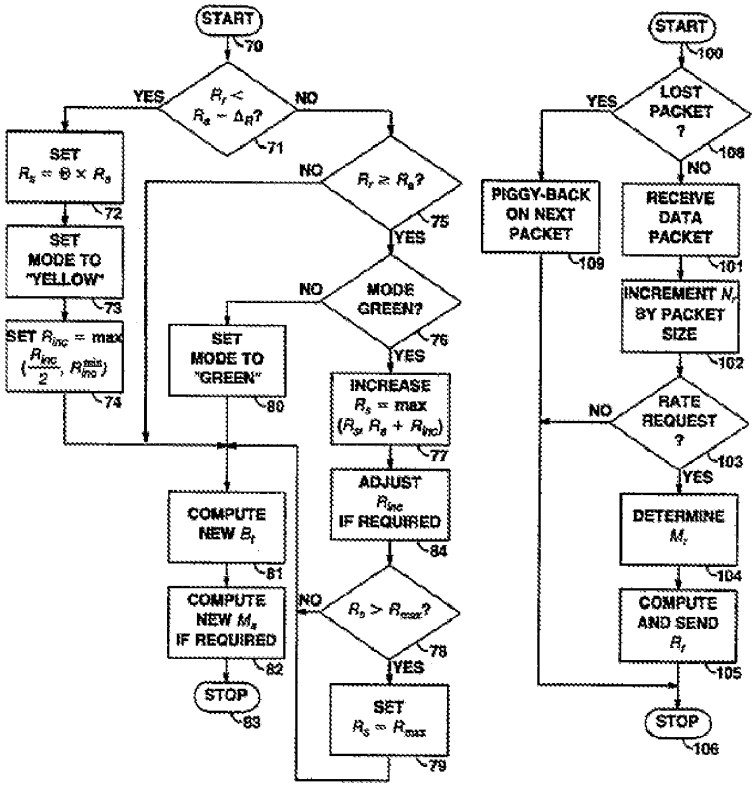


FIG. 1
PRIOR ART

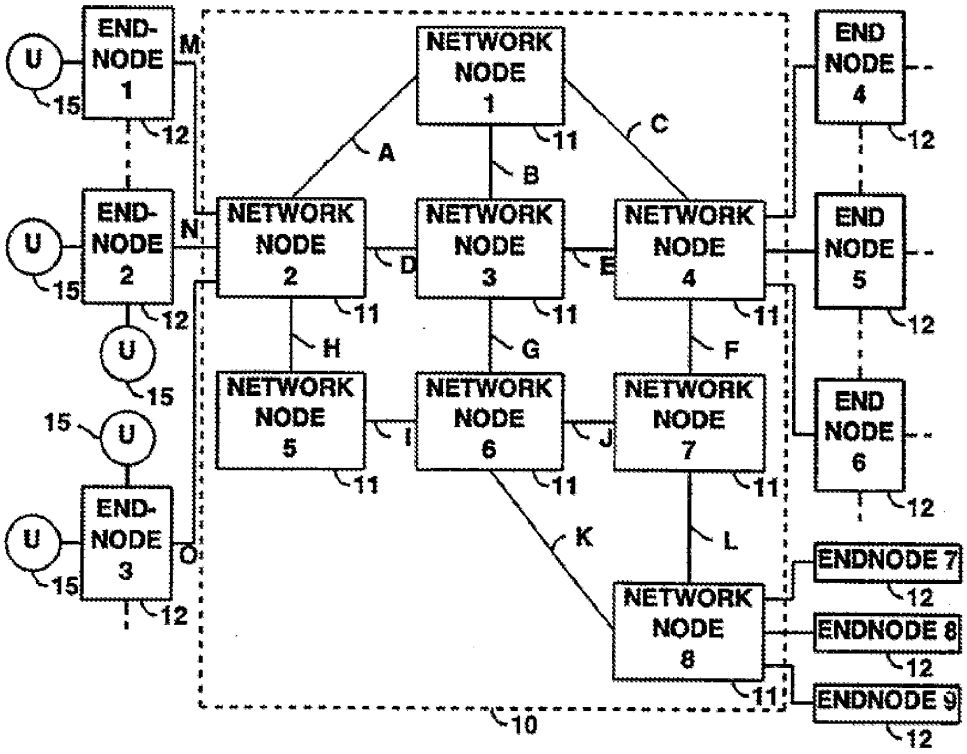


FIG. 2
PRIOR ART

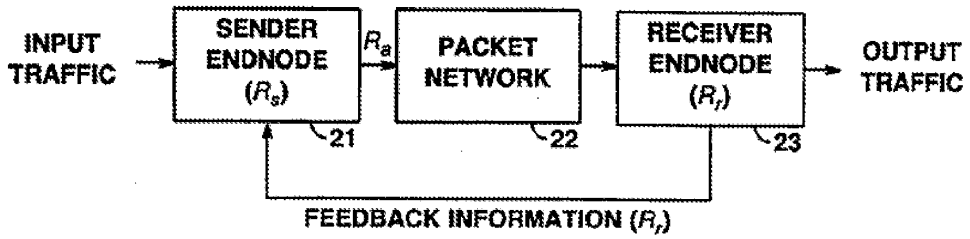


FIG. 3

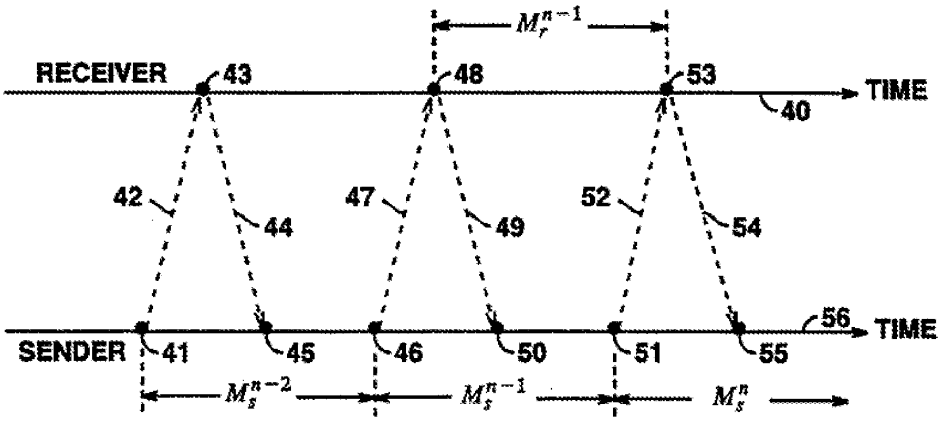

M_r^{n-1}
RECEIVER
43
48
53
TIME
40
42
44
47
49
52
54
56
TIME
SENDER
41
45
46
50
51
55
M_s^{n-2}
M_s^{n-1}
M_s^n

FIG. 4

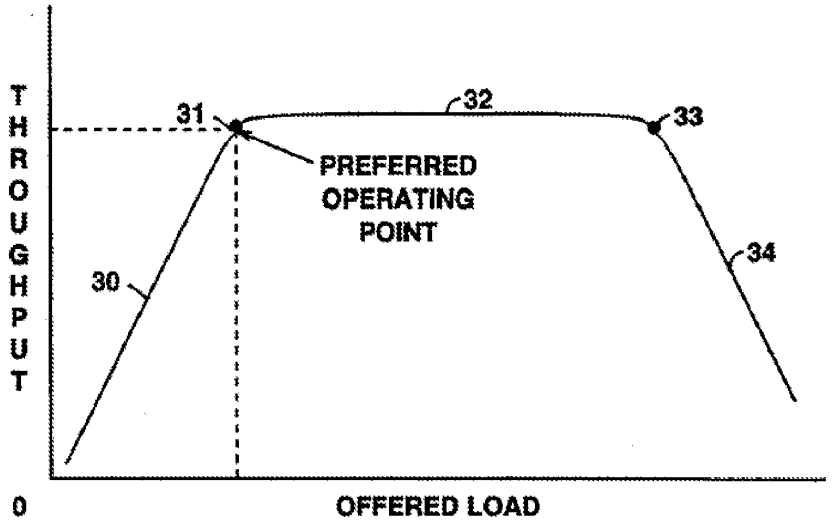

THROUGHPUT
31
32
33
PREFERRED
OPERATING
POINT
30
34
0
OFFERED LOAD

FIG. 5

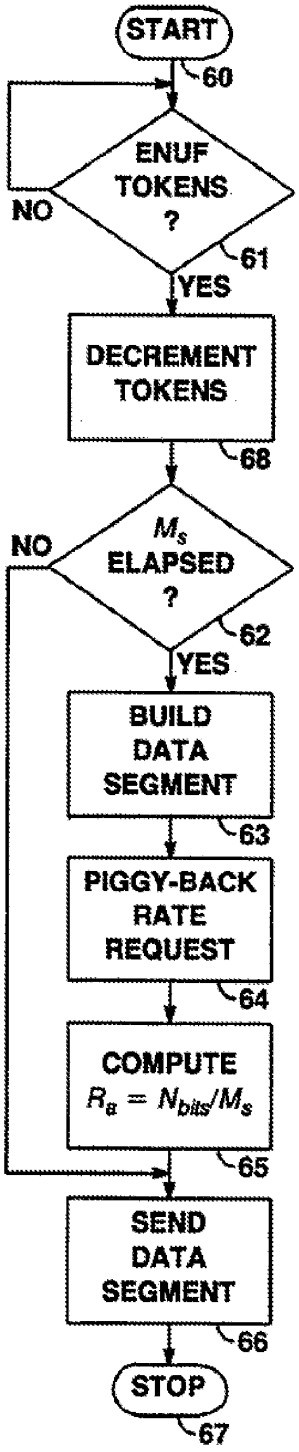

START
60
ENUF TOKENS ?
NO
61
YES
DECREMENT TOKENS
68
M_s ELAPSED ?
NO
62
YES
BUILD DATA SEGMENT
63
PIGGY-BACK RATE REQUEST
64
COMPUTE $R_a = N_{bits}/M_s$
65
SEND DATA SEGMENT
66
STOP
67

FIG. 7

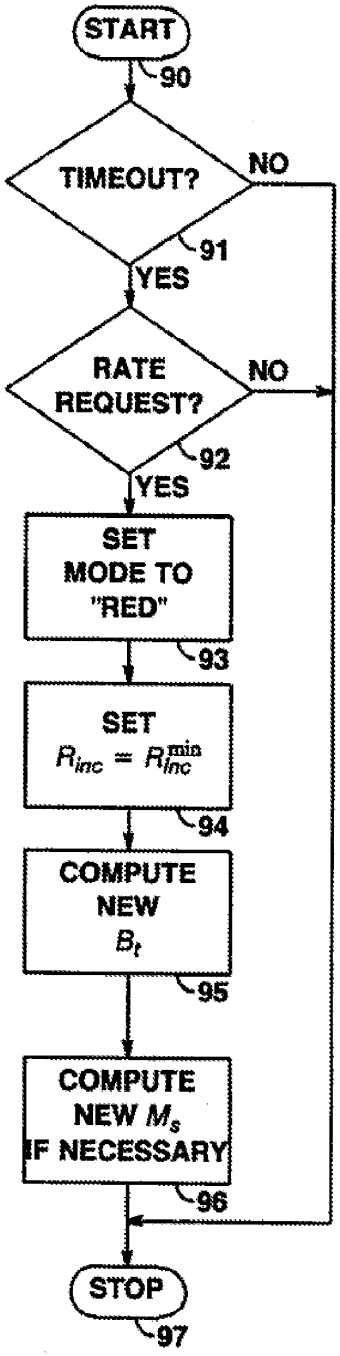
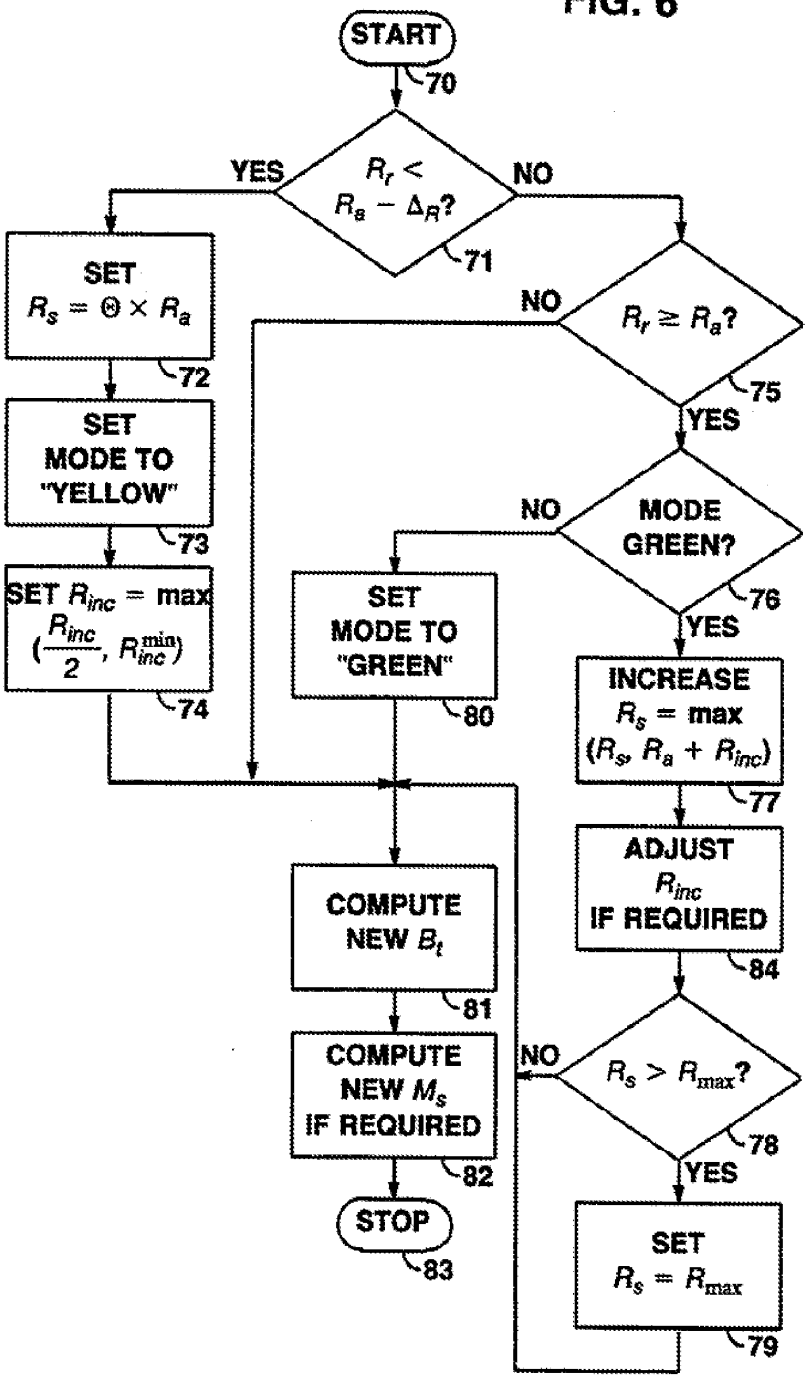

START
90
TIMEOUT?
NO
91
YES
RATE REQUEST?
NO
92
YES
SET MODE TO "RED"
93
SET $R_{inc} = R_{inc}^{min}$
94
COMPUTE NEW B_t
95
COMPUTE NEW M_s IF NECESSARY
96
STOP
97

FIG. 6


START
70
$R_r <$ $R_a - \Delta_R$?
71
YES
NO
SET $R_s = \Theta \times R_a$
72
SET MODE TO "YELLOW"
73
SET R_{inc} = max $(\frac{R_{inc}}{2}, R_{inc}^{min})$
74
$R_r \geq R_a$?
75
YES
NO
MODE GREEN?
76
YES
NO
SET MODE TO "GREEN"
80
INCREASE R_s = max $(R_s, R_a + R_{inc})$
77
ADJUST R_{inc} IF REQUIRED
84
COMPUTE NEW B_t
81
COMPUTE NEW M_s IF REQUIRED
82
$R_s > R_{max}$?
78
NO
YES
SET $R_s = R_{max}$
79
STOP
83

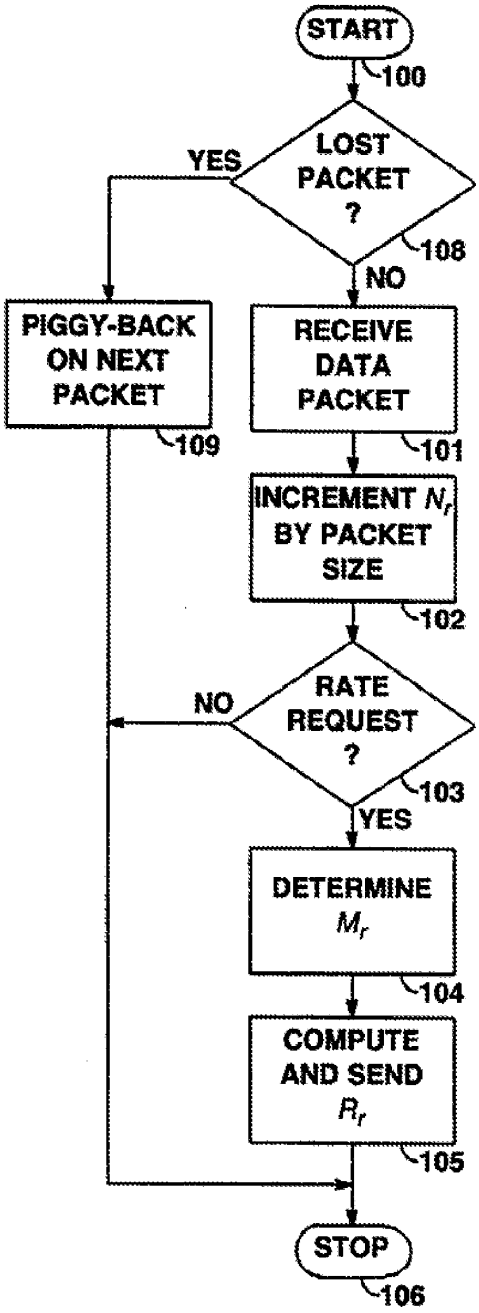

START
100
LOST PACKET ?
YES
NO
108
PIGGY-BACK ON NEXT PACKET
109
RECEIVE DATA PACKET
101
INCREMENT N_r BY PACKET SIZE
102
RATE REQUEST ?
NO
YES
103
DETERMINE M_r
104
COMPUTE AND SEND R_r
105
STOP
106

FIG. 8

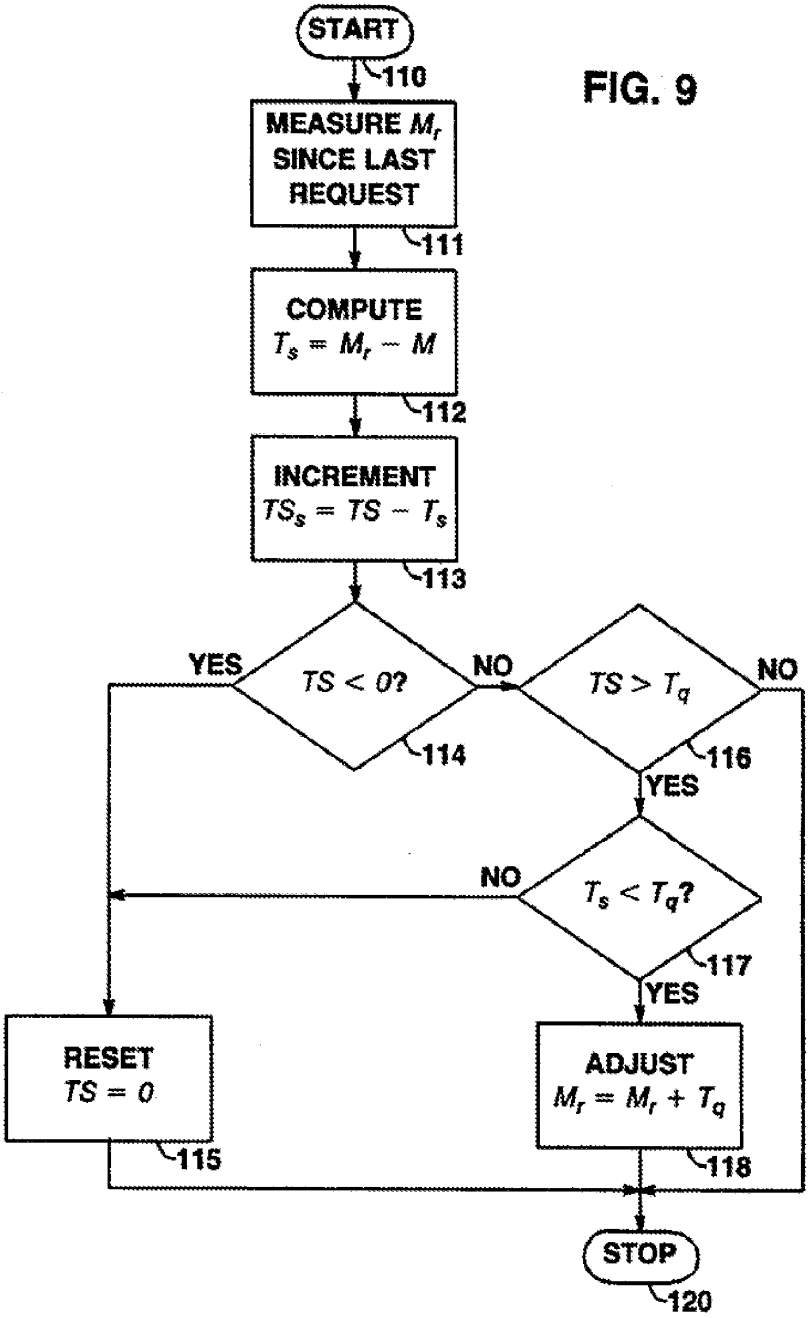

START
110
MEASURE M_r SINCE LAST REQUEST
111
COMPUTE $T_s = M_r - M$
112
INCREMENT $TS_s = TS - T_s$
113
YES
$TS < 0$?
NO
114
$TS > T_q$
NO
116
YES
NO
$T_s < T_q$?
117
YES
RESET $TS = 0$
115
ADJUST $M_r = M_r + T_q$
118
STOP
120

FIG. 9

ADAPTIVE RATE-BASED CONGESTION AND FLOW CONTROL IN PACKET COMMUNICATIONS NETWORKS

TECHNICAL FIELD

This invention relates to congestion and flow control in packet communications systems, and, more particularly, to improved end-to-end, closed loop, distributed control mechanisms for implementing such congestion and flow control.

BACKGROUND OF THE INVENTION

In packet communications systems, it is common to reserve bandwidth for high priority packets which are then transmitted in preference to lower priority packets. Such lower priority traffic therefore must be managed to take advantage of the bandwidth remaining after all of the higher priority traffic has been served. This remaining bandwidth can vary widely depending on the activity of the high priority traffic and it is therefore of considerable importance to manage the low priority traffic so as to optimize the use of the widely varying "left-over" bandwidth in the network, and, at the same time, avoid congestion in the network which reduces network throughput.

It has become common to utilize window-based flow control mechanisms to avoid congestion in the packet communications network. Such window-based mechanisms pro-allocate receiver buffer credits to packet sources and notify the corresponding sender how much data can be sent. Upon detection of congestion, either at an outgoing link (if the receiver is an intermediate node) or within a node, the receiver withholds buffer credits, forcing the sending partner to slow down the launching of packets or to stop transmission altogether. This process, also known as "back pressure" congestion control, is repeated hop by hop, eventually reaching the sources of traffic causing the congestion and forcing those sources to slow down.

Such window-based, back pressure mechanisms perform efficiently with low speed networks with reasonably high bit error rates. As networks move toward higher transmission speeds and more reliable transmission media such as optical fibers, the window-based mechanisms no longer perform adequately. The cost of such hop-by-hop mechanisms becomes prohibitively expensive and inefficient due to the fact that a sender can send an entire window's worth of data and be required to wait for the receipt of new buffer credits from the receiver before continuing. The resulting silent period is at least as long as two propagation delays and results in a direct loss of throughput during this silent interval. Furthermore, the window-based flow control does not smooth the transmission of data into the network and hence causes large oscillations in loading due to the clustering of packets, further degrading network performance. Using larger windows merely worsens the silent period throughput degradation.

In order to better accommodate modern high speed and reliable packet communications networks, it has been proposed to use an end-to-end congestion control mechanism which relies on the regular transmission of sample packets having time stamps included therein. One such mechanism is disclosed in "Adaptive Admission Congestion Control," by Z. Haas, *ACM SIGCOMM Computer Communications Review*, Vol 21(5), pages 58–76, October 1991. In the Haas article, successive time-stamped sample packets are used to calculate changes in network delays which are averaged to represent the state of the network. The averaged network delay is then used to control the admission of packets to the network by controlling the admission rate, either by controlling the inter-packet gap directly, or by adjusting the token rate in a standard leaky bucket scheme at the admission point.

One disadvantage of the Haas end-to-end congestion control mechanism is that Haas sends sampling packets at regular intervals regardless of the traffic load from a sender. Sending such sampling packets when the sender is idle is wasted effort and reduces the good throughput of the system. Furthermore, Haas must await the arrival of a plurality of sampling packets before initiating congestion control, thus providing too slow a response time to permit flow control as well as congestion control.

Another disadvantage of the Haas scheme is the so-called "accumulation effect." If the length of queues along the congestion path is built up gradually by small amounts, the overall delay can exceed the threshold allowed for the overall connection without being detected by the Haas endpoint detection scheme. The network can therefore become congested without timely correction when using the Haas congestion control scheme.

Yet another disadvantage of the Haas congestion control scheme is the fact that the inter-packet control gap is used to control the input packet rate. Sources of short packets are therefore penalized unfairly compared to sources of long packets when the inter-packet gap control technique of Haas is used to control congestion. Finally, and most importantly, the Haas congestion control scheme requires relatively frequent transmission of sampling packets to provide timely control information. Indeed, the overhead for such sampling packets can reach up to twenty percent of the entire throughput of the network, making the Haas congestion control scheme provide a lower throughput than an uncontrolled network when the traffic load is less than eighty percent. If the transmission rate of Haas' sampling packets were to be reduced to approximate the round trip delay period, on the other hand, the scheme simply would not work at all due to the paucity of control information available at the sender. That is, the averaging step used to reduce the noise in the control signal would make the scheme so unresponsive to the congestion to be controlled that the low sampling rate would be unable to correct the congestion.

SUMMARY OF THE INVENTION

In accordance with the illustrative embodiment of the present invention, the end-to-end, closed loop flow and congestion control in packet networks is accomplished by calculating, for every requested sampling interval, the lower of either the rate at which the receiver is accepting data from the network (congestion control), or the rate at which the receiver is able to deliver data to the end user (flow control). Thus, the network transfer rate is used to detect congestion while the end user acceptance rate is used to control the flow of data into the network so as not to exceed the user acceptance rate. The lower of these two rates is, of course, used to control entry into the network. Such a rate control mechanism is provided for each direction of transmission on each connection, and each end of a connection can request flow and congestion control information by

piggy-backing a request for control information on the current data packet being sent to a receiver. The receiver calculates the network transfer rate by dividing the number of bits received since the last request by the length of the interval since that last request. Rate information is also piggy-backed on data packets for return to the sender. Requests are timed to reflect the data transmittal rate into the network, requests being more frequent when traffic is heavy and less frequent when traffic is light, and thus conserving network resources when not needed for flow or congestion control.

More particularly, to implement the flow and congestion control of the present invention, three different operating modes are defined for the traffic flow control mechanism. That is, the sender uses the traffic feedback information from the receiver to set the sender into one or the other of the three operating modes which can be called the "green," "yellow" and "red" operating modes. The sender is set to the green mode when the rate information received from the receiver indicates that the receiver rate is equal to or greater than the average data input rate at the sender, less a sensitivity threshold used to compensate for the accumulated delay effect described above. The sender is set to the yellow operating mode when the received rate is less than the average data input rate less the sensitivity threshold, and the sender is set to the red operating mode if a timeout occurs following the expected waiting period for a reply to the most recent request for rate information.

The green operating mode is used when the receiver is able to receive data at a higher rate than the currently permitted rate. In the green operating mode, the sending rate can therefore be increased by some incremental amount which does not exceed the maximum rate which will be tolerated by the network, where the maximum rate is the rate negotiated by the sender when the connection is set up. If the current operating mode is other than green, and the most recent response from the receiver calls for a shift to the green operating mode, this shift is delayed at least one request-response period. This requirement for confirmation of shifts to the green operating mode by at least two responses calling for such a shift to the green operating mode prevents oscillations in operating mode due to transients in the data rates.

If the receiving rate returned in the response message is slower than the current sending rate by more than the same sensitivity threshold, the operating mode is set to yellow. In the yellow operating mode, the sending rate is reduced by some reduction factor, preferably a reduction factor between 0.9 and 1.0. In the yellow operating mode, it is assumed that the network is saturating and can carry no more traffic. The sender continues in the yellow operating mode, reducing the sending rate for each new response message indicating a lower receiving rate than the current sending rate (less the sensitivity threshold) until at least two responses are received dictating the need to shift to the green operating mode.

If a response timeout occurs while the sender is waiting for a response to its most recent request for rate information, the sender immediately assumes that congestion has delayed the response and the sending rate is therefore cut drastically, e.g., by half, or to the latest received receiving rate, whichever is lower. The sender remains in the red operating mode until two or more request responses call for a return to the green operating mode.

In accordance with one feature of the present invention, the incremental increase in the sending rate during the green operating mode can be defined to be non-linear to permit rapid increases in sending rate until saturation is approached, and thereafter, the incremental increases can be kept small to avoid wide fluctuations around a desirable operating point. Conversely, incremental decreases in the sending rate during the yellow operating mode can likewise be defined non-linearly to permit small decreases for small departures from an optimal operating point and ever larger decreases as the departure from the optimal operating point itself increases in magnitude.

The flow and congestion control mechanism of the present invention is highly adaptive to network conditions in that it is responsive to the offered traffic load and to the congestion, and results in maximizing throughput of the network while at the same time minimizing network congestion. The flow control mechanism of the present invention also smooths the input traffic to discourage very large data bursts and the resulting large queues in the network. Finally, the rate-based control mechanism of the present invention provides equal and fair access to all connections regardless of their relative average packet length.

A major advantage of the congestion control scheme of the present invention over prior art schemes such as Haas' is the ease with which the scheme of the present invention can be implemented and the low overhead, in both processor cycles and in network bandwidth, with which that implementation can be operated. Moreover, the same mechanism which provides congestion control also provides flow control, thereby avoiding the necessity of a separate flow control mechanism for each connection. The flow and congestion control mechanism of the present invention is overlaid on all connections at the connection source and destinations for each connection and requires no changes in the operation of the network itself. That is, the flow and congestion control mechanism of the present invention is entirely implemented at user access endnodes of the network. Moreover, the control mechanism of the present invention can be realized either in the transport layer of the network protocol or in the network layer, providing great flexibility for networks of differing architectures. Piggy-backing rate requests and responses on regular data packets both conserves network resources and forces rate control message interchanges to reflect the data packet rate.

Finally, the flow and congestion control mechanism of the present invention can be readily modified to provide protection against congestion arising from the accumulation effect. In particular, by keeping track of the delay accumulations in a connection, the reported receiving rate can be adjusted to reduce the accumulated delays. In one embodiment, a reduction in the receiving rate is accomplished indirectly by extending the measuring period over which the receiving rate is calculated, thereby causing the receiving rate to be less than it would otherwise be.

BRIEF DESCRIPTION OF THE DRAWINGS

A complete understanding of the present invention may be gained by considering the following detailed description in conjunction with the accompanying drawings, in which:

FIG. 1 shows a general block diagram of a packet communications system in which the flow and conges-

tion control mechanism of the present invention might find use;

FIG. 2 shows a general block diagram of a closed loop, end-to-end control mechanism in which the flow and congestion control scheme of the present invention might be implemented;

FIG. 3 shows a graphical representation of a typical exchange of requests for rate feedback information from a sender and the responses to such requests from the receiver, implementing the dynamic flow and congestion control scheme of the present invention;

FIG. 4 shows a graphical representation of the typical traffic saturation curve for a packet communications network such as that shown in FIG. 1 and indicating the preferred operating point on the saturation curve;

FIG. 5 shows a flow chart of the operation of a sending endnode in generating and launching a request for flow and congestion control rate information from the corresponding sending endnode in the packet network of FIG. 1, implementing the dynamic control scheme of the present invention;

FIG. 6 shows a flow chart of the operation of a sending endnode in response to the receipt of a response to a request for rate information from the corresponding receiving endnode in the packet network of FIG. 1, implementing the dynamic control scheme of the present invention;

FIG. 7 shows a flow chart of the operation of the sending endnode in response to the time out of the waiting period for receipt of a response to a request for rate information from the corresponding receiving endnode in the packet network of FIG. 1, implementing the dynamic control scheme of the present invention;

FIG. 8 shows a flow chart of the operation of the receiving endnode in responding to a request for rate information from the corresponding sending endnode in the packet network of FIG. 1, implementing the dynamic control scheme of the present invention; and

FIG. 9 show a flow chart of the operation of the receiving endnode in calculating the measurement period used for determining the rate information which is returned to the corresponding sending endnode in the packet network of FIG. 1, implementing the dynamic control scheme of the present invention.

To facilitate reader understanding, identical reference numerals are used to designate elements common to the figures.

DETAILED DESCRIPTION

Referring more particularly to FIG. 1, there is shown a general block diagram of a packet transmission system **10** comprising eight network nodes **11** numbered 1 through 8. Each of network nodes **11** is linked to others of the network nodes **11** by one or more communication links A through L. Each such communication link may be either a permanent connection or a selectively enabled (dial-up) connection. Any or all of network nodes **11** may be attached to end nodes, network node 2 being shown as attached to end nodes **1**, **2** and **3**, network node 7 being shown as attached to end nodes **4**, **5** and **6**, and network node 8 being shown as attached to end nodes 7, 8 and 9. Network nodes **11** each comprise a data processing system which provides data communications services to all connected nodes, network nodes and end nodes, as well as providing decision points within the node. The network nodes **11** each comprise one or more decision points within the node, at which point incoming data packets are selectively routed on one or more of the outgoing communication links terminated within that node or at another node. Such routing decisions are made in response to information in the header of the data packet. The network node also provides ancillary services such as the calculation of new routes or paths between terminal nodes, the provision of access control to packets entering the network at that node, and the provision of directory services and topology database maintenance at that node.

Each of end nodes **12** comprises either a source of digital data to be transmitted to another end node, a utilization device for consuming digital data received from another end node, or both. Users **15** of the packet communications network **10** of FIG. 1 utilize an end node device **12** connected to the local network node **11** for access to the packet network **10**. The local network node **11** translates the user's data into packets formatted appropriately for transmission on the packet network of FIG. 1 and generates the header which is used to route the packets through the network **10**.

In order to transmit packets on the network of FIG. 1, it is necessary to calculate a feasible path or route through the network from the source node to the destination node for the transmission of such packets. To avoid overload on any of the links on this route, the route is calculated in accordance with an algorithm that insures that adequate bandwidth is available for the new connection. One such algorithm is disclosed in the copending application, Ser. No. 07/874,917, filed Apr. 28, 1992, now U.S. Pat. No. 5,233,604, issued Aug. 3, 1993 and assigned to applicant's assignee. Once such a route is calculated, a connection request message is launched on the network, following the computed route and updating the bandwidth occupancy of each link along the route to reflect the new connection. Traffic offered to the network of FIG. 1 may be of difference Classes of Service (COSs) or require different Quality of Service (QOSs). That is, the connection calculated for different sources may require different levels of bandwidth, different delay guarantees, or different loss probabilities. Real time signals such as voice and television, for example, require higher levels of priority than accounting information or other non-real time signals. Such connection characteristics are negotiated along with the bandwidth requirements and enforced by assigning higher priorities to the transmission of the real time signals and, under congestion, discarding the non-real time packets before discarding the real time packets.

In packet communications networks such as that shown in FIG. 1 it is necessary to manage the traffic so as to enforce the Class of Service and Quality of Service guarantees for the high priority packet sources while, at the same time, to optimize the transmission of the lower priority traffic through the network. It is assumed that the higher priority controls guarantee the transmission of the high priority traffic so as to meet the negotiated service level. The present invention is directed at the control of the lower priority traffic so as to optimize the transmission of this low priority traffic and, at the same time, to avoid congestion in the network.

In FIG. 2 there is shown a general block diagram of an adaptive rate based flow and congestion control mechanism in accordance with the present invention for controlling the traffic in a packet communications network such as that shown in FIG. 1. The control mechanism of FIG. 2 comprises a sender endnode **21** and a receiver endnode **23**, both connected to packet network **10** (FIG. 1). Endnodes **21** and **23** are, of course two of

the endnodes 12 of FIG. 1 and represent the endnodes at the end of a connection between a sending user 15 and a receiving user 15. Indeed, two flow and congestion control mechanisms such as that shown in FIG. 2 is provided for each connection through network 10 and, one for each direction of transmission on a connection. Before proceeding to a detailed discussion of the operation of FIG. 2, the following definitions will be made to assist in this discussion.

PARAMETER DEFINITIONS

- R_s is the instantaneous maximum bit rate at which the sender endnode 21 is allowed to transmit data into the network 10.
- R_a is the average actual bit rate at which data is sent into the network 10 by the sender endnode 21 in a particular measurement interval. Generally, $R_a \leqq R_s$ because there are gaps between packets when sender endnode 21 has nothing to send.
- R_r is the minimum of two rates; first, the rate at which the receiver endnode 23 accepts data from network 10; and, second, the rate at which the user connected to endnode 23 can process data. Both values are measured by receiving endnode 23 and only the minimum rate of the two is sent back to sender endnode 21 as receiver rate R_r.
- M_r is the length of the measurement interval at the receiving endnode 23. M_r is used in computing the rate measurement R_r to be returned to the sending endnode 21.
- B_s is the number of bits that sender endnode 23 is allowed to send continuously in a single burst at the rate R_s.
- B_t is the burst time for the burst B_s, i.e., $B_t = B_s/R_s$.
- M_s is the length of the measurement interval at the sending endnode 21. M_s is used in computing the actual rate R_a of transmission from endnode 21, i.e., $R_a = N_s/M_s$, where N_s is the number of bits sent by sender endnode 21 during the measurement interval M_s. Measurement intervals of length M_s occur in a serial sequence 1, 2, 3, . . . , n, where n represents the nth interval in this sequence. M_s is set independently for each connection and is on the order of magnitude of the round trip delay for the connection.
- T_{out} is the time-out value for the normally expected round trip delay of transmissions from the sender endnode 21 to the receiver endnode 23 and back again. If the actual round trip delay exceeds T_{out}, it is assumed that the path is congested and that the packet has either been dropped en route or that the packet has been delayed in a long queue at an intermediate node. T_{out} can a fixed constant which is calculated at the time a connection is being set up or can be estimated dynamically from network loading parameters.
- T_y is network time delay for a rate request message experienced by the message in its transmission between endnode 21 to receiving endnode 23. T_y can be thought of as the network queuing time for a rate request message, and varies over time.
- T_q is the maximum network queuing time allowed for a given connection along the connection path through the network. T_q is the sum of the individual queuing times for each hop along the connection path and is computed at the time the connection is set up, based on the Quality of Service requirements. T_q is used as a threshold not to be exceeded in managing the traffic in the network in accordance with the present invention.

The closed loop feedback control mechanism of FIG. 2 operates generally to return to the sending endnode 21 a value for R_r which is the minimum of two rates. One of these rates is the rate at which the receiver is accepting data from the network; the other rate is the rate at which receiver endnode 23 can deliver data to the end user. The former indicates the status of the path through the network and is used by the sending endnode 21 to regulate the input traffic so as to avoid overloading the path and thereby causing congestion in the network. The latter is used by the sending endnode 21 to avoid overloading the receiving user's capacity to receive data, thereby providing end-to-end flow control. Using this information, the sending endnode 21 can determine when congestion is about to develop in the network, and take appropriate action to avoid such congestion, or to detect overload of the receiving facilities and to slow traffic so as to match the reception capacity of the receiving user. When significant congestion actually does occur, sending endnode 21 takes even more drastic action to return the network load to a more optimum operating point.

More particularly, at the end of the nth measurement interval of length M_s, sending node 21 sends, piggy-backed on the current data packet being sent to receiving endnode 23, a request for rate information from receiving endnode 23. When this request is received by receiving endnode 23, the receiving rate $R_r(n)$ is calculated. $R_r(n)$ is the lower of either the number of bits received from network 10, or the number of bits delivered to the user, divided by the time interval M_r since the last (n−1)th request. The calculated receiving rate $R_r(n)$ is returned to the sending endnode 21, for example, by piggy-backing this rate information on a normal data packet being returned to endnode 21. These message exchanges can be better seen in FIG. 3.

Referring more particularly to FIG. 3, there are shown two time lines 40 and 56, corresponding, respectively, to time at the receiving endnode 23 of FIG. 2 (time line 40) and to time at the sending endnode 21 of FIG. 2 (time line 56). At time 41, sending endnode 21 sends a rate request message 42 to receiving endnode 23 for rate information, piggy-backed on the current data packet. This request is received at receiving endnode 23 at time 43 and the minimum flow control or congestion control rate R_r is sent on rate response message 44 back to sending endnode 21, also piggy-backed on a data packet. This rate response message 44 is received at sending endnode 21 at time 45. At a future time 46, sending endnode 21 sends another rate request message 47 which is received at receiving endnode 23 at time 48 which then launches a second rate response message 49 which is received at sending endnode 21 at time 50. Yet another future time 51, sending endnode 21 sends another rate request message 52 which is received at receiving endnode 23 at time 53 which then launches another rate response message 54 which is received at sending endnode 21 at time 55. The exchange of rate request messages and rate response messages continues between endnodes 21 and 23 so long as the data source at endnode 21 remains active. (Of course, a similar exchange of rate requests and rate responses takes place between these two endnodes to control traffic flow and congestion in the opposite direction.)

As shown in FIG. 3, the sending measurement intervals M_s extend between times 41-46, 46-51 and time 51

to the next request, not shown in FIG. 3. Similarly, the receiving measurement intervals M_r extend between times 43–48 and 48–53. The value of M_s determines the level of adaptability of the control mechanism; the smaller M_s, the tighter the control. On the other hand, the smaller M_s, the higher the cost in processing. The cost of transmission is virtually insignificant since the rate requests and rate responses can be piggy-backed on user data packets. It should be noted that M_s also depends on the sending rate for data packets. The higher the data sending rate, the lower the value of M_s, and the lower the data sending rate, the longer the interval M_s becomes, request messages stopping altogether when the data sending rate is zero. In general, the sending measurement interval M_s is set to be on the order of a single round trip delay for information between sending endnode 21 of FIG. 2 and receiving endnode 23. The response of sending endnode 21 to the rate response messages from receiving endnode 23 can be better understood by considering FIG. 4 The sending rate R_s controls the bit rate at which data packets are transmitted from sending endnode 21. If the control mechanism is a leaky bucket mechanism, the sending rate R_s is used as the green token source rate γ with the green token pool size equal to B_s, as taught in the copending application Ser. No. 07/943,097, filed Sep. 10, 1992, U.S. Pat. No. 5,311,513, and assigned to applicants' assignee.

In FIG. 4 there is shown a graphical representation of the offered load for a given path through a packet communications network such as network 10 of FIG. 1 versus the data throughput for that path. As can be seen in FIG. 4, the throughput is equal to the load for a linear portion 30 of the saturation curve of FIG. 4. At some knee point 31, saturation occurs in the network due to the fact that essentially all network resources are filled. For a portion 32 after knee 31, of the curve of FIG. 4, increase in offered load produce no increase in throughput, and path congestion begins to occur. At a cliff point 33 thereafter, the portion 34 of the curve of FIG. 4 shows that a decrease in throughput occurs for further increases in offered load due to the fact that many packets are dropped and most of the network resources are consumed in retransmitting packets. So-called "reactive" congestion control depends on detecting congestion and reacting thereto, usually by detecting the timeout of an expected message transfer time. Prior art congestion control schemes tended to be reactive in nature. "Preventive" congestion control, on the other hand, attempts to detect the approach to congestion and reduce the offered load to prevent congestion in the first place. The present invention is a preventive congestion control scheme.

It is clear from the saturation curve of FIG. 4 that the preferred operating point is the knee point 31 of FIG. 4. Below this offered load, the path is capable of handling a greater load than is being offered, and efficiency would increase if the offered load were to increase. On the other hand, loads greater than that represented at knee point 31 would produce no additional throughput and would simply increase the congestion in the network until throughput started to decrease at point 33. The goal of the congestion control mechanism of the present invention, then, is to maintain all paths through the network at or near the optimum operating point 31 of FIG. 4. That is, if the feedback rate indicates that the monitored path is below point 31, the sending endnode 21 increases the sending rate to bring the sending rate up closer to point 31. Conversely, if the feedback rate indicates that the monitored path is above point 31, the sending node 21 reduces the sending rate to bring this sending rate down closer to point 31. To assist in this control function, three different operating modes are defined for determining how the sending rate R_s is to be adjusted in response to the varying values of the receiver rate R_r.

In particular, the sending node 21 is set to a first mode, called the GREEN mode, when the receiver rate R_r is equal to or greater than the sender rate R_s by some incremental amount called the sensitivity threshold. That is, set the mode to GREEN if $R_r \cong R_s - \Delta_R$, where Δ_r is the sensitivity threshold. In the GREEN mode, the sending rate is increased to bring it closer to point 31 of FIG. 4. In the GREEN mode, the monitored path is capable of handling more offered load than it is now experiencing and hence the load can be increased. The sensitivity threshold will be further discussed below.

The sending node is set to a second operating mode, called the YELLOW mode, when the receiver rate R_r is less than the sender rate R_s, less the sensitivity threshold, i.e., $R_r < R_s - \Delta_R$. In the YELLOW mode, the sending rate is decreased to bring it closer to the point 31 of FIG. 4. In the YELLOW mode, the monitored path is becoming congested due to saturation, and cannot further increase throughput. The sending rate (offered load) is therefore reduced to reduce the congestion.

The sending node is set to a third operating mode, called the RED mode, when congestion has become so severe that loss of throughput is possible. The RED mode is entered when a timeout period (T_{out}) expires while waiting for a response to the most recent rate request. During the GREEN and YELLOW modes, the sending rate is changed slowly to smooth the input traffic. In the RED mode, the sending rate is reduced drastically to rapidly move the offered load back towards the optimum operating point 31.

The amount of increase in the sending rate during the GREEN mode is termed the increase rate R_{inc}. That is, every time the receiver rate R_r is equal to or higher than the sending rate R_s (less the sensitivity threshold Δ_R), the sending rate is increased by R_{inc}, i.e., $R_s \rightarrow R_s + R_{inc}$. R_s, however, is never allowed to increase beyond the initially negotiated rate R_{max} for that connection, reflecting the negotiated Quality of Service (QOS). The sending rate R_s therefore remains at R_{max} so long as the operating mode remains GREEN. In order to reduce oscillations in the system, the switch to the GREEN mode can be delayed until at least two successive rate messages require a switch to the green mode, i.e., two successive receiver rate messages with $R_r \cong R_r$. Note that R_{inc} can be defined non-linearly to be large initially to quickly bring the sending rate up to the optimum point 31. Once point 31 is reached, further increases in the sending rate are kept small to avoid wide fluctuations around the preferred operating point 31. If more bandwidth becomes available in the network (reduced congestion), the value of R_{inc} can be gradually increased again to take advantage of this newly available capacity.

If the receiver rate R_r is lower than the sender rate R_s by more than the sensitivity threshold ($R_r < R_s - \Delta_R$), the YELLOW mode is invoked and the sending rate R_s reduced. In the YELLOW mode, the sending rate is reduced by a weighting factor Θ, i.e., $R_s \rightarrow \Theta \times R_s$, where Θ is a value between 0.9 and 1.0 ($0.9 < \Theta < 1.0$), based on heuristics or modeling of the system controlled. In the YELLOW operating mode,

the path is saturated (congestion control), or the receiving user cannot handle the receiver rate (flow control).

When a rate response time-out occurs, the RED mode is invoked and the sending rate R_s is reduced drastically (or set to the receiver capability rate, whichever is lower). For example the sending rate can be reduced by half in the RED mode ($R_s \rightarrow 0.5 \times R_s$). It is necessary in the RED mode to reduce the offered load quickly and extensively to rapidly reduce the congestion.

While the dynamic flow and congestion control mechanism of the present invention could be implemented by special purposes circuitry, the preferred embodiment is implemented by programming a general purpose computer to carry out the scheme. The attached APPENDIX provides detailed pseudo code for implementing the inventive control mechanism, while FIGS. **5** through **9** comprise flow charts of this implementation. The flow charts of FIGS. **5-9** (and the pseudo code of the APPENDIX) are event driven in that the procedures of the flow charts and pseudo code are initiated by events such as the receipt of a data packet or the assembly of a packet for transmission.

In FIG. **5** there is shown a flow chart of the data sending algorithm at the sending endnode **21** of FIG. **2**. Starting at start box **60**, decision box **61** is entered where it is determined if enough green tokens have been accumulated to send a data packet. If not, decision box **61** is re-entered to await the accumulation of sufficient green tokens. If enough green tokens have been accumulated, box **68** is entered to decrement the green token count by the number of bits in the packet to be transmitted. Decision box **62** is then entered where it is determined if the sending measuring time M_s has elapsed. If not, box **66** is entered where the next data segment is transmitted to the receiving endnode. The process then terminates in stop box **67**. If M_s has elapsed, box **63** is entered where the next data segment is constructed for transmission into the network **10** (FIG. **1**). In box **64**, a rate request is piggy-backed on the data segment constructed in box **64**. This rate request need be no more that setting a single rate request bit in the header of the data segment to the rate request value. e.g., "1". The rate request may also include the value of M_s (FIG. **2**), if the accumulation effect is to be corrected as discussed in connection with FIG. **9**. Box **65** is then entered where the value R_a of the actual data transmission rate is calculated for the measurement interval M_s. This value will be used in the flow chart of FIG. **6** in computing the new sending rate R_s when the rate response message is received. Box **66** is then entered where the data segment is launched on the network **10** (FIG. **1**), using the currently operative sending rate R_s. The process terminates in stop box **67**.

In FIG. **6** there is shown a flow chart of the processing which takes place at the sending endnode **21** upon receipt of a new rate response message from the receiving endnode **23**. Starting in start box **70**, decision box **71** is entered to determine whether the receiving rate R_r, received in the rate response message, is less than the actual transmission rate R_a and the sensitivity threshold Δ_R. The value of Δ_R is a compromise between control instability (Δ_R too small and the control mechanism reacts too quickly to any queuing delay in the network **10**) and operation near the cliff **33** of FIG. **4** (Δ_R too large, resulting in marginal congestion). Δ_R, for example can be calculated as

$$\Delta_R = \frac{T_q}{M_s + T_q} R_a$$

If the value of R_r is less than R_a less Δ_R, as determined in decision box **71**, box **72** is entered where the data sending rate R_s is reduced by a factor Θ. The value of Θ is based on the network topology of network **10** and is generally between 0.9 and 1.0, e.g., 0.95. In box **73** the mode is set to "YELLOW" to indicate that the sending rate is being reduced to avoid congestion (congestion control) or to avoid overloading the receiving user capacity (flow control). Box **74** is then entered to calculate the value for R_{inc}, i.e., the incremental rate at which the sending rate will be increased next time the receiving rate R_r justifies sending rate increases. As shown in box **74**, the previous value of R_{inc} is reduced by half, but not below some minimum value R_{inc}^{min}. The value of R_{inc}^{min} is set to insure a significant increase in the sending rate R_s and has a value on the order of a few hundred bits per second to a few thousand bits per second. Box **81** is then entered where a new value of burst time B_t is calculated, where $B_t = B_s/R_s$. That is, the burst time must be update each time the sending rate is changed to maintain a constant number of bits in a single burst. The burst time B_t is used to control the transmission of data from sending endnode **21** in FIG. **2**. Box **82** is thereafter entered to compute a new sending node measuring period M_s, if required. That is, M_s can be reset based on a dynamic measurement of the round trip time. The process then terminates in stop box **83**.

If it is determined, in decision box **71**, that the receiving rate R_r is greater than or equal to $R_a - \Delta_R$, decision box **75** is entered to determine whether or not R_r is equal to or greater than the actual transmission rate R_a. If so, decision box **76** is entered to determine if the operating mode is already "GREEN". If not, the mode is set to "GREEN" in box **80** and box **81** entered to calculate a new value for B_t, a new value for M_s, if required, in box **82** and the process terminated in stop box **83**.

If it is determined that the mode is already green in decision box **76**, box **77** is entered to increase the sending rate R_s to $R_a + R_{inc}$. That is, the sending rate is increased to the actual transmission rate increased by the incremental value calculated in box **74**. If the existing sending rate is already higher than this incremented rate, however, the sending rate is left at this higher sending rate. Decision box **78** is then entered to determine whether this newly calculated sending rate R_s is greater than the maximum transmission rate R_{max} permitted over this connection. That is, each connection, when it is set up, is limited to a maximum transmission rate, depending on the requested Quality of Service. If the transmission rate R_s exceeds R_{max}, it is reset to R_{max} in box **79**. Boxers **81** and **82** are then entered to calculate B_t and M_s, respectively, and the process then terminated in stop box **83**. If the new sending rate R_s is not greater than R_{max}, as determined by decision box **78**, box **81** is entered directly.

It can be seen that the flow chart of FIG. **6** requires two consecutive responses requiring increases in R_s ("GREEN" operating mode) before the increase is effected. That is, the first time an increase might be made, box **80** is entered and no increase is made. The next time a response calls for and increase, decision box **76** determines that the "GREEN" operating mode has

already been set and box 77 can be entered to execute the sending rate increase. The values of Δ_R, Θ, R_{inc}, R_{inc}^{min}, and R_{max} can be adjusted to fine tune the operation of the rate-based, end-to-end, dynamic flow and congestion control system of the present invention.

In FIG. 7 there is shown a flow chart of the time out procedure used at the sending endnode 21 of FIG. 2. Starting at start box 90, decision box 91 is entered where it is determined if a timeout has occurred. That is, has the time elapsed since the last rate request message was sent (FIG. 5) exceeded the expected request-response period T_{out}? If the time out period has not expired, stop box 97 is entered to terminate the process. If the time out period has expired, as determined by decision box 91, decision box 92 is entered to determine whether a rate request has been sent which has not been responded to. If not, the process is again terminated in stop box 97. If a unanswered rate request has been sent, as determined by decision box 92, box 93 is entered to set the operating mode to "RED." In this mode, a drastic reduction in the sending rate R_s is called for since congestion has already occurred to a sufficient degree to delay a rate response beyond the time out period T_{out}. As previously noted, in the RED mode, the sending rate R_s is drastically reduced, e.g., cut in half, to relieve the existing congestion. In box 94 the value of R_{inc} is set to the minimum value R_{inc}^{min}. That is, the value used to increase the sending rate R_s when in the next GREEN operating mode is set to a minimum to make these increase as small as possible at the beginning of the rate increase process. In box 95 a new value of the maximum sending burst time B_t is calculated, based on the new value of R_s established in box 93. Thereafter, box 96 is entered to calculate a new value for the sending measurement interval M_s, if the round trip delay has changed substantially. The process terminates in stop box 97.

In FIG. 8 there is shown a flow chart of the processing of rate requests received at the receiving endnode 23 of FIG. 2. Starting in start box 100, decision box 108 is entered to determine if a packet has been lost. The detection of a lost packet is dependent on, for example, packet sequence numbers included in each packet, and which can be used to detect the dropping of a packet at an overloaded queue somewhere in the connection path. If a lost packet is detected in decision box 108, box 109 is entered where a lost packet message is piggybacked on the next transmitted packet from the receiving endnode 23 to the sending endnode 21. This message is treated by sending endnode 21 a message which is treated as a time out, invoking the procedure of FIG. 7.

If no packet loss is detected in decision box 108, box 101 is entered where the next data packet is received from network 10. Box 102 is then entered where the number of bits N_r received in the current measurement period M_r is incremented by the number of bits in the packet received in box 101. The value of N_r will be used in box 105 to calculate the new value of R_r to be returned to the sending endnode 21. Decision box 103 is then entered to determine whether a rate request has been received and not yet responded to. If not, the process terminates in stop box 106. If an unanswered rate request message has been received, as determined by decision box 103, box 104 is entered where the new receive period M_r is calculated. Normally, the value of M_r is determined simply by measuring the period since the reception of the last rate request message. As will be described in connection with FIG. 9, however, the value of M_r can be adjusted to correct for the "accumulation effect" to be described below. Using this value of M_r, the value of the receiving rate R_r is calculated and returned to the sending endnode 21 in box 105. Preferable, the receiving rate R_r is piggy-backed on a data segment for return to the sending endnode 21. The process of FIG. 8 then terminates in stop box 106.

In packet communications networks such as that illustrated in FIG. 1, the queue lengths at the nodes along a connection path often are built up gradually by small amounts. This effect is called the "accumulation effect" and can result in accumulating sufficient queuing delays to exceed the overall connection delay negotiated at the time of connection set up without being detected at the connection endnodes 21 or 23 (FIG. 2). If the drift in the transit delay is not detected, the congestion control system is unable to respond thereto and the network can become congested because of this accumulation effect without correction. As will be discussed in connection with FIG. 9, the determination of the receiving measuring period M_r, in accordance with one feature of the present invention, has been modified to detect and respond to the accumulation effect and prevent such undetected congestion. The flow and congestion control scheme is modified such that the receiver rate measuring period is artificially extended to make the rate response value much lower that it would otherwise be if no delay accumulation had occurred. The effect of extending the measuring period is to reduce the calculated receiving rate R_r and thereby force the sending endnode 21 to reduce the sending rate R_s. The result of such a reduction in the sending rate R_s is the reduction of the accumulated delays along the connection path, reducing the effect of the accumulation effect.

In general, the accumulation effect is detected by detecting changes in the network shift/delay time T_s for each rate request message from the sending endnode 21 (FIG. 2). The value of T_s, in turn, can be approximated by determining the difference between the sending measuring period and the receiving measuring period. That is, the difference between the receiving endnode measurement period M_r and the corresponding sending endnode measurement period M_s is measured to approximate changes over time of the network delay. As noted in connection with FIG. 6, the value of M_r can be sent to the receiving endnode 23 piggy-backed on a data segment along with the request for the receiving rate R_r. T_s can be thought of as the network queuing time of rate request messages. Due to local congestion at intermediate nodes, the queuing time at these nodes can increase or decrease over time, thus allowing the value of T_s to vary over time. The successive values of T_s for successive rate request messages are accumulated (in a variable called TS) and compared to a ceiling (called T_q) calculated at the time the connection is set up. If TS falls below zero, no accumulation has occurred and the accumulation process is restarted to detect future increases in the network queuing time. On the other hand, if T_s exceeds T_q, the receiving endnode measuring period M_r is increased (for example, by T_q) to reduce the rate R_r fed back to the receiving endnode and thereby force the reduction of the sending rate R_s. This reduction in the sending rate reduces the probability of further congestion due to the longer network queuing time and thereby avoids the congestion due to the accumulation effect. This accumulation effect avoidance algorithm is implemented in FIG. 9.

In FIG. 9 there is shown a flow chart of the process for computing and transmitting a new receive rate R_r, as called for in box **105** of FIG. 8. Starting in start box **110**, box **111** is entered where the length of the measurement period M_r since the last request is measured. This measurement involves no more than subtracting the arrival time of the previous rate request message from the arrival time of the current rate request message. Box **112** is then entered where the value of the network queuing shift T_s is calculated as the difference between the current receiving endnode measuring cycle M_r and the previous sending endnode measuring cycle M_s. Box **113** is then entered to accumulate the new value of T_s with the previously accumulated values in TS. Decision box **114** is then entered to determine if the value of TS is less than zero. If so, no accumulation avoidance is necessary and box **115** is entered to reset the value of TS to zero. The process then terminates in stop box **120**.

If the value of TS is not less than zero, as determined in decision box **114**, decision box **116** is entered to determine if the value of TS exceeds the precalculated delay ceiling T_q. If not, the process is terminated in stop box **120**. If, however, the value of TS is equal to or exceeds the value of T_q, as determined by decision box **116**, decision box **117** is entered to determine if the latest measured value of T_s is less than T_q. If so, box **118** is entered to adjust the receiving measuring period. As suggested in box **118**, the measuring period M_r can be adjusted by adding to it the value of the delay ceiling T_q. This value of the increment to M_r insures that the resulting receiving rate R_r will be reduced sufficiently to trigger a reduction in the sending rate R_s. Other values for this increment can also be used if the result is a reduction in the sending rate R_r. The process then ends in stop box **120**. As show in FIG. 8, however, this new value for M_r is used in calculating R_r.

If the value of T_s is equal to or greater than T_q, as determined by decision box **117**, it must be assumed that an error has occurred, such as the loss of a rate request message, and box **115** is re-entered to reset TS to zero to start a new drift measurement cycle. The process then terminates in stop box **120** without adjusting the value of M_r.

It should also be clear to those skilled in the art that further embodiments of the present invention may be made by those skilled in the art without departing from the teachings of the present invention.

APPENDIX
Pseudo-Code Algorithms for Adaptive Rate-Based Flow and Congestion Control

Data Sending Algorithm at Sending Endnode

```
IF (More Data Can Be Sent in Interval B_I) THEN
  IF (M_s Has Elapsed) THEN
    Build Data Segment
      Piggy-Back Receiver Feedback Request
      Compute R_s Sent During M_s = Bits Sent Divided by M_s
  END
  Send Data Segment
END
```

Feedback Processing Algorithm at Sending Endnode

```
IF (R_r < R_s - Δ_R) THEN
  Set Rate R_s = θ × R_r
  Set MODE to "YELLOW"
  Set R_inc = max( R_inc / 2 , R_inc^min )
ELSE
```

-continued

APPENDIX
Pseudo-Code Algorithms for Adaptive Rate-Based Flow and Congestion Control

```
  IF (R_r ≧ R_s) and (MODE = GREEN) THEN
    R_s = max(R_s, R_s + R_inc)
    But R_s ≦ min(R_s, R_max)/* Rate Cannot Exceed
      Maximum Rate Allowed */
  END
  Set MODE to "GREEN"
END
Compute New B_I
Compute New M_s, if Required.
```

Time-Out Algorithm at Sending Endnode

```
IF (Time...Out and No...Rate...Request) THEN
  Set Mode to "RED"
  Cut R_s by half, R_s = min(0.5 × R_s, R_r)
  Set R_inc = R_inc^min
  Compute new B_I
  Compute new M_r if necessary
END
```

Rate Calculation Algorithm at Receiving Endnode

```
Receive Packet
Process Packet
  N_r = N_r + Packet Size
IF (Rate Request) THEN
  IF (No Packet Loss Detected) THEN
    Compute M_r
    /* See Interval Computation Algorithm Below */
    Compute R_r = N_r/M_r
    Return Acknowledgement Message Including R_r
  ELSE
    Return Message Indicating Packet Loss
  END
END
```

Interval Computation Algorithm at Receiving Endnode (Accumulation Effect Avoidance)

```
Measure Elapsed Time M_r Since Last Rate Request
Compute T_s = M_r - M_s
Accumulate TS = TS + T_s
IF (TS < 0) THEN
  Reset TS = 0   /* Start a New Timing Cycle */
ELSE
  IF (TS > T_q) THEN
    IF (T_s < T_q) THEN
      Send Message Forcing Sender to Reduce Rate
      Adjust M_r = M_r + T_q
    END
    Reset TS = 0      /* Start a New Timing Cycle */
  END
END
```

What is claimed is:

1. A closed loop rate-based flow and congestion control system for a packet communications network providing a connection for the transmission of packets between a sending endnode and a receiving endnode of said network, said control system comprising

means at said receiving endnode for determining the lower of either the rate at which said packets are delivered from said network to said receiving endnode or the rate at which said packets are delivered from said receiving endnode to a connected user,

means for transmitting said lower rate from said receiving endnode to said sending endnode of said network, and

means for adjusting the rate at which packets are launched from said sending endnode on said connection in response to said lower rate.

2. The flow and congestion control system according to claim **1** wherein

said rates are determined over a measurement period approximately equal to the round trip delay of packets transmitted through said network between said sending endnode and said receiving endnode.

3. The flow and congestion control system according to claim 2 further comprising

means for determining changes in the transmission delay in said connection, and

means responsive to accumulations in said transmission delay for extending said measurement period.

4. The flow and congestion control system according to claim 1 further comprising

means for piggy-backing said lower rate on one of said packets being transmitted from said receiving endnode to said sending endnode.

5. The flow and congestion control system according to claim 1 further comprising

means at said sending endnode for generating a request for rate information,

means for transmitting said request to said receiving endnode, and

means at said receiving endnode for enabling said means for determining said lower rate in response to said request.

6. The flow and congestion control system according to claim 5 wherein said means for transmitting said request comprises

means for piggy-backing said request on one of said packets being transmitted from said sending endnode to said receiving endnode.

7. The flow and congestion control system according to claim 1 wherein said means for adjusting the packet launching rate comprises

means for increasing said packet launching rate in response to said lower rate exceeding the current packet launching rate.

8. The flow and congestion control system according to claim 7 wherein said means for increasing said packet launching rate includes

means for increasing said packet launching rate in equal increments.

9. The flow and congestion control system according to claim 7 wherein said means for increasing said packet launching rate includes

means for increasing said packet launching rate in non-equal increments.

10. The flow and congestion control system according to claim 1 wherein said means for adjusting the packet launching rate comprises

means for decreasing said packet launching rate in response to the current packet launching rate exceeding said lower rate.

11. The flow and congestion control system according to claim 10 wherein said means for decreasing said packet launching rate includes

means for decreasing said packet launching rate in non-equal increments.

12. A closed loop rate-based method for controlling data packet flow and congestion in a packet communications network providing a connection for the transmission of packets between a sending endnode and a receiving endnode of said network, said method comprising the steps of

determining at said receiving endnode the lower of either the rate at which said packets are delivered from said network to said receiving endnode or the rate at which said packets are delivered from said receiving endnode to a connected user,

transmitting said lower rate from said receiving endnode to said sending endnode of said network, and

adjusting the rate at which packets are launched from said sending endnode on said connection in response to said lower rate.

13. The flow and congestion control method according to claim 12 wherein said rate determining step further comprises the step of

determining said rates over a measurement period approximately equal to the round trip delay of packets transmitted through said network between said sending endnode and said receiving endnode.

14. The flow and congestion control method according to claim 13 further comprising the steps of

determining changes in the transmission delay in said connection, and

extending said measurement period in response to accumulations in said transmission delay.

15. The flow and congestion control system according to claim 12 further comprising the step of

piggy-backing said lower rate on one of said packets being transmitted from said receiving endnode to said sending endnode.

16. The flow and congestion control method according to claim 12 further comprising the steps of

generating at said sending endnode a request for rate information,

transmitting said request to said receiving endnode, and

enabling said step of determining said lower rate in response to said request.

17. The flow and congestion control method according to claim 16 wherein said step of transmitting said request comprises the step of

piggy-backing said request on one of said packets being transmitted from said sending endnode to said receiving endnode.

18. The flow and congestion control method according to claim 12 wherein said step of adjusting the packet launching rate comprises the step of

increasing said packet launching rate in response to said lower rate exceeding the current packet launching rate.

19. The flow and congestion control method according to claim 18 wherein said step of increasing said packet launching rate further includes the step of

increasing said packet launching rate in equal increments.

20. The flow and congestion control method according to claim 18 wherein said step of increasing said packet launching rate further include the step of

increasing said packet launching rate in non-equal increments.

21. The flow and congestion control method according to claim 12 wherein said step of adjusting the packet launching rate further comprises the step of

decreasing said packet launching rate in response to the current packet launching rate exceeding said lower rate.

22. The flow and congestion control system according to claim 21 wherein said means for decreasing said packet launching rate includes

means for decreasing said packet launching rate in non-equal increments.

* * * * *

Exhibit 1009

Declaration of Mr. Scott Delman

Customer No.: 000027683

Haynes and Boone, LLP
IP Section
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Telephone [214] 651.5000
Fax [214] 200.0853

DECLARATION OF SCOTT DELMAN

I, Scott Delman, am over twenty-one (21) years of age. I have never been convicted of a felony, and I am fully competent to make this declaration. I declare the following to be true to the best of my knowledge, information, and belief:

1. I am Director of Publications for the Association for Computing Machinery ("ACM").

2. Neither I nor ACM itself is being compensated for this declaration.

3. Among my responsibilities as Director of Publications, I act as a custodian of records for ACM's publications.

4. I make this declaration based on my personal knowledge, information contained in the business records of ACM, or confirmation with other responsible ACM personnel with such knowledge.

5. As part of its ordinary course of business, ACM publishes technical papers, including ACM's conference proceedings, journals, and other full-text publications. ACM publications are distributed to ACM's individual and institutional subscribers, ACM members, and/or ACM conference attendees. Additionally, ACM publications are available for public download through ACM's Digital Library (as described further herein), which has existed since July 1997.

6. As part of its ordinary course of business, since July 1997, ACM has maintained a Digital Library where ACM publications are available to the public, either for free or purchase, depending on the article. The Digital Library includes keyword search functionality permitting users to search for articles using keyword search terms and to filter search results by whether the term appears in the title, full-text, index terms, and/or abstract of the article. In addition, users may search by author name, and may filter search results by the particular type of journal, type of proceeding, and/or the date of publication.

7. Exhibit A to this declaration is a true and correct copy, from the ACM DL, of an article authored by Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson, titled "Practical network support for IP traceback" (the "Paper") which was published in the ACM publication *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (August 2000), https://doi.org/10.1145/347059.347560. This document is maintained in the ordinary course of ACM's business.

8. Exhibit B to this declaration is a true and correct copy of the ACM DL record for the Paper. This document is maintained in the ordinary course of ACM's business.

9. The Paper is currently available to the public for purchase and download from the ACM DL.

1601 Broadway, 10th Floor
New York, NY 10019-7434

Tel: +1-212-869-7440
Fax: +1-212-944-1318

acmhelp@acm.org
www.acm.org

10. The ACM DL record for any given publication is automatically populated with metadata associated with the publication of that article, which ACM maintains in the ordinary course of ACM's business. The ACM DL record for the Paper (annexed at Exhibit B) reflects that the Paper has been downloaded 4,029 times, from the date of its publication (as reflected in Exhibit B) through March 30, 2021.

11. ACM's records reflect that 407 individuals registered for the SIGCOMM `00 conference at which the Paper was presented and distributed. ACM's records further reflect that *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* was distributed to all attendees of the SIGCOMM '00 conference in hard copy, in addition to being made available electronically through the ACM DL.

12. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under 18 U.S.C. § 1001.

I declare under penalty of perjury that the foregoing statements are true and correct.

Executed on: April 1, 2021

Scott Delman, Director of Publications

1601 Broadway, 10th Floor
New York, NY 10019-7434

Tel: +1-212-869-7440
Fax: +1-212-944-1318

acmhelp@acm.org
www.acm.org

Declaration of Mr. Scott Delman
IPR2021-00785

ATTACHMENT A

Practical Network Support for IP Traceback

Stefan Savage, David Wetherall, Anna Karlin and Tom Anderson

Department of Computer Science and Engineering
University of Washington
Seattle, WA, USA

Abstract

This paper describes a technique for tracing anonymous packet flooding attacks in the Internet back towards their source. This work is motivated by the increased frequency and sophistication of denial-of-service attacks and by the difficulty in tracing packets with incorrect, or "spoofed", source addresses. In this paper we describe a general purpose traceback mechanism based on probabilistic packet marking in the network. Our approach allows a victim to identify the network path(s) traversed by attack traffic without requiring interactive operational support from Internet Service Providers (ISPs). Moreover, this traceback can be performed "post-mortem" – after an attack has completed. We present an implementation of this technology that is incrementally deployable, (mostly) backwards compatible and can be efficiently implemented using conventional technology.

1. INTRODUCTION

Denial-of-service attacks consume the resources of a remote host or network, thereby denying or degrading service to legitimate users. Such attacks are among the hardest security problems to address because they are simple to implement, difficult to prevent, and very difficult to trace. In the last several years, Internet denial-of-service attacks have increased in frequency, severity and sophistication. Howard reports that between the years of 1989 and 1995, the number of such attacks reported to the Computer Emergency Response Team (CERT) increased by 50 percent per year [25]. More recently, a 1999 CSI/FBI survey reports that 32 percent of respondents detected denial-of-service attacks directed against their sites [16]. Even more worrying, recent reports indicate that attackers have developed tools to coordinate distributed attacks from many separate sites [14].

Unfortunately, mechanisms for dealing with denial-of-service have not advanced at the same pace. Most work in this area has focused on *tolerating* attacks by mitigating their effects on the victim [38, 2, 26, 29, 9]. This approach can provide an effective stop-gap measure, but does not eliminate the problem nor does it discourage attackers. The other option, and the focus of this paper, is to trace attacks back towards their origin – ideally stopping an attacker at the source.

A perfect solution to this problem is complicated by the potential use of indirection to "launder" the true causal origin of an attack. For example, an attack may consist of packets sent from many different slave machines, themselves under the control of a remote master machine. Such indirection may be achieved either explicitly (by compromising the individual slave hosts directly) or implicitly (by sending false requests to the slaves on behalf of the victim – a so-called *reflector*). More challenging still, the true origin and identity of the attacker can be similarly concealed through chains of false computer accounts, call forwarding, and so forth. Consequently, we regard a complete solution – particularly one able to address the forensic needs of law enforcement – as an open problem.

Instead, we address the more limited operational goal of simply identifying the machines that *directly* generate attack traffic and the network path this traffic subsequently follows. We call this the *traceback problem* and it is motivated by the operational need to control and contain attacks. In this setting, even incomplete or approximate information is valuable because the efficacy of measures such as packet filtering improve as they are applied further from the victim and closer to the source.

However, even for our restricted problem, determining the source generating attack traffic is surprisingly difficult due to the stateless nature of Internet routing. Attackers routinely disguise their location using incorrect, or "spoofed", IP source addresses. As these packets traverse the Internet their true origin is lost and a victim is left with little useful information. While there are several ad hoc traceback techniques in use, they all have significant drawbacks that limit their practical utility in the current Internet.

In this paper we present a new approach to the traceback problem that addresses the needs of both victims and network operators. Our solution is to probabilistically mark packets with partial path information as they arrive at routers. This approach exploits the observation that attacks generally comprise large numbers of packets. While each marked packet represents only a "sample" of the path it has traversed, by combining a modest number of such packets a victim can reconstruct the entire path. This allows victims to locate the approximate source of attack traffic without requiring the assistance of outside network operators. Moreover, this determination can be made even after an attack has completed. Both facets of our solution represent substantial improvements over existing capabilities for dealing with flooding-style denial-of-service attacks.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGCOMM'00, Stockholm, Sweden.
Copyright 2000 ACM 1-58113-224-7/00/0008...$5.00.

Cloudflare, Inc. Exhibit 1003 Page 901

A key practical deployment issue with any modification of Internet routers is to ensure that the mechanisms are efficiently implementable, may be incrementally deployed, and are backwards compatible with the existing infrastructure. We describe a traceback algorithm that adds little or no overhead to the router's critical forwarding path and may be incrementally deployed to allow traceback within the subset of routers supporting our scheme. Further, we demonstrate that we can encode the necessary path information in a way that peacefully co-exists with existing routers, host systems and more than 99% of today's traffic.

The rest of this paper is organized as follows: In Section 2, we describe related work concerning IP spoofing and solutions to the traceback problem. Section 3 outlines our basic approach and section 4 characterizes several abstract algorithms for implementing it. In Section 5 we detail a concrete encoding strategy for our algorithm that can be implemented within the current Internet environment. We also present experimental results demonstrating the effectiveness of our solution. In section 6 we discuss the main limitations and weaknesses of our proposal and potential extensions to address some of them. Finally, we summarize our findings in Section 7.

2. RELATED WORK

It has been long understood that the IP protocol permits anonymous attacks. In his 1985 paper on TCP/IP weaknesses, Morris writes:

> "The weakness in this scheme [the Internet Protocol] is that the source host itself fills in the IP source host id, and there is no provision in ... TCP/IP to discover the true origin of a packet." [31]

In addition to denial-of-service attacks, IP spoofing can be used in conjunction with other vulnerabilities to implement anonymous one-way TCP channels and covert port scanning [31, 3, 24, 44].

There have been several efforts to reduce the anonymity afforded by IP spoofing. Table 1 provides a subjective characterization of each of these approaches in terms of management cost, additional network load, overhead on the router, the ability to trace multiple simultaneous attacks, the ability trace attacks after they have completed, and whether they are preventative or reactive. We also characterize our proposed traceback scheme according to the same criteria. In the remainder of this section we describe each previous approach in more detail.

2.1 Ingress filtering

One way to address the problem of anonymous attacks is to eliminate the ability to forge source addresses. One such approach, frequently called *ingress filtering*, is to configure routers to block packets that arrive with illegitimate source addresses [20]. This requires a router with sufficient power to examine the source address of every packet and sufficient knowledge to distinguish between legitimate and illegitimate addresses. Consequently, ingress filtering is most feasible in customer networks or at the border of Internet Service Providers (ISP) where address ownership is relatively unambiguous and traffic load is low. As traffic is aggregated from multiple ISPs into transit networks, there is no longer enough information to unambiguously determine if a packet arriving on a particular interface has a "legal" source address. Moreover, on many deployed router architectures the overhead of ingress filter becomes prohibitive on high-speed links.

The principal problem with ingress filtering is that its effectiveness depends on widespread, if not universal, deployment. Unfortunately, a significant fraction of ISPs, perhaps the majority, do not implement this service – either because they are uninformed or have been discouraged by the administrative burden[1], potential router overhead and complications with existing services that depend on source address spoofing (e.g. some versions of Mobile IP [33] and some hybrid satellite communications architectures). A secondary problem is that even if ingress filtering were universally deployed at the customer-to-ISP level, attackers could still forge addresses from the hundreds or thousands of hosts within a valid customer network [14].

It is clear that wider use of ingress filtering would dramatically improve the Internet's robustness to denial-of-service attacks. At the same time it is prudent to assume that such a system will never be fullproof – and therefore traceback technologies will continue to be important.

2.2 Link testing

Most existing traceback techniques start from the router closest to the victim and interactively test its upstream links until they determine which one is used to carry the attacker's traffic. Ideally, this procedure is repeated recursively on the upstream router until the source is reached. This technique assumes that an attack remains active until the completion of a trace and is therefore inappropriate for attacks that are detected after the fact, attacks that occur intermittently, or attacks that modulate their behavior in response to a traceback (it is prudent to assume the attacker is fully informed). Below we describe two varieties of link testing schemes, *input debugging* and *controlled flooding*.

2.2.1 Input debugging

Many routers include a feature called *input debugging*, that allows an operator to filter particular packets on some egress port and determine which ingress port they arrived on. This capability is used to implement a trace as follows: First, the victim must recognize that it is being attacked and develop an *attack signature* that describes a common feature contained in all the attack packets. The victim communicates this signature to a network operator, frequently via telephone, who then installs a corresponding input debugging filter on the victim's upstream egress port. This filter reveals the associated input port, and hence which upstream router originated the traffic. The process is then repeated recursively on the upstream router, until the originating site is reached or the trace leaves the ISP's border (and hence its administrative control over the routers). In the later case, the upstream ISP must be contacted and the procedure repeats itself. While such tracing is frequently performed manually, several ISPs have developed tools to automatically trace attacks across their own networks [41].

The most obvious problem with the input debugging approach, even with automated tools, is its considerable management overhead. Communicating and coordinating with network operators at multiple ISPs requires the time, attention and commitment of both the victim and the remote personnel – many of whom have no direct economic incentive to provide aid. If the appropriate network

[1] Some modern routers ease the administrative burden of ingress filtering by providing functionality to automatically check source addresses against the destination-based routing tables (e.g. `ip verify unicast reverse-path` on Cisco's IOS). This approach is only valid if the route to and from the customer is symmetric – generally at the border of single-homed stub networks.

Cloudflare, Inc. Exhibit 1003 Page 902

	Management overhead	Network overhead	Router overhead	Distributed capability	Post-mortem capability	Preventative/ reactive
Ingress filtering	Moderate	Low	Moderate	N/A	N/A	Preventative
Link testing						
Input debugging	High	Low	High	Good	Poor	Reactive
Controlled flooding	Low	High	Low	Poor	Poor	Reactive
Logging	High	Low	High	Excellent	Excellent	Reactive
ICMP Traceback	Low	Low	Low	Good	Excellent	Reactive
Marking	Low	Low	Low	Good	Excellent	Reactive

Table 1: Qualitative comparison of existing schemes for combating anonymous attacks and the probabilistic marking approach we propose.

operators are not available, if they are unwilling to assist, or if they do not have the appropriate technical skills and capabilities, then a traceback may be slow or impossible to complete [21].

2.2.2 *Controlled flooding*

Burch and Cheswick have developed a link testing traceback technique that does not require any support from network operators [6]. We call this technique *controlled flooding* because it tests links by flooding them with large bursts of traffic and observing how this perturbs traffic from the attacker. Using a pre-generated "map" of Internet topology, the victim coerces selected hosts along the upstream route into iteratively flooding each incoming link on the router closest to the victim. Since router buffers are shared, packets traveling across the loaded link – including any sent by the attacker – have an increased probability of being dropped. By observing changes in the rate of packets received from the attacker, the victim can therefore infer which link they arrived from. As with other link testing schemes, the basic procedure is then applied recursively on the next upstream router until the source is reached.

While the scheme is both ingenious and pragmatic, it has several drawbacks and limitations. Most problematic among these is that controlled flooding is itself a denial-of-service attack – exploiting vulnerabilities in unsuspecting hosts to achieve its ends. This drawback alone makes it unsuitable for routine use. Also, controlled flooding requires the victim to have a good topological map of large sections of the Internet in addition to an associated list of "willing" flooding hosts. As Burch and Cheswick note, controlled flooding is also poorly suited for tracing distributed denial-of-service attacks because the link-testing mechanism is inherently noisy and it can be difficult to discern the set of paths being exploited when multiple upstream links are contributing to the attack. Finally, like all link-testing schemes, controlled flooding is only effective at tracing an on-going attack and cannot be used "post-mortem".

2.3 Logging

An approach suggested in [37] and [41] is to log packets at key routers and then use data mining techniques to determine the path that the packets traversed. This scheme has the useful property that it can trace an attack long after the attack has completed. However, it also has obvious drawbacks, including potentially enormous resource requirements (possibly addressed by sampling) and a large scale inter-provider database integration problem. We are unaware of any commercial organizations using a fully operational traceback approach based on logging[2].

[2]Historically, the T3-NFSNET *did* log network-to-network traffic statistics and these were used on at least one occasion to trace IP spoofing attacks to an upstream provider [43].

2.4 ICMP Traceback

Since the first writing of this paper, a new traceback proposal bas emerged based on the use of explicit router-generated ICMP traceback messages [4]. The principle idea in this scheme is for every router to sample, with low probability (e.g., 1/20,000), one of the packets it is forwarding and copy the contents into a special ICMP traceback message including information about the adjacent routers along the path to the destination. During a flooding-style attack, the victim host can then use these messages to reconstruct a path back to the attacker. This scheme has many benefits compared to previous work and is in many ways similar to the packet marking approach we have taken. However, there are several disadvantages in the current design that complicate its use. Among these: ICMP traffic is increasingly differentiated and may be filtered or rate limited differently from normal traffic, the ICMP Traceback message relies on an input debugging capability (i.e. the ability to associate a packet with the input port and/or MAC address on which it arrived) that is not available in some router architectures, if only some of the routers participate it seems difficult to positively "connect" traceback messages from participating routers separated by a non-participating router, and finally, it requires a key distribution infrastructure to deal with the problem of attackers sending false ICMP Traceback messages. That said, we believe that the scheme is promising and that hybrid approaches combining it with some of the algorithms we propose are likely to be quite effective.

3. OVERVIEW

Burch and Cheswick mention the possibility of tracing flooding attacks by "marking" packets, either probabilistically or deterministically, with the addresses of the routers they traverse [6]. The victim uses the information in the marked packets to trace an attack back to its source. This approach has not been previously explored in any depth, but has many potential advantages. It does not require interactive cooperation with ISPs and therefore avoids the high management overhead of input debugging. Unlike controlled flooding, it does not require significant additional network traffic and can potentially be used to track multiple attacks. Moreover, like logging, packet marking can be used to trace attacks "post-mortem" – long after the attack has stopped. Finally, we have found that marking algorithms can be implemented without incurring any significant overhead on network routers. The remainder of this paper focuses on fully exploring and characterizing this approach.

3.1 Definitions

Figure 1 depicts the network as seen from a victim V. For the purposes of this paper, V may be a single host under attack, or a network border device such as a firewall or intrusion detection system that represents many such hosts. Every potential *attack origin* A_i is a leaf in a tree rooted at V and every router R_i is an internal

Cloudflare, Inc. Exhibit 1003 Page 903

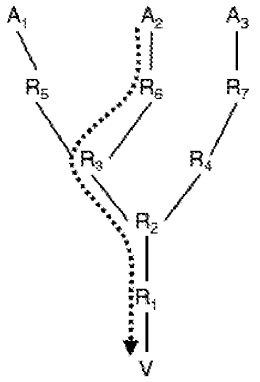


Figure 1: Network as seen from the victim of an attack, V. Routers are represented by R_i, and potential attackers by A_i. The dotted line represents a particular *attack path* between an attacker and the victim.

node along a path between some A_i and V. The *attack path* from A_i is the unique ordered list of routers between A_i and V. For instance, if an attack originates from A_3 then to reach V it must first traverse the path R_6, R_3, R_2, and R_1 – as shown by the dotted line in Figure 1.

The *exact traceback* problem is to determine the attack path and the associated attack origin for each attacker. However, solving this problem is complicated by several practical limitations. The exact attack origin may never be revealed (even MAC source addresses may be spoofed) and a wily attacker may send false signals to "invent" additional routers in the traceback path. We address these issues in section 6, but for now we restrict our discussion to solving a more limited problem. We define the *approximate traceback* problem as finding a candidate attack path for each attacker that contains the true attack path as a suffix. We call this the *valid suffix* of the candidate path. For example, $(R_6, R_5, R_3, R_2, R_1)$ is a valid approximate solution to Figure 1 because it contains the true attack path as a suffix. We say a solution to this problem is *robust* if an attacker cannot prevent the victim from discovering candidate paths containing the valid suffix.

All marking algorithms have two components: a *marking procedure* executed by routers in the network and a *path reconstruction procedure* implemented by the victim. A router "marks" one or more packets by augmenting them with additional information about the path they are traveling. The victim attempts to reconstruct the attack path using only the information in these marked packets. The *convergence time* of an algorithm is the number of packets that the victim must observe to reconstruct the attack path.

3.2 Basic assumptions

The design space of possible marking algorithms is large, and to place our work in context we identify the assumptions that motivate and constrain our design:

- an attacker may generate any packet,
- multiple attackers may conspire,
- attackers may be aware they are being traced,
- packets may be lost or reordered,
- attackers send numerous packets,
- the route between attacker and victim is fairly stable,
- routers are both CPU and memory limited, and
- routers are not widely compromised.

The first four assumptions represent conservative assessments of the abilities of the modern attackers and limitations of the network. Designing a traceback system for the Internet environment is extremely challenging because there is very little that can be trusted. In particular, the attacker's ability to create arbitrary packets significantly constrains potential solutions. When a router receives a packet, it has no way to tell whether that packet has been marked by an upstream router or if the attacker simply has forged this information. In fact, the only invariant that we can depend on is that a packet from the attacker must traverse all of the routers between it and the victim.

The remaining assumptions reflect the basis for our design and deserve additional discussion. First, denial-of-service attacks are only effective so long as they occupy the resources of the victim. Consequently, most attacks are comprised of thousands or millions of packets. Our approach relies on this property because we mark each packet with only a small piece of path state and the victim must observe many such packets to reconstruct the complete path back the the attacker. If many attacks emerge that require only a single packet to disable a host (e.g. ping-of-death [11]), then this assumption may not hold (although we note that even these attacks require multiple packets to *keep* a machine down).

Second, measurement evidence suggests that while Internet routes do change, it is extremely rare for packets to follow many different paths over the short time-scales of a traceback operation (seconds in our system) [32]. This assumption greatly simplifies the role of the victim, since it can therefore limit its consideration to a single primary path for each attacker. If the Internet evolves to allow significant degrees of multi-path routing then this assumption may not hold.

Third, while there have been considerable improvements in router implementation technology, link speeds have also increased dramatically. Consequently, we assert that any viable implementation must have low per-packet overhead and must not require per-flow state. Significantly simpler schemes than ours can be implemented if we assume that routers are not resource constrained.

Finally, since a compromised router can effectively eliminate any information provided by upstream routers, it is effectively indistinguishable from an attacker. In such circumstances, the security violation at the router must be addressed first, before any further traceback is attempted. In normal circumstances, we believe this is an acceptable design point. However, if non-malicious, but information hiding, routing infrastructures become popular, such as described in [22, 35], then this issue may need to be revisited.

4. BASIC MARKING ALGORITHMS

In this section we describe a series of marking algorithms – starting from the most simple and advancing in complexity. Each algorithm attempts to solve the approximate traceback problem in a manner consistent with our assumptions.

Cloudflare, Inc. Exhibit 1003 Page 904

Marking procedure at router R:
for each packet w, append R to w

Path reconstruction procedure at victim v:
for any packet w from attacker
extract path $(R_i..R_j)$ from the suffix of w

Figure 2: Node append algorithm.

4.1 Node append

The simplest marking algorithm – conceptually similar to the IP Record Route option [34] – is to append each node's address to the end of the packet as it travels through the network from attacker to victim (see Figure 2). Consequently, every packet received by the victim arrives with a complete ordered list of the routers it traversed – a built-in attack path.

The node append algorithm is both robust and extremely quick to converge (a single packet), however it has several serious limitations. Principal among these is the infeasibly high router overhead incurred by appending data to packets in flight. Moreover, since the length of the path is not known *a priori*, it is impossible to ensure that there is sufficient unused space in the packet for the complete list. This can lead to unnecessary fragmentation and bad interactions with services such as MTU discovery [30]. This problem cannot be solved by reserving "enough" space, as the attacker can completely fill any such space with false, or misleading, path information.

4.2 Node sampling

To reduce both the router overhead and the per-packet space requirement, we can sample the path one node at a time instead of recording the entire path. A single static "node" field is reserved in the packet header – large enough to hold a single router address (i.e. 32 bits for IPv4). Upon receiving a packet, each router chooses to write its address in the node field with some probability p. After enough packets have been sent, the victim will have received at least one sample for every router in the attack path. As stated in section 3, we assume that the attacker sends enough packets and the route is stable enough that this sampling can converge.

Although it might seem impossible to reconstruct an ordered path given only an unordered collection of node samples, it turns out that with a sufficient number of trials, the order can be deduced from the relative number of samples per node. Since routers are arranged serially, the probability that a packet will be marked by a router and then left unmolested by all downstream routers is a strictly decreasing function of the distance to the victim. If we constrain p to be identical at each router, then the probability of receiving a marked packet from a router d hops away is $p(1-p)^{d-1}$. Since this function is monotonic in the distance from the victim, ranking each router by the number of samples it contributes will tend to produce the accurate attack path. The full algorithm is shown in Figure 3.

Putting aside for the moment the difficulty in changing the IP header to add a 32-bit node field, this algorithm is efficient to implement because it only requires the addition of a write and checksum update to the forwarding path. Current high-speed routers already must perform these operations efficiently to update the *time-to-live* field on each hop. Moreover, if $p > 0.5$ then this algorithm is robust against a single attacker because there is no way for an attacker to insert a "false" router into the path's valid suffix by contributing

Marking procedure at router R:
for each packet w
let x be a random number from [0..1)
if $x < p$ then,
write R into w.node

Path reconstruction procedure at victim v:
let $NodeTbl$ be a table of tuples (node,count)
for each packet w from attacker
z := lookup w.node in $NodeTbl$
if z != NIL then
increment z.count
else
insert tuple (w node,1) in $NodeTbl$
sort $NodeTbl$ by count
extract path $(R_i..R_j)$ from ordered node fields in $NodeTbl$

Figure 3: Node sampling algorithm.

more samples than a downstream router, nor to reorder valid routers in the path by contributing more samples than the difference between any two downstream routers.

However, there are also two serious limitations. First, inferring the total router order from the distribution of samples is a slow process. Routers far away from the victim contribute relatively few samples (especially since p must be large) and random variability can easily lead to misordering unless a very large number of samples are observed. For instance, if $d = 15$ and $p = 0.51$, the receiver must receive more than 42,000 packets on average before it receives a *single* sample from the furthest router. To guarantee that the order is correct with 95% certainty requires more than seven times that number.

Second, if there are multiple attackers then multiple routers may exist at the same distance – and hence be sampled with the sample probability. Therefore, this technique is not robust against multiple attackers.

4.3 Edge sampling

A straightforward solution to these problems is to explicitly encode *edges* in the attack path rather than simply individual nodes. To do this, we would need to reserve *two* static address-sized fields, *start* and *end*, in each packet to represent the routers at each end of a link, as well as an additional small field to represent the distance of an edge sample from the victim.

When a router decides to mark a packet, it writes its own address into the start field and writes a zero into the distance field. Otherwise, if the distance field is already zero this indicates that the packet was marked by the previous router. In this case, the router writes its own address into the end field – thereby representing the edge between itself and the previous router. Finally, if the router doesn't mark the packet then it always increments the distance field. This somewhat baroque signaling mechanism allows edge sampling to be incrementally deployed – edges are constructed only between participating routers.

The mandatory increment is *critical* to minimize spoofing by an attacker. When the packet arrives at the victim its distance field represents the number of hops traversed since the edge it contains

Cloudflare, Inc. Exhibit 1003 Page 905

Marking procedure at router R:

```
for each packet w
    let x be a random number from [0..1)
    if x < p then
        write R into w.start and 0 into w.distance
    else
        if w.distance = 0 then
            write R into w.end
        increment w.distance
```

Path reconstruction procedure at victim v:

```
let G be a tree with root v
let edges in G be tuples (start,end,distance)
for each packet w from attacker
    if w.distance = 0 then
        insert edge (w.start,v,0) into G
    else
        insert edge (w.start,w.end,w.distance) into G
remove any edge (x,y,d) with d ≠ distance from x to v in G
extract path (R_i..R_j) by enumerating acyclic paths in G
```

Figure 4: Edge sampling algorithm.

was sampled.[3] Any packets written by the attacker will necessarily have a distance greater or equal to the length of the true attack path. Therefore, a single attacker is unable to forge any edge between themselves and the victim (for a distributed attack, of course, this applies only to the closest attacker) and the victim does not have to worry about "chaff" while reconstructing the valid suffix of the attack path. Consequently, since we no longer use the sampling rank approach to distinguish "false" samples, we are free to use arbitrary values for the marking probability p.

The victim uses the edges sampled in these packets to create a graph (much as in Figure 1) leading back to the source, or sources, of attack. The full algorithm is described in Figure 4. Because the probability of receiving a sample is geometrically smaller the further away it is from the victim, the time for this algorithm to converge is dominated by the the time to receive a sample from the furthest router, $\frac{1}{p(1-p)^{d-1}}$ in expectation, for a router d hops away. However, there is a small probability that we will receive a sample from the furthest router, but not from some nearer router. We can bound this effect to a factor of $ln(d)$ by the following argument: We conservatively assume that samples from all of the d routers appear with the same likelihood as the furthest router. Since these probabilities are disjoint, the probability that a given packet will deliver a sample from some router is at least $dp(1-p)^{d-1}$. Finally, as per the well-known *coupon collector* problem, the number of trials required to select one of each of d equi-probable items is $d(ln(d) + O(1))$ [19]. Therefore, the number of packets, X, required for the victim to reconstruct a path of length d has the following bounded expectation:

$$E(X) < \frac{ln(d)}{p(1-p)^{d-1}}$$

For example, if $p = \frac{1}{10}$, and the attack path has a length of 10, then a victim can typically reconstruct this path after receiving 75 packets from the attacker. While this choice of $p = \frac{1}{d}$ is optimal, the convergence time is not overly sensitive to this parameter for the path lengths that occur in the Internet. So long as $p \leq \frac{1}{d}$, the results are generally within a small constant of optimal. In the rest of this paper we will use $p = \frac{1}{25}$ since few paths exceed this length [7, 42, 17]. For comparison, the previous example converges with only 108 packets using $p = \frac{1}{25}$.

This same algorithm can efficiently discern multiple attacks because attackers from different sources produce disjoint edges in the tree structure used during reconstruction. The number of packets needed to reconstruct each path is independent, so the number of packets needed to reconstruct all paths is a linear function of the number of attackers. Finally, edge sampling is also robust. That is it is impossible for any edge closer than the closest attacker to be spoofed, due to the robust distance determination. Conversely, in a distributed attack this also means that it is impossible to trust the contents of any edge *further* away than the closest attacker. As with the ICMP Traceback approach [4], an additional mechanism incorporating a shared secret is required to completely address the problem of attackers spoofing edges.

Of course, a significant practical limitation of this approach is that it requires additional space in the IP packet header and therefore is not backwards compatible. In the next section we discuss a modified version of edge-sampling that addresses this problem, albeit at some cost in performance and a reduction in robustness during large distributed attacks.

5. ENCODING ISSUES

The edge sampling algorithm requires 72 bits of space in every IP packet (two 32-bit IP addresses and 8 bits for distance to represent the theoretical maximum number of hops allowed using IP). It would be possible to directly encode these values into an MPLS label stack [36], to enable traceback within a single homogeneous ISP network. However, our focus is on a heterogeneous environment based purely on IP datagrams. One obvious approach is to store the edge sample data in an IP option, but this is a poor choice for many of the same reasons that the node append algorithm is infeasible – appending additional data to a packet in flight is expensive and there may not be sufficient space to append this data. We could also send this data out-of-band – in a separate packet – but this would add both router and network overhead plus the complexity of a new and incompatible protocol.

Instead, we have developed a modified version of edge sampling that dramatically reduces the space requirement in return for a modest increase in convergence time and a reduction in robustness to multiple attackers. Following an analysis of our algorithm we explore the practical implementation issues and discuss one concrete encoding of this scheme based on overloading the 16-bit IP *identification* field used for fragmentation. Any solution involving such overloading necessarily requires compromises and we stress that our solution reflects only one design point among many potential implementation tradeoffs for this class of algorithm and *does not* necessarily reflect an optimal balance among them.

5.1 Compressed edge fragment sampling

We use three techniques to reduce per-packet storage requirements while preserving robustness. First, we encode each edge in half the space by representing it as the *exclusive-or* (XOR) of the two IP addresses making up the edge, as depicted in Figure 5. When

[3] It is important that distance field is updated using a saturating addition. If the distance field were allowed to wrap, then the attacker could spoof edges close to the victim by sending packets with a distance value close to the maximum.

Cloudflare, Inc. Exhibit 1003 Page 906

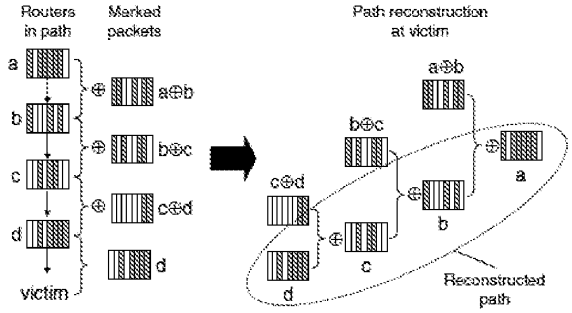


Figure 5: Edge data can be communicated in half the space by sending the XOR of the two nodes (i.e. router IP addresses) making up an edge, rather than sending each node separately. Over time the victim receives the messages d, $c \oplus d$, $b \oplus c$, and $a \oplus b$. By XORing these messages together the original path can be reconstructed.

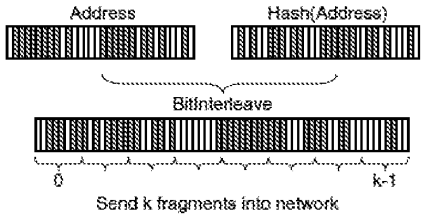


Figure 6: Each router calculates a uniform hash of its IP address once, at startup, using a well-known function. This hash is interleaved with the original IP address (the original address on odd bits, the hash on even bits). The resulting quantity is then broken into k fragments, which the router selects among randomly when marking a packet. Although it is not shown, each of these fragments is further labeled with its offset. The next downstream router uses this offset to select the appropriate fragment to XOR – thereby encoding part of an edge.

some router decides to mark a packet it writes its address, a, into the packet. The following router, b, notices that the distance field is 0 and (assuming it does not mark the packet itself) reads a from the packet, XORs this value with its own address and writes the resulting value, $a \oplus b$, into the packet. We call the resulting value the *edge-id* for the edge between a and b. The edge-ids in the packets received by the victim always contain the XOR of two adjacent routers, except for samples from routers one hop away from the victim, which arrive unmodified. Since $b \oplus a \oplus b = a$, marked packets from the final router can be used to decode the previous edge id, and so on, hop-by-hop until we reach the first router.

Our second modification further reduces our per-packet space requirements by subdividing each edge-id into some number, k, of smaller non-overlapping fragments. When a router decides to mark a packet, it selects one of these fragments at random and stores it in the packet. We use a few additional bits ($log_2 k$) to store the offset of this fragment within the original address – this is necessary to ensure that both fragments making up an edge-id are taken from the same offset. If enough packets are sent by the attacker, the victim will eventually receive all fragments from all edge-ids.

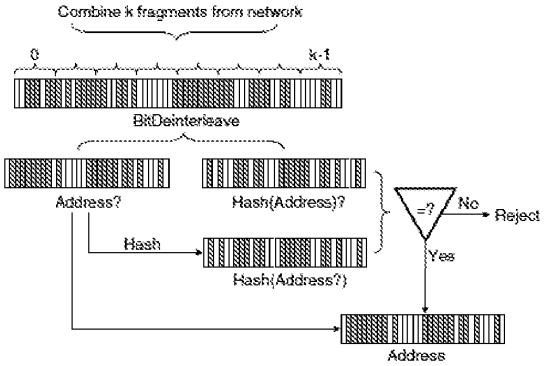


Figure 7: When reconstructing a candidate edge, the victim combines k fragments to produce a bit string. By deinterleaving this string, the address portion and the hash portion are extracted. We recalculate the hash over this address portion using the same hash function used by the router. If the resulting hash is the same as the hash portion extracted, then the address is accepted as valid. This procedure protects against accidentally combining fragments of different edges.

Finally, unlike full IP addresses, edge-id fragments are not unique and multiple fragments from different edge-ids may have the same value. If there are multiple attackers, a victim may receive multiple edge fragments with the same offset and distance. To reduce the probability that we accidentally reconstruct a "false" edge-id by combining fragments from different paths, we add a simple error detection code to our algorithm. We *increase* the size of each router address, and hence each edge-id, by bit-interleaving its IP address with a random hash of itself (depicted in Figure 6). As described earlier, this value is split into fragments, each fragment is selected randomly and stored with an offset, and downstream routers use XOR to combine fragments at the same offset to make up edge-id fragments. The victim constructs *candidate edge-ids* by combining all combinations of fragments at each distance with disjoint offset values. As shown in Figure 7, a candidate edge-id is only accepted if the hash portion matches the data portion for each of its two nodes. As we increase the size of the hash, the probability of a collision is reduced. We provide a describe the full procedure in Figure 8.

The expected number of packets for this algorithm to converge is similar to the edge sampling approach, except now we need k fragments for each edge-id, rather than just one, a total of kd fragments. If we again assume conservatively that each of these fragments is delivered equi-probably with probability $p(1-p)^{d-1}$, the expected number of packets required for path reconstruction is bounded by:

$$E(X) < \frac{k \cdot ln(kd)}{p(1-p)^{d-1}}$$

For example, if there are 8 fragments per edge-id, an attacker is 10 hops away, and $p = \frac{1}{25}$, then a victim can reconstruct the full path after receiving slightly less than 1,300 packets on average. Using techniques similar to those used to show sharp concentration results for the coupon collectors problem, we can further show that the approximate the number of packets required to ensure that a path

Cloudflare, Inc. Exhibit 1003 Page 907

Marking procedure at router R:

let R' = BitInterleave(R, Hash(R))
let k be the number of non-overlapping fragments in R'
for each packet w
  let x be a random number from [0..1)
  if $x < p$ then
    let o be a random integer from [0..$k-1$]
    let f be the fragment of R' at offset o
    write f into w.frag
    write 0 into w.distance
    write o into w.offset
  else
    if w.distance = 0 then
      let f be the fragment of R' at offset w.offset
      write $f \oplus w$.frag into w.frag
    increment w.distance

Path reconstruction procedure at victim v:

let $FragTbl$ be a table of tuples (frag,offset,distance)
let G be a tree with root v
let edges in G be tuples (start,end,distance)
let $maxd := 0$
let $last := v$
for each packet w from attacker
  $FragTbl$.Insert(w.frag,w.offset,w.distance)
  if w.distance > $maxd$ then
    $maxd := w$.distance
for $d := 0$ to $maxd$
  for all ordered combinations of fragments at distance d
    construct edge z
    if $d \neq 0$ then
      $z := z \oplus last$
    if Hash(EvenBits(z)) = OddBits(z) then
      insert edge (z,EvenBits(z),d) into G
      $last$:= EvenBits(z);
remove any edge (x,y,d) with $d \neq$ distance from x to v in G
extract path (R_0..R_j) by enumerating acyclic paths in G

Figure 8: Compressed edge fragment sampling algorithm.

can be reconstructed with probability $1 - \frac{1}{e}$ is:

$$\frac{k \cdot ln(kdc)}{p(1-p)^{d-1}}$$

packets. To completely reconstruct the previous path with 95% certainty should require no more than 2150 packets. Many denial-of-service attacks send this many packets in a few seconds.

Finally, we explore the robustness of this algorithm with respect to multiple attackers. For a random hash of length h, the probability of accepting an arbitrarily constructed candidate edge-id is $\frac{1}{2^h}$. In the event that there are m attackers, then at any particular distance d, in the worst case there may be up to m distinct routers.[4] Consequently the probability that any edge-id at distance d is accepted incorrectly is at most:

$$1 - (1 - \frac{1}{2^h})^{m^k}$$

[4] In practice, the number of distinct routers is likely to be smaller for the portion of the path closest to the receiver, since many attackers will still share significant portions of their attack path with one another.

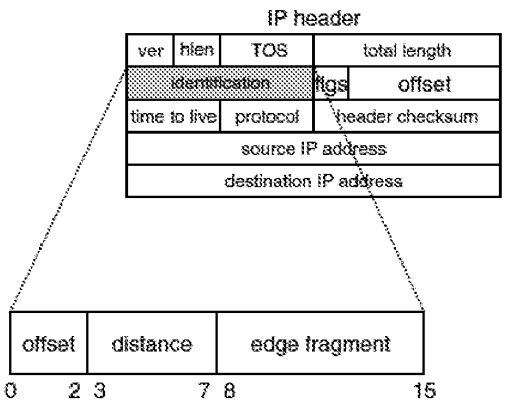


Figure 9: Encoding edge fragments into the IP identification field.

since there are m^k possible combinations of fragments in the worst case. For $h = 32$ and $k = 4$ this means that 100 distinct routers at the same distance (i.e. disjoint attack paths) will be resolved with no errors with a probability of better than 97%. For $h = 32$ and $k = 8$, (the values we use for our implementation) the same certainty can only be provided for 10 distinct routers at the same distance. Our use of the XOR function further complicates reconstruction since all combinations of XOR values must be tried as attack paths diverge. This is somewhat mitigated as the probability of propagating an error from a single edge all the way to the attacker is is extremely small because the resulting edge-id, when XORed with the previous edge-id, must again produce a correct hash.

The most significant drawback to this scheme is the large number of combinations that must be considered as the multiple attack paths diverge. While these combinations can be computed off-line, for large values of k and m even this can become intractable. For example, even with $k = 8$ and $m = 10$, if the separate attack paths diverge such that there are 10 completely independent edges per attacker, this will require roughly a billion combinations to be considered. Consequently, there is a design tension in the size of k – per-packet space overhead is reduced by a larger k, while computational overhead and robustness benefits from a smaller k.

5.2 IP header encoding

To allow for practical deployment requires that we "overload" existing header fields in a manner that will have minimal impact on existing users. This is a difficult task, especially given that even after prodigious effort we require 16 bits of space. Nonetheless, we believe it possible to obtain this space by overloading the 16-bit IP identification field. This field is currently used to differentiate IP fragments that belong to different packets. We describe our proposed encoding below, and then discuss the issues of backwards-compatibility that it raises. However, we note that because the issue of backwards-compatible encoding is largely separate from our traceback algorithms, we could adopt any reasonable encoding that comes to light.

Figure 9 depicts our choice for partitioning the identification field: 3 offset bits to represent 8 possible fragments, 5 bits to represent

Cloudflare, Inc. Exhibit 1003 Page 908

the distance, and 8 bits for the edge fragment. We use a 32-bit hash, which doubles the size of each router address to 64 bits. This implies that 8 separate fragments are needed to represent each edge – each fragment indicated by a unique offset value. Finally, 5 bits is sufficient to represent 31 hops, which is more than almost all Internet paths [7, 42, 17].[5]

The observant reader will note that this layout is chosen to allow the highest performance software implementation of our algorithm, which already had a low per-packet router overhead. In the common case, the only modification to the packet is to increment its distance field. Because of its alignment within the packet, this increment precisely offsets the required decrement of the time-to-live field implemented by each router [1]. Consequently, the header checksum *does not need to be altered at all* and the header manipulation overhead could be even lower than in current software-based routers – simply an addition to the distance field, a decrement to the TTL field, and a comparison to check if either has overflowed. In the worst case, our algorithm must read the IP identification field, lookup an edge fragment and XOR it, and fold the write-back into the existing checksum update procedure (a few ALU operations). Of course, for modern ASIC-based routers these optimizations are unnecessary.

As we reuse of the IP identification field, we must address issues of backwards-compatibility for IP fragment traffic. Ultimately, there is no perfect solution to this problem and we are forced to make compromises that disadvantage fragmented traffic. Fortunately, recent measurements suggest that less than 0.25% of packets are fragmented [40, 10]. Moreover, it has long been understood that network-layer fragmentation is detrimental to end-to-end performance [27] so modern network stacks implement automatic MTU discovery to prevent fragmentation regardless of the underlying media [30]. Consequently, we believe that our encoding will interoperate seamlessly with existing protocol implementations in the vast majority of cases.

However, there is a small but real fraction of legitimate traffic that is fragmented, and we wish to ensure that it is not affected by our modifications to the extent that this is possible. Normally if a packet is fragmented, its identification field is copied to each fragment so the receiver can faithfully reassemble the fragments into the original packet. Our marking procedure can violate this property in one of two ways: by writing different values into the identification fields of fragments from the same datagram or by writing the same values into the identification fields of fragments from different datagrams. These two problems present different challenges and have different solutions.

First, a datagram may be fragmented *upstream* from a marking router. If the fragment is subsequently marked and future fragments from the same datagram are not marked consistently then reassembly may fail or data may be corrupted. While the simplest solution to this problem is to simply not mark fragments, an adversary would quickly learn to evade traceback by exploiting this limitation. In fact, some current denial-of-service attacks already use IP fragments to exploit errors in host IP reassembly functions [12]. Instead, we propose an alternative marking mechanism for fragments. We use a separate marking probability, q, for fragments. When we decide to mark a fragment, we prepend a new ICMP "echo reply" header, along with the *full* edge data – truncating the tail of the packet. This ICMP packet is considered "marked" and its distance field is set to zero, thereby guaranteeing that the distance field reflects the number of edges traversed on the way to the victim. The packet is consequently "lost" from the standpoint of the receiver, but the edge information is delivered in a way that does not impact legacy hosts. Because we can use the full edge sampling algorithm, q can be more than an order of magnitude smaller than p and yet achieve the same convergence time. This solution increases the loss rate of fragmented flows somewhat (more substantially for longer paths) but preserves the integrity of the data in these flows.

[5] It is also reasonable to turn off marking on any routers that cannot be directly connected to an attacking host (e.g. core routers). This both reduces the convergence time, and increases the "reach" of the distance field.

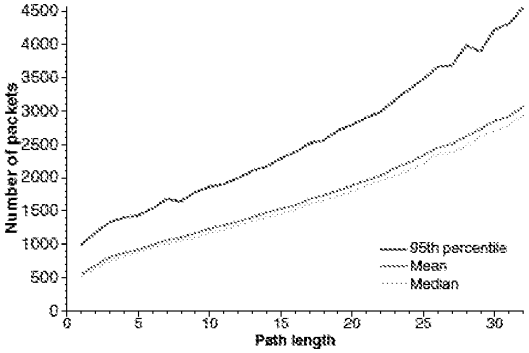


Figure 10: Experimental results for number of packets needed to reconstruct paths of varying lengths. The marking probability, p, is set to $\frac{1}{25}$. Each path length result represents the results of 1,000 independent simulation runs.

A more insidious problem is presented by fragmentation that occurs *downstream* from a marking router. If a marked packet is fragmented, but one of the fragments is lost, then the remaining fragments may linger in the victim's reassembly buffer for an extended period [5]. Future packets marked by the same router can have the same IP identification value and consequently may be incorrectly reassembled with the previous fragments. One possibility is to leave this problem to be dealt with by higher layer checksums. However, not all higher layer protocols employ checksums, and in any case it is dangerous to rely on such checksums because they are typically designed only for low residual error rates. Another solution is to set the *Don't Fragment* flag on every marked packet. Along rare paths that require fragmentation, this solution will degrade communication between hosts not using MTU path discovery, and may filter marked packets if a reduced MTU edge is close to the victim, but it will never lead to data corruption.

5.3 Experience

We have implemented the marking and reconstruction portions of our algorithm and have tested it using a simulator that creates random paths and originates attacks. In Figure 10 we graph the mean, median and 95th percentile for the number of packets required to reconstruct paths of varying lengths over 1,000 random test runs for each length value. We assume a marking probability of $\frac{1}{25}$. Note that while the convergence time is theoretically exponential in the path length, all three lines appear linear due to the finite path length and appropriate choice of marking probability.

We see that most paths can be resolved with between one and two

Cloudflare, Inc. Exhibit 1003 Page 909

thousand packets, and even the longest paths can be resolved with a very high likelihood within four thousand packets. To put these numbers in context, most flooding-style denial of service attacks send many hundreds or thousands of packets each second. The analytic bounds we described earlier are conservative, but in our experience they are no more than 30% higher than our experimental results.

6. LIMITATIONS AND FUTURE WORK

There are still a number of limitations and loose ends in our approach. We discuss the most important of these here:

- backwards compatibility,
- distributed attacks,
- path validation, and
- approaches for determining the attack origin.

6.1 Backwards compatibility

The IP header encoding as we have described it has several practical limitations. It negatively impacts users that require fragmented IP datagrams and is currently incompatible with parts of IPsec [28] (the Authentication Header provides cryptographic protection for the identification field and therefore the field cannot be safely modified by routers). These problems are hardly unique to traceback and are inherent limitations that come about from attempting to coexist with or co-opt protocol features that did not anticipate a new use. One way to address this issue, originally proposed by John Hawkinson, is to selectively enable traceback support in response to operational needs. A "request for traceback" from a particular network could be encoded as a BGP attribute in the network's route advertisement. Routers receiving such an advertisement would enable traceback support on packets destined for that network. Since a network requesting such support is presumably already suffering under an attack, any minor service degradation for fragmented or authenticated flows would be acceptable.

Finally, our scheme does not address implementation in IPv6, the proposed successor to IPv4, which does not have an identification field [18]. While we do not attempt to propose a complete encoding here, we believe that the same techniques we have proposed could also be employed within IPv6, perhaps by overloading the 24-bit *flow label* field (without any further modifications this would result in roughly a factor of three increase in the number of packets required to reconstruct a path).

6.2 Distributed attacks

For large distributed attacks, the practical implementation we have described has serious limitations due to the difficulty in correctly grouping fragments together. Consequently, the probability of misattributing an edge, as well as the amount of state needed to evaluate this decision, increases very quickly with the fan-out of an attack. There is significant future work in designing alternative encoding methods that scale their robustness as they receive more data.

6.3 Path validation

Some number of the packets sent by the attacker are unmarked by intervening routers. The victim cannot differentiate between these packets and genuine marked packets. Therefore an attacker could insert "fake" edges by carefully manipulating the identification fields in the packets it sends. While the distance field prevents an attacker from spoofing edges between it and the victim – what we call the *valid suffix* – nothing prevents the attacker from spoofing extra edges past the end of the true attack path.

There are several ways to identify the valid suffix within a path generated by the reconstruction procedure. With minimal knowledge of Internet topology one can differentiate between routers that belong to transit networks (e.g. ISPs) and those which belong to stub networks (e.g. enterprise networks). Generally speaking, a valid path will never enter a stub network and then continue into a transit network. Moreover, simple testing tools such as `traceroute` should enable a victim to determine if two networks do, in fact, connect. More advanced network maps [8, 23] can resolve this issue in an increasing number of cases.

A more general mechanism is to provide each router with a "secret" that is sent along with each marked packet (perhaps in the single unallocated bit in the IP flags field). When the victim wants to validate a router in the path, it contacts the associated network (possibly out of band, via telephone or e-mail) and obtains the secret used by the router at the time of the attack. To guard against replay, the secret can be time-varying and hashed with the packet contents. Since the attacker will not know the router's secret, they will not be able to include the proper bit in their forged edge-id fragments. By eliminating edge-ids for which the secret in their constituent fragments can not be validated, we can prune a candidate attack path to only include the valid suffix.

6.4 Attack origin detection

While our IP-level traceback algorithm could be an important part of the solution for stopping denial-of-service attacks, it is by no means a complete solution. Our algorithm attempts to determine the approximate origin of attack traffic – in particular, the earliest traceback-capable router involved in forwarding attack traffic from the source that directly generated it. As mentioned earlier, there are a number of reasons why this may differ from the true source of the attack: attackers can hide their true identities by "laundering" attacks through third parties, either indirectly (e.g. smurf attacks [13] or DNS reflectors [15]) or directly via compromised "stepping stone" machines or IP-in-IP tunnels. While there is ongoing work on following attackers through intermediate hosts [45, 39], there are still significant challenges in developing a generally applicable and universally deployable solution to this problem. One interesting possibility enabled by the packet marking approach is to extend traceback across "laundering points". For example, identifying marks could be copied from a DNS request packet into the associated DNS reply, thereby allowing the victim to trace the full causal path.

Even in absence of such "laundering", our approach does not reveal the actual host originating the attack. Moreover, since hosts can forge both their IP source address and MAC address the origin of a packet may never be explicitly visible. On shared media such as FDDI rings, this problem can only be solved by explicit testing. However, on point-to-point media, the input port a packet arrives on is frequently enough to determine its true origin. On other media, there may be a MAC address, cell number, channel, or other hint that would help to locate the attack origin. In principle, our algorithm could be modified to report this information by occasionally marking packets with a special edge-id representing a link between the router and the input port on which the packet arrived (or other

Cloudflare, Inc. Exhibit 1003 Page 910

"hint" information). We have not explored the design of such a feature in any depth.

Finally, traceback is only effective at finding the source of *attack traffic*, not necessarily the *attacker* themselves. Stopping an attack may be sufficient to eliminate an immediate problem, but long term disincentives may require a legal remedy and therefore the forensic means to determine an attacker's identity. Even with perfect traceback support, unambiguously identifying a sufficiently skilled and paranoid attacker is likely to require cooperation from law enforcement and telecommunications organizations.

7. CONCLUSION

In this paper we have argued that denial-of-service attacks motivate the development of improved traceback capabilities and we have explored traceback algorithms based on packet marking in the network. We have shown that this class of algorithm, best embodied in *edge sampling*, can enable efficient and robust multi-party traceback that can be incrementally deployed and efficiently implemented. As well, we have developed variant algorithms that sacrifice convergence time and robustness for reduced per-packet space requirements. Finally, we have suggested one potential deployment strategy using such an algorithm based on overloading existing IP header fields and we have demonstrated that this implementation is capable of fully tracing an attack after having received only a few thousand packets. We believe our solution represents a valuable first step towards an automated network-wide traceback facility. Several areas remain to be addressed in future work, such as the combination of widely distributed attacks and points of indirection such as reflectors.

Acknowledgments

This paper has benefited from conversations with many different people – far more than can be acknowledged completely here. Still we would like to particularly thank Bill Cheswick and Hal Burch for early access to their work in this area, Sean McCreary and KC Claffy for access to their packet trace data, Alec Wolman for help with jgraph, Przemek Pardyak, Mike Swift, Neil Spring, Gretta Bartels, Robert Grimm and Geoff Voelker for commenting on early drafts of the paper, and constructive feedback from John Hawkinson, Robert Stone, Jeff Mogul, Robert Moskowitz, Greg Minshall, Tony Li, Curtis Villamizar, Steve Corbato, and countless others. Finally, we thank both the anonymous reviewers and our shepherds, Vern Paxson and Craig Partridge, for their efforts.

8. REFERENCES

[1] F. Baker. Requirements for IP Version 4 Routers. RFC 1812, June 1995.

[2] G. Banga, P. Druschel, and J. Mogul. Resource Containers: A New Facility for Resource Management in Server Systems. In *Proceedings of the 1999 USENIX/ACM Symposium on Operating System Design and Implementation*, pages 45–58, Feb. 1999.

[3] S. M. Bellovin. Security Problems in the TCP/IP Protocol Suite. *ACM Computer Communications Review*, 19(2):32–48, Apr. 1989.

[4] S. M. Bellovin. ICMP Traceback Messages. Internet Draft: draft-bellovin-itrace-00.txt, Mar. 2000.

[5] R. Braden. Requirements for Internet Hosts – Communication Layers. RFC 1122, Oct. 1989.

[6] H. Burch and B. Cheswick. Tracing Anonymous Packets to Their Approximate Source. Unpublished paper, Dec. 1999.

[7] R. L. Carter and M. E. Crovella. Dynamic Server Selection Using Dynamic Path Characterization in Wide-Area Networks. In *Proceedings of the 1997 IEEE INFOCOM Conference*, Kobe, Japan, Apr. 1997.

[8] B. Cheswick and H. Burch. Internet Mapping Project. `http://cm.bell-labs.com/who/ches/map/index.html`, 2000.

[9] Cisco Systems. Configuring TCP Intercept (Prevent Denial-of-Service Attacks). Cisco IOS Documentation, Dec. 1997.

[10] K. Claffy and S. McCreary. Sampled Measurements from June 1999 to December 1999 at the AMES Inter-exchange Point. Personal Communication, Jan. 2000.

[11] Computer Emergency Response Team. CERT Advisory CA-96.26 Denial-of-Service Attack via pings. `http://www.cert.org/advisories/CA-96.26.ping.html`, Dec. 1996.

[12] Computer Emergency Response Team. CERT Advisory CA-97.28 IP Denial-of-Service Attacks. `http://www.cert.org/advisories/CA-97.28.smurf.html`, Dec. 1997.

[13] Computer Emergency Response Team. CERT Advisory CA-98.01 "smurf" IP Denial-of-Service Attacks. `http://www.cert.org/advisories/CA-98.01.smurf.html`, Jan. 1998.

[14] Computer Emergency Response Team. CERT Advisory CA-2000-01 Denial-of-Service Developments. `http://www.cert.org/advisories/CA-2000-01.html`, Jan. 2000.

[15] Computer Emergency Response Team. CERT Incident Note IN-2000-04 Denial-of-Service Attacks using Nameservers. `http://www.cert.org/incident_notes/IN-200-04.html`, Apr. 2000.

[16] Computer Security Institute and Federal Bureau of Investigation. 1999 CSI/FBI Computer Crime and Security Survey. Computer Security Institute publication, Mar. 1999.

[17] Cooperative Associationfor Internet Data Analysis. Skitter Analysis. `http://www.caida.org/Tools/Skitter/Summary/`, 2000.

[18] S. Deering. Internet protocol, version 6 (ipv6). RFC 2460, Dec. 1998.

[19] W. Feller. *An Introduction to Probability Theory and Its Applications (2nd edition)*, volume 1. Wiley and Sons, 1966.

[20] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing. RFC 2267, Jan. 1998.

[21] J. Glave. Smurfing Cripples ISPs. Wired Technolgy News: (`http://www.wired.com/news/news/technology/story/9506.html`), Jan. 1998.

Cloudflare, Inc. Exhibit 1003 Page 911

[22] I. Goldberg and A. Shostack. Freedom Network 1.0 Architecture and Protocols. Zero-Knowledge Systems White Paper, Nov. 1999.

[23] R. Govindan and H. Tangmunarunkit. Heuristics for Internet Map Discovery. In *Proceedings of the 2000 IEEE INFOCOM Conference*, Tel Aviv, Israel, Mar. 2000.

[24] L. T. Heberlein and M. Bishop. Attack Class: Address Spoofing. In *1996 National Information Systems Security Conference*, pages 371–378, Baltimore, MD, Oct. 1996.

[25] J. D. Howard. *An Analysis of Security Incidents on the Internet*. PhD thesis, Carnegie Mellon University, Aug. 1998.

[26] P. Karn and W. Simpson. Photuris: Session-Key Management Protocol. RFC 2522, Mar. 1999.

[27] C. Kent and J. Mogul. Fragmentation Considered Harmful. In *Proceedings of the 1987 ACM SIGCOMM Conference*, pages 390–401, Stowe, VT, Aug. 1987.

[28] S. Kent and R. Atkinson. Security architecture for the internet protocol. RFC 2401, Nov. 1998.

[29] C. Meadows. A Formal Framework and Evaluation Method for Network Denial of Service. In *Proceedings of the 1999 IEEE Computer Security Foundations Workshop*, Mordano, Italy, June 1999.

[30] J. Mogul and S. Deering. Path MTU Discovery. RFC 1191, Nov. 1990.

[31] R. T. Morris. A Weakness in the 4.2BSD Unix TCP/IP Software. Technical Report Computer Science #117, AT&T Bell Labs, Feb. 1985.

[32] V. Paxson. End-to-End Routing Behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, Oct. 1997.

[33] C. Perkins. IP Mobility Support. RFC 2002, Oct. 1996.

[34] J. Postel. Internet Protocol. RFC 791, Sept. 1981.

[35] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, May 1998.

[36] E. C. Rosen, Y. Rekhter, D. Tappan, D. Farinacci, G. Fedorkow, T. Li, and A. Conta. MPLS Label Stack Encoding. Internet Draft: draft-ietf-mpls-label-encaps-07.txt (expires March 2000), Sept. 1998.

[37] G. Sager. Security Fun with OCxmon and cflowd. Presentation at the Internet 2 Working Group, Nov. 1998.

[38] O. Spatscheck and L. Peterson. Defending Against Denial of Service Attacks in Scout. In *Proceedings of the 1999 USENIX/ACM Symposium on Operating System Design and Implementation*, pages 59–72, Feb. 1999.

[39] S. Staniford-Chen and L. T. Heberlein. Holding Intruders Accountable on the Internet. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, pages 39–49, Oakland, CA, May 1995.

[40] I. Stoica and H. Zhang. Providing Guaranteed Services Without Per Flow Management. In *Proceedings of the 1999 ACM SIGCOMM Conference*, pages 81–94, Boston, MA, Aug. 1999.

[41] R. Stone. CenterTrack: An IP Overlay Network for Tracking DoS Floods. In *to appear in Proceedings of thje 2000 USENIX Security Symposium*, Denver, CO, July 2000.

[42] W. Theilmann and K. Rothermel. Dynamic Distance Maps of the Internet. In *Proceedings of the 2000 IEEE INFOCOM Conference*, Tel Aviv, Israel, Mar. 2000.

[43] C. Villamizar. Personal Communication, Feb. 2000.

[44] M. Vivo, E. Carrasco, G. Isern, and G. O. Vivo. A review of port scanning techniques. *ACM Computer Communications Review*, 29(2):41–48, Apr. 1999.

[45] Y. Zhang and V. Paxson. Stepping Stone Detection. In *to appear in Proceedings of thje 2000 USENIX Security Symposium*, Denver, CO, July 2000.

Cloudflare, Inc. Exhibit 1003 Page 912

ATTACHMENT B

ACM DIGITAL LIBRARY | Association for Computing Machinery
ACM – Association for Computing Mac...
Journals Magazines Proceedings Books SIGs Conferences People
Search ACM Dig... Advanced Search
COMM

ARTICLE

Practical network support for IP traceback

Authors: Stefan Savage, David Wetherall, Anna Karlin, Tom Anderson
Authors Info & Affiliations

Publication: SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication • August 2000 • Pages 295–306 • https://doi.org/10.1145/347059.347560

595 4,029
eReader PDF

ABSTRACT

This paper describes a technique for tracing anonymous packet flooding attacks in the Internet back towards their source. This work is motivated by the increased frequency and sophistication of denial-

SIGCOMM '00: Proceedings of the conference on Applications,...
Practical network support for IP traceback
Pages 295–306
Feedback

Cloudflare, Inc. Exhibit 1003 Page 914

ACM DIGITAL LIBRARY | Association for Computing Machinery
ACM – Association for Computing Mac...
Journals Magazines Proceedings Books SIGs Conferences People
Search ACM Dig... Advanced Search
COMA

interactive operational support from Internet Service Providers (ISPs). Moreover, this traceback can be performed ``post-mortem'' -- after an attack has completed. We present an implementation of this technology that is incrementally deployable, (mostly) backwards compatible and can be efficiently implemented using conventional technology.

References

1. F. Baker. Requirements for IP Version 4 Routers. RFC 1812, June 1995.]]

2. G. Banga, P. Druschel, and J. Mogul. Resource Containers: A New Facility for Resource Management in Server Systems. In Proceedings of the 1999 USENIX/ACM Symposium on Operating System Design and Implementation, pages 45-58, Feb. 1999.]]

3. S. M. Bellovin. Security Problems in the TCP/IP Protocol Suite. ACM Computer Communications Review, 19(2):32-48, Apr. 1989.]]

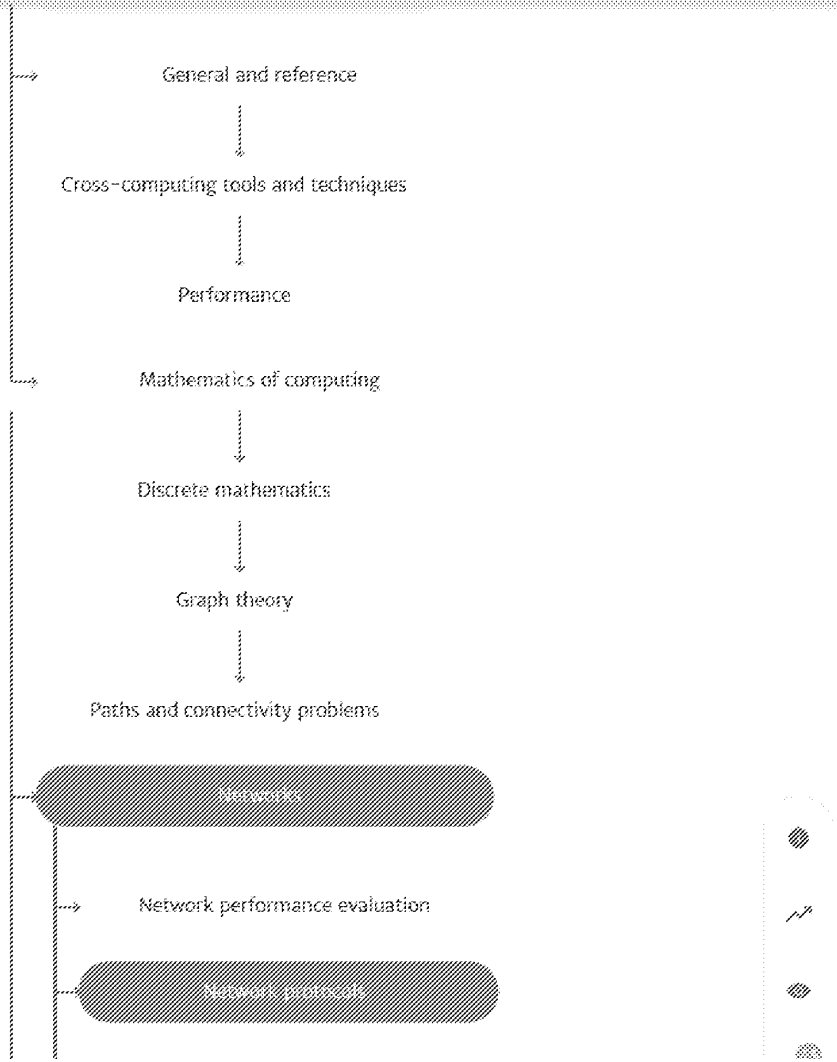
Show All References

Index Terms

Feedback

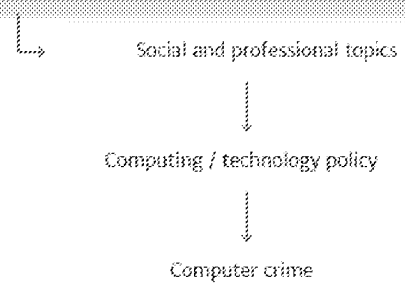
Cloudflare, Inc. Exhibit 1003 Page 915

ACM DIGITAL LIBRARY
Association for Computing Machinery
ACM - Association for Computing Mac...
Journals Magazines Proceedings Books SIGs Conferences People
Search ACM Dig...
Advanced Search
CCSMA



Feedback

Cloudflare, Inc. Exhibit 1003 Page 916



Comments

It appears that Recommendations are not enabled on the current forum. If you are a moderator you can enable this feature in the settings.

Cloudflare, Inc. Exhibit 1003 Page 917

ACM DIGITAL LIBRARY
Association for Computing Machinery
ACM - Association for Computing Mac...
Journals Magazines Proceedings Books SIGs Conferences People
Search ACM Dig...
Advanced Search
GOMA
Tweet Share
Sort by Newest
Nothing in this discussion yet.
View Table Of Contents
Categories
Journals
Magazines
Books
Proceedings
SIGs
Conferences
Collections
People
About
About ACM Digital Library
Subscription Information
Author Guidelines
Using ACM Digital Library
All Holdings within the ACM Digital Library
ACM Computing Classification System
Join
Join ACM
Join SIGs
Subscribe to Publications
Institutions and Libraries
Connect
Contact
Facebook
Twitter
LinkedIn
The ACM Digital Library is published by the Association for Computing Machinery. Copyright © 2021 ACM, Inc.
Terms of Usage | Privacy Policy | Code of Ethics
ACM DIGITAL LIBRARY
Association for Computing Machinery
Feedback

Cloudflare, Inc. Exhibit 1003 Page 918

Exhibit 1010

U.S. Patent Application Publication 2002/0032871 by Gerald R. Malan et al. ("Malan")

Customer No.: 000027683

Haynes and Boone, LLP
IP Section
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Telephone [214] 651.5000
Fax [214] 200.0853

US 20020032871A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2002/0032871 A1**

Malan et al. (43) **Pub. Date: Mar. 14, 2002**

(54) **METHOD AND SYSTEM FOR DETECTING, TRACKING AND BLOCKING DENIAL OF SERVICE ATTACKS OVER A COMPUTER NETWORK**

(75) Inventors: **Gerald R. Malan**, Ann Arbor, MI (US); **Farnam Jahanian**, Ann Arbor, MI (US)

Correspondence Address:
David R. Syrowik
Brooks & Kushman P.C.
1000 Town Center, 22nd Floor
Southfield, MI 48075-1351 (US)

(73) Assignee: **The Regents of The University of Michigan**, 3003 S. State Street, Ann Arbor, MI 48109

(21) Appl. No.: **09/855,808**

(22) Filed: **May 15, 2001**

Related U.S. Application Data

(63) Non-provisional of provisional application No. 60/231,479, filed on Sep. 8, 2000, which is a non-provisional of provisional application No. 60/231,480, filed on Sep. 8, 2000, which is a non-provisional of provisional application No. 60/231,481, filed on Sep. 8, 2000.

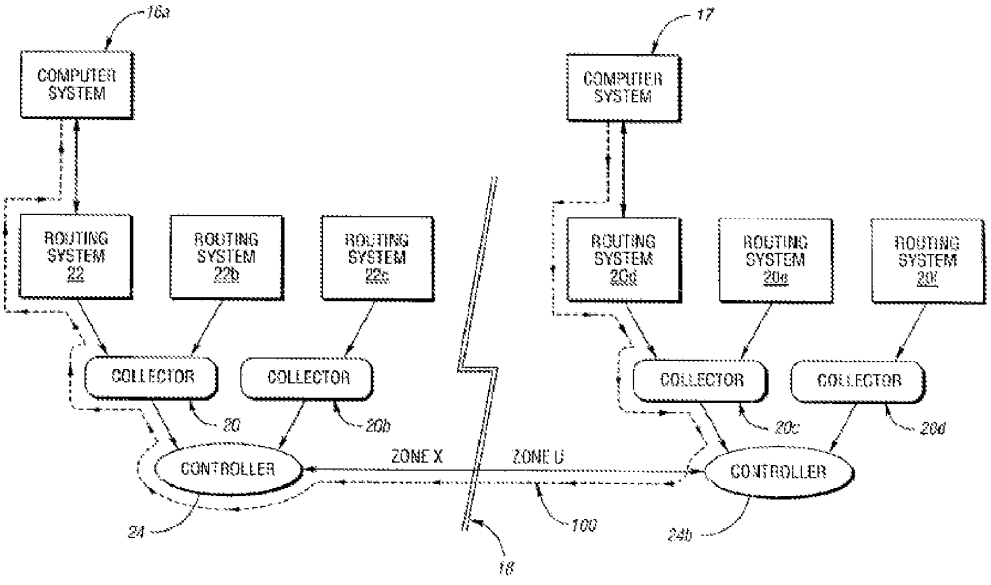
Publication Classification

(51) **Int. Cl.**[7] **G06F 11/30**

(52) **U.S. Cl.** **713/201**

(57) **ABSTRACT**

A system and method is provided for detecting, tracking and blocking denial of service ("DoS") attacks, which can occur between local computer systems and/or between remote computer systems, network links, and/or routing systems over a computer network. The system includes a collector adapted to receive a plurality of data statistics from the computer network and to process the plurality of data statistics to detect one or more data packet flow anomalies. The collector is further adapted to generate a plurality of signals representing the one or more data packet flow anomalies. The system further includes a controller that is coupled to the collector and is adapted to receive the plurality of signals from the collector. The controller is constructed and arranged to respond to the plurality of signals by tracking attributes related to the one or more data packet flow anomalies to at least one source, and to block the one or more data packet flow anomalies using a filtering mechanism executed in close proximity to the at least one source.



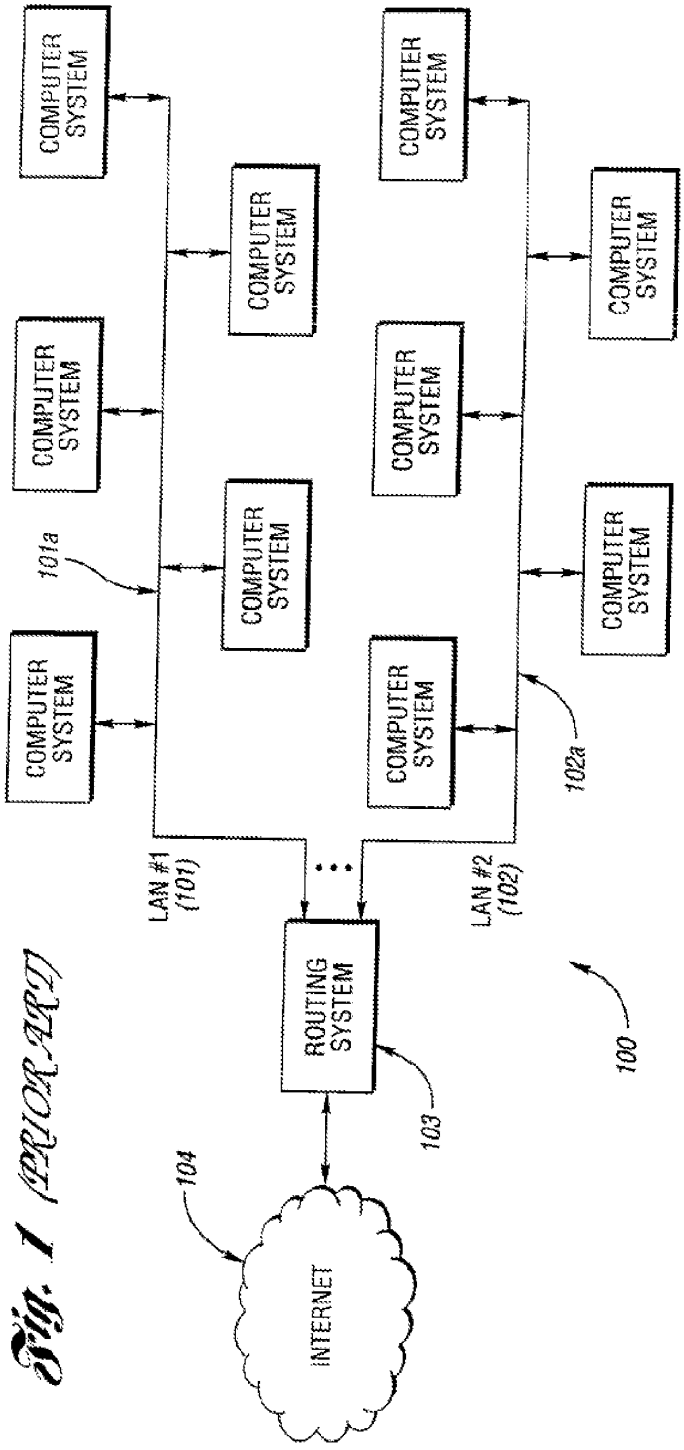


Fig. 1 (PRIOR ART)

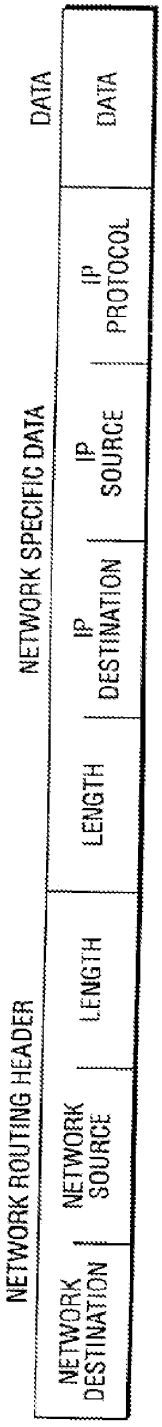


Fig. 2 (PRIOR ART)

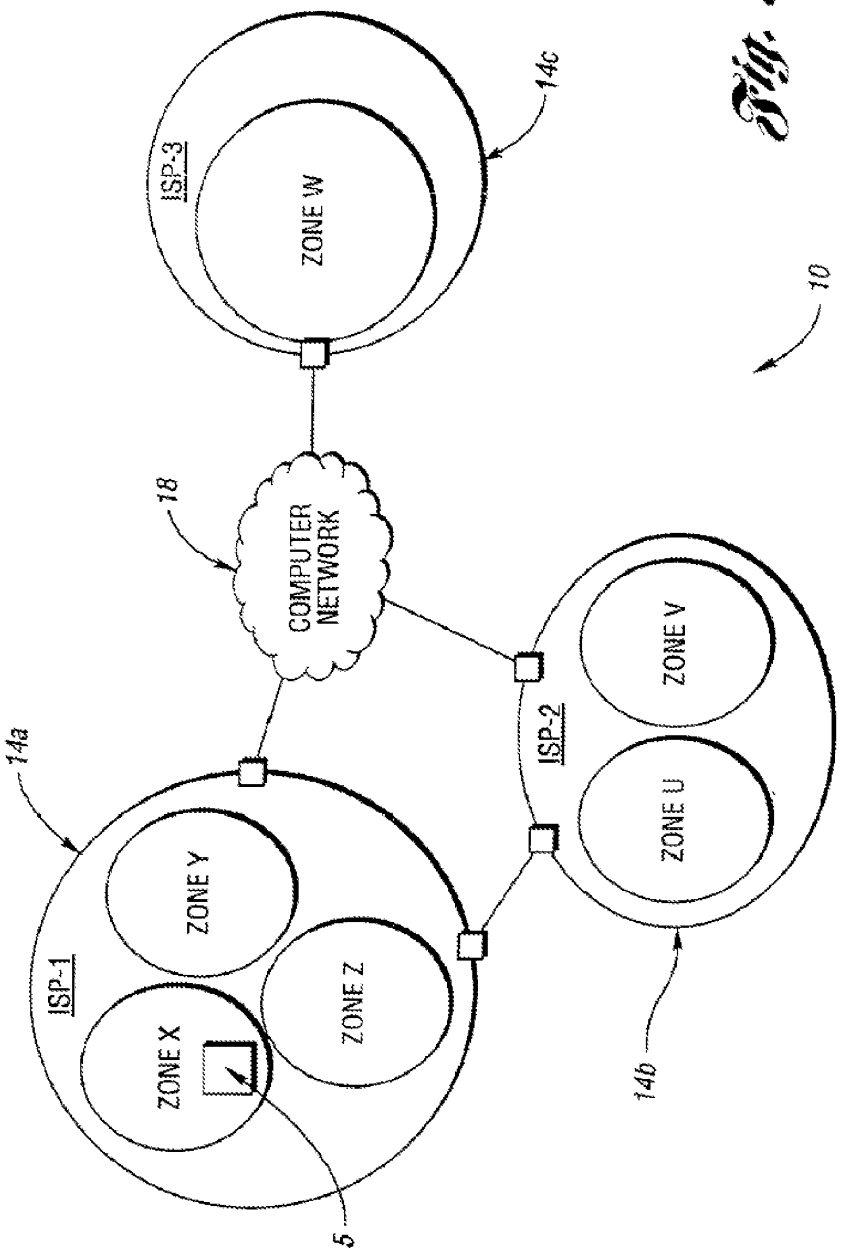

ISP-3
ZONE W
14c
10
18
COMPUTER NETWORK
ISP-2
ZONE V
ZONE U
14a
ISP-1
ZONE Y
ZONE Z
ZONE X
14b
5

Fig. 3

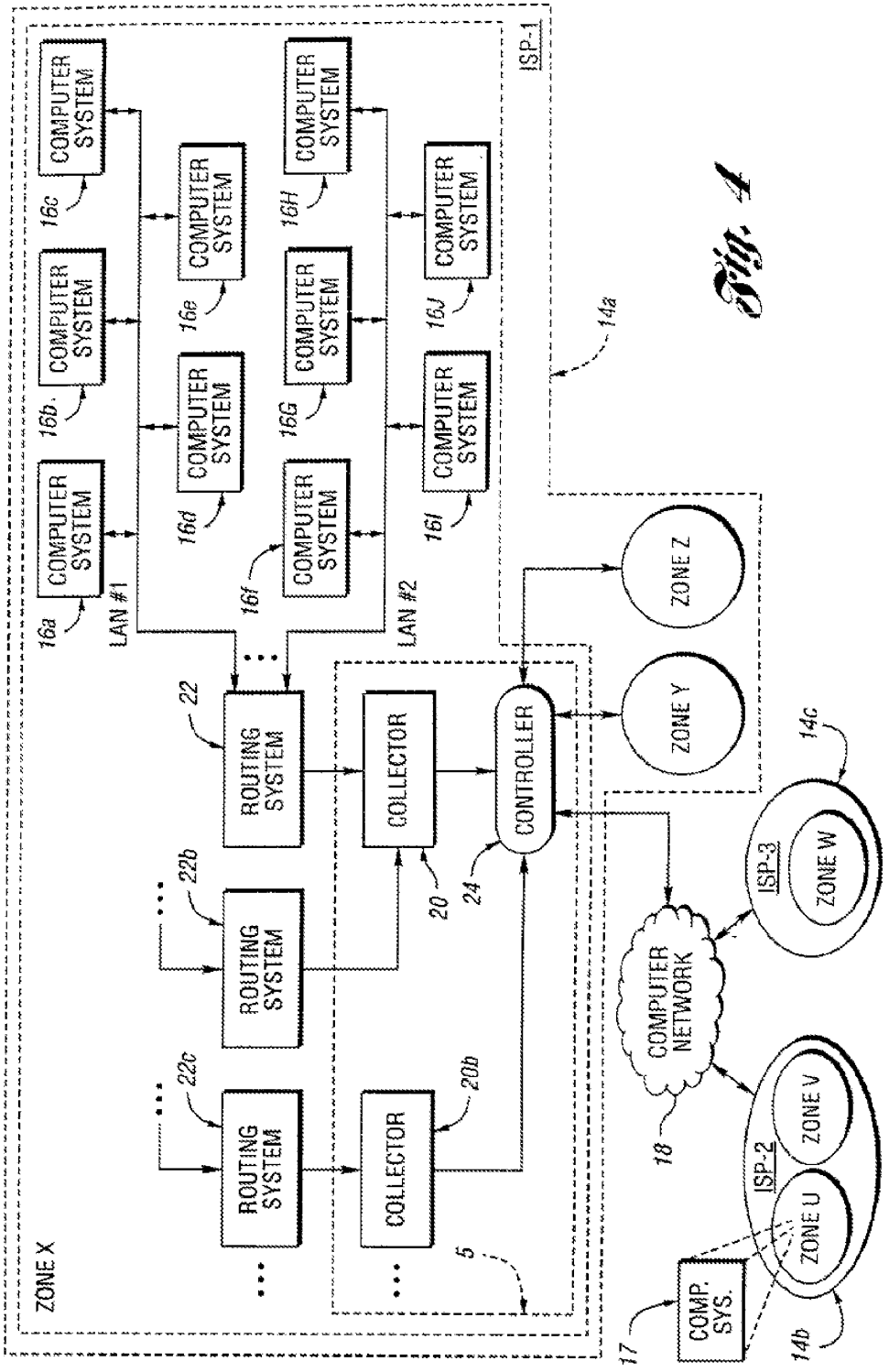

ZONE X
ISP-1
LAN #1
LAN #2
COMPUTER SYSTEM
16a
16b
16c
16d
16e
16f
16G
16H
16I
16J
ROUTING SYSTEM
22
22b
22c
COLLECTOR
20
20b
CONTROLLER
24
5
14a
ZONE Z
ZONE Y
ZONE W
ZONE V
ZONE U
ISP-3
ISP-2
14c
14b
COMPUTER NETWORK
18
COMP. SYS.
17

Fig. 4

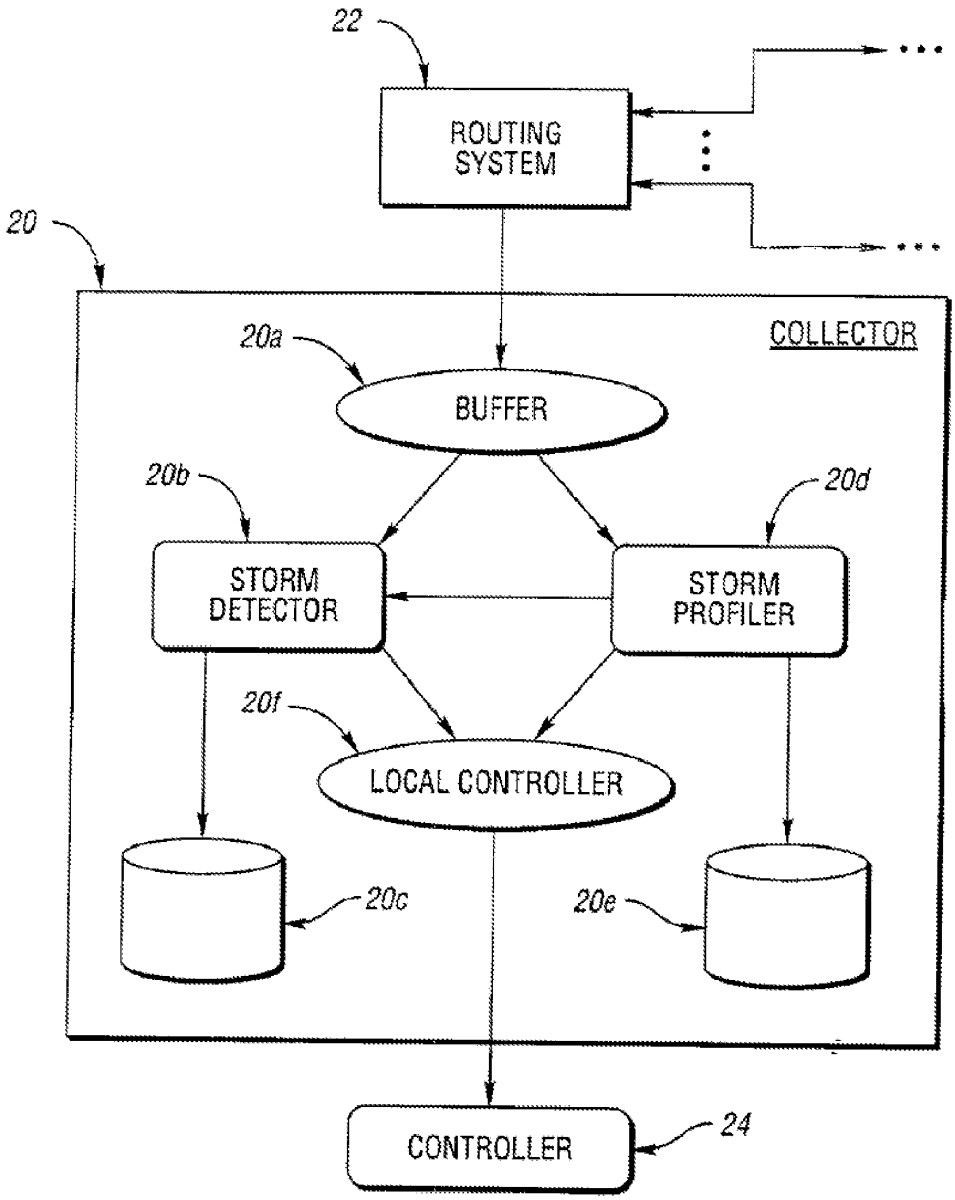

22
ROUTING SYSTEM
20
COLLECTOR
20a
BUFFER
20b
20d
STORM DETECTOR
STORM PROFILER
20f
LOCAL CONTROLLER
20c
20e
CONTROLLER
24

Fig. 5

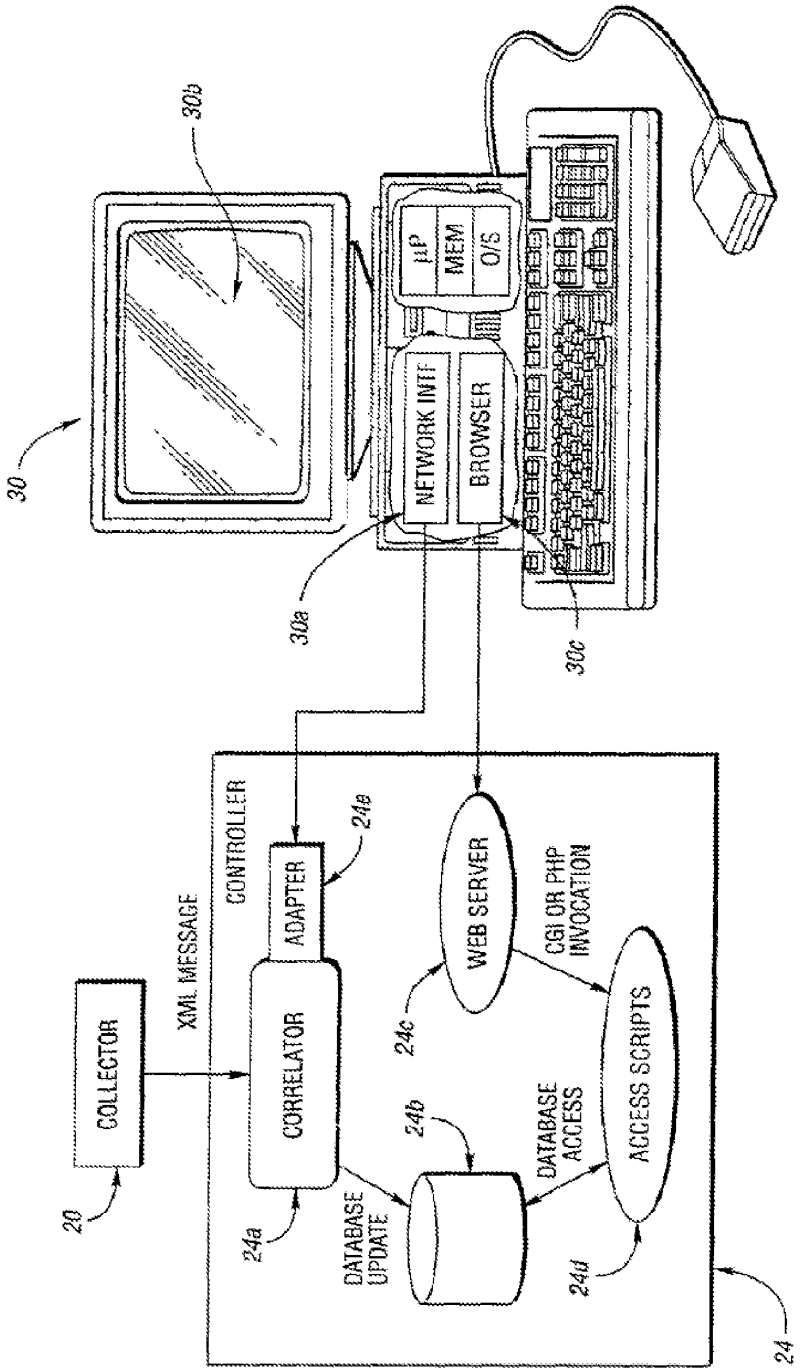

30b
30
μP
MEM
O/S
NETWORK INTF
BROWSER
30a
30c
CONTROLLER
ADAPTER
24e
WEB SERVER
CGI OR PHP INVOCATION
XML MESSAGE
COLLECTOR
CORRELATOR
24c
24b
DATABASE ACCESS
ACCESS SCRIPTS
DATABASE UPDATE
24a
24d
20
24

Fig. 6

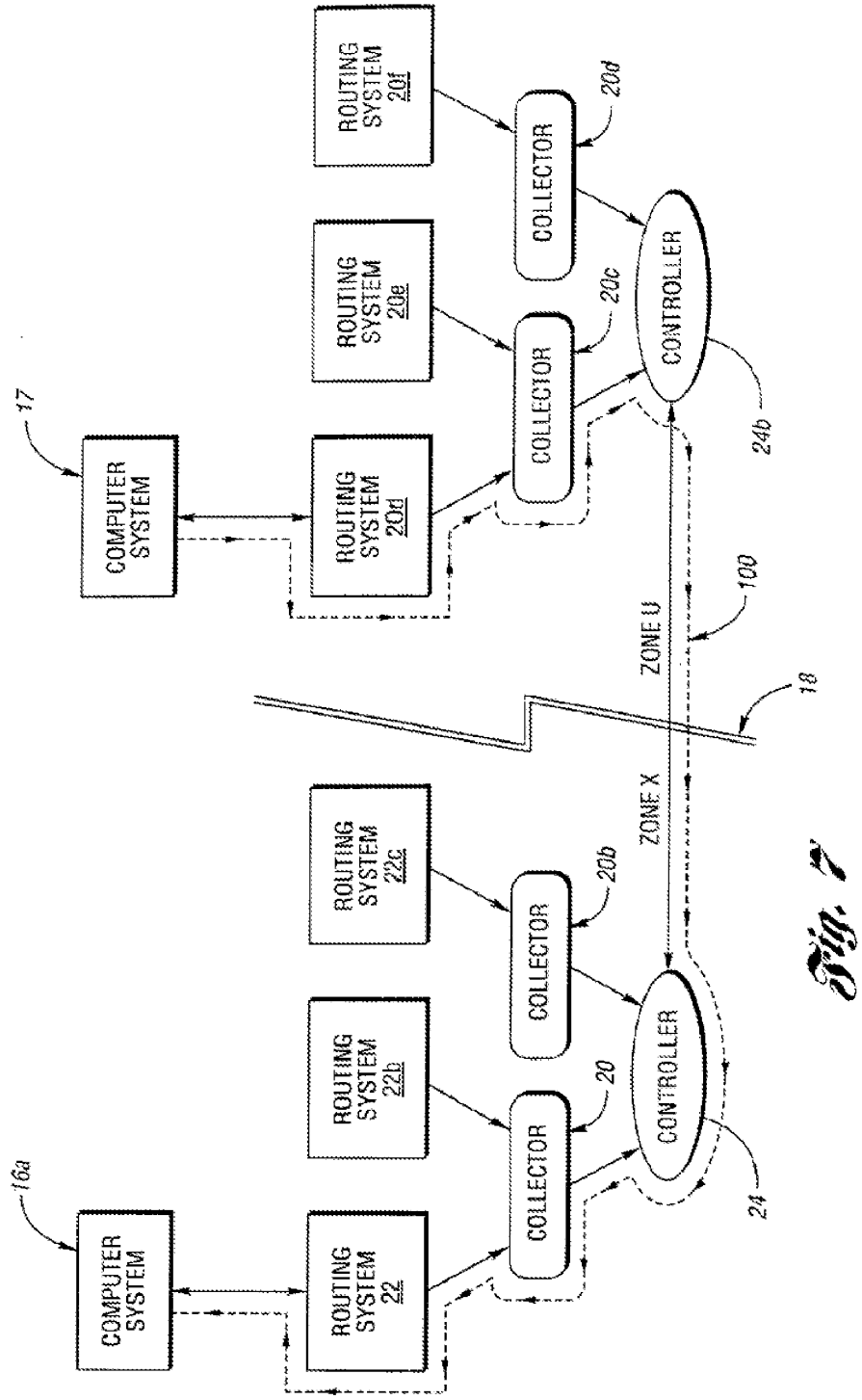

ROUTING SYSTEM 20f
ROUTING SYSTEM 20e
ROUTING SYSTEM 20d
COLLECTOR
COLLECTOR
CONTROLLER
24b
17
COMPUTER SYSTEM
ROUTING SYSTEM 22c
ROUTING SYSTEM 22b
ROUTING SYSTEM 22
COLLECTOR
COLLECTOR
CONTROLLER
24
16a
ZONE U
ZONE X
100
18
20
20b
20c

Fig. 2

METHOD AND SYSTEM FOR DETECTING, TRACKING AND BLOCKING DENIAL OF SERVICE ATTACKS OVER A COMPUTER NETWORK

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. provisional application Ser. No. 60/231,479, filed Sep. 8, 2000; U.S. provisional application Ser. No. 60/231,480, filed Sep. 8, 2000, and U.S. provisional application Ser. No. 60/231,481, filed Sep. 8, 2000, all of which are hereby incorporated by reference in their entirety.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[0002] This invention was made with Government support under Contract No. F30602-99-1-0527 awarded by DARPA. The Government has certain rights to the invention.

BACKGROUND OF THE INVENTION

[0003] 1. Field of the Invention

[0004] The present invention relates generally to data processing systems and more particularly to a method and system for detecting, tracking and blocking denial of service attacks over a local or remote computer network.

[0005] 2. Background Art

[0006] Computer systems are often interconnected into vast computer networks. The computer systems connected on such networks communicate with each other by sending information through their electronic connections. The networks can be organized into various types of topologies. **FIG. 1** illustrates one such topology that includes a network **100** having several local area networks **101-102** and that are connected to a routing system **103**. The computer systems of each local area network are connected to the communications link **101***a*-**102***a*. When a source computer system on a local area network **101** or **102** sends information to a destination computer system on the same local area network **101** or **102**, the source computer system prepares a packet that includes the address of the destination computer system and transmits the packet on the communications link **101***a* or **102***a*. The other computer systems on that same local area network **101** or **102** (i.e., connected to the communications link **101***a* or **102***a*) read the packet that was transmitted. The destination computer system detects that its address is included in that packet, and its processes the packet accordingly. Because of geographic and speed considerations, local area networks **101-102** typically only include a limited number of computer systems that are in close proximity. For example, a company with offices in several locations may have a local area network at each location. However, the users of the computer systems may need to send packets to one another regardless to which of local area networks **101-102** the users' computer systems are connected.

[0007] To allow packets to be sent from one local area network **101** or **102** to another local area network **101** or **102**, routing systems **103** have been developed. A routing system **103** is typically a dedicated special-purpose computer system to which each local area network **101-102** is connected. The routing system **103** maintains a cross-reference between computer system addresses and the local area network **101-102** to which each computer system is connected. The routing system **103** monitors the packets sent on each local area network **101-102** to detect (using the cross-reference) when a computer system on one local area network **101-102** is sending a packet to a computer system on another local area network **101** or **102**. When the routing system **103** detects such a packet, it forwards that packet onto the communications link **101***a* or **102***a* for the local area network **101** or **102** to which the destination computer system is connected. In this way, the routing system **103** interconnects each of the local area networks **101** and **102** into an overall network **100**. Similar routing techniques are used to interconnect networks other than local area networks **101-102**. For example, such routing techniques can be used on wide area networks (not shown) and on the Internet **104**.

[0008] Many different protocols have been developed to allow two computer systems to exchange information. If two computer systems support the same protocol, then they can exchange information. Certain protocols have been tailored to support the exchange of certain types of information efficiently. For example, the Internet protocol ("IP") was specified by the Department of Defense to facilitate the exchange of information between geographically separated computer systems. The IP specifies a destination in a packet format that identifies source and destination computer systems for data to exchange, but does not specify the format of the data itself. Several additional protocols may be used in conjunction with the IP to specify the format of the data. Two such additional protocols are the transmission control protocol ("TCP"), and the user datagram protocol ("UDP"). TCP and UDP further specify sub-protocols, such as the hyper-text transmission protocol ("HTTP") and the file transfer protocol ("FTP"), which specify the format of the data of the packet.

[0009] **FIG. 2** is a diagram illustrating a typical packet sent on a local area network. The packet includes a network routing header followed by protocol specific data. The network routing header may include the destination computer address, the source computer address, and the length of the packet. The protocol specific data includes identification of the protocol and the IP destination address, the IP source address, and the length of the IP portion of the packet. The data portion of the packet contains the sub-protocol identification plus other data of the packet. One specific field of the TCP and UDP sub-protocol is the port number. This port number is used to identify application protocols, which define network services that are available to remote systems.

[0010] One problem occurs when a first computer system maliciously sends a flood of packets to a target or second computer system, routing system or network link to overwhelm the reception resources or capacity of the target, which can result in either loss of connectivity to or failure of the target. This flood of packets based attack is commonly known as a denial of service attack ("DoS").

[0011] The most insidious types of DoS attacks occur when the initiator or first computer system hides their origin by forging the source Internet Protocol (IP) address on the attack packets. As a result, administrators and security officers of the target cannot determine the origin of the DoS attack. Further, the administrators and security officers of the target will not likely be able to avoid or shut down the DoS attack.

[0012] Conventional routing systems **103** have attempted to avoid DoS attacks by employing various types of packet filtering techniques in the form of firewalls at the entrance to the local area network **101-102**. Current implementations of packet filtering permit packets to be delivered to computer systems if the packet's format conforms to access list tables, which include a fixed format. This method is limited to the set of protocols and services defined in the particular access list table. Further, this method does not allow the introduction of different protocols or services which are not specified in the access list table. Finally, while firewall solutions may reduce unauthorized information from accessing a target, the firewall solutions do not reduce the impact that denial of service attacks can have on the availability of the target's bandwidth.

[0013] Other packet filtering schemes include a network administrator configuring a routing system **103** to restrict the type and timing of packets that are sent over the network **100**. For example, a network administrator may want to restrict packets that are generated by a computer game from being transmitted over the network **100** during normal business hours. A packet for a computer game may be identifiable, for example, by a TCP destination address, that indicates which application on the computer system identified by the IP destination address that is to receive the packet. Thus, the network administrator would configure the routing system **103** to not forward any such packets during normal business hours. Also, the network administrator may want to filter out packets based on their source and destination addresses. For example, a company CEO may only want to receive packets from certain source computer systems and not every computer system on the network **100**.

[0014] Present known filtering systems, such as packet filtering described above, have often proven either to be ineffective in preventing DoS attacks, or have severely limited access to communication services for communicating with other networks. In general, existing filtering systems disable certain critical communication services between the computer systems that deteriorate inter and intra computer system communications. Moreover, identifying the characteristics related to the DoS attacks can be impractical for network engineers and operators to accomplish by inspection alone, because of the voluminous amount of information associated with the characteristics. Finally, solutions for filtering attack traffic close to the local area network do not affect denial of service attacks that are directed at the heart of a service provider's routing infrastructure, such as attacks on network links or the routing infrastructure directly.

[0015] Previously works in this area of technology includes the following:

[0016] MCI's DoS Tracker: The DoS tracker's approach was a recursive script that would iterate over a set of routers. Network operators would invoke this script when a DoS attack had already been detected and identified at a specific point in the network (a customer's access router for example). The script would login to a router over its command line interface (CLI), and then turn on debugging. It would then examine the router's debugging output to identify interfaces that were affected by the denial of service attack.

[0017] The work was abandoned due to the performance impact caused by using the debugging feature, and the inability to continue the tracking across a network's core.

[0018] UUNet's Center Track: The Center Track work involves building a measurement overlay network by building tunnels from each of a network's edge routers to a set of measurement routers. Center Track is only used once an attack is detected by an external tool (or a customer calling on the phone and complaining). All of the target's traffic is off-ramped onto the Center Track overlay network, where its origin can be tracked using direct measurement or router debugging tools.

[0019] Network-based Intrusion Detection: Network-based Intrusion Detection (NID) systems are systems that are similar in that they look at a copy of the data in a network and identify malicious attacks. NID systems use passive packet capture techniques to examine the contents of every packet on a network and recreate both transport and application layer information to identify well-known attacks. However, because NID systems detect a wide spectrum of attacks, they do not scale to the highest bandwidth areas, like network service provider networks.

[0020] U.S. Pat. No. 4,817,080 to Soha discloses a system that measures traffic statistics by looking at packet contents. The system collects distributed measurements and forwards them to a centralized point.

[0021] U.S. Pat. No. 5,781,534 to Perlman et al. discloses apparatus for determining characteristics of a path by utilizing active probing along a network path to determine its characteristics. These characteristics are added to the packet as it traverses the network.

[0022] U.S. Pat. No. 5,968,176 to Nessett et al. discloses a system that utilizes many network elements to provide an umbrella countermeasure.

[0023] U.S. Pat. No. 5,991,881 to Conklin et al. discloses a system which flags intrusions and updates the status of the intruder's progress. This system only stores the packets with the source address of the attacker.

[0024] U.S. Pat. No. 6,078,953 to Vaid et al. discloses a system which classifies packets at the border of the network to provide quality of service. It polices traffic at the edge of the network.

[0025] U.S. Pat. No. 6,088,804 to Hill et al. discloses a system which correlates distributed attacks to build a path of the attack through the network. The system uses a training signature for attack identification. That is, the system is trained on attacks, and then compares current activity to this known misuse.

[0026] U.S. Pat. No. 6,134,662 to Levy et al. discloses a physical layer security manager for memory-mapped serial communications interface.

[0027] Therefore, an unsolved need remains for a system and method for detecting, tracking and blocking DoS attacks which can occur between local computer systems and/or between remote computer systems over a computer network, that overcomes the above-described limitations and deficiencies of the prior art.

SUMMARY OF THE INVENTION

[0028] In accordance with principles of the present invention, a system and method is provided for detecting, tracking and blocking DoS attacks, which can occur between local computer systems and/or between remote computer systems, network links, and/or routing systems over a computer network.

[0029] In one embodiment of the present invention, a system includes a collector adapted to receive a plurality of data statistics from the computer network and to process the plurality of data statistics to detect one or more data packet flow anomalies and to generate a plurality of signals representing the one or more data packet flow anomalies. The system further includes a controller which is coupled to the collector. The controller is constructed and arranged to receive and respond to the plurality of signals by tracking attributes related to the one or more data packet flow anomalies to at least one source. The controller is further constructed and arranged to block the one or more data packet flow anomalies using one or more filtering mechanisms executed in close proximity to the at least one source.

[0030] The one or more filtering mechanisms can include a plurality of filter list entries, such as access control list entries as well as firewall filter entries, and/or a plurality of rate limiting entries, such as committed access rate (CAR) entries.

[0031] In aspect of the present invention, the collector includes a buffer coupled to the computer network and a detector coupled to the buffer. The collector further includes a profiler coupled to the buffer and to the detector. The buffer is adapted to receive and process the plurality of data statistics to generate at least one record that is communicated to the profiler. The profiler processes the record to generate a predetermined threshold. The detector is adapted to receive and process the predetermined threshold and the at least one record to detect if attributes associated with the record exceed the predetermined threshold, which represents the one or more data packet flow anomalies.

[0032] The profiler may include means for aggregating the data statistics to obtain a traffic profile of network flows.

[0033] The data statistics may be aggregated based on at least one invariant feature of the network flows.

[0034] The data statistics may also be aggregated based on temporal, static network and dynamic routing parameters.

[0035] The at least one invariant feature may include source and destination endpoints.

[0036] The collector further includes a local controller coupled to the detector and to the profiler. The local controller is adapted to receive and respond to the one or more data packet flow anomalies by generating the plurality of signals, which represents the one or more data packet flow anomalies.

[0037] The detector includes a database for storing the at least one record, predetermined threshold, the one or more data packet flow anomalies, and related information. Similarly, the profiler includes a database for storing a plurality of data packet flow profiles and related information.

[0038] In an aspect of the present invention, the controller includes a correlator coupled to the collector. The correlator is adapted to receive and normalize the plurality of signals representing the one or more data packet flow anomalies. The correlator is further adapted to generate an anomaly table including the attributes related to the one or more data packet flow anomalies. The correlator includes a database for storing the anomaly table. Additionally, the correlator includes an adapter that is constructed and arranged to communicate the anomaly table to a computer device for further processing.

[0039] The controller further includes a web server and access scripts that cooperate with the web server to enable the computing device to access the database defined on the controller to view the anomaly table.

[0040] In accordance with the present invention, the method for detecting, tracking and blocking one or more denial of service attacks over a computer network includes the steps of collecting a plurality of data statistics from the computer network; processing the plurality of data statistics to detect one or more data packet flow anomalies; generating a plurality of signals representing the one or more data packet flow anomalies; and receiving and responding to the plurality of signals by tracking attributes related to the one or more data packet flow anomalies to at least one source.

[0041] The method further includes the step of blocking the one or more data packet flow anomalies in close proximity to the at least one source.

[0042] The step of collecting the plurality of data statistics includes buffering the plurality of data statistics; processing the plurality of data statistics to generate at least one record; and receiving and profiling the at least one record to generate a predetermined threshold.

[0043] The step of collecting the plurality of data statistics further includes detecting if attributes related to the at least one record exceed the predetermined threshold representing the one or more data packet flow anomalies.

[0044] The step of collecting the plurality of data statistics further includes responding locally to the one or more data packet flow anomalies by generating the plurality of signals representing the one or more data packet flow anomalies.

[0045] The step of receiving and responding to the plurality of signals includes correlating the plurality of signals representing the one or more data packet flow anomalies; and generating an anomaly table including the attributes related to the one or more data packet flow anomalies.

[0046] The step of receiving and responding to the plurality of signals further includes the step of communicating the anomaly table to a computing device for further processing.

[0047] The above objects and other objects, features, and advantages of the present invention are readily apparent from the following detailed description of the best mode for carrying out the invention when taken in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0048] **FIG. 1** is a high level block diagram of a conventional computer network system;

[0049] **FIG. 2** is an exemplary data packet format which can be adapted for communication over the conventional computer network system shown in **FIG. 1**;

[0050] **FIG. 3** is a high level block diagram of a computer network system according to one embodiment of the present invention;

[0051] **FIG. 4** is a partially exploded view of the computer network system shown in **FIG. 3**;

[0052] **FIG. 5** is a high level block diagram of the collector shown in **FIG. 4**;

[0053] **FIG. 6** is a high level block diagram of the controller shown in **FIG. 4**; and

[0054] **FIG. 7** is a high level block diagram exemplifying a DoS attack.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT(S)

[0055] For purposes of illustration and to facilitate a further understanding of the present invention, described below is a reference to an Internet-based computer network system and a method for processing data. However, as understood by one skilled in the art, the present invention is not limited to Internet-based systems and can include systems employing other computer networks as well as stand alone systems.

[0056] In accordance with principles of the present invention, a system and method is set forth for detecting, tracking and blocking DoS attacks, which can occur between local computer systems and/or between remote computer systems, network links, and/or routing systems over a computer network.

[0057] Referring to **FIG. 3**, a system **5** for detecting, tracking and blocking DoS attacks is incorporated in the computer network system **10** in accordance with one embodiment of the present invention. The system **5** can be located on a single server computer (not shown), which is in communication with components of the computer network system **10** or distributed over a plurality of server computers (not shown), which are also in communication with components of the computer network system **10**.

[0058] The computer network system **10** includes a plurality of Internet Service Provider computer networks **14***a*, **14***b* and **14***c* (hereinafter ISP computer network(s)") coupled over a computer network **18**. The ISP computer networks **14***a*, **14***b* and **14***c* can also be coupled directly to each other. Each of the ISP computer networks **14***a*, **14***b* and/or **14***c* can include a plurality of computer network zones. As exemplified in **FIG. 3**, the ISP computer network **14***a* includes computer network Zone X, Zone Y and Zone Z. The ISP computer network **14***b* includes computer network Zone U and Zone V. The ISP computer network **14***c* includes computer network Zone W.

[0059] **FIG. 4** shows a partially expanded view of the system **5**, which is incorporated in the partially expanded view of the computer network system **10**. In **FIG. 4**, Zone X of the ISP computer network **14***a* includes a number of local area networks ("LAN(s)") coupled to a central routing system **22**. Each LAN is coupled with a plurality of computer systems **16***a*, **16***b*, **16***c*, **16***e*, **16***f*, **16***g*, **16***h*, **16***i* and **16***j* (hereinafter collectively referred to as "computer system(s) **16**"). The computer network Zones Y and Z, which are also located on the ISP computer network **14***a*, can be similarly constructed and arranged as computer network Zones X. Further, the computer network Zones U and V, which are located on the ISP computer network **14***b* and the computer network Zone W, which is located on the ISP computer network **14***c*, can also be similarly constructed and arranged as computer network Zones X.

[0060] The system **5** includes a collector **20**, an optional collector **20***b* and a zone controller **24**. In Zone X, the collector **20** is coupled to the central routing system **22**. The collector **20** is further coupled to a zone controller **24**, which provides a primary interface to Zone X of the ISP computer network **14***a*. The computer network Zones Y and Z, which are also located on the ISP computer network **14***a* can be similarly constructed and arranged as computer network Zone X. Further, the computer network Zones U and V, which are located on the ISP computer network **14***b* and the computer network Zone W, which is located on the ISP computer network **14***c*, can also be similarly constructed and arranged as computer network Zones X.

[0061] In another embodiment, the collector **20** can be coupled to one or more other router systems, such as the routing system **22***b*, as exemplified in **FIG. 4**. In addition, the zone controller **24** can be coupled to one or more other collectors, such as the collector **20***b*, as also exemplified in **FIG. 4**. Further, the collector **20***b*, can be coupled to one or more other routing systems, such as the routing system **22***c*.

[0062] The zone controller **24** located in Zone X of the ISP Computer network **14***a* provides a primary interface to the computer network Zone Y and to the computer network Zone Z, which are both located on the ISP computer network **14***a*. The zone controller **24** further provides a primary interface to the computer network Zone U and the computer network Zone V, which are located on the ISP computer network **14***b*, over the computer network **18**. Similarly, the zone controller **24** further provides a primary interface to computer network Zone W, which is located on the ISP computer network **14***c*, over the computer network **18**.

[0063] In an embodiment of the present invention, the computer systems **16** located in computer network Zone X of the ISP computer network **14***a* can each comprise a conventional computer server such as an "NT-Server" which can be provided by Microsoft of Richmond, Wash. or a "Unix Solaris Server" which can be provided by Sun Micro Systems of Palo Alto, Calif. These computer systems **16** can be programmed with conventional Web-page interface software such as: "Visual Basic", "Java", "JavaScript", "HTML/DHTML", "C++", "J+", "Perl" or "Perlscript", or "ASP". These computer systems can further be programmed with an operating system, Web server software, Web Application software, such as an e-commerce application and computer network interface software.

[0064] Each of the routing systems **22**, **22***b* and **22***c*, as shown in **FIG. 4**, can be a conventional router, such as a "Cisco 12000", available from Cisco Corporation of San Jose, Calif. Further, each of the routing systems can be adapted to run data packet flow statistical software, such as Netflow™ software, also available from Cisco Corporation of San Jose, Calif. Alternatively, each of the routing systems, as shown in **FIG. 4**, can be another conventional router, such as an "M-40", available from Juniper Corporation of Sunnyvale, Calif. Further, each of the routing systems can be adapted to run data packet flow statistical software, such as Juniper Cflowd™ software, also available from Juniper

Corporation of Sunnyvale, Calif. The packet flow statistical software running on each of the routing systems **22**, **22***b* and **22***c* enable each of the routing systems **22**, **22***b* and **22***c* to gather and store data packet flow statistical information. The data packet flow statistical information can include the number of packets which have been communicated between computer systems **16**, the duration of communication between each of the computer systems **16**, the total number of packets communicated over each LAN (which is typically used for capacity planning) as well as other various data packet flow statistical information.

[0065] **FIG. 5** shows the collector **20** in detail. The collector includes an input buffer **20***a* coupled to the routing system **22**. The input buffer is coupled to a storm detector **20***b* and to a storm profiler **20***d*. The storm detector **20***b* includes a detector database and the storm profiler **20***d* includes a profiler database **20***e*. The collector **20** further includes a local controller **20***f*, which is coupled to the storm detector **20***b* and to a storm profiler **20***d*. The local controller **20***f* is further coupled to the zone controller **24**.

[0066] The collector **20** is adapted to receive the data packet flow statistical information from the routing system **22** and to process the data packet flow statistical information to detect data packet flow anomalies. The collector **22***b* of Zone X, as well as other various collectors (not shown), which are included in the other various Zones U, V, W, Y and Z are similarly constructed and arranged as the collector **20** of Zone X.

[0067] The input buffer **20***a*, located on collector **20**, is adapted to normalize or categorize the data packet flow statistical information and to generate a number of records including the normalized data packet flow statistical information. The storm detector **20***b* is adapted to detect the data packet flow anomalies by comparing the records to an anomaly pattern and/or a predetermined threshold. If components of the normalized data packet flow statistical information exceed the predetermined threshold, a data packet flow anomaly is detected. Thereafter, the detected data packet flow anomaly and data associated with the data packet flow anomaly, such as the source and destination addresses of the flow information can be stored in the detector database **20***e*.

[0068] The storm profiler module **20***d* is adapted to receive the normalized data packet flow statistical information or records from the input buffer **20***a* and to generate the predetermined threshold, which is concomitantly communicated to the storm detector module **20***b*. In this configuration, the predetermined threshold defined in the storm detector is adaptively adjusted based on changing trends or profiles of the normalized data packet flow statistical information received by the storm profiler **20***d*. The changing trends or profiles of the normalized data packet flow statistical information, for example, can include changes in the average bandwidth allocated to each of the computer systems **16** during a particular period of time or changes to the number of computer systems **16** communicating information at the same instant of time.

[0069] The local controller **20***f*, which is coupled to both the storm detector **20***b* and to the storm profiler **20***f*, is adapted to receive the data packet flow anomaly from the storm detector **20***b*, as well as data associated with the data packet flow anomaly, as previously described. After receiving the data packet flow anomaly and the associated data from the storm detector, the local controller **20***f* generates a signal or an alert message. The alert message can include pertinent information related to the anomaly. The pertinent information related to the anomaly can include the characteristics of the anomaly, the source and destination of the anomaly, the protocols involved and their sub-protocols, the detection mechanism used to identify the anomaly, the predetermined threshold, routing systems in the path of the anomaly, as well as the magnitude or severity of the anomaly. The alert message is communicated to the zone controller **24** to enable the zone controller **24** to further process the alert message and to enable the zone controller **24** to communicate the alert message to other Zones U, V, W, X, Y and Z and/or ISPs **14***b* and **14***c*.

[0070] In an embodiment, the collector takes samples of several types of statistics, which are obtained by the router **22**, such as single packet statistics and flow-based statistics. Single packet statistics provide essential information about a set of packets entering a forwarding node or router **22**. Some of the single packet statistics can include: destination and source IP addresses, incoming interface, protocol, ports, and length. After collection of these single-packet statistics, the collector can process the statistics as described above to adaptively adjust the predetermined threshold defined in the storm detector, which detects the packet anomalies.

[0071] Flow-based statistics include a set of packets that are related to the same logical traffic flow. The concept of flow-based statistics is generally defined as a stream of packets that all have the same characteristics, such as, source address, destination address, protocol type, source port, and destination port. The flow-based statistics may be either uni-directional or bidirectional. Single-packet statistics can be aggregated to generate a single flow-based statistic. An example of the single flow-based statistic can include a flow duration, number of packets included over a predetermined duration, mean bytes per packet, etc.

[0072] Referring further to **FIG. 6**, the zone controller **24** includes a correlator **24***a* coupled to the collector **20**. The correlator **24***a* includes a communication interface adapter **24***e*. The zone controller **24** further includes an alert message database **24***b*, which is coupled to the correlator module **24***a*. A web server **24***c* and access scripts software **24***d* are also defined on the controller **24**.

[0073] The zone controller **24** is adapted to receive a plurality of alert messages from the collector **20**, and to process the alert messages by aggregating the alert messages based on the pertinent information related to the anomaly, as described above. The zone controller **24** of Zone X, as well as other various controllers (not shown), which are included in the other various Zones U, V, W, Y and Z are similarly constructed and arranged as the controller **24** of Zone X.

[0074] More precisely, the correlator **24***a* is adapted to receive and categorize the alert messages and to generate a number of tables including the categorized alert messages. The tables including the categorized alert messages are stored in the alert message database **24***b*, which is coupled to the correlator module **24***a*. The correlator module **24***a* is further adapted to compare the alert messages to determine if trends exist. One example of a trend can be a plurality of alert messages that are traceable through the computer network system **10** to a particular computer system **16**.

Another example of trend can be a plurality of alert messages that include similar characteristics.

[0075] The communication interface adapter **24***e* operates to provide a communication interface to an external computer device **30**, such as a notebook computer, desktop computer, server or personal digital assistant ("PDA"). The personal computing device **30** can be adapted to run network management interface software **30***a*, such as HP Openview™, which can be obtained from Hewlett-Packard Company of Palo Alto, Calif. The network management interface software **30***a* is adapted to interface with the alert message database **24***b* and to provide a graphical user interface ("GUI") on the display **30***b* of the computing device **30**. Thereafter, a network administrator can view and respond to the alert messages.

[0076] Alternatively, the personal computing device **30** can include a conventional web browser **30***c*, which is similarly adapted to interface with the alert message database **24***b* via a web server **24***c* and access scripts module **24***d* and to provide a graphical user interface ("GUI") on the display **30***b* of the computing device **30**. Similar to that described above, the network administrator can view and respond to the alert messages.

[0077] Once the controller has received the alert message from the collector **20**, the controller **24** can apply several approaches to trace the DoS attack back to its origin, such as, directed tracing or distributed correlation. In directed tracing, information related to the computer network system topology is processed to work backwards towards the source or origin of the DoS attack. Directed tracing relies on the fact that both the router system's incoming interface statistic for a DoS attack and information related to the computer network system **10** topology are known to determine what routers are upstream on a particular link that carried the DoS attack packet. With this knowledge, upstream routers (not shown) can then be queried for their participation in transiting the attack packet. It is useful to note that since these upstream routers are looking for a specific attack signature, it is much easier to find the statistics related to the attack packet.

[0078] In distributed correlation, the controller **24** compares the attack signature or characteristic information related to the DoS attack with similar information detected at other routers **22***b* and **22***c* in the computer network system **10**. DoS attack signatures that substantially match are grouped and implicitly form the path from the source of the DoS attack to the target. This contrasts with the directed tracing approach, as previously described, where a general attack profile is extracted from every router's statistics to uncover the global path for the DoS attack packet.

[0079] After detection and tracing of the DoS attack packet, the controller **24** blocks DoS attacks as close to their Source as possible. By taking a global view of the ISP computer networks **14***a*, **14***b* and **14***c*, the controller **24** is able to coordinate the configuration of the routing systems **22**, **22***b* and/or **22***c* to filter certain types of traffic by employing either custom filtering hardware (not shown) or filtering mechanisms included in the routing systems. The custom filtering hardware can be incrementally deployed in the network. Example filtering mechanisms can include Access Control List entries ("ACLs"), and Committed Access Rate ("CAR") limiters, which can be provided by Cisco Systems Corporation of San Jose, Calif. An example of filtering hardware can include Internet Processor **11**, which can be provided by Juniper Networks Corporation of Sunnyvale, Calif., which can be utilized to download coarse-grained filters that will remove unwanted DoS attacks in real-time.

[0080] Referring again to **FIG. 4**, in one specific example, a DoS attack from a computer system **17** located in Zone U of ISP computer network **14***b* to one specific computer system **16***a* of Zone X can be detected, tracked and blocked by the system **5** of the present invention.

[0081] In this example, the DoS attack executed by the computer system **17** includes a SYN-packet flood DoS attack with spoofed source addresses. SYN-packets are TCP/IP packets that initiate data transfer sessions. As such, a SYN-packet flood denies legitimate traffic access to the targeted computer system **16***a*, because it uses up available bandwidth and consumes predefined computer system **16***a* resources. A spoofed source addresses is one in which the attacking computer system **17** hides it actual computer network location from the targeted computer system **16***a* by forging the return address on the TCP/IP data packet (**FIG. 2**). This makes it difficult to identify the source of the traffic when examining forensic data at the targeted computer system **16***a*.

[0082] Referring further to **FIG. 7**, the specific trajectory of the SYN-packet flood attack from the computer system **17** of Zone U located in the ISP-2 computer network **14***b* to computer system **16***a* of Zone X located in the ISP-1 computer network **14***a* is illustrated by the DoS attack path **100**. The DoS attack path **100** commences at the attacking computer system **17** and extends through the routing system **22***d*, through the collector **20***c*, through the controller **24***b*, through the computer network **18**, through the controller **24**, through the collector **20**, through the routing system **22** and to the targeted computer system **16***a*.

[0083] After the SYN-packets flow through the routing system **22**, the routing system **22** generates flow statistics, which are exported to the collector **20**. These flow statistics describe the traffic flow characteristics between computer system **17** (DoS attacker) and the computer system **16***a* (target of DoS attack). The SYN-packet flood attack is represented in these exported flow statistics as the computer system **16***a* receiving an unusually high number of TCP sessions. This anomalous traffic is detected at the collector **20** and an alert message is communicated to the controller **24**. After the controller **24** receives the alert message, it schedules a periodic sampling of anomaly statistics from collector **20**, which can be represented by a pair of request and reply messages communicated between the collector **20** and the controller **24**.

[0084] Referring again to **FIG. 5**, during this SYN-packet flood attack, the collector **20** collects flow statistics related to the SYN-packets and stores the flow statistics in the buffer **20***a*, which is located on the collector **20**. The buffer **20***a* normalizes the incoming flow-statistics to form records. The records are places into a shared table. The storm detector module **20***b* analyzes the records in this shared table and detects anomalous traffic. In this example, the storm detector **20***b* detects the pattern of records as a SYN-packet flood attack, because the number of records exceeds a predetermined threshold defined on the storm detector **20***b*. The

storm profiler **20***d* also analyzes the records and based on this analysis, the storm profiler **20***d* adaptively adjusts the predetermined threshold defined on the storm detector **20***b*. After detecting the SYN-packet flood attack, the storm detector **20***b* sends an alert message along with a signature (e.g. a fingerprint of the alert) to the local controller **20***f*. The local controller **20***f* adds the signature of the alert to a table in memory, which represents the on-going local anomalies. When one of these local ongoing anomalies reaches a significant level of interest (e.g. a second predetermined threshold), such as a long duration or high severity, the local controller **20***f* notifies an anomaly-profiler module (not shown) to add a new anomaly to the set of current-anomalies that it measures. Thereafter, the anomaly-profiler module analyzes the normalized flow statistics in buffer **20***a* that are related to the anomaly and begins to collect long-term statistics about the anomaly. Furthermore, the anomaly-profiler places periodic snapshots of these long-term statistics into the storm profiler database **20***e*, which is located on the collector **20**. At the same time, the local controller forwards the alert to the controller **24** as an alert message. The controller **24** can periodically request updated anomaly information, which in this example relates to a SYN-packet flood attack, from the local controller **20**. The local controller **20** can respond by providing the controller **24** with the most recently collected long-term statistics related to the anomaly.

[0085] As shown in **FIG. 6**, the specific operation of the controller **24** includes receiving the alert messages, anomaly fingerprints and anomaly statistical summaries from the collector **20** at the correlator **24***a* located on the controller **24**. Upon receipt of the alert message from collector **20**, the correlator **24***a* schedules a periodic request for updated anomaly statistical summaries. The correlator **24***a* translates the updated anomaly statistical summaries and correlates their features using attributes in the anomaly fingerprint to identify system-wide anomalies. These controller-specific anomaly statistics are then translated into system-wide representation anomalies, which are subsequently stored in the database **24***b*.

[0086] In the SYN-packet flood based attack example, the correlator **24***a* located on the controller **24** sends a simple network management protocol ("SNMP") alert message to the network management interface **30***a* located on the personal computing device **30**. This alert message notifies the network administrator and/or security operators as to the presence of the SYN-packet based flood attack. Included in this alert message is the network address, such as the universal resource locator ("URL") that describes the anomaly's location in the database **24***b* of the controller **24**. The network management interface **30***a* can share the URL associated with the SYN-packet based flood attack with the web browser **30***c* also located on the personal computing device **30**. The browser **30***c* can use a hyper text transfer protocol ("HTTP") type transfer using the URL to visualize the statistics related to the SYN-packet based flood attack, and to generate ACL and CAR entries for remediation of the SYN-packet based flood attack When the web server **24***c* receives the URL from the browser **30***c*, the web server **24***c* invokes server-side access scripts **24***d*, which generates queries to the database **24***b* for generating a dynamic HTML web page. The network administrator and/or security operators can view the SYN-packet based flood attack anomalies on the web page, which is displayed on the display **30***b* of the computing device **30**.

[0087] Although not shown, in an embodiment, the system **5** for detecting, tracking and blocking denial of service attacks can be located on a removable storage medium. The removable storage medium can be transported and selectively loaded onto the routing systems **22**, **22***b* and/or **22***c*. Alternatively, the system **5** for detecting, tracking and blocking denial of service attacks can be partially located on the routing systems **22**, **22***b* and/or **22***c* and partially located on other servers (not shown). For example, the collector **20** can be located on routing system **22** and the collector **20***b* can be located on routing system **22***c*. Further, zone controller **24** can be co-located with either the collector **20**, the collector **20***b*, or , zone controller **24** can be located on another server (not shown).

[0088] Having thus described at least one illustrative embodiment of the invention, various alterations, modifications and improvements will readily occur to those skilled in the art. Such alterations, modifications and improvements are intended to be within the scope and spirit of the invention. Accordingly, the foregoing description is by way of example only and is not intended as limiting. The invention's limit is defined only in the following claims and the equivalents thereto.

What is claimed is:

1. A system for detecting, tracking and blocking one or more denial of service attacks over a computer network, the system comprising:

a collector adapted to receive a plurality of data statistics from the computer network and to process the plurality of data statistics to detect one or more data packet flow anomalies and to generate a signal representing the one or more data packet flow anomalies; and

a controller coupled to the collector to receive the signal;

wherein the controller is constructed and arranged to respond to the signal by tracking attributes related to the one or more data packet flow anomalies to at least one source, and wherein the controller is constructed and arranged to block the one or more data packet flow anomalies.

2. The system of claim 1, wherein the collector includes a buffer coupled to the computer network and being adapted to process the plurality, of data statistics to generate at least one record.

3. The system of claim 2, wherein the collector further includes a profiler coupled to the buffer and being adapted to receive and process the record to generate a predetermined threshold.

4. The system of claim 3, wherein the profiler includes means for aggregating the data statistics to obtain a traffic profile of network flows.

5. The system of claim 4, wherein the data statistics are aggregated based on at least one invariant feature of the network flows.

6. The system of claim 4, wherein data statistics are aggregated based on temporal, static network and dynamic routing parameters.

7. The system of claim 5, wherein the at least one invariant feature includes source and destination endpoints.

8. The system of claim 3, wherein the collector further includes a detector coupled to the buffer and to the profiler, the collector being adapted to receive and process the record and the predetermined threshold to detect if attributes associated with the record exceed the predetermined threshold representing the one or more data packet flow anomalies.

9. The system of claim 8, wherein the collector further includes a local controller coupled to the detector and to the profiler and being adapted to receive and respond to the one or more data packet flow anomalies by generating the signal representing the one or more data packet flow anomalies.

10. The system of claim 9, wherein the detector includes a database for storing the at least one record, predetermined threshold, the one or more data packet flow anomalies, and related information.

11. The system of claim 10, wherein the profiler includes a database for storing a plurality of data packet flow profiles and related information.

12. The system of claim 1, wherein the controller includes a filtering mechanism for blocking the one or more data packet flow anomalies.

13. The system of claim 12, wherein the filtering mechanism includes a plurality of filter list entries.

14. The system of claim 12, wherein the filtering mechanism includes a plurality of rate limiting entries.

15. The system of claim 1, wherein the controller includes a correlator coupled to the collector and being adapted to receive and normalize the plurality of signals representing the one or more data packet flow anomalies and to generate an anomaly table including the attributes related to the one or more data packet flow anomalies.

16. The system of claim 15, wherein the correlator includes a database for storing the anomaly table.

17. The system of claim 16, wherein the correlator further includes an adapter that is constructed and arranged to communicate the anomaly table to a computing device for further processing.

18. The system of claim 16, wherein the controller further includes:

a web server; and

access scripts that cooperate with the web server to enable the computing device to access the database defined on the controller to view the anomaly table.

19. A system comprising:

at least one routing system;

a plurality of computer systems coupled to the routing system; and

means for detecting one or more denial of service attacks communicated to the plurality of computer systems over the at least one routing system.

20. The system of claim 19, further including a means for tracking the one or more denial of service attacks communicated to the plurality of computer systems over the at least one routing system.

21. The system of claim 20, further including a means for blocking the one or more denial of service attacks communicated to the plurality of computer systems over the at least one routing system.

22. The system of claim 21, wherein the means for detecting includes a means for collecting a plurality of data statistics from the at least one routing system.

23. The system of claim 22, wherein the means for detecting further includes a means for processing the plurality of data statistics to detect one or more data packet flow anomalies.

24. The system of claim 23, wherein the means for detecting further includes a means of generating a plurality of signals representing the one or more data packet flow anomalies.

25. The system of claim 24, wherein the means for tracking includes a means for receiving and responding to the plurality of signals by tracking attributes related to the one or more data packet flow anomalies to at least one source.

26. The system of claim 19, further including a means for communicating the one or more denial of service attacks to a computing device for further processing.

27. A method for detecting, tracking and blocking one or more denial of service attacks over a computer network, the system comprising the steps of:

collecting a plurality of data statistics from the computer network;

processing the plurality of data statistics to detect one or more data packet flow anomalies;

generating a plurality of signals representing the one or more data packet flow anomalies; and

receiving and responding to the plurality of signals by tracking attributes related to the one or more data packet flow anomalies to at least one source.

28. The method of claim 27, further including the step of blocking the one or more data packet flow anomalies in close proximity to the at least one source.

29. The method of claim 28, wherein the step of collecting the plurality of data statistics includes:

buffering the plurality of data statistics;

processing the plurality of data statistics to generate at least one record; and

receiving and profiling the at least one record to generate a predetermined threshold.

30. The method of claim 29, wherein the step of collecting the plurality of data statistics further includes;

detecting if attributes related to the at least one record exceed the predetermined threshold representing the one or more data packet flow anomalies.

31. The method of claim 30, wherein the step of collecting the plurality of data statistics further includes:

responding locally to the one or more data packet flow anomalies by generating the plurality of signals representing the one or more data packet flow anomalies.

32. The method of claim 27, wherein the step of receiving and responding to the plurality of signals includes:

correlating the plurality of signals representing the one or more data packet flow anomalies; and

generating an anomaly table including the attributes related to the one or more data packet flow anomalies.

33. The method of claim 32, wherein the step of receiving and responding to the plurality of signals further includes the step of communicating the anomaly table to a computing device for further processing.

* * * * *

Exhibit 1011

U.S. Patent 7,062,782 by Robert J. Stone et al. ("Stone")

Customer No.: 000027683

Haynes and Boone, LLP
IP Section
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Telephone [214] 651.5000
Fax [214] 200.0853

US007062782B1

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

(10) **Patent No.: US 7,062,782 B1**
(45) **Date of Patent: Jun. 13, 2006**

(54) **OVERLAY NETWORK FOR TRACKING DENIAL-OF-SERVICE FLOODS IN UNRELIABLE DATAGRAM DELIVERY NETWORKS**

(75) Inventors: **Robert J. Stone**, Centerville, VA (US); **Matthew J. Sibley**, Columbia, MD (US)

(73) Assignee: **UUNet Technologies, Inc.**, Ashburn, VA (US)

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

(21) Appl. No.: **09/469,505**

(22) Filed: **Dec. 22, 1999**
(Under 37 CFR 1.47)

(51) **Int. Cl.**
G06F 11/00 (2006.01)
G06F 11/22 (2006.01)
G06F 12/14 (2006.01)

(52) **U.S. Cl.** **726/22**; 713/151; 713/153

(58) **Field of Classification Search** 713/201, 713/151, 153; 709/224, 200; 370/216; 726/22
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,862,362 A * 1/1999 Somasegar et al. 703/21
5,991,881 A * 11/1999 Conklin et al. 713/201
5,996,021 A 11/1999 Civanlar et al. 709/238
6,014,628 A * 1/2000 Kovarik, Jr. 705/1
6,157,649 A * 12/2000 Peirce et al. 370/401
6,182,226 B1 * 1/2001 Reid et al. 713/201
6,185,689 B1 * 2/2001 Todd et al. 713/201
6,282,546 B1 * 8/2001 Gleichauf et al. 707/102
6,301,668 B1 * 10/2001 Gleichauf et al. 713/201
6,321,338 B1 * 11/2001 Porras et al. 713/201
6,327,242 B1 * 12/2001 Amicangioli et al. 370/216
6,363,489 B1 * 3/2002 Comay et al. 713/201
6,442,694 B1 * 8/2002 Bergman et al. 713/201
6,484,203 B1 * 11/2002 Porras et al. 709/224

(Continued)

OTHER PUBLICATIONS

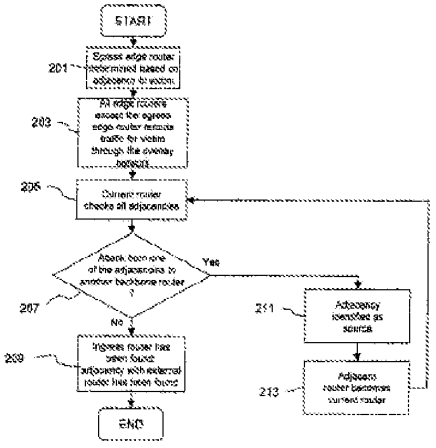
Robert Stone. CenterTrack: An IP Overlay for Tracking DoS Floods. Oct. 1, 1999. p. 1-9.*
Using the Border Gateway Protocol for Interdomain Routing, http://www.cisco.com, accessed Nov. 5, 1999, 66 pages, Chapter 12.
Robert Stone, Center Track: An IP Overlay Network for Tracking DoS Floods, Oct. 1, 1999, 9 pages, UUNET Technologies, Inc.
Robert Stone, Center Track: An IP Overlay Network for Tracking Denial—of—Service Floods, Power Point Presentation, Oct. 5, 1999, 19 pages, UUNET Technologies, Inc.

Primary Examiner—Ayaz Sheikh
Assistant Examiner—Christian LaForgia

(57) **ABSTRACT**

An approach for tracking denial-of-service (DoS) flood attacks using an overlay IP (Internet Protocol) network is disclosed. One or more tracking routers form an overlay tracking network over the network of an Internet Service Provider (ISP). The ISP network includes numerous transit routers and edge routers. The tracking routers communicate directly with all the edge routers using IP tunnels. The edge routers within the ISP network perform security diagnostic functions, in part, to identify a DoS flood attack that has been launched by one or more attackers. To track down an attacker, an egress edge router identifies the DoS flood attack datagrams, rerouting these datagrams to the overlay tracking network. The tracking routers perform hop-by-hop input debugging to identify the ingress edge router associated with the source of the DoS flood attack.

29 Claims, 9 Drawing Sheets



U.S. PATENT DOCUMENTS

6,499,107 B1 * 12/2002 Gleichauf et al. 713/201
6,535,227 B1 * 3/2003 Fox et al. 345/736
6,578,147 B1 * 6/2003 Shanklin et al 713/200
6,609,205 B1 * 8/2003 Bernhard et al. 713/201
6,611,872 B1 * 8/2003 McCanne 709/238
6,654,882 B1 * 11/2003 Froutan et al. 713/153
6,687,732 B1 * 2/2004 Bector et al. 709/200
6,687,833 B1 * 2/2004 Osborne et al. 713/201
2002/0032871 A1 * 3/2002 Malan et al. 713/201
2003/0135468 A1 * 7/2003 Barbir et al. 705/64

* cited by examiner

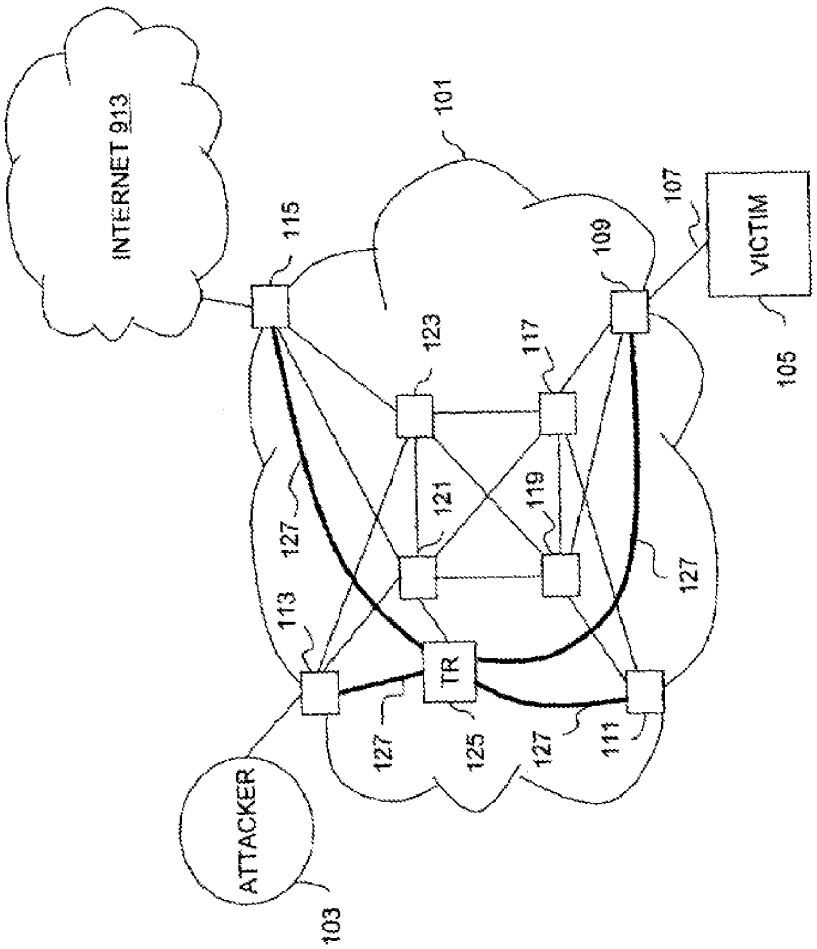

INTERNET 913
101
115
107
VICTIM
109
105
123
117
121
119
127
113
TR
125
111
ATTACKER
103

FIG. 1

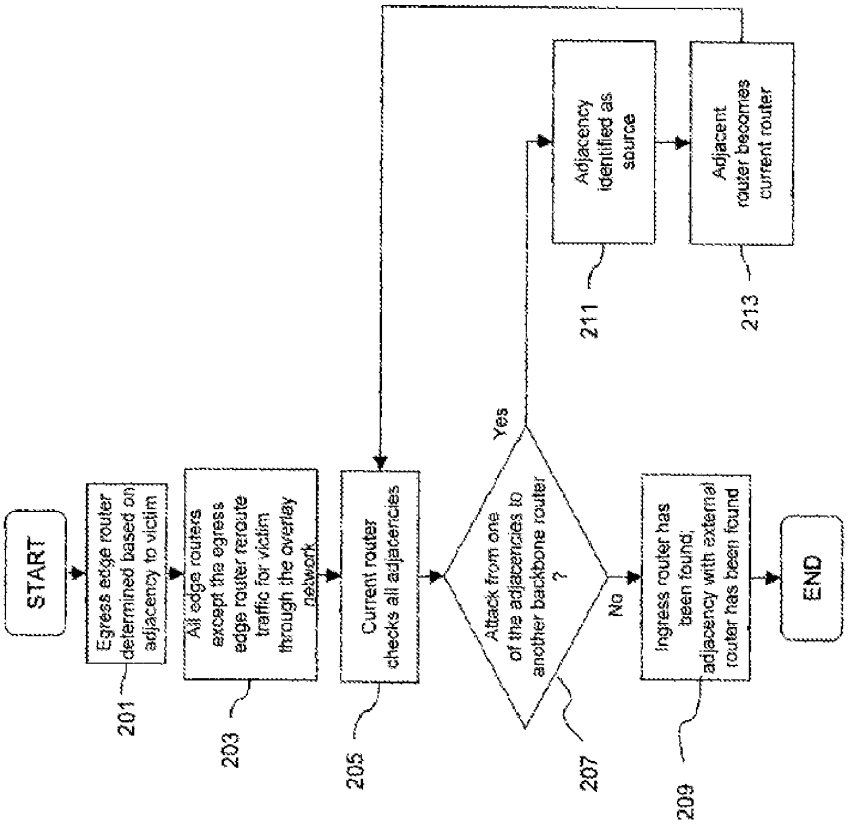

START
Egress edge router determined based on adjacency to victim
201
All edge routers except the egress edge router reroute traffic for victim through the overlay network
203
Current router checks all adjacencies
205
Attack from one of the adjacencies to another backbone router ?
207
Yes
No
Adjacency identified as source
211
Adjacent router becomes current router
213
Ingress router has been found; adjacency with external router has been found
209
END

FIG. 2

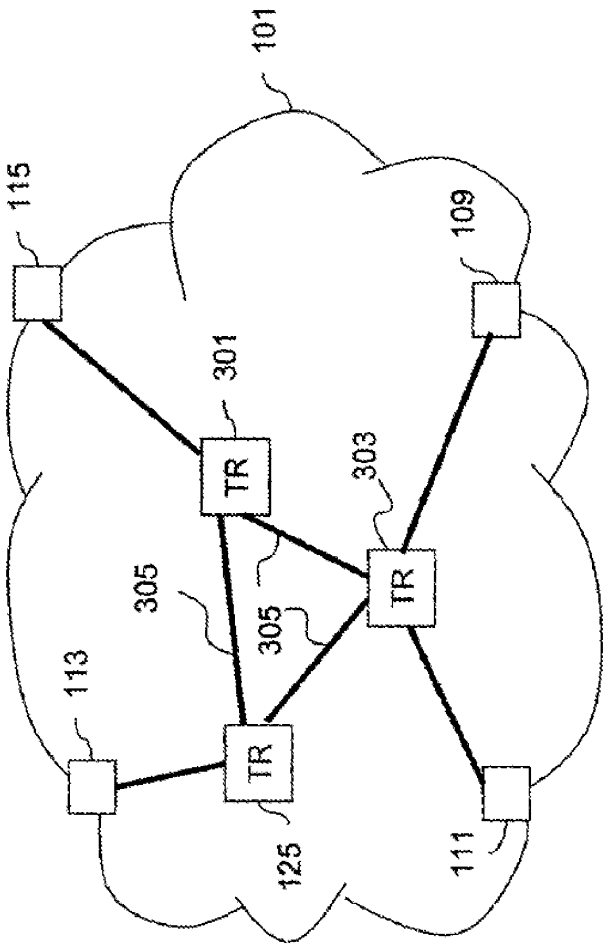


FIG. 3

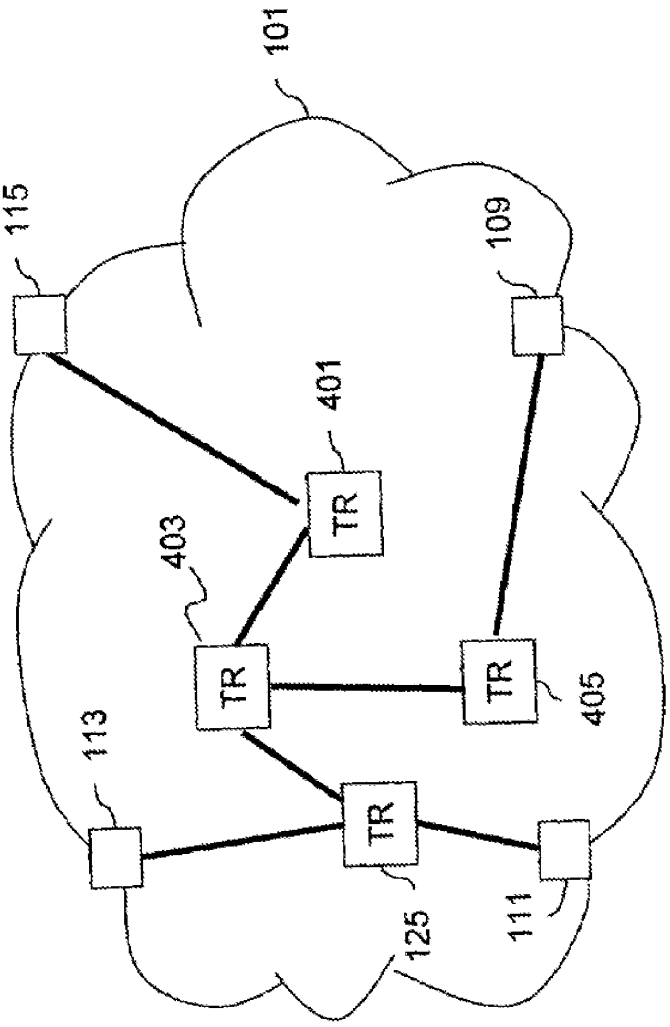


FIG. 4

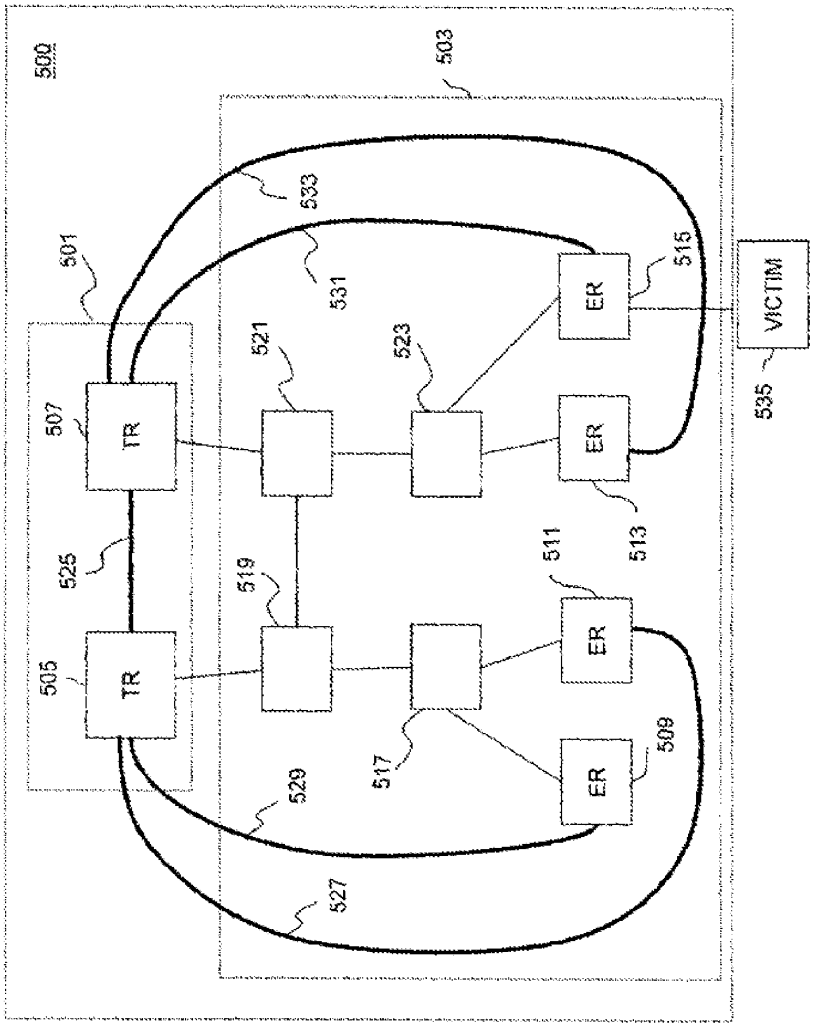

500
503
501
533
531
515
VICTIM
535
521
523
507
TR
ER
525
519
511
513
505
TR
ER
517
509
ER
529
527

FIG. 5

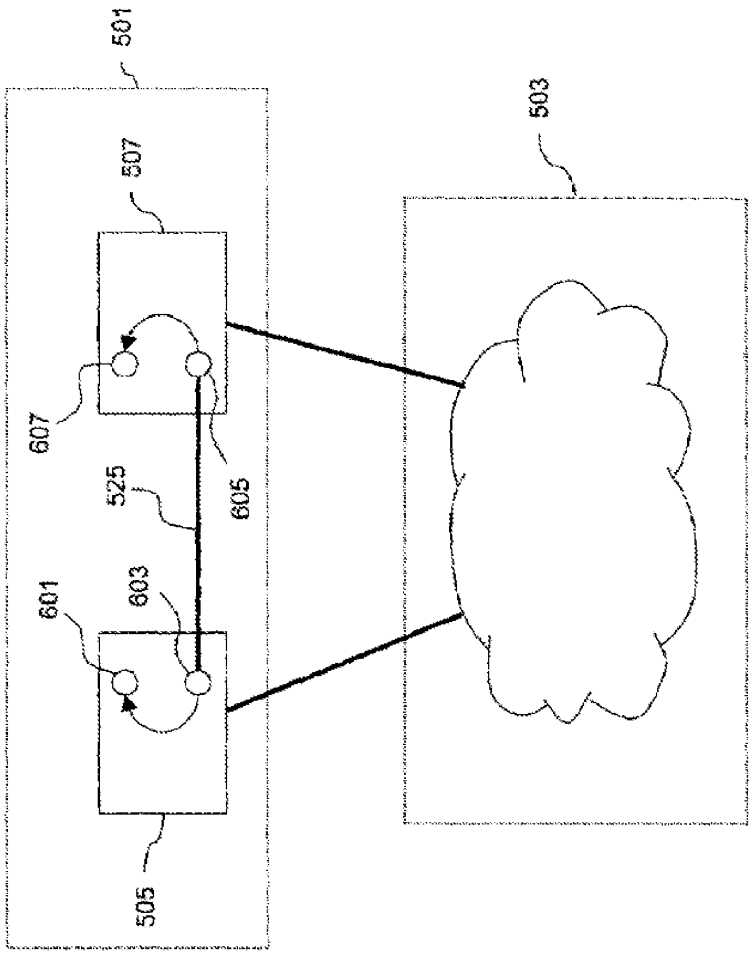


FIG. 6

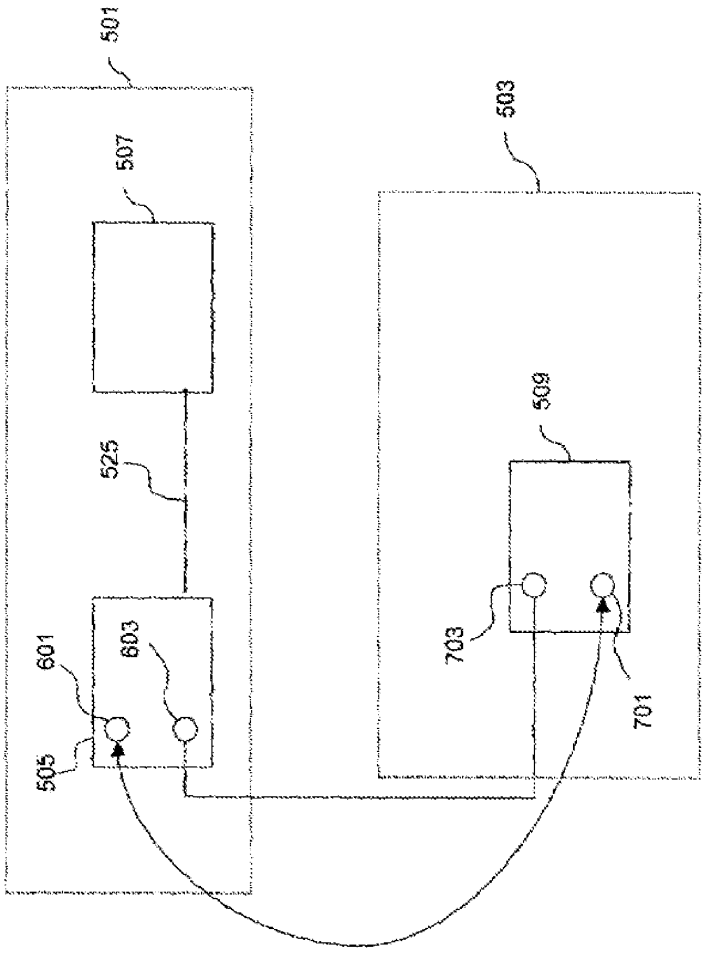


FIG. 7

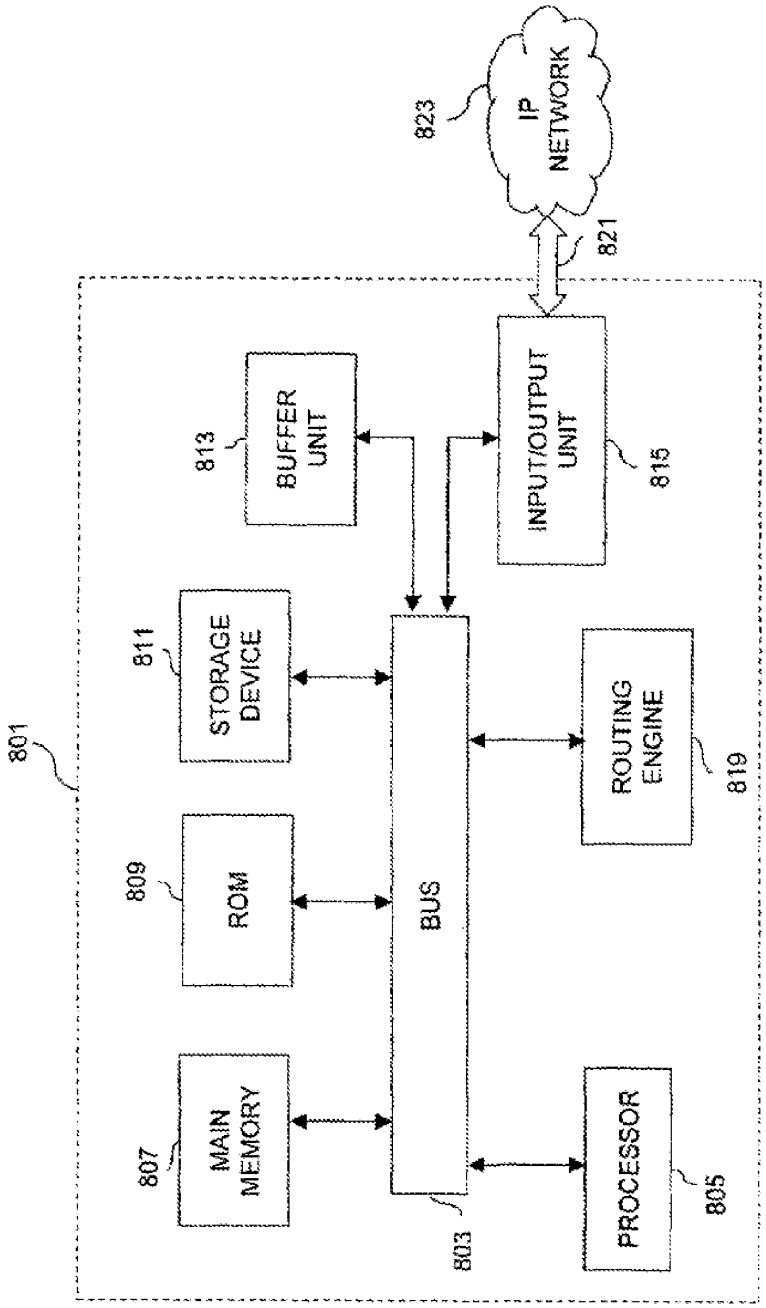

823
IP NETWORK
821
801
813
BUFFER UNIT
INPUT/OUTPUT UNIT
815
811
STORAGE DEVICE
809
ROM
BUS
ROUTING ENGINE
819
807
MAIN MEMORY
803
PROCESSOR
805

FIG. 8

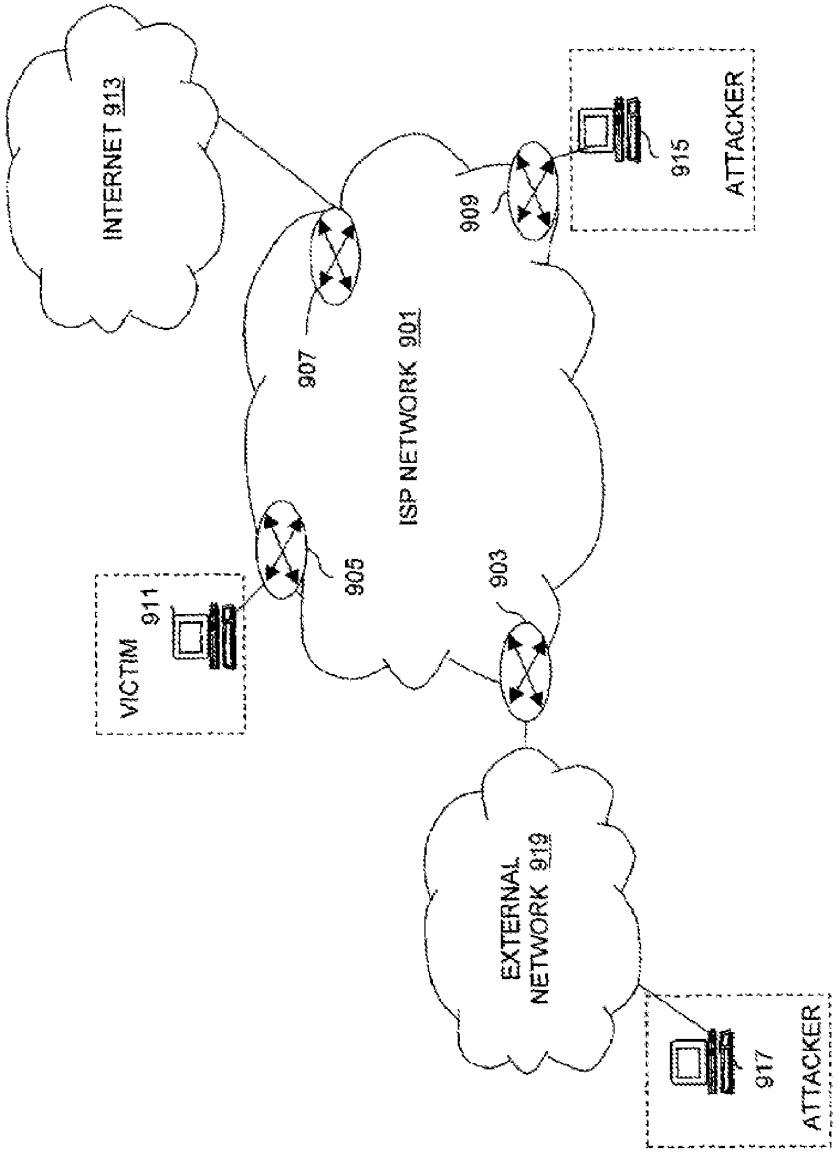


FIG. 9 PRIOR ART

OVERLAY NETWORK FOR TRACKING DENIAL-OF-SERVICE FLOODS IN UNRELIABLE DATAGRAM DELIVERY NETWORKS

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to data communications, and more specifically relates to network security.

2. Discussion of the Background

The phenomenal growth of the Internet has presented Internet Service Providers (ISPs) with the continual challenge of responding to the millions of users' demand for reliable, fast and dependable access to this global resource. Satisfying these demands is imperative to maintaining a competitive edge in an intensely competitive market. To further intensify this challenge, ISPs and their customers frequently are victims of various types of packet flood attacks that negatively impacts service availability.

Packet flood attacks are a type of denial-of-service (DoS) attack. A DoS attack is initiated by an attacker to deliberately interfere or disrupt a subscriber's datagram delivery service. A packet flood attack differs from other types of denial-of-service attacks in that a flood attack requires constant and rapid transmission of packets to the victim in order to be effective. The flood attack overwhelms the victim's connection and consumes precious bandwidth on the ISPs' backbone networks. Examples of packet flood attacks specific to Unreliable Datagram Delivery Service Networks utilizing IP (Internet Protocol) include ICMP (Internet Control Message Protocol) flood, "SMURF" (or Directed Broadcast Amplified ICMP Flood), "Fraggle" (or Directed Broadcast UDP (User Datagram Protocol) Echo Flood), and TCP (Transmission Control Protocol) SYN flood. These attacks effectively prevent the subscribers from accessing the Internet; in some circumstances, the effects of these attacks may cause a victim host to freeze, thereby requiring a system reboot. In addition to being a nuisance, a system freeze can result in lost of data if precautions were not taken in advance. Because of the severe and direct impact it has on its subscribers, an ISP needs an effective mechanism to prevent or minimize these DoS attacks.

Like many other types of DoS attacks, the attacker can forge the source address of the flood packets without reducing the effectiveness of the attack. Finding the source of forged datagrams in a large, high-speed, unreliable datagram delivery service network is difficult when source-based forwarding decisions are not employed and sufficient capability in most high-speed, high-capacity router implementations is not available. Typically in this case, not enough of the routers in such a network are capable of performing the packet forwarding diagnostics that are required to determine the source. Because the source addresses of the attack packets are almost always forged, it is non-trivial to determine the true origin of such attacks. As a result, tracking down the source of a flood-type denial-of-service attack is usually difficult or impossible in networks that meet these criteria.

FIG. **9** shows a conventional high-speed network of an Internet Service Provider. An ISP network **901** includes a number of routers, of which edge routers **903**, **905**, **907**, and **909** are shown. To access the Internet **913**, user station **911** initiates a communications session with the ISP network **901**. To transmit packets to the Internet **913**, user station **911** generates the datagrams, which enter the ISP network **901** through edge router **905**. These packets are then forwarded through one or more transit routers (not shown) within the ISP network **901**, ultimately reaching edge router **907**, which in turn forwards the packets to the Internet **913**.

Assume now that user station **915** wants to prevent user station **911** from accessing the Internet **913**. the attacking user station **915**, for example, can launch a DoS flood attack such as a SMURF attack. In the role of attacker, user station (or host) **915** transmits a large amount of ICMP echo (PING) traffic using the directed broadcast addresses of previously discovered amplification subnets as the destination, in which the attacker **915** has a spoofed source address of the user station **911**, which in this case is the victim. Accordingly, all of the hosts connected to the amplification subnets (which could be connected to the ISP network **901** or the Internet **913**) will reply to the ICMP echo requests. A large public network such as the Internet serves an enormous number of hosts and it is trivial to find sufficient amplification subnets capable of overwhelming most victims. It is particularly trivial to overwhelm the connection of a typical subscriber, who has a 56 kbps modem. The only recourse that the subscriber on user station **911** has is to notify the ISP of the service disruption.

As seen in FIG. **9**, an attacker **917** may reside within another network. User station **917** can initiate a flood attack through an external network **919**, which is connected to ISP network **901** via edge router **903**. External network **919** may belong to another ISP, which has a connection to the Internet **913** (not shown). Under this circumstance, ISP of network **901** after isolating the source of the attack to the external network **919** simply notifies the administrator of external network **919** that one of its subscribers is initiating a packet flood attack.

In recognition of this problem, ISPs have developed various solutions to eliminate or mitigate the damaging effects of DoS flood attacks. One approach is to equip every router within the ISP network **901**, which includes transit routers (not shown) and edge routers **903**, **905**, **907** and **909**, with an input debugging feature to trace the source of the flood attacks to a particular edge router that is associated with the DoS attack. To perform this tracing, personnel within the ISP manually performs hop-by-hop tracking to locate the edge router that serves as an ingress to the flood attack (which in this case is edge router **909**). One drawback with this approach is the fact that every router, both transit and edge routers, is required to have the input debugging feature, which entails considerable expense. Further, many transit routers do not support this feature, as their main purpose is to forward packets at an extremely high rate; consequently, new hardware and software may need to be developed, thereby potentially delaying implementation of the appropriate security measures. Hardware and software development, in turn, may require adopting proprietary standards, and thus, may increase the complexity of the network because of interoperability issues, for example. Another drawback stems from the fact that the hop-by-hop tracking is performed manually, potentially locating the attacker after significant damage has already occurred. In other words, the flood attack may be over by the time the ISP is able to address the attack. Yet another drawback is that legitimate traffic could be impeded because input debugging is performed on the core network.

Based on the foregoing, there is a clear need for improved approaches for tracking DoS flood attacks.

There is also a need to limit the number and complexity of features that are required on routers.

There is also a need to minimize operational risks to the network.

There is also a need to expedite the deployment of security measures against DoS attacks by using existing hardware and software.

There is a further need to identify the source of the DoS floods without interrupting the flow of legitimate traffic.

Based on a need to deploy countermeasures against DoS attacks, an approach for tracking down the ingress edge router associated with the DoS attack using existing hardware and software infrastructure is highly desirable.

SUMMARY OF THE INVENTION

According to one aspect of the invention, a method is provided for tracking denial-of-service (DoS) floods. The method comprises rerouting a flood attack to a tracking router. The tracking router forms an overlay tracking network with respect to an egress edge router. The method also includes identifying an ingress edge router that forwarded the DoS flood attack datagram. Under this approach, the cost and deployment time for introducing a network security mechanism to counteract denial-of-service flood attacks is minimized.

According to another aspect of the invention, a communication system for tracking denial-of-service floods comprises a plurality of edge routers, which include an ingress edge router and an egress edge router. Each of the edge routers is configured to perform security diagnostic functions, in part, to identify a DoS flood attack datagram. The ingress edge router is associated with a source of the DoS flood attack datagram. A tracking router, which is adjacent to the egress edge router, performs security diagnostic functions. The ingress edge router reroutes the DoS flood attack datagram to the tracking router as to permit identification of the ingress edge router. The tracking router forms an overlay tracking network with respect to the plurality of edge routers. The above arrangement advantageously minimizes network complexity by utilizing standardized protocols and equipment.

In yet another aspect of the invention, a computer-readable medium carrying one or more sequences of one or more instructions for tracking denial-of-service floods. The one or more sequences of one or more instructions include instructions which, when executed by one or more processors, cause the one or more processors to perform the steps of receiving a datagram; identifying the DoS flood attack datagram; and identifying a previous hop router associated with the DoS flood attack datagram to ultimately locate an ingress router and an ingress adjacency associated with the DoS flood attack. This approach advantageously enables the identification of the ingress edge router associated with the DoS flood attack without interrupting the flow of legitimate traffic.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete appreciation of the invention and many of the attendant advantages thereof will be readily obtained as the same becomes better understood by reference to the following detailed description when considered in connection with the accompanying drawings, wherein:

FIG. **1** is a diagram of an overlay tracking network in accordance with an embodiment of the present invention;

FIG. **2** is a flow diagram of the hop-by-hop input debugging operation as performed by the system of FIG. **1**;

FIG. **3** is a diagram of the tracking routers in a fully meshed configuration, according to one embodiment of the present invention;

FIG. **4** is a diagram of the tracking routers in a two-level hierarchical arrangement, according to one embodiment of the present invention;

FIG. **5** is a diagram of an overlay tracking network utilizing two tracking routers, in accordance with one embodiment of the present invention;

FIG. **6** is a diagram of the tracking routers communicating using IP (Internet Protocol) tunneling according to the system of FIG. **5**;

FIG. **7** is a diagram of the IP tunnel between a tracking router and an edge router according to the system of FIG. **5**;

FIG. **8** is a block diagram of an exemplary architecture of a general router utilized in the system of FIG. **5**; and

FIG. **9** is a diagram of a conventional approach to addressing denial-of-service (DoS) flood attacks.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the following description, for the purposes of explanation, specific details are set forth in order to provide a thorough understanding of the invention. However, it will be apparent that the invention may be practiced without these specific details. In some instances, well-known structures and devices are depicted in block diagram form in order to avoid unnecessarily obscuring the invention.

The present invention provides a valuable solution for the tracking of DoS flood attacks using existing hardware and software infrastructure. An overlay network, which consists of IP (Internet Protocol) tunnels, is created to selectively reroute flood attack datagrams directly from edge routers to tracking routers. It is recognized that the tunnels may be created, in general, over any unreliable datagram delivery service protocol. The edge routers and the tracking routers are equipped with security diagnostic features to perform, according to one embodiment, input debugging. The tracking routers determine the ingress edge router by observing which tunnel the datagrams arrive on. These datagrams may then be examined and dropped or forwarded to the appropriate egress point. This approach simplifies the process of determining the ingress adjacency of a flood attack while bypassing any equipment, which may be incapable of performing the necessary diagnostic functions.

Although the present invention is discussed with respect to an Internet Protocol (IP) network, it should be appreciated that one of ordinary skill in the art would recognize that the present invention has applicability to other communication protocols.

FIG. **1** shows a diagram of an overlay tracking network, according to an embodiment of the present invention. The network **101** of an Internet Service Provider (ISP) includes many different types of routers, depending on the roles these routers play in forwarding datagrams or packets within the network **101**. The term "datagram" is used herein interchangeably with "packet" to signify a unit of information that contains delivery information (e.g., destination address) to permit forwarding through a packet switched network. As shown, ISP network **101** serves subscribers **103** and **105**. Subscriber **103**, in this example, assumes the role of an attacker, which introduces packet flood attacks that are destined for subscriber **105** to ISP network **101**. As a recipient of the DoS flood attack datagrams, subscriber **105** assumes the role of victim. Both the attacker **103** and the victim **105** can be individual hosts, end systems, or other networks (such as another ISP). It should be noted that multiple attackers can transmit flood attacks to a single victim or multiple victims. For explanatory purposes, a

single attacker **103** and a single victim **105** are shown. In this instance, attacker **103** desires to deny service to victim **105** by launching a series of packet flood attacks through a system of routers within network **101** to victim **105**, thereby saturating the connection **107** of victim **105**.

Within the system of FIG. **1**, two basic classifications of routers exist: backbone routers, which are routers within the ISP network **101**; and external routers, which are routers which are not a part of the ISP network **101**. That is, external routers (not shown) belong to a subscriber or another ISP. Backbone routers **109**–**125** are further classified according to their adjacency. Edge routers **109**, **111**, **113**, and **115** are backbone routers that are adjacent to one or more external routers. In addition, transit routers **117**, **119**, **121**, and **123** are backbone routers that are adjacent only to other backbone routers. According to the present invention, another type of backbone router is defined as a tracking router **125**. A tracking router **125** is logically adjacent to edge routers and other tracking routers (not shown).

At this point, the terminology with respect to adjacency is introduced to better understand the present invention. An adjacency between a tracking router **125** and an edge router **109**, **111**, **113** and **115**, or a tracking router **125** and another tracking router (not shown), is called a tracking adjacency. In the example of FIG. **1**, an ISP network **101** is assumed to receive attacks outside the network **101** in which the attacks are targeted at a victim **105** that is outside the network **101**. Therefore, the malicious packets that comprise the flood attack are transmitted across the edge of the network at two points. The first point is termed ingress edge adjacency, i.e., the "source" of the attack; the second point is at an egress edge adjacency (the "destination" of the attack). Also, the term edge adjacency refers to an adjacency between an edge router and an external router. The ingress edge router is therefore the edge router that has the ingress edge adjacency, which in this example is edge router **113**. The egress edge router is the edge router that has the egress edge adjacency; which, in this scenario, is edge router **109**.

Subscriber **103**, as attacker, launches DoS flood attacks using bogus source addresses so that victim **105** cannot detect the originator of the DoS flood attack datagrams simply by inspecting the received datagrams. The victim **105** experiences an inability to receive legitimate datagrams from, and possibly transmit through the connection **107**. Further, as indicated previously, victim **105** may be required to reboot, assuming the victim **105** is a host computer. Victim **105** communicates its service disruption to the ISP, which must restore service by locating the attacker **103** and redirecting the bogus traffic away from the victim **105**.

The ISP network **101** identifies an attack signature of the attack launched by **103**. Attack signature refers to some pattern that can be used to help distinguish malicious packets from normal traffic. At the very least, an attack signature is defined by the IP address or address range of the victim **105**, which is being attacked.

In the system of FIG. **1**, tracking router **125** forms an overlay network with edge routers **109**, **111**, **113**, and **115** to track down the ingress edge router that is responsible for forwarding the packet flood attacks into ISP network **101**. According to one embodiment of the present invention, edge routers **109**, **111**, **113** and **115** as well as the tracking router **125** perform input debugging. Input debugging refers to the diagnostic features that determine the adjacency that originated a DoS flood attack datagram matching an attack signature on an individual router. Essentially, input debugging reveals the previous hop along the data path of the DoS flood attack datagrams. A tracking hop is defined as one usage of input debugging on a particular router. The number of tracking hops is expressed in terms of the maximum hop diameter of the backbone network **101** (d) and the hop diameter of the overlay tracking network (d_o). The use of input debugging in an overlay tracking network provides an ability for an ISP to expediently determine the ingress edge router.

The links **127** between tracking router **125** and the edge routers **109**, **111**, **113** and **115** are logical IP tunnels, according to an embodiment of the present invention. Alternatively, the links **127** may be actual physical connections or virtual connections (e.g., ATM (Asynchronous Transfer Mode) or frame relay); the virtual connections may be over any lower layer (n−1) protocol. IP tunnels **127** (shown as bold lines) are created over an existing IP network **101**. Because IP is supported by all edge routers **109**, **111**, **113**, and **115**, the IP tunnels are isolated from any underlying infrastructure change of network **101**. This advantageously reduces complexity of the overlay tracking network, as well as enhance speed of deployment.

Upon creation of the overlay tracking network, a method known as hop-by-hop tracking is employed to determine the ingress edge router that originated the DoS flood attack datagram. Hop-by-hop tracking involves recursively using the input debugging feature within edge routers **109**, **111**, **113**, and **115** as well as tracking router **125** to identify the ingress edge adjacency. This approach is further detailed in FIG. **2**, below.

FIG. **2** shows a flow diagram of the hop-by-hop input debugging operation employed by the overlay tracking network of FIG. **1**. Because each edge router **109**, **111**, **113**, and **115** is equipped with security diagnostic functions, these routers **109**, **111**, **113**, and **115** can identify DoS flood attack datagrams based upon the attack signature. Turning back to the example in FIG. **1**, egress edge router **109** is determined based on detection of a DoS attack (which may be performed by the victim or tools employed by the network provider) egressing through that router, per step **201**. All edge routers, with the exception of the egress edge router, reroute all packets destined for the victim through the overlay network to the egress edge router due to a dynamic routing update. The egress edge router specifically does not receive this update, ignores the update, or otherwise does not honor the update. The result of this is that all packets received by edge routers **111**, **113**, **115** destined for the victim **105** are routed through the tracking router **125** to the egress edge router **109**.

Thereafter, hop-by-hop tracking is then used, starting with the tracking router closest to the victim **105**, which will always be logically adjacent to the egress edge router **109**. In the system of FIG. **1**, because there is only one tracking router, two iterations of hop-by-hop tracking are performed. In step **205**, the current tracking router, which is tracking router **125** in this example, checks all the tunnels **127**. Next, tracking router **125** determines whether a DoS flood attack datagram exists on any of the **127**, as in step **207**. If no attack is identified from any of the tunnels **127**, then the attack is originated from the egress edge router **109**. In this rare case, the ingress edge router is the same as the egress edge router. Input debugging is then performed, as in step **209**, on the edge router **109** to find the source of the DoS flood attack datagrams; this would be another external adjacency (not shown). However, if step **207** determines that the DoS flood attack datagram is originating from one of the tunnels, the corresponding adjacency is identified as the source, per step **211**. In step **213**, the adjacent edge router associated with the identified adjacency becomes the current router for the

purposes of the hop-by-hop tracking scheme. Accordingly, step **205** and subsequent steps are repeated as many times as necessary until the ingress edge router **113** is located.

The above method of FIG. **2** provides an example involving one attacker **103**; however, one of ordinary skill in the art would recognize that an attack from multiple sources can be tracked using the above method. Where multiple source adjacencies are identified, each adjacency is investigated separately (possibly in parallel) until multiple edge adjacencies are identified. Such a capability is useful since multiple source attacks are common. That is, attackers often collaborate to disrupt the service of a single victim especially when each individual attacker lacks the resources to launch an effective attack.

The embodiment of FIG. **1** involves a single tracking router **125** participating in the overlay tracking network. The overlay network can scale if the number of edge routers is larger than the number of tracking adjacencies than a single router can service. In other words, if a single tracking router cannot support sufficient number of tracking adjacencies (e.g. IP tunnels, Border Gateway Protocol (BGP) sessions, and other resources necessary to maintain a particular number of adjacencies), then multiple routers are necessary. Although BGP is discussed herein, one of ordinary skill in the art would recognize that any inter-administrative-domain routing/signaling protocol may be used. FIG. **3** shows a diagram of an overlay tracking network with three tracking routers in a fully meshed configuration. In FIG. **3**, transit routers **117**, **119**, **121**, and **123** (of FIG. **1**) are not shown to simplify the diagram. The overlay tracking network includes two additional tracking routers **301** and **303**, which possess the same functionalities as tracking router **125**. The single level full mesh configuration of tracking routers **125**, **301** and **303** are interconnected via IP tunnels (shown as bold lines) **305** as are the links to the edge routers **109**, **111**, **113**, and **115**.

For a single level network of tracking routers, assuming each tracking router can handle C tracking adjacencies, then the number of tracking routers (NC) is as follows:

$$N_C=[(C+1)-((C+1)^2-4N_E)^{1/2}]/2 \qquad \text{Eq. (1)}$$

N_E is the number of edge routers present in the network **101**. Thus, the number of tracking routers increases at the rate of $O(N_E^{1/2})$ until a useful maximum of $[(C-1)/2]$ is reached.

In the alternative to the full mesh overlay network of FIG. **3**, FIG. **4** shows an overlay tracking network with tracking routers arranged in a two-level hierarchy. Another tracking router **403** is introduced at a second level in the hierarchy. The overlay tracking network includes tracking router **125** in communication with tracking router **403**, which is at the higher hierarchical level. Tracking routers **401** and **405** are also connected to tracking router **403**. Tracking routers **125**, **401** and **405** are at the same hierarchical level and directly attach to edge routers **109**, **111**, **113** and **115**, as shown in FIG. **4**. Under this topology, approximately NC adjacencies between tracking routers and other tracking routers are substituted with N_C-1 adjacencies to a single second level tracking router **403**. This increases the diameter of the overlay tracking network by an additional hop, and frees up approximately $N_C^2-N_C$ adjacencies. This topology is desirable when the network **101** grows at a pace faster than the tracking routers' adjacency capacity and NC is reaching its limit. The tracking adjacencies capacity of each tracking router, C, can be calculated according to equation (2):

$$C=[N_C-1+N_E/N_C] \qquad \text{Eq. (2)}$$

FIG. **5** shows an overlay tracking network with two tracking routers, according to an embodiment of the present invention. An ISP network **500** includes two autonomous systems (AS) **501** and **503**, which essentially separate the tracking routers **505** and **507** from the rest of the backbone routers **509**–**523** in terms of dynamic routing policy. An autonomous system is a grouping of routers that operate under the same administrative domain. Basically, each domain has knowledge of the addresses that are assigned to the other domain, and of the manner in which the other domain accepts packets. Each administrative domain can enforce an elaborate access and attribute modification policy over routing requests that it receives from the other domain. However, each domain is not aware of how the packets are internally routed in the other domain.

As shown in FIG. **5**,AS **501** includes tracking routers **505** and **507**, while AS **503** includes edge routers **509**, **511**, **513**, and **515** as well as transit routers **517**, **519**, **521** and **523**, which follow a two-level hierarchy. Transit routers **519** and **521** represent the top level in the transit router hierarchy, and transit routers **517** and **523** are at the bottom level of the hierarchy. The tracking routers **505** and **507** are connected to the top level transit routers **519** and **521**, respectively. An IP tunnel **525** exists between the two tracking routers **505** and **507**. Tracking router **505** communicates over IP tunnel **527** with edge router **511** and communicates with edge router **509** via IP tunnel **529**. The remaining edge routers **513** and **515** are linked to tracking router **507** via IP tunnels **533** and **531**, respectively. In this manner, all edge routers **509**, **511**, **513** and **515** are reachable from each other through the overlay tracking network, which is created, in part, by tracking routers **505** and **507**. A victim **535** is linked to edge router **515**.

Once tunnels **525**, **527**, **529**, **531**, and **533** are built, an Interior Gateway Protocol (IGP), such as IS—IS (Intermediate System-to-Intermediate System), distributes link-state information about the tunnels **525**, **527**, **529**, **531**, and **533**. The IS—IS intradomain routing exchange protocol is an ISO (International Organization for Standardization) standard 10589. Because tracking routers **505** and **507** are on a different autonomous system than edge routers **509**, **511**, **513** and **515**, tracking routers **505** and **507** utilize BGP to announce routes to the edge routers **509**, **511**, **513** and **515**. BGP provides loop-free intradomain routing between AS **501** and AS **503**. BGP is defined in IETF (Internet Engineering Task Force) RFC 1771, which is incorporated by reference herein in its entirety. Within the same AS, the routers run internal BGP (IBGP), while routers of different autonomous systems run external BGP (EBGP). One of ordinary skill in the art would recognize that a number of other routing/signaling protocols (e.g., ISIS) may be utilized, other than EBGP. IBGP routers (i.e., routers within the same AS) use loop back interfaces to eliminate a dependency that arises when an IP address of a physical interface is used to configure BGP. That is, the loop back interface is used so that BGP does not have to rely on the availability of a particular interface for establishing TCP (Transmission Control Protocol) connections. BGP specifies TCP as the transport layer protocol for the transmission of routing information between peer routers. Such routing information includes AS numbers reflecting a path to the destination network. Based upon this routing information, BGP constructs a loop free map of the autonomous systems **501** and **503**.

In this example, a static route with a destination of the egress edge router **515** is added on the tracking router that is closest to the victim **535**, which in this case is tracking

router **507**. The static route is then redistributed into the overlay tracking networks BGP, which is advertised to all tracking routers as well as to all edge routers **509**, **511**, **513** and **515**. To prevent the routing loop, a static route with a destination of the appropriate next-hop external router for the victim **535** is added on the egress edge router **515**. The static route on the egress edge router **515** takes precedence over the route from the overlay tracking network, causing the edge router **515** to continue routing traffic to the victim **535**. With the static routes in place, a new path is created through the overlay tracking network hop-by-hop input debugging is then applied beginning with the tracking router that is closest to the victim **535** to find the attacker.

FIG. **6** illustrates the interconnection between two tracking routers using IP tunneling in accordance with an embodiment of the present invention. It is noted that if a tracking router learns that a tunnel's termination address is reachable through the tunnel **525** itself, then the tunnel **525** collapses and becomes useless. To prevent collapse of tunnel **525**, it is necessary to ensure that no tunnel termination address can be announced through the tunnel **525**. Thus, according to an exemplary embodiment, one approach is to number tunnel termination interfaces out of a distinct range of addresses, called the tunnel termination address space, using this range to filter the routing announcements. In addition, traffic that should be on the overlay tracking network are to be routed directly out of a tracking router's physical interface.

For this reason, tunnel interfaces are numbered out of a distinct address range (i.e., tunnel interface address space) in which the least specific prefix for this address block are routed to a "black hole" interface to prevent matching a default route. Once the address ranges are defined, tunnel interfaces never announce or accept prefixes from the tunnel termination address space. Further, the physical interfaces of the tracking routers **505** and **507** never announce or accept prefixes that are part of the tunnel interface address space. Accordingly, tunnels are prevented from collapsing by configuring the route announcement filtering in the manner described above.

As shown in FIG. **6**, tracking router **505** has an overlay loop back interface **601** and a primary loop back interface **603**. Similarly, tracking router **507** has an overlay loop back interface **607** and a primary loop back interface **605**. In both cases, the overlay loop back interfaces are used instead of the tunnel interface addresses. The overlay loop back address provides the next-hop identity on the overlay tracking network. The primary loop back addresses provide the next-hop identity on the backbone network within AS **503**. Tunnel **525** is established using the primary loop backs **603** and **605** as termination points. Reachability for the overlay loop backs is established over the tunnel **525**, which is not numbered. This reachability for the overlay loop backs is established by running IS—IS.

Once the tracking routers **505** and **507** of AS **501** become aware of each others overlay loop backs these routers **505** and **507** establish IBGP sessions with each other over tunnel **525** using the overlay loop backs. It should be noted that IS—IS does not announce any information about the primary loop backs **603** and **605**, thereby preventing collapse of tunnel **525**. Further, IBGP is prevented from announcing these primary loop backs **603** and **605** by filtering the address block that all primary loop backs **603** and **605** are numbered out of. In addition, IS—IS is configured to simply not know about the primary loop backs **603** and **605**.

FIG. **7** shows a tunnel connection between a tracking router and an edge router, according to an embodiment of the present invention. After the tracking routers **505** and **507** are fully connected with tunnels, each edge router (of which only edge router **509** is shown) is linked to a tracking router using IP tunnels. Similar to the interconnection between tracking routers **505** and **507**, primary loop backs are used as termination points for the tunnels. Edge router **509** has an overlay loop back **701** as well as a primary loop back **703**. For the edge routers, reachability for the overlay loop backs is established using static routes. Once the overlay loop backs are known, EBGP sessions are established between edge router **509** and tracking router **505** via overlay loop backs **601** and **701**. A static route also exists between tracking router **505** and edge router **509** via primary loop backs **603** and **703**. The overlay tracking network does not accept any routes that are announced from an edge router; however, edge routers accept any routes from the overlay tracking network, setting the local preference high to force the route through the overlay tracking network.

It should be noted that other methods of configuring BGP or other routing/signaling protocol attributes to achieve the desired results are possible. Also, it is possible to implement IP-over-IP tunnels without the use of an overlay loopback interface if each tunnel interface is explicitly numbered.

FIG. **8** illustrates a computer system **801** that possesses routing capabilities upon which an embodiment according to the present invention may be implemented. Such a computer system **801** may be configured as a node to perform identification of the DoS flood attack datagrams. Computer system **801** includes a bus **803** or other communication mechanism for communicating information, and a processor **805** coupled with bus **803** for processing the information. Computer system **801** also includes a main memory **807**, such as a random access memory (RAM) or other dynamic storage device, coupled to bus **803** for storing information and instructions to be executed by processor **805**. In addition, main memory **807** may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor **805**. Notably, routing tables may be stored in main memory **807**. Computer system **801** further includes a read only memory (ROM) **809** or other static storage device coupled to bus **803** for storing static information and instructions for processor **805**. A storage device **811**, such as a magnetic disk or optical disk, is provided and coupled to bus **803** for storing information and instructions.

Embodiments are related to the use of computer system **801** to selectively forward DoS flood attack datagrams. According to one embodiment, rerouting of the DoS flood attack datagrams is provided by computer system **801** in response to processor **805** executing one or more sequences of one or more instructions contained in main memory **807** in conjunction with the routing engine **819**. Such instructions may be read into main memory **807** from another computer-readable medium, such as storage device **811**. Execution of the sequences of instructions contained in main memory **807** causes processor **805** and routing engine **819** to perform the process steps described herein (as in FIG. **2**). One of the responsibilities of processor **805** is to determine whether the received datagrams match a predetermined attack signature. The received datagrams can be stored in buffer unit **813** for processing by processor **805**.

One or more processors in a multi-processing arrangement may also be employed to execute the sequences of instructions contained in main memory **807**. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions. Thus, embodiments are not limited to any specific combination of hardware circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor **805** and the routing engine **819** for transmitting and receiving datagrams to/from the IP network **823**. Such a medium may take many forms, **1**, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device **811**. Volatile media includes dynamic memory, such as main memory **807**. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus **803**. Transmission media can also take the form of acoustic or light waves, such as those generated during radio wave and infrared data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor **805** for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions relating to network parameters/characteristics into its dynamic memory and send the instructions over a packet-switched network or a telephone line using a modem. A modem local to computer system **801** can receive the data on the telephone line and use an infrared transmitter to convert the data to an infrared signal. An infrared detector coupled to bus **803** can receive the data carried in the infrared signal and place the data on bus **803**. Bus **803** carries the data to main memory **807**, from which processor **805** retrieves and executes the instructions. The instructions received by main memory **807** may optionally be stored on storage device **811** either before or after execution by processor **805**.

Computer system **801** also includes an input/output unit **815** coupled to bus **803**. The input/output unit **815** provides a two-way data communication coupling to a network link **821** that is connected to the IP network **823**. For example, input/output unit **815** may be a communication interface card to attach to any packet-switched network or telecommunication network using the following exemplary interfaces: SONET (Synchronous Optical Network) (e.g., OC-3c, OC-12c, OC-48c, etc.), ATM (Asynchronous Transfer Mode) (25 Mbps, 622 Mbps, etc.), T3/T1, DSL (Digital Subscriber Line), and ISDN (Integrated Services Digital Network). Wireless links may also be implemented. In any such implementation, input/output unit **815** sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information. Network link **821** typically provides data communication through one or more networks (i.e., the private network **617**) to other data devices. Networks, which are a part of the private network **617**, use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link **821** and through input/output unit **815**, which carry the digital data to and from computer system **801**, are exemplary forms of carrier waves transporting the information.

Computer system **801** can send messages and receive data, including program code, through the network(s), network link **821** and input/output unit **815**. The received code may be executed by processor **805** as it is received, and/or stored in storage device **811**, or other non-volatile storage for later execution. In this manner, computer system **801** may obtain application code in the form of a carrier wave.

The techniques described herein provide several advantages over prior approaches to tracking DoS flood attacks. An overlay network of IP tunnels selectively reroutes flood attack datagrams directly from edge routers to the tracking routers. Using security diagnostic features, the edge routers and the tracking routers identify DoS flood attack datagrams and perform input debugging to determine the ingress edge router. These datagrams are examined and dropped or forwarded to the appropriate egress point. This approach simplifies the process of determining the ingress adjacency of a flood attack, using the existing hardware and software infrastructure.

Obviously, numerous modifications and variations of the present invention are possible in light of the above teachings. It is therefore to be understood that within the scope of the appended claims, the invention may be practiced otherwise than as specifically described herein.

What is claimed as new and desired to be secured by Letters Patent of the United States is:

1. A method for tracking denial-of-service floods, the method comprising:

rerouting a DoS flood attack datagram to a tracking router, wherein the tracking router forms an overlay tracking network with respect to an egress edge router; and

identifying, by the tracking router, an ingress edge router that forwarded the DoS flood attack datagram.

2. The method according to claim **1**, further comprises executing security diagnostic functions.

3. The method according to claim **2**, wherein the security diagnostic functions comprise input debugging.

4. The method according to claim **1**, wherein the overlay tracking network is within an autonomous system that is different from another autonomous system corresponding to the ingress edge router and the egress edge router.

5. The method according to claim **4**, further comprising providing routing information by the overlay tracking network to the ingress edge router and the egress edge router using an inter-administrative-domain routing/signaling protocol.

6. The method according to claim **5**, wherein the inter-administrative-domain routing/signaling protocol is BGP (Border Gateway Protocol).

7. The method according to claim **1**, further comprising communicating between the edge routers and the tracking router via tunnels that are created over an unreliable datagram delivery service protocol.

8. The method according to claim **1**, further comprising communicating between the edge routers and the tracking router via virtual connections over a separate lower layer protocol.

9. The method according to claim **1**, further comprising communicating between the edge routers and the tracking router via physical connections.

10. The method according to claim **1**, further comprising routing the DoS flood attack datagram from the ingress edge router to the tracking router, wherein the egress edge router has a static route to the victim.

11. The method according to claim **10**, further comprising announcing the static route to the edge routers using an inter-administrative-domain routing/signaling protocol.

12. The method according to claim 11, wherein the inter-administrative-domain routing/signaling protocol in the announcing step is EBGP (External Border Gateway Protocol).

13. The method according to claim 11, further comprising establishing another static route between the egress router and an external router associated with a victim node, the victim node receiving the DoS flood attack datagram.

14. A communication system for tracking denial-of-service (DoS) floods, the communication system comprising:

a plurality of edge routers including an ingress edge router and an egress edge router, each of the edge routers being configured to perform security diagnostic functions, in part, to identify a DoS flood attack datagram, wherein the ingress edge router is associated with a source of the DoS flood attack datagram; and

a tracking router adjacent to the egress edge router, the tracking router being configured to perform the security diagnostic functions, the ingress edge router rerouting the DoS flood attack datagram to the tracking router as to permit identification of the ingress edge router, wherein the tracking router forms an overlay tracking network with respect to the plurality of edge routers.

15. The system according to claim 14, wherein the security diagnostic functions comprise input debugging.

16. The system according to claim 14, wherein the overlay tracking network is within an autonomous system that is different from another autonomous system corresponding to the plurality of edge routers.

17. The system according to claim 16, wherein the tracking router communicates routing information by the overlay network to the one edge router using an inter-administrative-domain routing/signaling protocol.

18. The system according to claim 17, wherein the inter-administrative-domain routing/signaling protocol is BGP.

19. The system according to claim 14, wherein the tracking router communicates with the edge routers via tunnels that are created over an unreliable datagram delivery service protocol.

20. The system according to claim 14, wherein the tracking router communicates with the edge routers via virtual connections over a separate lower layer protocol.

21. The system according to claim 14, wherein the tracking router communicates with the edge routers via physical connections.

22. The system according to claim 14, wherein the overlay tracking network further comprises additional tracking routers.

23. The system according to claim 22, wherein the tracking routers are interconnected via tunnels that are created over an unreliable datagram delivery service protocol.

24. The system according to claim 22, wherein the tracking routers are interconnected via virtual connections over a separate lower layer protocol.

25. The system according to claim 22, wherein the tracking routers are interconnected via physical connections.

26. The system according to claim 14, wherein the ingress edge router routes the DoS flood attack datagram to the tracking router due to a dynamic routing update from the tracking router.

27. The system according to claim 26, further comprising an external router coupled to the egress edge router via another static route, wherein the external router is associated with a victim node, the victim node receiving the DoS flood attack datagram.

28. A computer-readable medium carrying one or more sequences of one or more instructions for tracking denial-of-service floods (DoS), the one or more sequences of one or more instructions including instructions which, when executed by one or more processors, cause the one or more processors to perform the steps of:

receiving a DoS flood attack datagram on an overlay network formed by a tracking router;

identifying the DoS flood attack datagram; and

identifying, by the tracking router, a previous hop router associated with the DoS flood attack datagram to determine an ingress adjacency associated with the DoS flood attack.

29. The computer-readable medium according to claim 28, wherein the computer readable medium further includes instructions for causing the one or more processors to perform the steps of:

instructing the previous hop router to identify a respective previous hop router associated with the DoS flood attack datagram.

* * * * *

Exhibit 1012

U.S. Patent 7,120,931 by David Cheriton ("Cheriton")

Customer No.: 000027683

Haynes and Boone, LLP
IP Section
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Telephone [214] 651.5000
Fax [214] 200.0853

US007120931B1

(12) **United States Patent**
Cheriton

(10) **Patent No.: US 7,120,931 B1**
(45) **Date of Patent: Oct. 10, 2006**

(54) **SYSTEM AND METHOD FOR GENERATING FILTERS BASED ON ANALYZED FLOW DATA**

(75) Inventor: **David Cheriton**, Palo Alto, CA (US)

(73) Assignee: **Cisco Technology, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 694 days.

(21) Appl. No.: **09/652,454**

(22) Filed: **Aug. 31, 2000**

(51) **Int. Cl.**
G06F 9/00 (2006.01)
G06F 15/16 (2006.01)
G06F 17/00 (2006.01)

(52) **U.S. Cl.** **726/13**; 726/12; 713/152

(58) **Field of Classification Search** 709/229, 709/233, 238, 240; 713/200, 201, 153, 154, 713/152; 370/235; 726/12, 13
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,774,668	A *	6/1998	Choquier et al.	709/223
5,848,233	A *	12/1998	Radia et al.	713/201
5,958,015	A *	9/1999	Dascalu	709/229
5,968,176	A *	10/1999	Nessett et al.	713/201
6,154,775	A *	11/2000	Coss et al.	709/225
6,222,856	B1 *	4/2001	Krishnan et al.	370/468
6,243,667	B1 *	6/2001	Kerr et al.	703/27
6,266,706	B1 *	7/2001	Brodnik et al.	709/242
6,321,338	B1 *	11/2001	Porras et al.	713/201
6,389,532	B1 *	5/2002	Gupta et al.	713/163
6,453,345	B1 *	9/2002	Trcka et al.	709/224
6,496,935	B1 *	12/2002	Fink et al.	713/201
6,510,509	B1 *	1/2003	Chopra et al.	712/13
6,542,508	B1 *	4/2003	Lin	370/395.43
6,550,012	B1 *	4/2003	Villa et al.	713/201
6,567,408	B1 *	5/2003	Li et al.	370/395.31
6,598,034	B1 *	7/2003	Kloth	706/47
6,625,150	B1 *	9/2003	Yu	370/389
6,651,099	B1 *	11/2003	Dietz et al.	709/224
6,667,985	B1 *	12/2003	Drummond-Murray	370/415
6,771,661	B1 *	8/2004	Chawla et al.	370/468
6,772,347	B1 *	8/2004	Xie et al.	713/201
6,804,820	B1 *	10/2004	Meric et al.	719/321
6,816,903	B1 *	11/2004	Rakoshitz et al.	709/226
2002/0023089	A1 *	2/2002	Woo	707/101
2002/0032793	A1 *	3/2002	Malan et al.	709/232
2002/0035628	A1 *	3/2002	Gil et al.	709/224
2002/0152209	A1 *	10/2002	Merugu et al.	707/7
2003/0023733	A1 *	1/2003	Lingafelt et al.	709/229
2003/0226038	A1 *	12/2003	Raanan et al.	713/201
2004/0215977	A1 *	10/2004	Goodman et al.	713/201

FOREIGN PATENT DOCUMENTS

EP 910197 A2 * 4/1999

(Continued)

OTHER PUBLICATIONS

Allied Telesyn, "Application Note 2037 Nemesis Firewall", May 2000.*

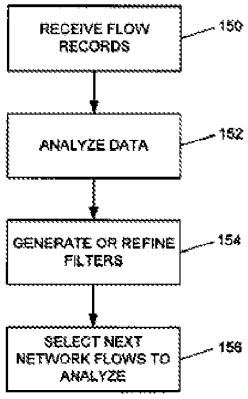
(Continued)

Primary Examiner—Jacques H. Louis Jacques
Assistant Examiner—Michael J. Simitoski
(74) *Attorney, Agent, or Firm*—Cindy Kaplan

(57) **ABSTRACT**

A method and system for generating filters based on analyzed flow data are disclosed. A method generally comprises separating the data into different network flows, analyzing at least one of the network flows, and detecting potentially harmful network flows. A filter is generated to prevent packets corresponding to the detected potentially harmful network flows from passing through the network device.

31 Claims, 6 Drawing Sheets



FOREIGN PATENT DOCUMENTS

WO WO 9724841 A2 * 7/1997

OTHER PUBLICATIONS

Apisdorf, Joel et al. "OC3MON: flexible, affordable, high performance statistics collection", Sep. 1996 (http://www.nlanr.net/NA/Oc3mon/).*
Cyberplaces Online Internet Workshop, "Firewalls" (http://www.cyberplaces.com/workshop/archive/0304982.htm).*
Decasper, Dan et al. "Router Plugins A Software Architecture for Next Generation Routers", 1998 ACM, pp. 229-240.*
Deering, S. et al. "Internet Protocol, Version 6 (IPv6) Specification", Dec. 1998.*
Gupta, Pankaj et al. "Packet Classification on Multiple Fields", 1999 ACM, pp. 147-160.*
Ilvesmaki, Mika et al. "Adaptive Flow Classification in IP Switching---The Measurement Based Approach", 1998.*
Lakshman, T.V. et al. "High-Speed Policy-Based Packet Forwarding Using Efficient Multi-dimensional Range Matching", 1998 ACM, pp. 203-214.*
Newman, Peter et al. "Flow Labelled IP: A Connectionless Approach to ATM", 1996 IEEE.*
Frank, Jeremy. "Artificial Intelligence and Intrusion Detection: Current and Future Directions", Jun. 1994 in Proceedings of the 17th National Computer Security Conference *
Neumann, Peter G. et al. "Experience with EMERALD to Date", Apr. 1999.*
Porras, Phillip A. et al. "Life Traffic Analysis of TCP/IP Gateways", Dec. 1997.*
Sekar, R. et al. "A High-Performance Network Intrusion Detection System", 1999.*
Valdes, Alfonso et al. "Adaptive, Model-based Monitoring for Cyber Attack Detection", 2000.*
Yoo, InSeon et al. "Adaptive Detection of Worms/Viruses In Firewalls", 2003.*
Bellovin, Steven M. "Distributed Firewalls", Nov. 1999.*
Buddhikot, Milind M. et al. "Space Decomposition Techniques for Fast Layer-4 Switching".*
Feldmann, Anja et al. "Tradeoffs for Packet Classification", 2000 IEEE.*
Network Associates. "The Active Firewall, The End of the Passive Firewall Era", 1999.*
Romig, Steve et al. "Cisco Flow Logs and Intrusion Detection at the Ohio State University", Dec. 1999, <http://web.archive.org/web/19980415195433/http://signal9.com/technical/prodsummary/fwsummary.html>.*
Signal9. "ConSeal PC Firewall Techaical Summary", Apr. 1998, <http://web.archive.org/web/19980415195433/http://signal9.com/technical/prodsummary/fwsummary.html>.*
Smith, Robert N. et al. "Operating Firewalls Outside the LAN Perimeter", 1999 IEEE.*
Smith, Robert N. et al. "A Protocol and Simulation for Distributed Communicating Firewalls", 1999 IEEE.*
Smith, Robert N. et al. "Firewall Placement in a Large Network Topology", 1997 IEEE.*
Venkatesan, Ramkumar M. et al. "Threat-Adaptive Security Policy", 1997 IEEE.*
Fullmer, Mark et al. "The OSU Flow-tools Package and Cisco NetFlow Logs", LISA 2000, Dec. 2000, <http://www.usenix.org/events/lisa00/full_papers/fullmer/fullmer_html>.*
Nvarro, John-Paul et al. "Combining Cisco NetFlow Exports with Relational Database Technology for Usage Statistics, Intrusion Detection, and Network Forensics", LISA 2000, Dec. 2000.*
Wang, Feiyi et al. "SITAR: A Scalable Intrusion-Tolerant Architecture for Distributed Services", 2001 IEEE.*

* cited by examiner

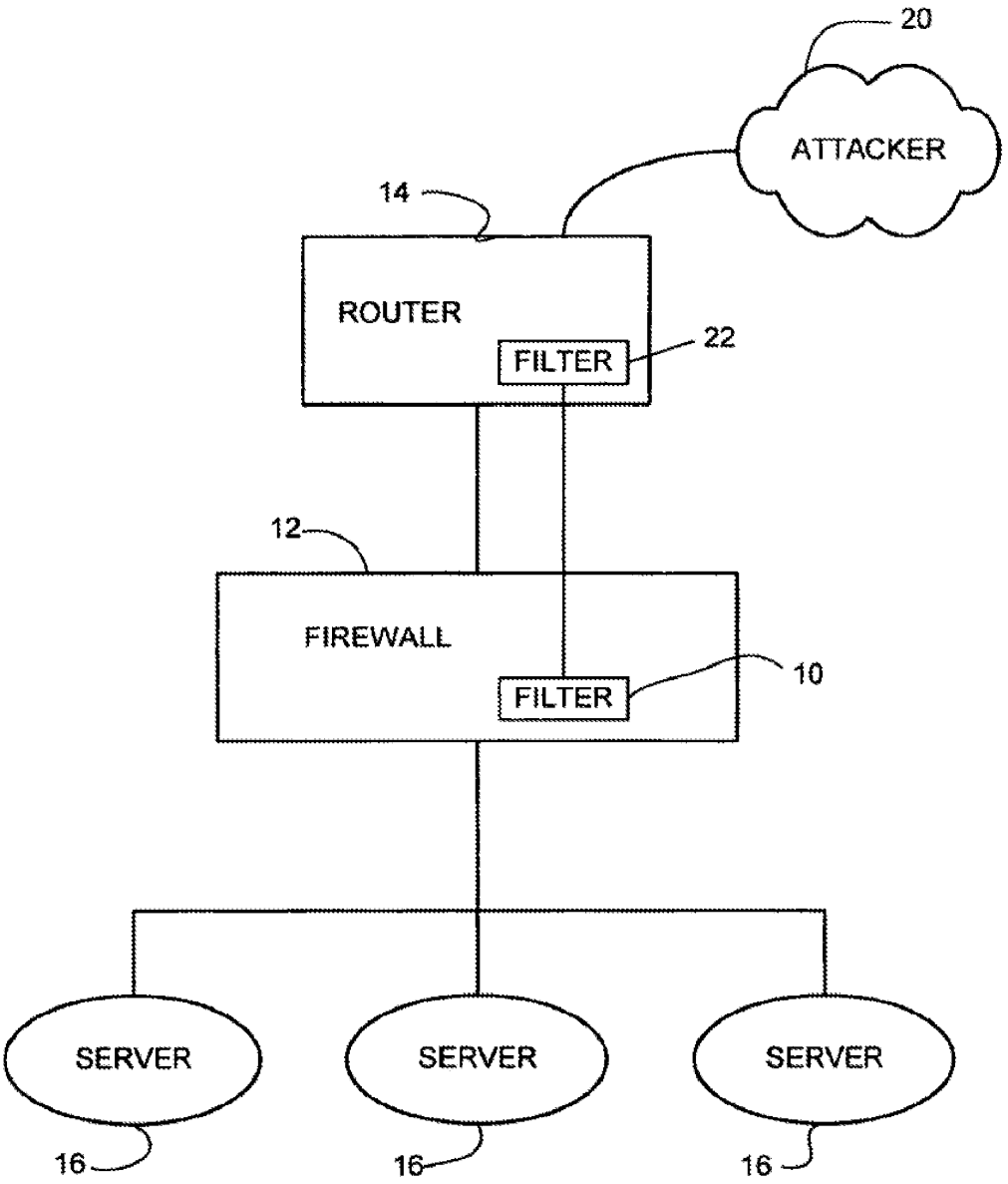

20
ATTACKER
14
ROUTER
FILTER
22
12
FIREWALL
FILTER
10
SERVER
SERVER
SERVER
16
16
16

Fig. 1

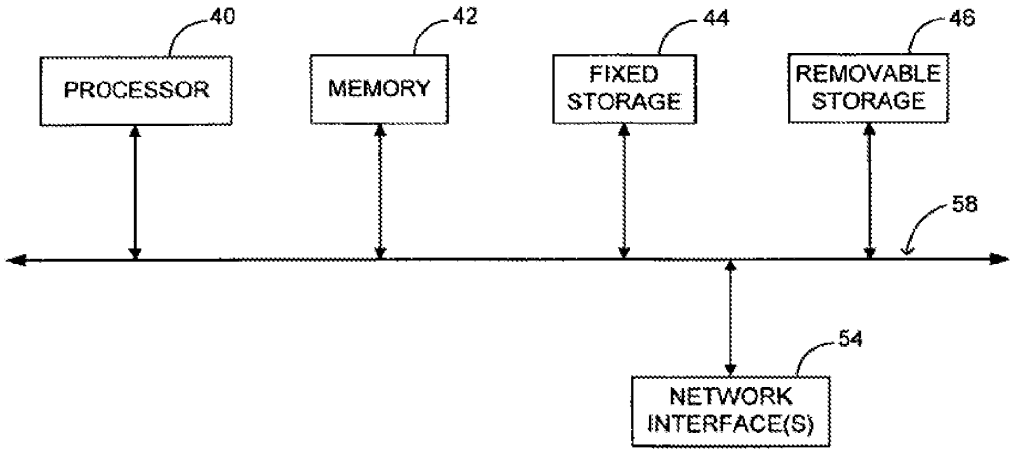

40
PROCESSOR
42
MEMORY
44
FIXED STORAGE
46
REMOVABLE STORAGE
58
54
NETWORK INTERFACE(S)

FIG. 2

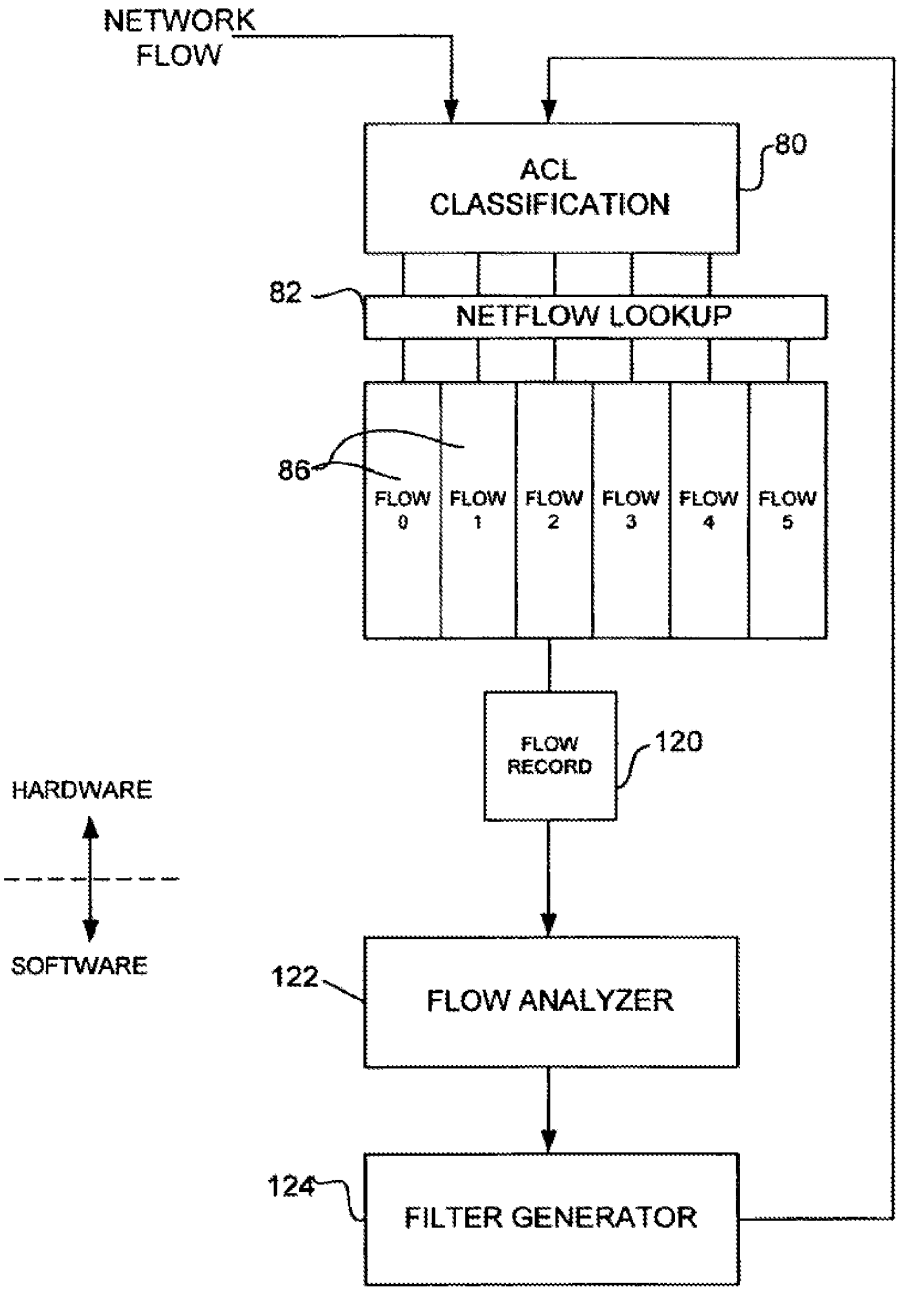

NETWORK FLOW
ACL CLASSIFICATION
80
82
NETFLOW LOOKUP
86
FLOW 0
FLOW 1
FLOW 2
FLOW 3
FLOW 4
FLOW 5
FLOW RECORD
120
HARDWARE
SOFTWARE
122
FLOW ANALYZER
124
FILTER GENERATOR

Fig. 3

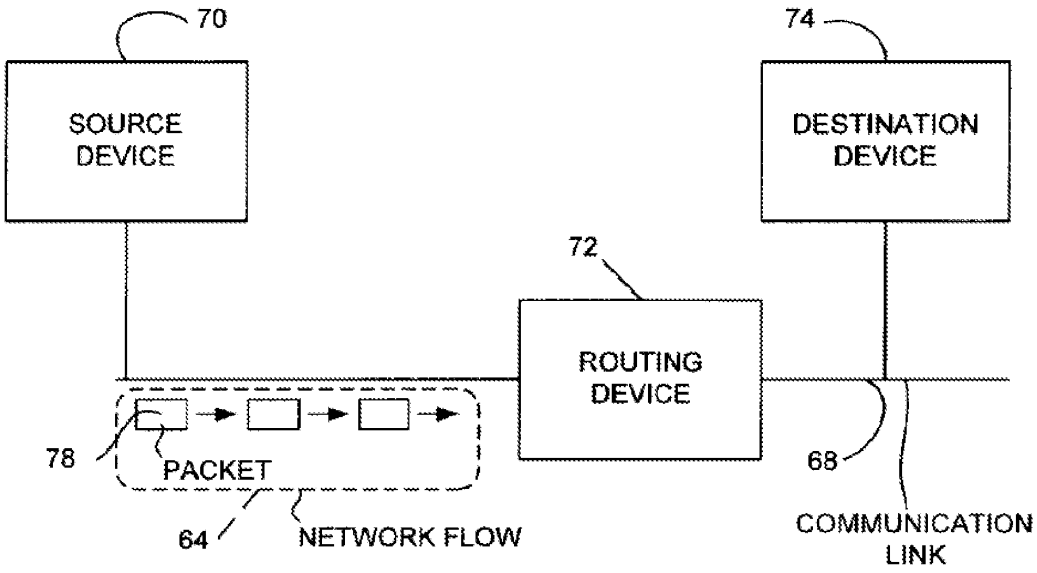

70
SOURCE DEVICE
74
DESTINATION DEVICE
72
ROUTING DEVICE
78
PACKET
64
NETWORK FLOW
68
COMMUNICATION LINK

Fig. 4

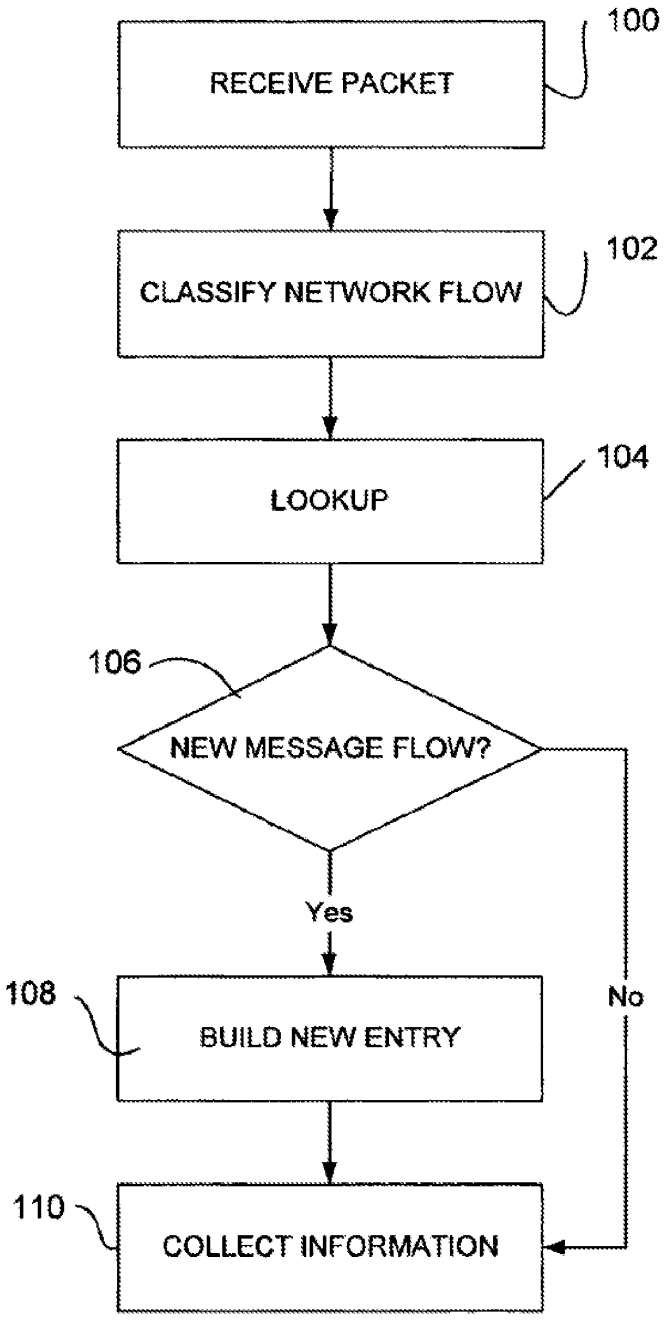

RECEIVE PACKET
100
CLASSIFY NETWORK FLOW
102
LOOKUP
104
106
NEW MESSAGE FLOW?
Yes
No
108
BUILD NEW ENTRY
110
COLLECT INFORMATION

Fig. 5

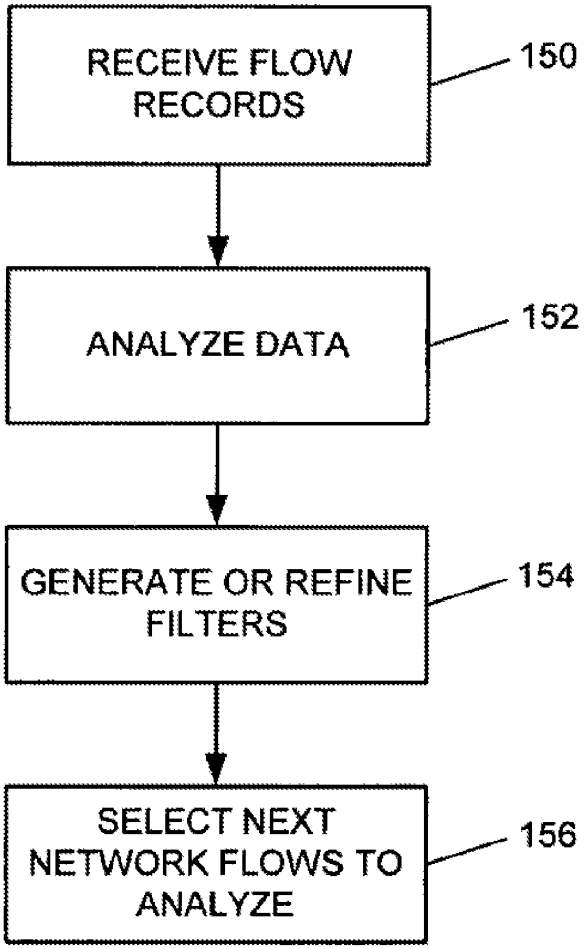

RECEIVE FLOW RECORDS
150
ANALYZE DATA
152
GENERATE OR REFINE FILTERS
154
SELECT NEXT NETWORK FLOWS TO ANALYZE
156

Fig. 6

SYSTEM AND METHOD FOR GENERATING FILTERS BASED ON ANALYZED FLOW DATA

BACKGROUND OF THE INVENTION

The present invention relates generally to filtering data in high-speed computer networks, and more specifically, to the generation and refinement of filters.

In the connected world of the Internet, destructive individuals can create major network security problems for administrators with systems exposed to public networks. The recent denial of service attacks on many of the web's most popular sites makes this clearer than ever before. A denial of service attack occurs when a malicious attacker sends continuous TCP/IP packets to a server, which quickly take up resources until there are no more resources available and a system hang or crash occurs. Commonly the targeted site may appear unavailable to the broader Internet because of the saturation of its network segment. Denial of service attacks can result in significant loss of time and money for many organizations.

Denial of service attacks are different from most other attacks because they are not targeted at gaining access to a network or information on the network. These attacks focus on making a service unavailable for normal use, which may be accomplished by exhausting some resource limitation on the network or within an operating system or application. Denial of service attacks are most frequently executed against network connectivity. The goal is to prevent hosts or networks from communicating on the network. There are several types of denial of service attacks, which can occur at various levels. When involving specific network server applications, such as a Hypertext Transfer Protocol (HTTP) server or a File Transfer Protocol (FTP) server, these attacks can focus on acquiring and keeping open all of the available connections supported by that server, effectively locking out valid users of the server or service. Denial of service attacks can also be implemented using other Internet protocols, such as UDP and Internet Control Message Protocol (ICMP).

The most common denial of service attack is the SYN attack. This is a network level attack. The attacker sends continuous TCP SYN packets to a server. Each TCP SYN packet hangs the connection until there are no more TCP resources available. The attacker begins the process of establishing a connection to the victim machine, but does it in such a way as to prevent the ultimate completion of the connection. In the meantime, the victim machine has reserved one of a limited number of data structures required to complete the impending connection. The result is that legitimate connections are denied while the victim machine is waiting to complete phony "half-open" connections. This type of attack does not depend on the attacker being able to consume network bandwidth. The intruder may be consuming kernel data structures involved in establishing a network connection. The implication is that an intruder can execute this attack from a dial-up connection against a machine on a very fast network.

High-speed networks make detecting and responding to certain types of failures and attacks difficult. The high speed makes it difficult to carefully examine every packet or even maintain state and monitor the state of every data steam without extensive hardware support. For example, a 50 MPPS switch may receive over one million flow streams per second, producing a logging data rate of 20 megabytes per second with just a 20 byte record per flow. This rate of log data is expensive to store in hardware and practically impossible to process in software. Monitoring only a subset of the traffic results in holes in detection and provides no defense against problems that exceed this subset capacity.

A more common approach is to use aggregate traffic monitoring and policing. For example, a server switch may rate limit ICMP traffic arriving on an external port to a predetermined maximum rate rather than preclude it altogether. However, an attack or a failure using ICMP may use up the entire rate, effectively blocking out other ICMP traffic, with no mechanism to determine what is causing the problem. The offending or suspicious data is therefore hidden in the aggregate. Furthermore, a high-rate attack or failure can originate upstream of a device experiencing the problem, thus compromising use of the associated link even if the device can filter out the traffic.

SUMMARY OF THE INVENTION

A method and system for generating filters based on analyzed flow data are disclosed. A method generally comprises separating the data into different network flows, analyzing at least one of the network flows, and detecting potentially harmful network flows. A filter is generated to prevent packets corresponding to the detected potentially harmful network flows from passing through the network device.

In another aspect of the invention a method for propagating filters to an upstream device comprises generating a filter at a first network device, sending information on the filter to a second network device located upstream from the first network device, and requesting the second network device to install the filter.

A computer program product for generating filters based on analyzed network flows generally comprises code that analyzes the network flow and code that detects potentially harmful network flows. The product further comprises code that automatically generates a filter to prevent packets corresponding to the detected potentially harmful network flows from passing through the network device and a computer readable storage medium for storing the codes.

A system of the present invention for automatically generating filters based on data entering a network device generally comprises a netflow device operable to receive streams of packets, separate the streams, and create a summary record containing information on each of the streams, and a flow analyzer operable to analyze the record and identify potentially harmful network flows. The system further includes a filter generator operable to generate a filter to prevent packets corresponding to the identified potentially harmful network flows from passing through the network device.

In another aspect of the invention a system generally comprises means for separating the data into different network flows, means for analyzing at least one of the network flows, means for detecting potentially harmful network flows, and means for generating a filter to prevent packets corresponding to the detected potentially harmful network flows from passing through the network device.

The above is a brief description of some deficiencies in the prior art and advantages of the present invention. Other features, advantages, and embodiments of the invention will be apparent to those skilled in the art from the following description, drawings, and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a diagram illustrating a firewall located between a router and a plurality of servers with an attacker sending data into the router.

FIG. **2** is a diagram illustrating a logical architecture of a computer system that may be used to execute software of this invention.

FIG. **3** is a diagram illustrating a system of the present invention for analyzing data transmitted to the firewall of FIG. **1** and generating a filter for the firewall.

FIG. **4** is a diagram of a network illustrating network flow on a communication link.

FIG. **5** is a flowchart illustrating a process for classifying and analyzing network flows in a netflow directory.

FIG. **6** is a flowchart illustrating a process of the present invention for refining filters to identify characteristics of packets involved in an attack or failure.

Corresponding reference characters indicate corresponding parts throughout the several views of the drawings.

DETAILED DESCRIPTION OF THE INVENTION

The following description is presented to enable one of ordinary skill in the art to make and use the invention. Descriptions of specific embodiments and applications are provided only as examples and various modifications will be readily apparent to those skilled in the art. The general principles described herein may be applied to other embodiments and applications without departing from the scope of the invention. Thus, the present invention is not to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features described herein. For purpose of clarity, details relating to technical material that is known in the technical fields related to the invention have not been described in detail.

Referring now to the drawings, and first to FIG. **1**, a filter **10** is shown inserted into a firewall **12** located between a router **14** and a plurality of servers **16**. An attacker **20** is shown positioned to transmit harmful packets to the router **14**. A system and method of the present invention utilize a combination of aggregate filters **10** and a mechanism such as a netflow directory of microflows to automatically and progressively refine the aggregate filters to identify detailed characteristics of packets involved in an attack or a failure. The system is used to detect harmful network flows which may include denial of service attacks or merely a high rate of data coming into the system which needs to be filtered to reduce the possibility of problems within the network. As described below, the system may also use an inter-router filter propagation protocol (FPP) to automatically propagate filter information upstream to filter data closer to the source of the data, as illustrated by filter **22** located at router **14**.

The present invention operates in the context of a data communication network including multiple network elements. Some of the nodes in a network that employs the present invention may be network devices such as routers and switches. For example, some of the nodes may be specially configured routers such as those available from Cisco Systems, Inc. of San Jose, Calif. As used herein the term router is used to refer to devices that forward packets based on network and higher layer information. The router may include, for example, a master central processing unit (CPU), interfaces, and a bus (e.g., a PCI bus). The CPU preferably includes a memory and a processor. When acting under the control of appropriate software or firmware, the CPU is responsible for such router tasks as routing table computations, network management, and general processing of packets. It preferably accomplishes all of these functions under the control of software including an operating system (e.g., a version of the Internetwork Operating System (IOS®) of Cisco Systems, Inc.) and any appropriate applications software. The CPU may include one or more processors such as a processor from the Motorola family or microprocessors of the MIPS family of microprocessors. In an alternative embodiment, the processor is specially designed hardware for controlling operations of the router. Memory can be non-volatile RAM and/or ROM. However, there are many different ways in which memory may be coupled to the system. In an alternative embodiment, a router or switch may be implemented on a general purpose network host machine such as the computer system shown in the block diagram of FIG. **2**.

FIG. **2** shows a system block diagram of a computer system that may be used to execute software of an embodiment of the invention. The computer system may include subsystems such as a central processor **40**, system memory **42**, removable storage **46** (e.g., CD-ROM drive), and a hard drive **44** which can be utilized to store and retrieve software programs incorporating computer code that implements aspects of the invention, data for use with the invention, and the like. The computer readable storage may also include tape, flash memory, or system memory. Additionally, a carrier wave (e.g., in a network including the Internet) may be a transmission medium. The computer system may further include a display screen, keyboard, and mouse which may include one or more buttons for interacting with a GUI (Graphical User Interface). Other computer systems suitable for use with the invention may include additional or fewer subsystems. For example, the computer system may include more than one processor **40** (i.e., a multi-processor system) or a cache memory.

The system bus architecture of the computer system is represented by arrows **58** in FIG. **2**. However, these arrows are only illustrative of one possible interconnection scheme serving to link the subsystems. For example, a local bus may be utilized to connect the central processor **40** to the system memory **42**. The components shown and described herein are those typically found in most general and special purpose computers and are intended to be representative of this broad category of data processors. The computer system shown in FIG. **2** is only one example of a computer system suitable for use with the invention. Other computer architectures having different configurations of subsystems may also be utilized.

Communication between computers within the network is made possible with the use of communication protocols, which govern how computers exchange information over a network. The computer may include an input/output circuit used to communicate information in appropriately structured form to and from the parts of computer and associated equipment. Connected to the input/output circuit are inside and outside high speed Local Area Network interfaces, for example. The inside interface will be connected to a private network, while the outside interface will be connected to an external network such as the Internet. Preferably, each of these interfaces includes a plurality of ports appropriate for communication with the appropriate media, and associated logic, and in some instances memory. The associated logic may control such communication intensive tasks as packet integrity checking and media control and management. The

high speed interfaces are preferably multiport Ethernet interfaces, but may be other appropriate interfaces such as FDDI interfaces.

Referring again to FIG. **1**, the firewall **12** is located between the router **14** and the plurality of servers **16**. The router **14** may be an Internet Service Provider (ISP) router, for example. It is to be understood that the system and method of the present invention may also be used in networks which are not connected to the Internet.

The firewall **12** is a system or group of systems that enforce a security policy between an organization's network and the Internet. The firewall **12** determines which inside services may be accessed from the outside, which outsiders are permitted access to the inside services, and which outside services may be accessed by insiders. For the firewall **12** to be effective, all traffic to and from the Internet must pass through the firewall, where it can be inspected. The firewall **12** permits only authorized traffic to pass, thus providing a perimeter form of defense for securing network access. The firewall **12** may be configured, for example, to allow specific IP source addresses to access specific IP destination addresses, or only allow HTTP packets and not allow TELNET or FTP. The firewall **12** is preferably a packet filtering firewall but may also be a proxy (application) firewall.

FIG. **3** illustrates a netflow directory of microflows which are used to analyze the high speed data entering the router **14** or firewall **12** to identify detailed characteristics of packets involved in an attack or a failure. The netflow mechanism is configured to create network flows for those matching specified aggregate filters as described below. Also included is a means for determining the specific flows created by the netflow mechanism and the aggregate filter responsible for the creation of each specific flow. The netflow mechanism may be, for example, a network flow switching and flow data export system such as disclosed in U.S. patent application Ser. No. 08/886,900, filed Jul. 2, 1997, which is incorporated herein by reference in its entirety.

FIG. **4** shows a network flow **64** on communication link **68** connecting a source device **70**, a routing device **72** and a destination device **74**. The network flow **64** consists of a unidirectional stream of packets **78** to be transmitted between pairs of transport service access points. The network flow **64** thus, broadly refers to a logical communication circuit between communication endpoints. The source device **70** may be the attacker **20**, the routing device may be the ISP router **14** or firewall **12**, and the destination device may be one of the plurality of servers **16** shown in FIG. **1**. The communication link **68** may comprise any form of physical media layer such as Ethernet, FDDI, or HDLC serial link. The routing device **72** may include specific hardware constructed or programmed for performing process steps described below, a general purpose processor operating under program control, or some combination thereof.

Data is received from the source device **70** in network flow **64** which is defined by a network layer address for the source device, a port number at the source device, a network layer address for the destination device **74**, a port number at the destination device, and a transmission protocol. For example, HTTP (Hypertext Transfer Protocol) web packets from a particular source host to a particular destination host constitute a separate flow from FTP (File Transfer Protocol) file transfer packets between the same pair of hosts. The transmission protocol type may identify a known transmission protocol, such as UDP, TCP, ICMP, or IGMP (Internet Group Management Protocol). The source device **70** may be identified by its IP (Internet Protocol) address, for example. The port number at the source device is identified by either a port number which is specific to a particular process, or by a standard port number for the particular transmission protocol type.

As shown in FIG. **3**, packets **78** within the network flow **64** are first sent to an ACL (Access Control List) classification device **80**. The ACL classification device **80** is configured to classify the received packets **78**. The flow may be classified, for example, by source node (IP address), destination node (IP address), detail destination node (destination address, source TCP/UDP port, destination TCP/UDP port, protocol), host matrix (source address/destination address pair), detail host matrix (source/destination address, port, protocol), source TCP/UDP (transportation layer source port), destination TCP/UDP port (transport layer destination port), protocol (protocol name), detail interface (input-output physical interface pair), or some combination thereof.

Security (ACL) processing may be applied only to the first packet **78** of a network flow **64**. For example, the ACL processing may determine whether or not to build a netflow entry on the first packet, and subsequent packets may bypass the ACL processing. Information from the first packet **78** is used to build an entry in a netflow cache. Subsequent packets in the flow are handled via a single streamlined task that handles data collection. After packets **78** pass through the ACL classification device **80**, the packets are sent to a netflow lookup device **82** which separates the streams into a plurality of flows (or bucket) **86**. Each bucket includes a linked list of entries which include information about a particular network flow **64**. The netflow mechanism maintains the flow cache by removing entries for network flows which are inactive or no longer considered valid.

FIG. **5** is a flowchart illustrating a process for analyzing packets utilizing the netflow directory. A packet **78** is first received at the ACL classifier **80** (step **100**). The ACL classifier **80** may examine a header of the packet **78** and identify the IP address for the source device, the IP address for the destination device, and the protocol type for the packet, for example, to classify the packet (step **102**). The ACL classifier **80** then selects a flow column **86** for the network flow **64**. Based on the results of the classification, the netflow lookup device **82** may perform a lookup in the flow cache for the identified network flow (step **104**). If the lookup is unsuccessful, the identified network flow **64** is a new network flow and the netflow mechanism may build a new entry in the flow cache (step **106**). The proper treatment of packets in the network flow is determined, for example, from the classification (steps **106** and **108**). The netflow mechanism then proceeds at step **110**, using the information from the new entry in the flow cache, just as if the identified network flow were an old network flow. If the lookup is successful, the identified network flow **64** is an old network flow and the lookup device **82** continues with step **110**. Since the netflow mechanism processes each packet **78** in the network flow **64** responsive to the entry for the network flow in the flow cache, the netflow mechanism is able to implement administrative policies which are designated for each network flow rather than for each packet. Thus, the network flows are analyzed and information on incoming packets is provided without examining each packet received. This flow collection aggregation allows for data to be stored by aggregate summary records instead of raw data records.

Once the flow **64** passes through the netflow directory, flow records **120** are created that provide information about

a particular network flow (FIG. **3**). The flow record **120** may include, for example, information about packets **78** in particular network flows **64**, including source address, port number, and protocol type, or other information relevant to diagnosing actual or potential network problems including attacks on the network. Since the amount of information from the high speed data is reduced in hardware by the netflow mechanism to a reasonable amount of data, the flow records **120** can now be analyzed by software. The flow records **120** are sent to a flow analyzer **122** where the flows are analyzed to identify characteristics of packets **78** involved in an attack or a failure. For example, if a large number of SYN packets or an unusual distribution of packets is identified as coming from a source (e.g., attacker **20**), it is likely that the source is involved in an attack. The attacker may first be identified as an organization such as a university, business, or an ISP, for example. However, it is likely that there is only one source within the organization that is sending harmful packets. Once a group of packets **78** are identified as harmful, the corresponding network flows **64** can be analyzed to further refine the filter. Therefore, instead of filtering out all data arriving from the identified organization, only the destructive packets received from the actual attacker are dropped.

FIG. **6** is a flowchart illustrating a process performed by the flow analyzer **122** and flow generator **124**. The flow records **120** are first received by the flow analyzer **122** at step **150**. The flow analyzer **122** then analyzes data received at step **152**. The filter generator **124** next generates or refines filters (step **154**). The filter generator **124** then selects the next group of network flows to be analyzed and passes this information to the ACL classifier **80** (step **156**).

The initial class of packets **78** to be analyzed is selected based on statistics associated with the aggregate filters, as described below. The data which is to be analyzed is periodically changed or updated to further refine a filter once it has been generated. For example, a first class of packets **78** may be analyzed for 0.5 second then a next class of packets analyzed for the next 0.5 seconds. The initial filters **10** may be configured according to user specified configurations or default values. The flow analyzer **122** and filter generator **124** then use the analyzed flow to determine if the existing filters need to be refined or new filters need to be generated. Based on the analyzed flow, the filter generator **124** will tell (or modify) the ACL classifier **80**, which then affects the netflow entries that are created. The class of packets **78** selected may be based on a class of packets which have been identified as potentially harmful, or may be randomly chosen. The ACL classifier **80** may, for example, begin by looking at flows **64** for all packets **78** received from a source with an IP address having the form 3.xxx.xxx.xxx, where xxx represents any possible value from zero to 255. If a problem is identified in one of the packets streams **64**, the ACL classifier **80** may be then instructed to look at flows for all packets **78** received from a source having an IP address of 3.141.xxx.xxx. This may be narrowed down further to refine the filter **10**.

The flow analyzer **122** monitors the statistics associated with these aggregate filters **10**. If the statistics associated with an aggregate filter entry indicate a potential problem (or just as a periodic check of the traffic distribution), creation of netflow entries is enabled for packets matching this entry. Consequently, the flow analyzer **122** receives a flow record **120** for each flow matching this aggregate. Using this specific flow information, the flow generator **124** determines how to refine the aggregate filter. For example, the flow label information may indicate that most ICMP packets are coming from a particular source address. In this case, the flow generator **124** can configure an aggregate filter **10** that matches ICMP packets from that source, establishing a separate policer for that filter or potentially just blocking the source. The original aggregate filter is preferably retained as well so that all other ICMP traffic matches to this original filter. The flow analyzer **122** can then monitor the statistics of the original aggregate filter with the offending host removed, to detect whether there are further anomalies within the aggregate flow.

The flow analyzer **122** may also be configured to recognize that the total rate of traffic matching an aggregate value may far exceed its ability to sample by examining the statistics for the entry. For example, the total port traffic into a web server may be too much to handle. In this case, the aggregate filters can be split into multiple subaggregates based on some quasi-random distinction. It may use, for example, four aggregate filters that select different traffic based on the lower-order two bits of the IP source address for the packets **78**. The flow analyzer **122** then samples using the netflow directory for each of the four aggregate filters in sequence.

The filters **10** may be refined to recognize either that the rate of packets **78** itself is a problem or else allow these packets to be redirected to the flow analyzer **122** for more careful examination, such as to identify specific aspects of an attack or failure. When such examination is in use, the filters **10** can employ a rate-limiting policer to prevent software from being overwhelmed.

The system shown in FIG. **3** may also be used to automatically recognize further structure to network traffic that does not necessarily represent an attack or a failure. For example, a web server may receive an excessive level of traffic from a search engine spider or an upstream web cache. By automatically detecting a high demand source of this nature, the filter generator **124** can automatically reconfigure the filters **10** to handle this demand. For example, a policy may indicate a maximum aggregate rate of HTTP traffic of 100 Mpbs and a maximum rate from any source of 25 Mbps. Rather than explicitly policing every flow, the system can be used to identify sources that appear to represent excessive traffic, allowing aggregate filters to be created that separate them out of the overall aggregate and throttle their traffic appropriately. These filters **10** can also be automatically removed when the associated traffic drops off, based on the statistics associated with the identified flow. Thus, for example, once a search engine finishes its searching at a web site, the filter **10** created for it indicates that traffic has dropped because of the lower rate and the specific filter can be reclaimed.

Once the initial filters **10** are defined within the firewall **12**, the system may be used to automatically propagate filter information upstream to filter data closer to the source, such as filter **22** located at router **14** shown in FIG. **1**. For example, an enterprise switch may identify excessive traffic as coming from a particular source subnet or host and communicate this upstream to a router, requesting this router to automatically install a filter for this traffic. The filter **22** prevents the excessive traffic from monopolizing a potentially slower tail circuit to the enterprise, just to be discarded at that point. This filter **22** can also be automatically removed, as previously described with respect to the filter **10** located at the firewall **12**. The downstream device is able to receive statistics from the upstream device for a specific filter that it requested, determine that the traffic requiring the filter is no longer present, and remove this upstream filter. An inter-switch/router filter propagation protocol (FPP) is

used to create, remove, monitor, and modify filters **10**, **22** between devices. FPP uses negative routing (i.e., it indicates what traffic not to forward). An upstream node receives the negative routing information from a downstream node and then refines the filter and puts it in a place such that it only affects traffic that it would forward to the requesting downstream node. That is, a node can not request filtering of traffic to other nodes. The upstream node maintains a packet and byte count of packets received that match the filter and are thus dropped. FPP provides an operation to request these statistics from the upstream node and optionally extend the lifetime of the filter. A filter is removed after a specified time period during which a downstream node has not requested extension of this filter or the downstream node explicitly requests removal of the filter. FPP preferably follows the type-length-value (TLV) structure of protocols such as BGP, and is designed to run over TCP, for example. It is to be understood that other transports and representation may be used for FPP without departing from the scope of the invention.

An upstream device can limit the filters that a downstream device can specify using this FPP to those affecting the traffic that the downstream device is to receive, based on local routing information. The upstream device can further limit the total number of such filters a downstream device can request, encouraging the downstream device to use this limited resource effectively. It can also refine these filters to match just that traffic destined for the requesting downstream device. For example, a downstream device on subnet 171.172.X.X. can request a filter on HTTP traffic from source network 36.131.X.X. The upstream device can enter the filter to match on SA 36.131.X.X. and DA 171.172.X.X so that the filter (policer) does not affect traffic from this source to other subnets connecting to the same upstream device. It can also require the downstream device to actively subscribe to each such filter, so that the upstream device can reclaim these filters easily if the downstream node reboots and forgets the requested filter information.

By using filters in several different devices (e.g., switches, routers) effectively pushing back upstream towards the sources, the total number of filters available to react against an attack can be substantially more than that supported by a single device. This multi-hop filter propagation also has the benefit of allowing an ISP router, for example, to automatically block the traffic at a peering point that is simply going to be dropped, rather than transporting it across the ISPs backbone just to be dropped. The peering ISP can then back-propagate these filters itself to drop the traffic sooner, and in the extreme, use it as a basis to shut down or investigate an offending source.

Reverse path forwarding (RPF) may be used to attempt to prevent source spoofing. Effective filtering of attackers depends on the prevention of the attacker from spoofing other source addresses that are not registered for use by this attacking node. RPF check can be used to detect and drop packets corresponding to source spoofing provided that the routing topology is restricted, such as largely hierarchical. RPF check may be used in cases where attacks and failures are a concern, with certain network topologies being preferred or avoided to make this as enforceable as possible. That is, configurations where traffic can simultaneously arrive at a switch from a given source from multiple input ports or VLANs is avoided. RPF is an input function applied on the input interface of a router at the upstream end of a connection. RPF checks to see if any packet received at a router interface arrives on one of the best return paths to the source of the packet. RPF does a reverse lookup in a forwarding table. If the packet was received from one the best reverse path routes, the packet is forwarded as normal. If there is no reverse path route on the same interface from which the packet was received, the source address may have been modified or forged.

Assuming that RPF checking is deployed, the automatic progressive filter refinement can respond automatically to denial of service attacks allowing the sites to effectively respond in seconds or less, rather than hours, even with a distributed denial of service attack. The attacking sources can be quickly identified and filters propagated on this traffic upstream and locally. The filter can be adapted to new sources as the attack moves.

Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations made to the embodiments without departing from the scope of the present invention. Accordingly, it is intended that all matter contained in the above description and shown in the accompanying drawings shall be interpreted as illustrative and not in a limiting sense.

What is claimed is:

1. A method for generating filters based on data entering a network device, comprising:

separating the data into a plurality of network flows;

creating separate aggregate network flow summaries for each of said network flows;

sending at least one of said aggregate network flow summaries to a flow analyzer at the network device;

analyzing said at least one aggregate network flow summary to detect characteristics of potentially harmful network flows;

generating or refining a filter based on said analyzed aggregate network flow summary to prevent packets corresponding to detected potentially harmful network flows from passing through said network device; and

selecting a new aggregate network flow summary to analyze and sending the selected aggregate network flow summary to the flow analyzer for analysis, wherein the new aggregate flow summary corresponds to a network flow associated with the generated or refined filter.

2. The method of claim **1** further comprising classifying network flow based on a source device sending a packet.

3. The method of claim **2** wherein the network flow is classified based on an IP address of the source device.

4. The method of claim **1** wherein analyzing said at least one aggregate network flow summary comprises monitoring statistics associated with said network flows.

5. The method of claim **1** further comprising propagating the generated filter to an upstream network device.

6. The method of claim **1** further comprising sending each of said plurality of network flows to a corresponding flow cache and implementing policies designated for each of said network flows.

7. The method of claim **6** wherein sending each of said plurality of network flows is performed by hardware and analyzing said at least one aggregate network flow summary is performed by software.

8. The method of claim **1** wherein analyzing said at least one aggregate network flow summary comprises utilizing software.

9. The method of claim **1** wherein selecting a new aggregate network flow summary to analyze comprises selecting a class of said network flows to analyze based on previously analyzed network flows.

10. The method of claim 1 wherein detecting potentially harmful network flows comprises identifying a source address associated with said harmful network flow and generating a filter comprises generating a filter to prevent packets from said identified source from passing through said network device.

11. The method of claim 1 wherein information resulting from analyzing at least one of said aggregate network flow summaries is reduced in hardware so that flow records can be analyzed by software.

12. The method of claim 1 wherein a group of potentially harmful packets is detected and further comprising analyzing said corresponding network flow and further refining said filter.

13. The method of claim 1 further comprising selecting a group of network flows to be analyzed.

14. The method of claim 13 further comprising passing information on the selected group of network flows to a classifier.

15. The method of claim 1 wherein a class of packets to be analyzed is selected based on statistics associated with the generated or refined filter.

16. The method of claim 1 further comprising splitting said filters if traffic into said filter exceeds a sampling capability of the filter.

17. The method of claim 1 wherein analyzing said aggregate network flow summary comprises analyzing for a specified interval of time.

18. A computer program product for generating filters based on analyzed network flows, comprising:

code that separates data into different network flows;

code that creates an aggregate network flow summary for one or more of said network flows;

code that selects one or more network flows for analysis;

code that sends said selected aggregate network flow summaries to a flow analyzer at the network device;

code that analyzes said selected network flows by reviewing said aggregate network flow summaries;

code that detects potentially harmful network flows;

code that automatically generates or refines a filter based on said analyzed network flow summary to prevent packets corresponding to said detected potentially harmful network flows from passing through the network device;

code that selects a new aggregate network flow summary to analyze and send the selected aggregate network flow summary to the flow analyzer for analysis; and

a computer-readable storage for storing the codes executable by a processor.

19. The computer program product of claim 18 further comprising code that propagates said filter to an upstream network device.

20. The computer program product of claim 18 further comprising code that refines said filter based on said analyzed network flow.

21. A system for automatically generating filters based on data entering a network device, comprising:

a netflow device operable to receive streams of packets, separate said streams, and create a summary record containing information on each of said streams;

a flow analyzer located at the network device and operable to receive said summary records from said netflow device and analyze said summary records and identify potentially harmful network flows; and

a filter generator operable to generate or refine a filter based on analyzed summary records to prevent packets corresponding to said identified potentially harmful network flows from passing through the network device, wherein said netflow device is operable to create a new summary record containing information on a stream of data associated with said generated or refined filter.

22. The system of claim 21 wherein the network device comprises hardware and the flow analyzer and filter generator comprise software.

23. The system of claim 21 wherein the network device comprises an ACL classifier, a lookup device, and a plurality of flow buckets.

24. The system of claim 21 further comprising a filter propagator operable to send information on said filters to an upstream device and request the upstream device to create a corresponding filter.

25. The system of claim 21 further comprising a netflow directory comprising a plurality of flow cache entries and configured to build new flow cache entries for network flows without a corresponding flow cache entry.

26. The system of claim 21 further comprising a classifier operable to classify said streams of packets based on one or more packets received at the netflow device.

27. A method for generating filters for network flow, comprising:

receiving data at a network device;

classifying network flows based on one or more packets received at the network device;

analyzing one or more of said network flows;

generating a filter for one or more of said network flows;

processing each of said network flows according to a corresponding policy;

selecting a class of network flows to analyze;

analyzing said selected class of network flows; and

refining said filter for said selected class of network flows, wherein refining said filter comprises modifying the classification of network flows.

28. The method of claim 27 wherein each of said filters are generated specifically for a corresponding network flow.

29. An apparatus for generating filters based on analyzed network flows, the apparatus comprising:

one or more processors; and

a memory that stores instructions to be executed by said one or more processors, said instructions comprising:

code that creates separate aggregate network flow summaries for each of said network flows;

code that sends at least one of said aggregate network flow summaries to a flow analyzer at the network device;

code that analyzes said at least one aggregate network flow summary to detect characteristics of potentially harmful network flows;

code that generates or refines a filter based on said analyzed aggregate network flow summary to prevent packets corresponding to detected potentially harmful network flows from passing through said network device; and

code that selects a new aggregate network flow summary to analyze and sending the selected aggregate network flow summary to the flow analyzer for analysis, wherein the new aggregate flow summary corresponds to a network flow associated with the generated or refined filter.

30. The apparatus of claim 29 wherein said instructions further comprise code that propagates said filter to an upstream network device.

31. The apparatus of claim 29 wherein said instructions further comprise code that classifies network flow based on a source device sending a packet.

* * * * *

Exhibit 1013

U.S. Patent 6,502,135 by Edmund C. Munger et al. ("Munger")

Customer No.: 000027683

Haynes and Boone, LLP
IP Section
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Telephone [214] 651.5000
Fax [214] 200.0853

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)

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

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

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

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

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.**[7] **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
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

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

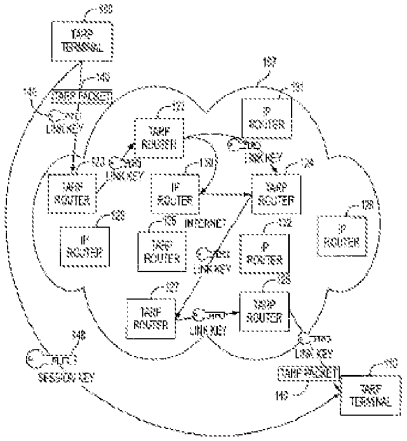
(List continued on next page.)

Primary Examiner—Krisna Lim
(74) *Attorney, Agent, or Firm*—Banner & Witcoff, Ltd.

(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.

17 Claims, 35 Drawing Sheets



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/ on Feb. 21, 2002, 3 pages.

J. Gilmore, "Swan: Securing the Internet against Wiretapping", printed from http://liberty.freeswan.org/freeswan_trees/freesswan-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 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", 2nd Edition, Chapter 13, IP Security, Jun. 8, 1998, pp. 390–400.

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

* cited by examiner

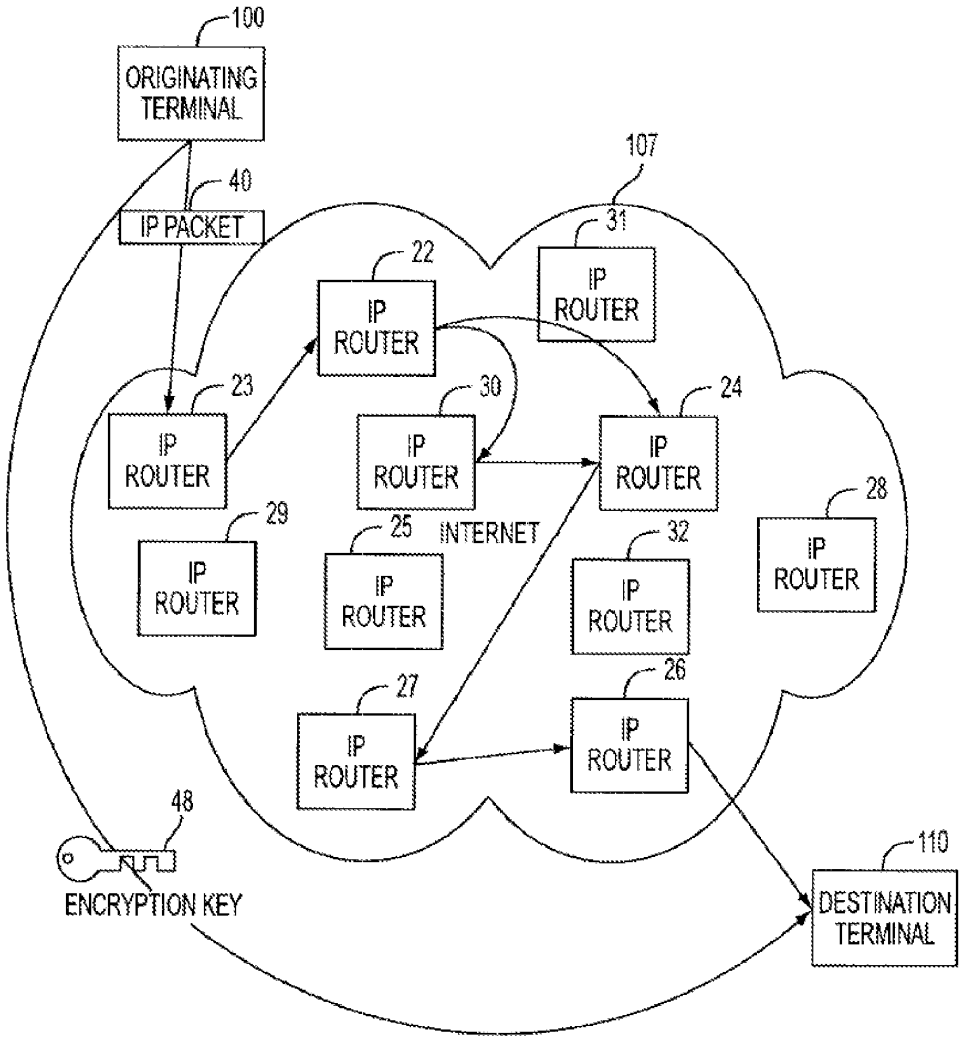

100
ORIGINATING TERMINAL
40
IP PACKET
107
31
IP ROUTER
22
IP ROUTER
23
IP ROUTER
30
IP ROUTER
24
IP ROUTER
28
IP ROUTER
29
IP ROUTER
25
IP ROUTER
INTERNET
32
IP ROUTER
26
IP ROUTER
27
IP ROUTER
48
ENCRYPTION KEY
110
DESTINATION TERMINAL

FIG. 1

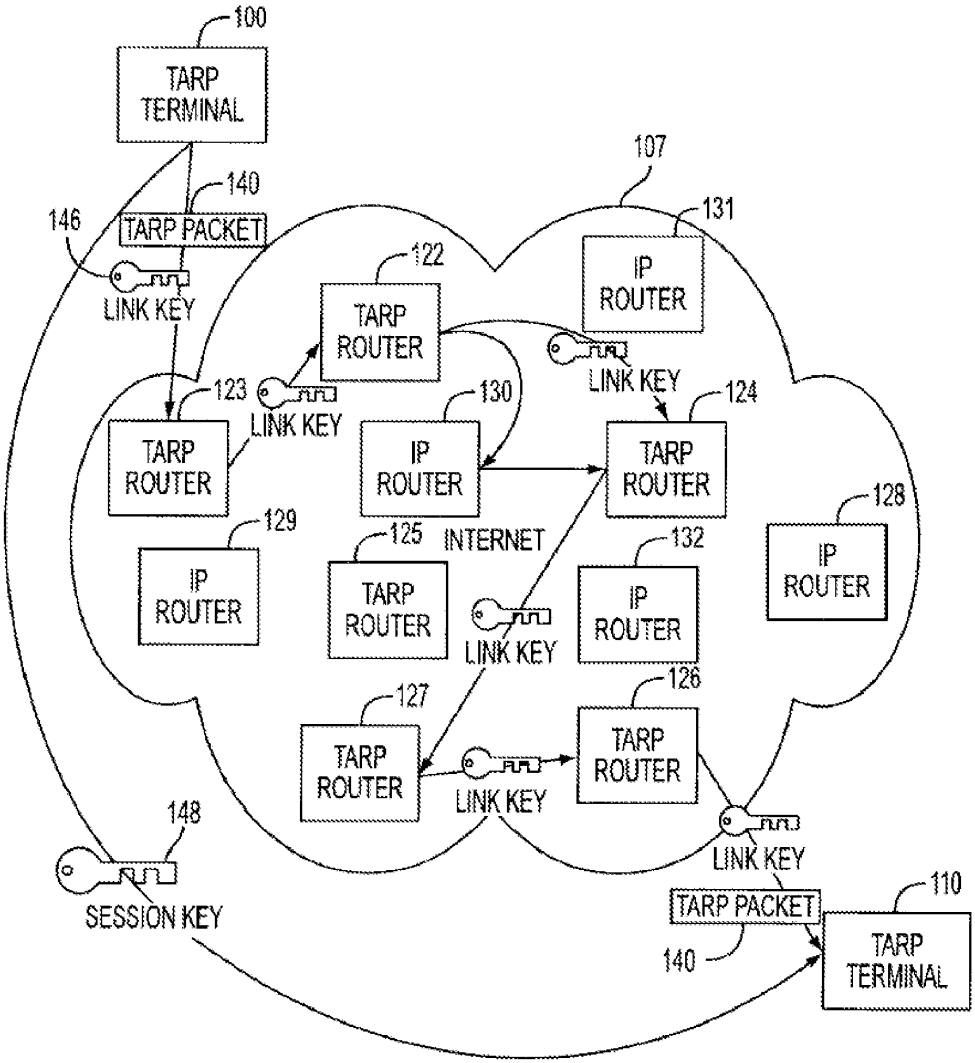

100
TARP TERMINAL
140
TARP PACKET
146
LINK KEY
107
131
IP ROUTER
122
TARP ROUTER
LINK KEY
123
LINK KEY
130
IP ROUTER
124
TARP ROUTER
TARP ROUTER
129
IP ROUTER
125
INTERNET
TARP ROUTER
LINK KEY
132
IP ROUTER
128
IP ROUTER
127
TARP ROUTER
LINK KEY
126
TARP ROUTER
LINK KEY
148
SESSION KEY
TARP PACKET
140
110
TARP TERMINAL

FIG. 2

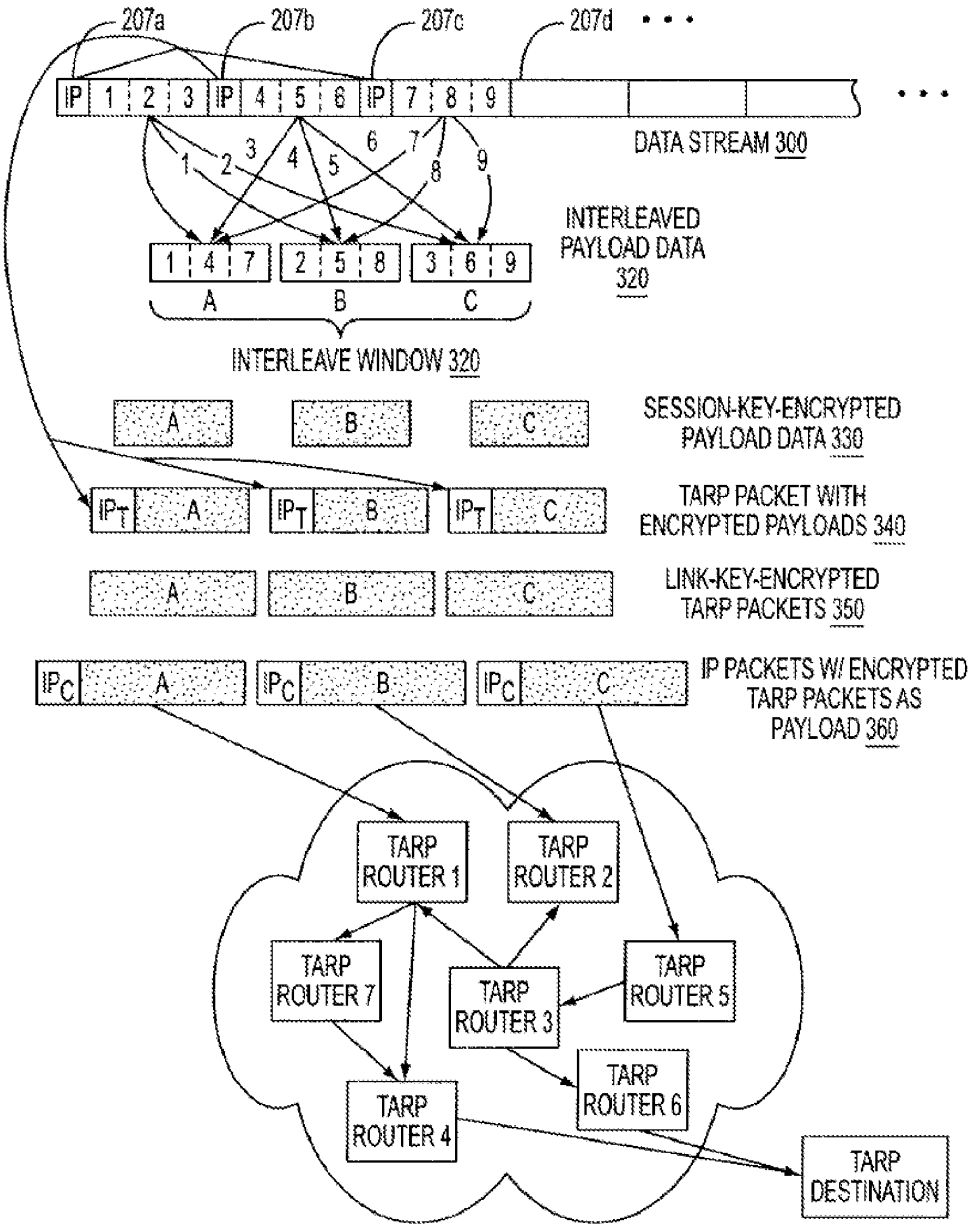

207a
207b
207c
207d
IP 1 2 3 IP 4 5 6 IP 7 8 9
DATA STREAM 300
1 2 3 4 5 6 7 8 9
INTERLEAVED PAYLOAD DATA 320
1 4 7 2 5 8 3 6 9
A B C
INTERLEAVE WINDOW 320
A B C
SESSION-KEY-ENCRYPTED PAYLOAD DATA 330
IPT A IPT B IPT C
TARP PACKET WITH ENCRYPTED PAYLOADS 340
A B C
LINK-KEY-ENCRYPTED TARP PACKETS 350
IPC A IPC B IPC C
IP PACKETS W/ ENCRYPTED TARP PACKETS AS PAYLOAD 360
TARP ROUTER 1
TARP ROUTER 2
TARP ROUTER 7
TARP ROUTER 3
TARP ROUTER 5
TARP ROUTER 6
TARP ROUTER 4
TARP DESTINATION

FIG. 3A

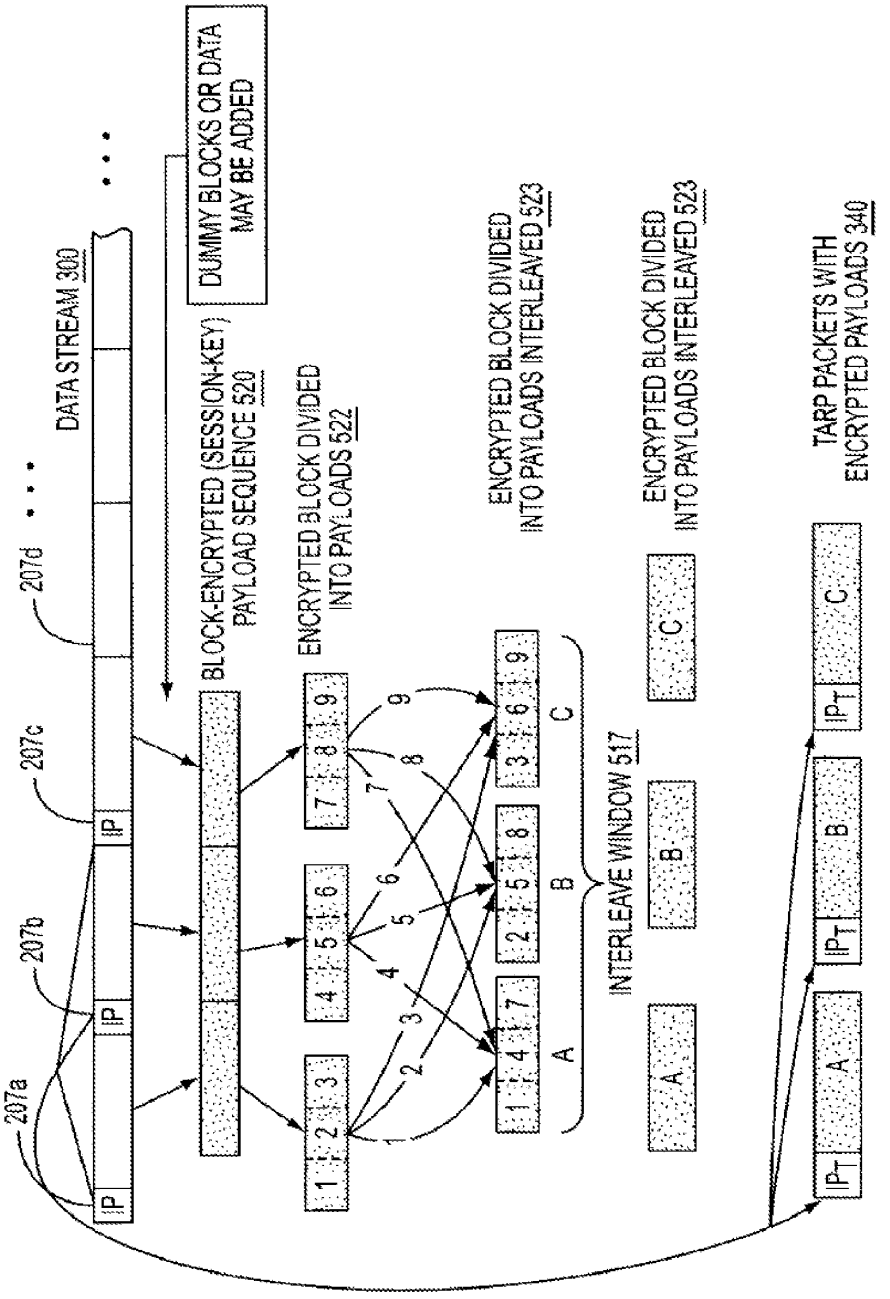

DATA STREAM 300
207a
207b
207c
207d
IP
BLOCK-ENCRYPTED (SESSION-KEY) PAYLOAD SEQUENCE 520
DUMMY BLOCKS OR DATA MAY BE ADDED
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS 522
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED 523
INTERLEAVE WINDOW 517
A
B
C
TARP PACKETS WITH ENCRYPTED PAYLOADS 340
IPT

FIG. 3B

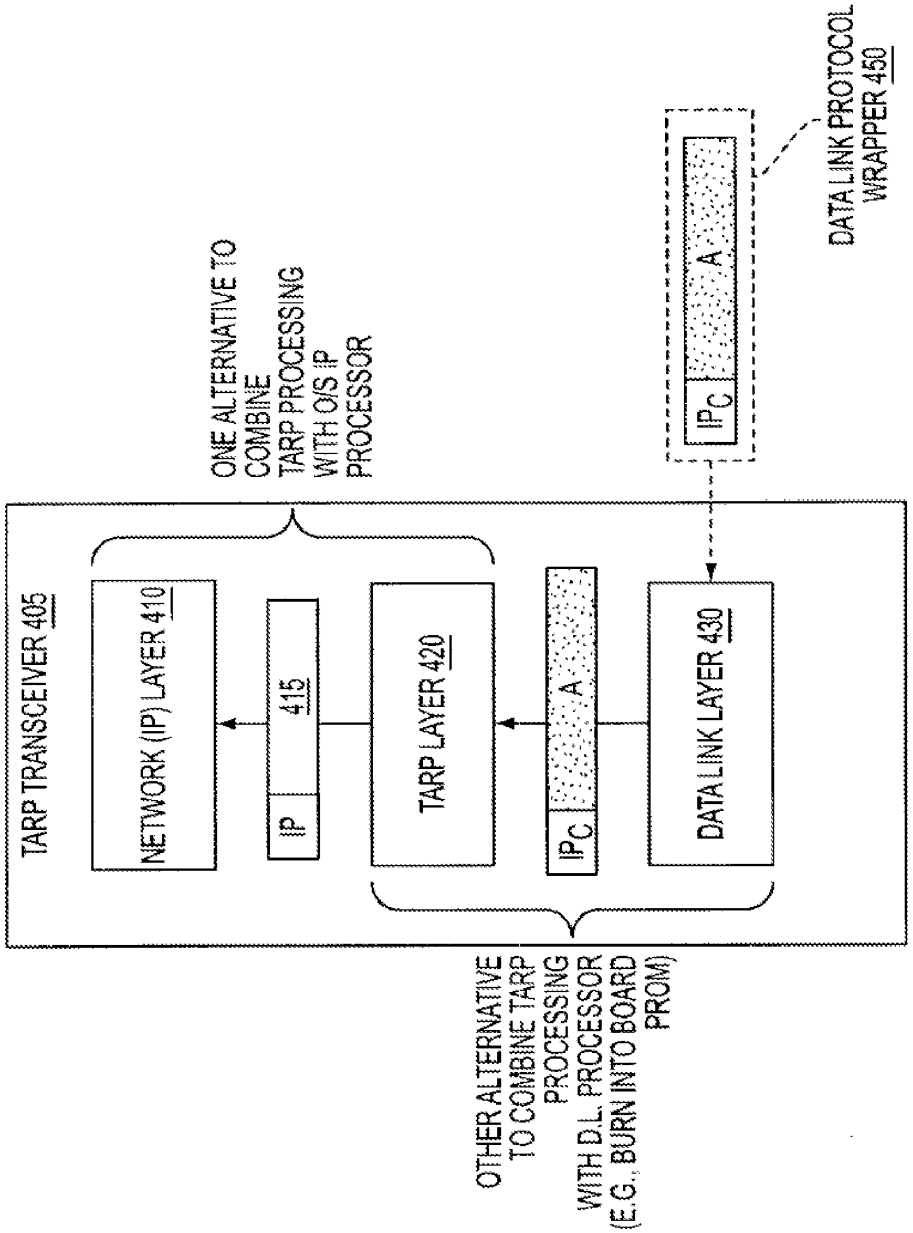

TARP TRANSCEIVER 405
NETWORK (IP) LAYER 410
IP
415
TARP LAYER 420
IPC
A
DATA LINK LAYER 430
ONE ALTERNATIVE TO COMBINE TARP PROCESSING WITH O/S IP PROCESSOR
OTHER ALTERNATIVE TO COMBINE TARP PROCESSING WITH D.L. PROCESSOR (E.G., BURN INTO BOARD PROM)
IPC
A
DATA LINK PROTOCOL WRAPPER 450

FIG. 4

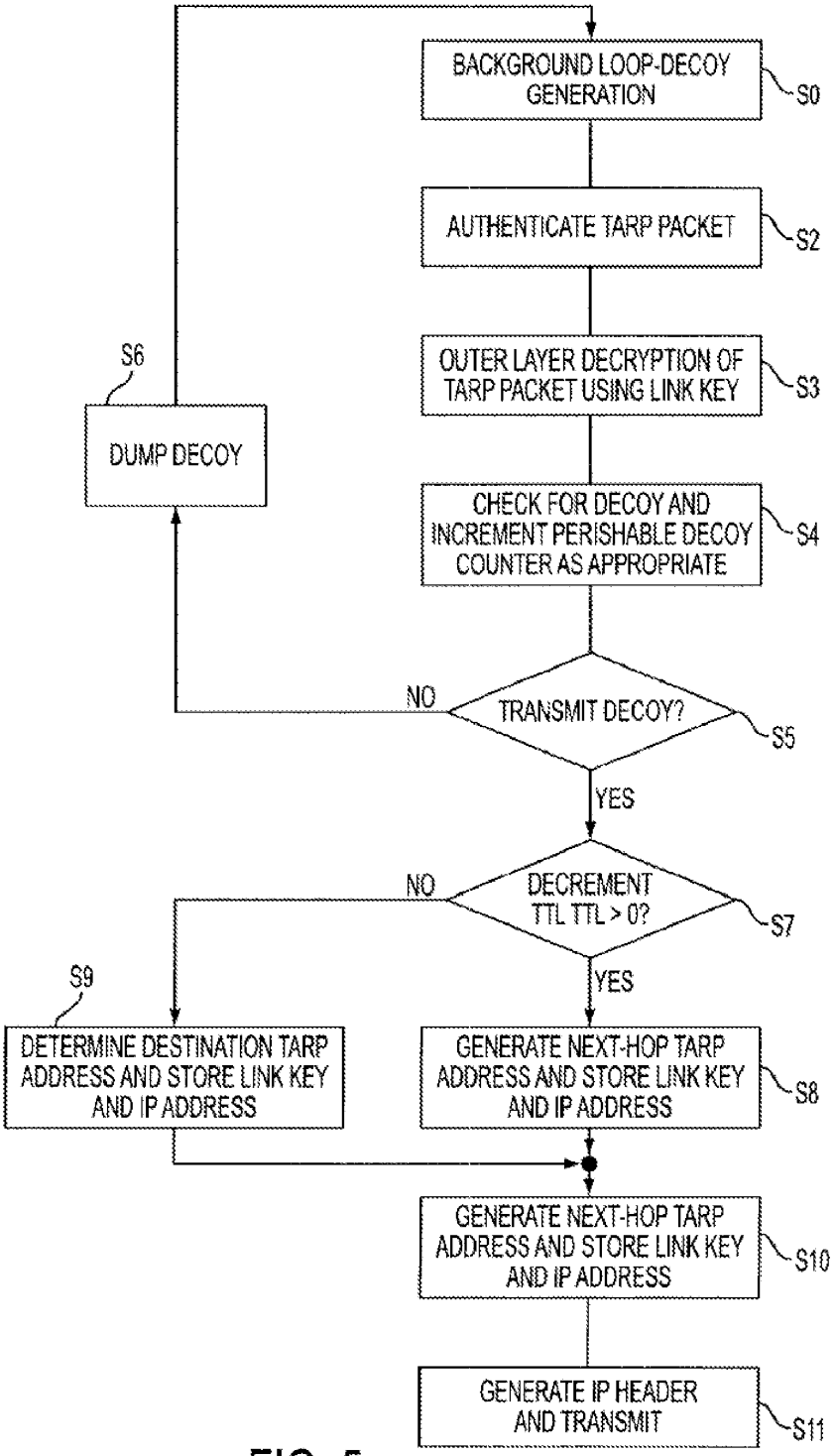

BACKGROUND LOOP-DECOY GENERATION
S0
AUTHENTICATE TARP PACKET
S2
OUTER LAYER DECRYPTION OF TARP PACKET USING LINK KEY
S3
CHECK FOR DECOY AND INCREMENT PERISHABLE DECOY COUNTER AS APPROPRIATE
S4
TRANSMIT DECOY?
S5
NO
YES
S6
DUMP DECOY
DECREMENT TTL TTL > 0?
S7
NO
YES
S9
DETERMINE DESTINATION TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S8
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S10
GENERATE IP HEADER AND TRANSMIT
S11

FIG. 5

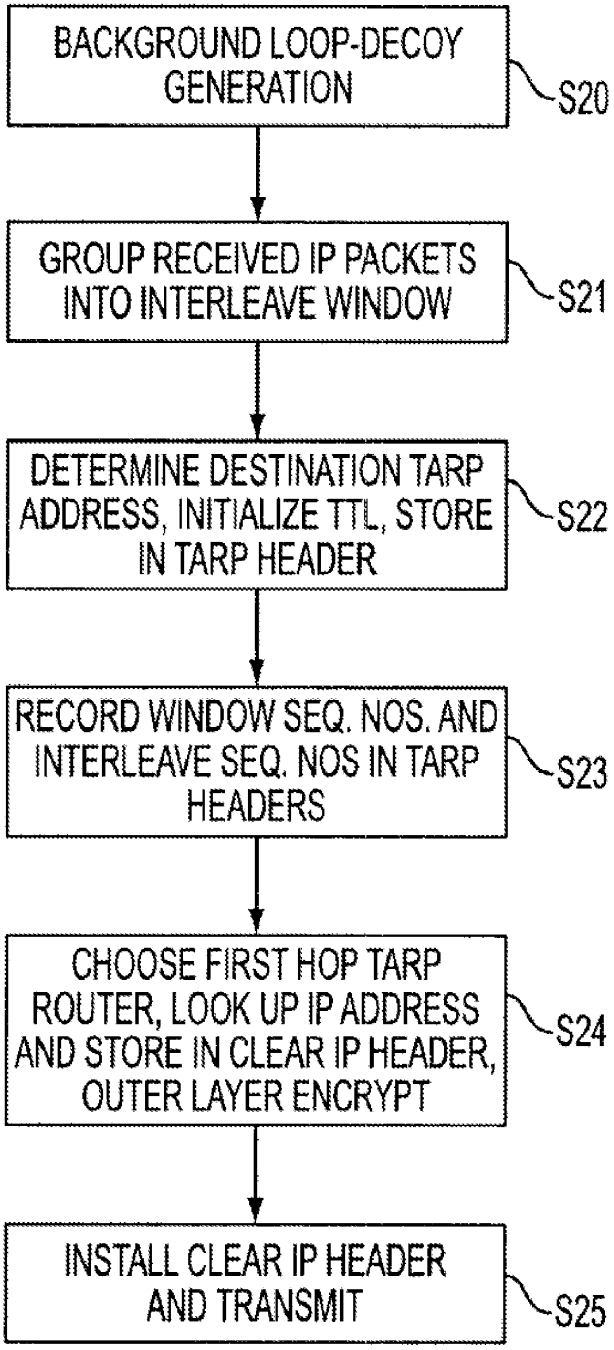

BACKGROUND LOOP-DECOY GENERATION
S20
GROUP RECEIVED IP PACKETS INTO INTERLEAVE WINDOW
S21
DETERMINE DESTINATION TARP ADDRESS, INITIALIZE TTL, STORE IN TARP HEADER
S22
RECORD WINDOW SEQ. NOS. AND INTERLEAVE SEQ. NOS IN TARP HEADERS
S23
CHOOSE FIRST HOP TARP ROUTER, LOOK UP IP ADDRESS AND STORE IN CLEAR IP HEADER, OUTER LAYER ENCRYPT
S24
INSTALL CLEAR IP HEADER AND TRANSMIT
S25

FIG. 6

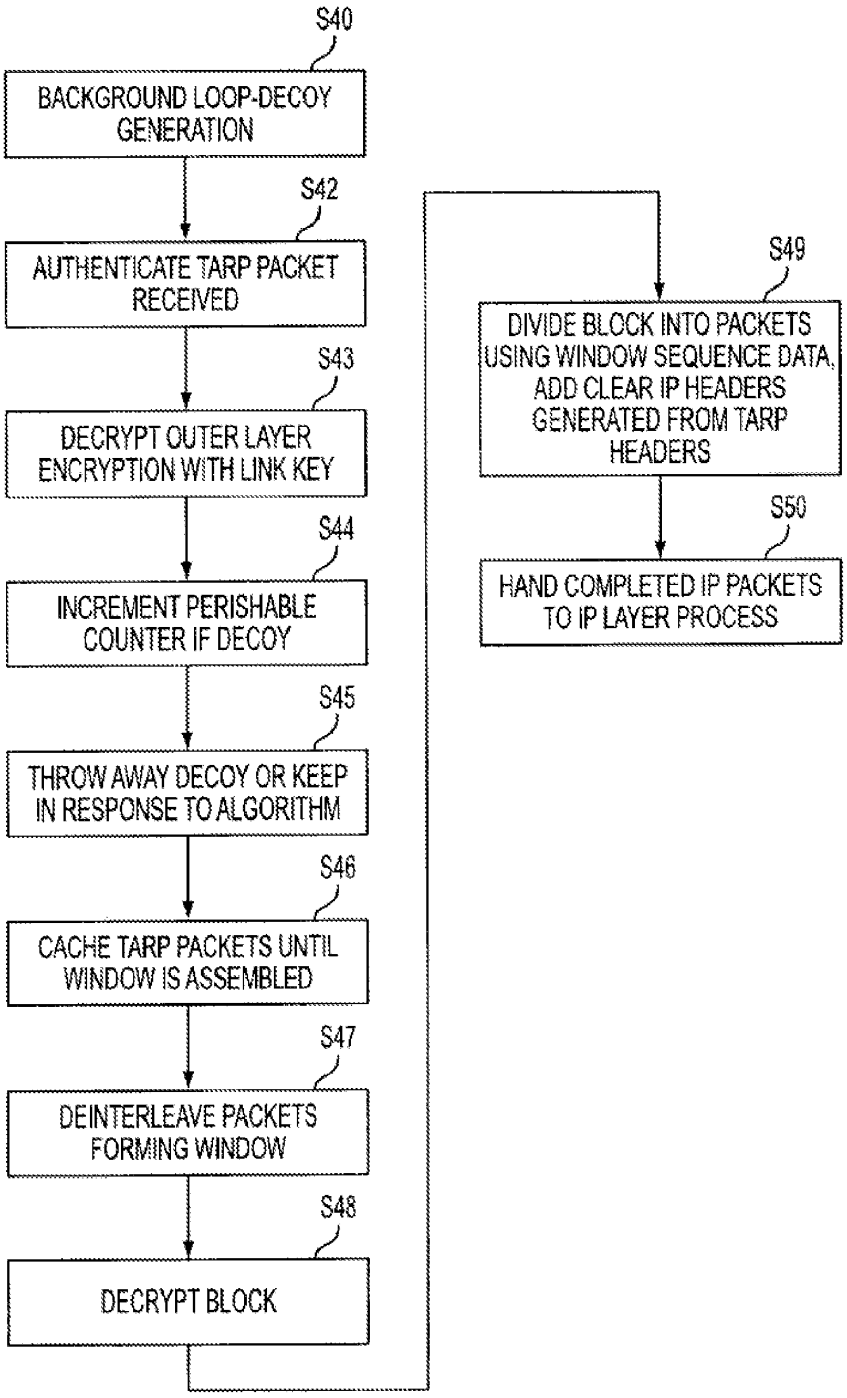

S40
BACKGROUND LOOP-DECOY GENERATION
S42
AUTHENTICATE TARP PACKET RECEIVED
S43
DECRYPT OUTER LAYER ENCRYPTION WITH LINK KEY
S44
INCREMENT PERISHABLE COUNTER IF DECOY
S45
THROW AWAY DECOY OR KEEP IN RESPONSE TO ALGORITHM
S46
CACHE TARP PACKETS UNTIL WINDOW IS ASSEMBLED
S47
DEINTERLEAVE PACKETS FORMING WINDOW
S48
DECRYPT BLOCK
S49
DIVIDE BLOCK INTO PACKETS USING WINDOW SEQUENCE DATA, ADD CLEAR IP HEADERS GENERATED FROM TARP HEADERS
S50
HAND COMPLETED IP PACKETS TO IP LAYER PROCESS

FIG. 7

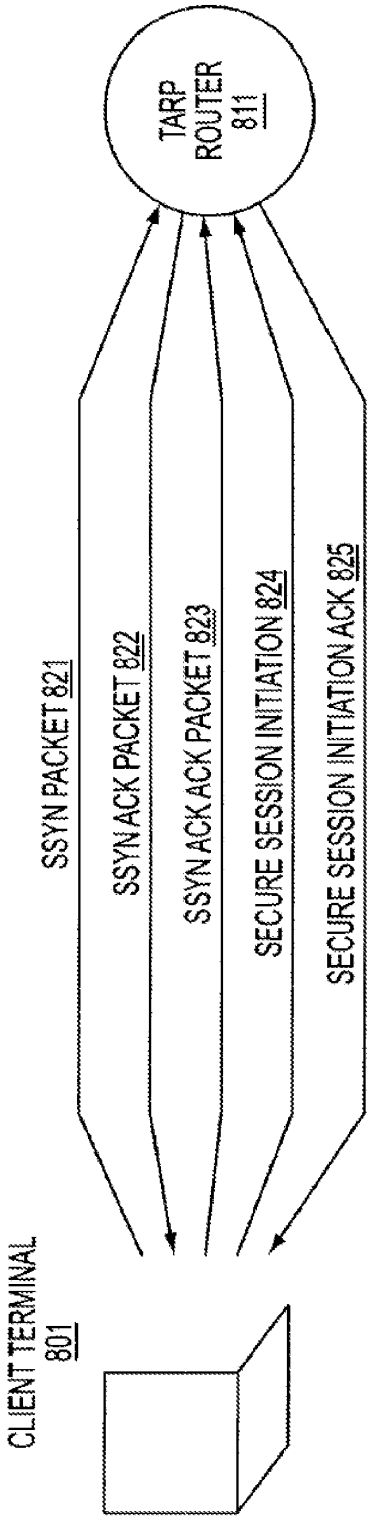

TARP ROUTER 811
SSYN PACKET 821
SSYN ACK PACKET 822
SSYN ACK ACK PACKET 823
SECURE SESSION INITIATION 824
SECURE SESSION INITIATION ACK 825
CLIENT TERMINAL 801

FIG. 8

CLIENT 1
901
TARP
ROUTER
911
TRANSMIT TABLE 921
131.218.204.98 , 131.218.204.65
131.218.204.221 , 131.218.204.97
131.218.204.139 , 131.218.204.186
131.218.204.12 , 131.218.204.55
RECEIVE TABLE 924
131.218.204.98 , 131.218.204.65
131.218.204.221 , 131.218.204.97
131.218.204.139 , 131.218.204.186
131.218.204.12 , 131.218.204.55
RECEIVE TABLE 922
131.218.204.161 , 131.218.204.89
131.218.204.66 , 131.218.204.212
131.218.204.201 , 131.218.204.127
131.218.204.119 , 131.218.204.49
TRANSMIT TABLE 923
131.218.204.161 , 131.218.204.89
131.218.204.66 , 131.218.204.212
131.218.204.201 , 131.218.204.127
131.218.204.119 , 131.218.204.49

FIG. 9

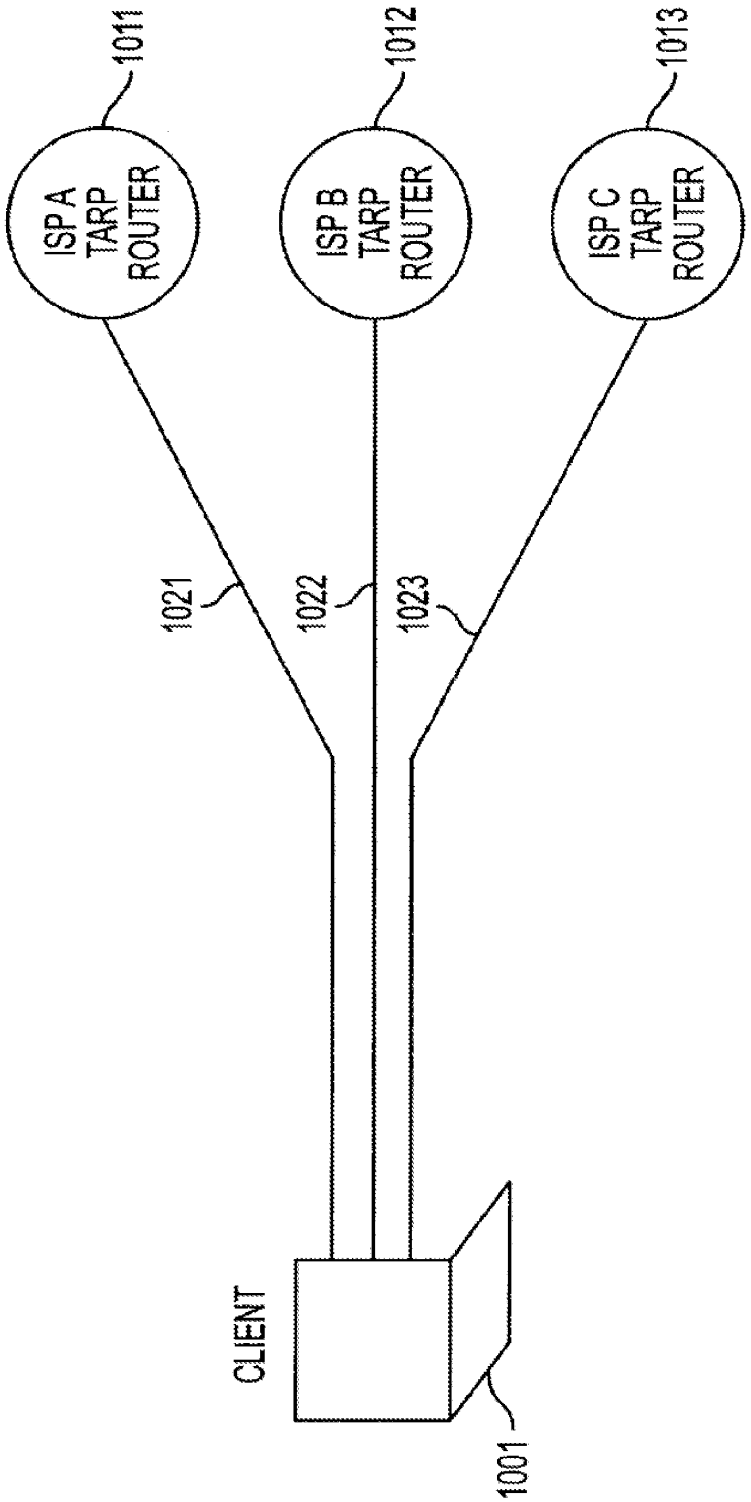

1011
1012
1013
ISP A TARP ROUTER
ISP B TARP ROUTER
ISP C TARP ROUTER
1021
1022
1023
CLIENT
1001

FIG. 10

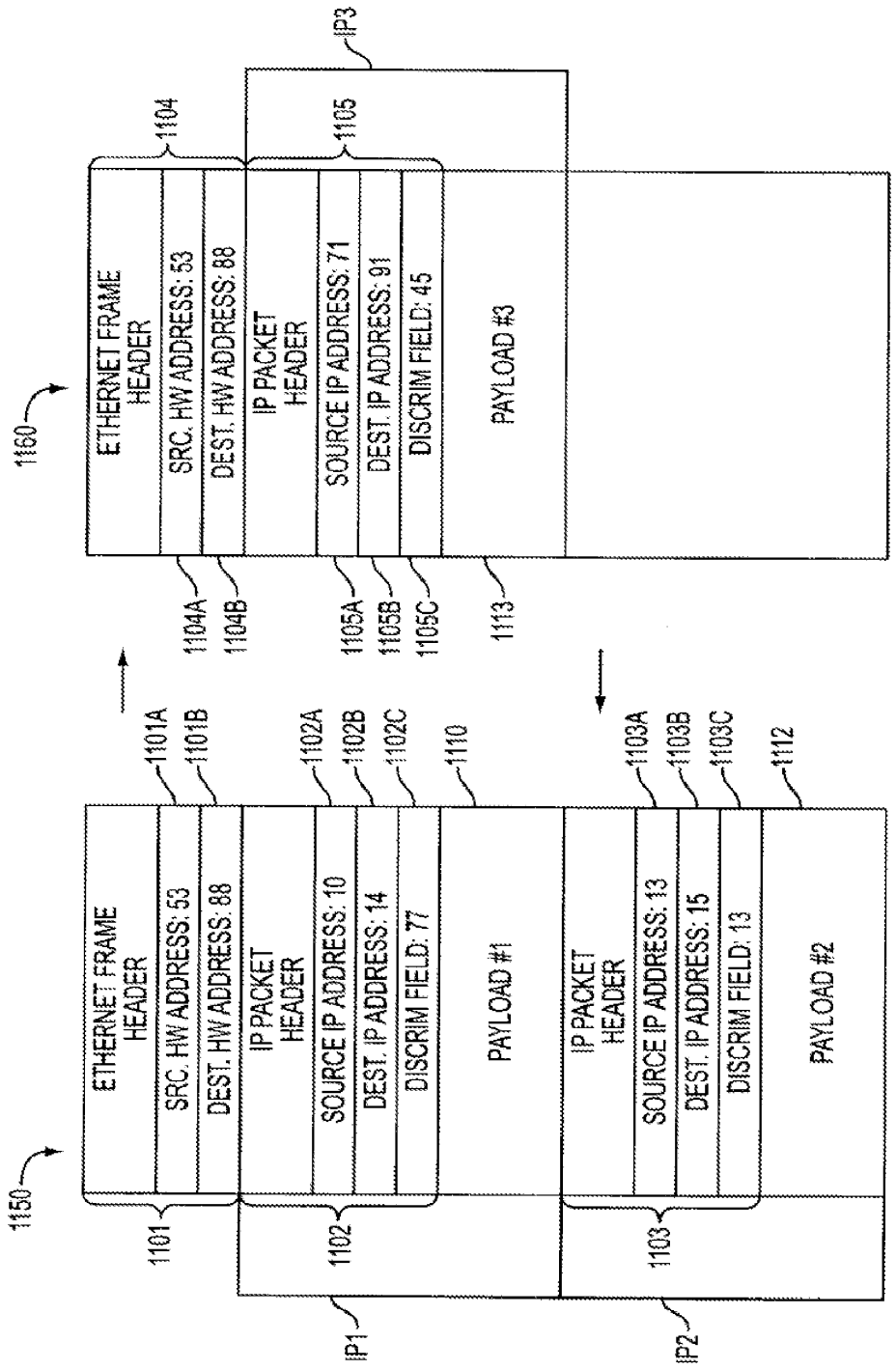

1150
1160
1101
1102
1103
1104
1105
IP1
IP2
IP3
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 10
DEST. IP ADDRESS: 14
DISCRIM FIELD: 77
PAYLOAD #1
IP PACKET HEADER
SOURCE IP ADDRESS: 13
DEST. IP ADDRESS: 15
DISCRIM FIELD: 13
PAYLOAD #2
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 71
DEST. IP ADDRESS: 91
DISCRIM FIELD: 45
PAYLOAD #3
1101A
1101B
1102A
1102B
1102C
1110
1103A
1103B
1103C
1112
1104A
1104B
1105A
1105B
1105C
1113

FIG. 11

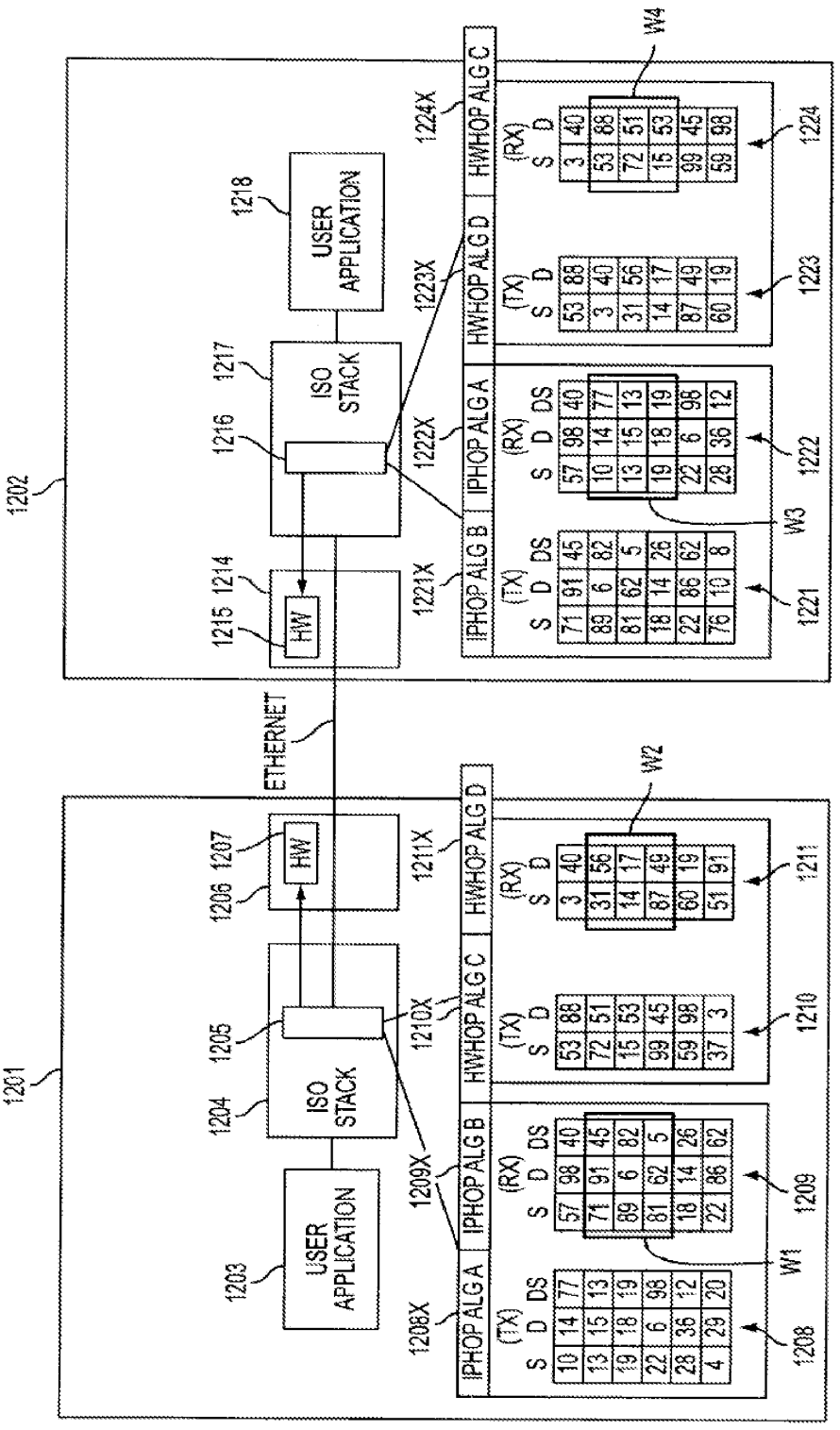

1201
1202
1203
USER APPLICATION
1204
ISO STACK
1205
1206
1207
HW
ETHERNET
1208
1209
1210
1211
1214
1215
1216
1217
1218
1221
1222
1223
1224
IPHOP ALG A
IPHOP ALG B
HWHOP ALG C
HWHOP ALG D
1206X
1209X
1210X
1211X
1221X
1222X
1223X
1224X
W1
W2
W3
W4
(TX)
(RX)
S
D
DS

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

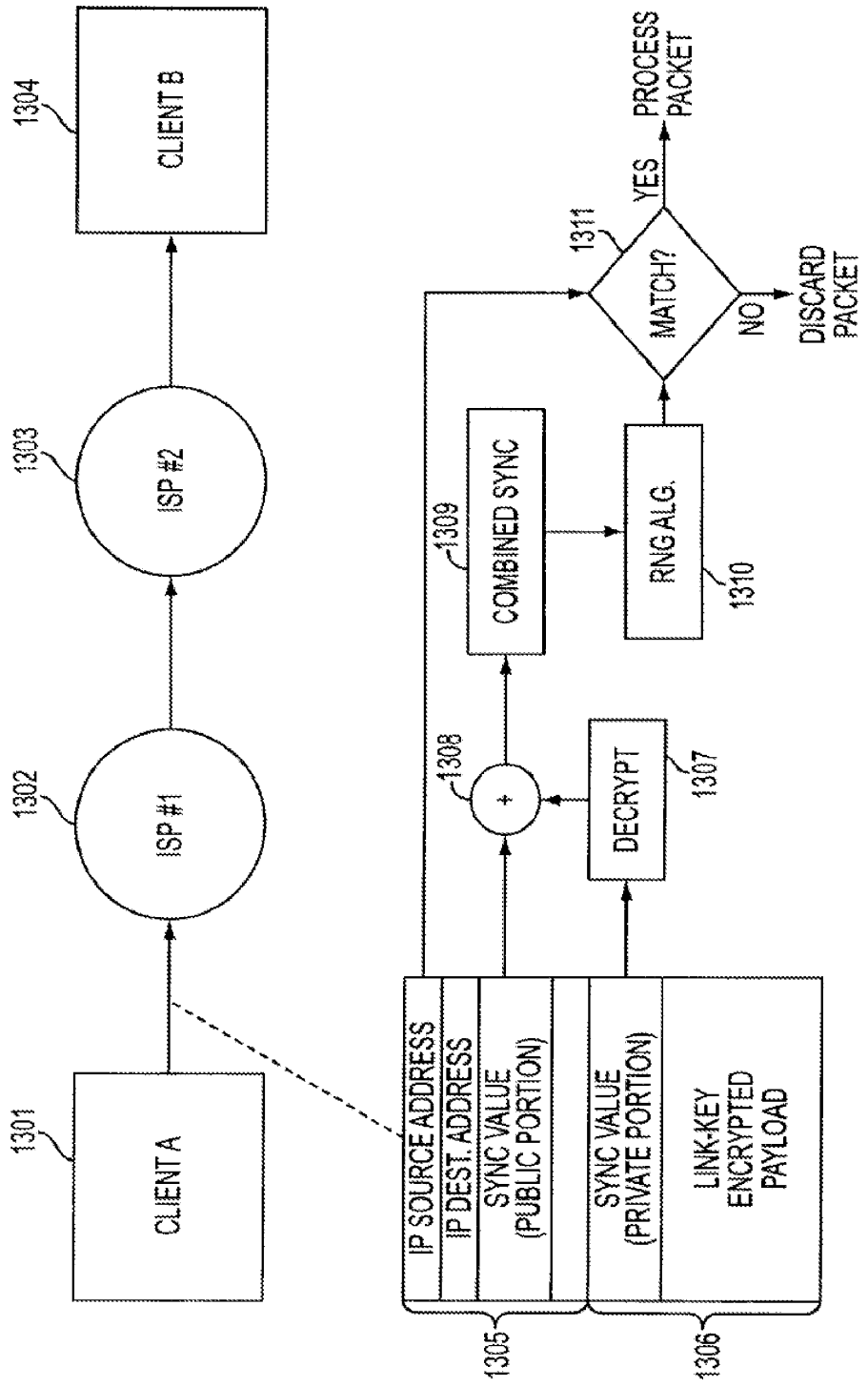

1301
CLIENT A
1302
ISP #1
1303
ISP #2
1304
CLIENT B
IP SOURCE ADDRESS
IP DEST. ADDRESS
SYNC VALUE (PUBLIC PORTION)
SYNC VALUE (PRIVATE PORTION)
LINK-KEY ENCRYPTED PAYLOAD
1305
1306
DECRYPT
1307
1308
COMBINED SYNC
1309
RNG ALG.
1310
1311
MATCH?
YES
PROCESS PACKET
NO
DISCARD PACKET

FIG. 13

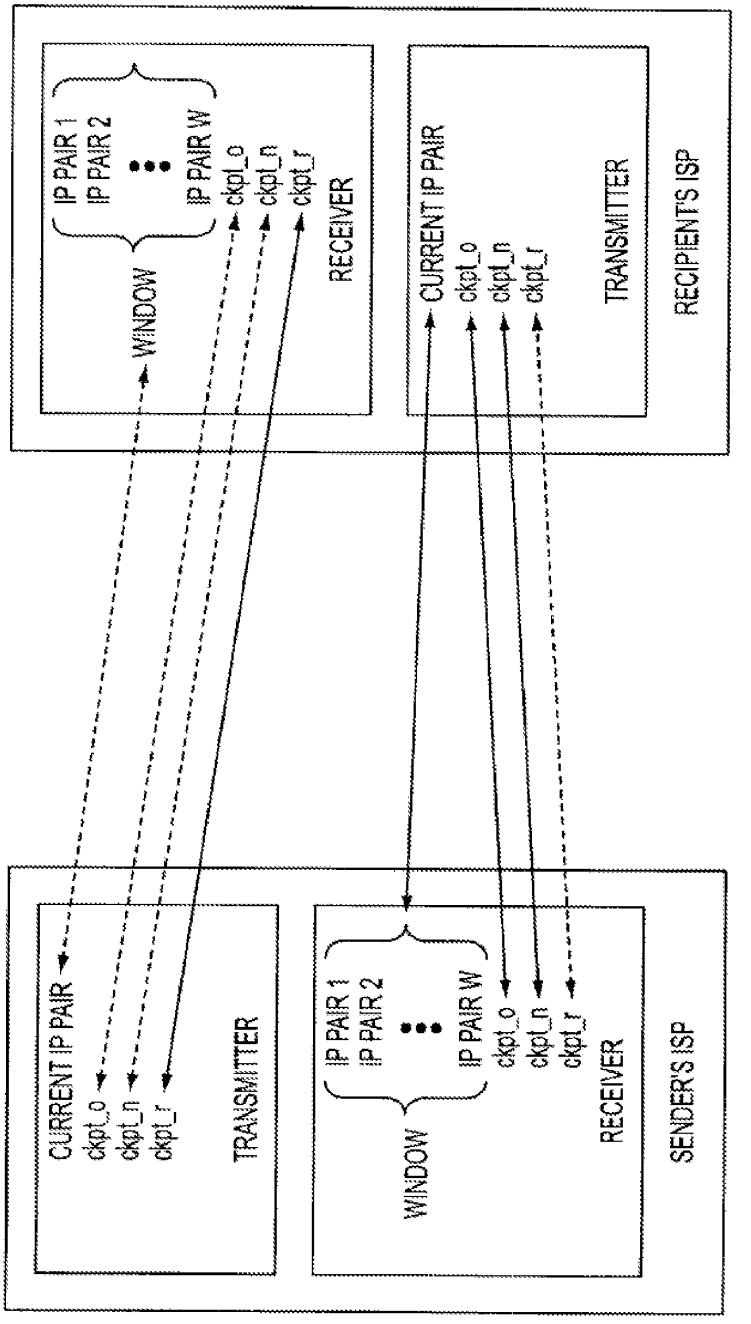

RECIPIENT'S ISP
TRANSMITTER
CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
RECEIVER
IP PAIR 1
IP PAIR 2
IP PAIR W
WINDOW
ckpt_o
ckpt_n
ckpt_r
SENDER'S ISP
TRANSMITTER
CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
RECEIVER
IP PAIR 1
IP PAIR 2
IP PAIR W
WINDOW
ckpt_o
ckpt_n
ckpt_r
KEPT IN SYNC FOR SENDER TO RECIPIENT SYNCHRONIZER
KEPT IN SYNC FOR RECIPIENT TO SENDER SYNCHRONIZER

FIG. 14

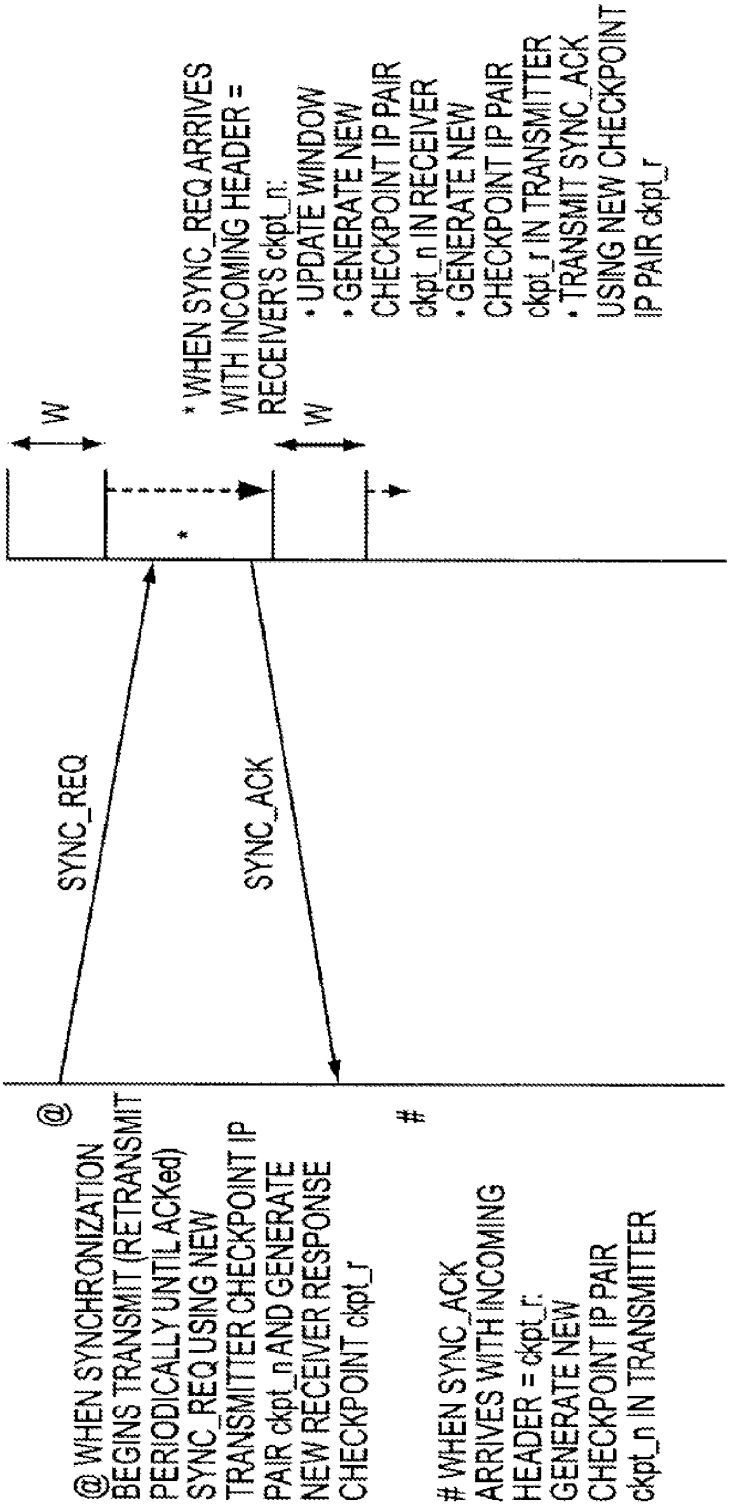

@ WHEN SYNCHRONIZATION BEGINS TRANSMIT (RETRANSMIT PERIODICALLY UNTIL ACKed) SYNC_REQ USING NEW TRANSMITTER CHECKPOINT IP PAIR ckpt_n AND GENERATE NEW RECEIVER RESPONSE CHECKPOINT ckpt_r
WHEN SYNC_ACK ARRIVES WITH INCOMING HEADER = ckpt_r: GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN TRANSMITTER
SYNC_REQ
SYNC_ACK
W
* WHEN SYNC_REQ ARRIVES WITH INCOMING HEADER = RECEIVER'S ckpt_n:
• UPDATE WINDOW
• GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN RECEIVER
• GENERATE NEW CHECKPOINT IP PAIR ckpt_r IN TRANSMITTER
• TRANSMIT SYNC_ACK USING NEW CHECKPOINT IP PAIR ckpt_r

FIG. 15

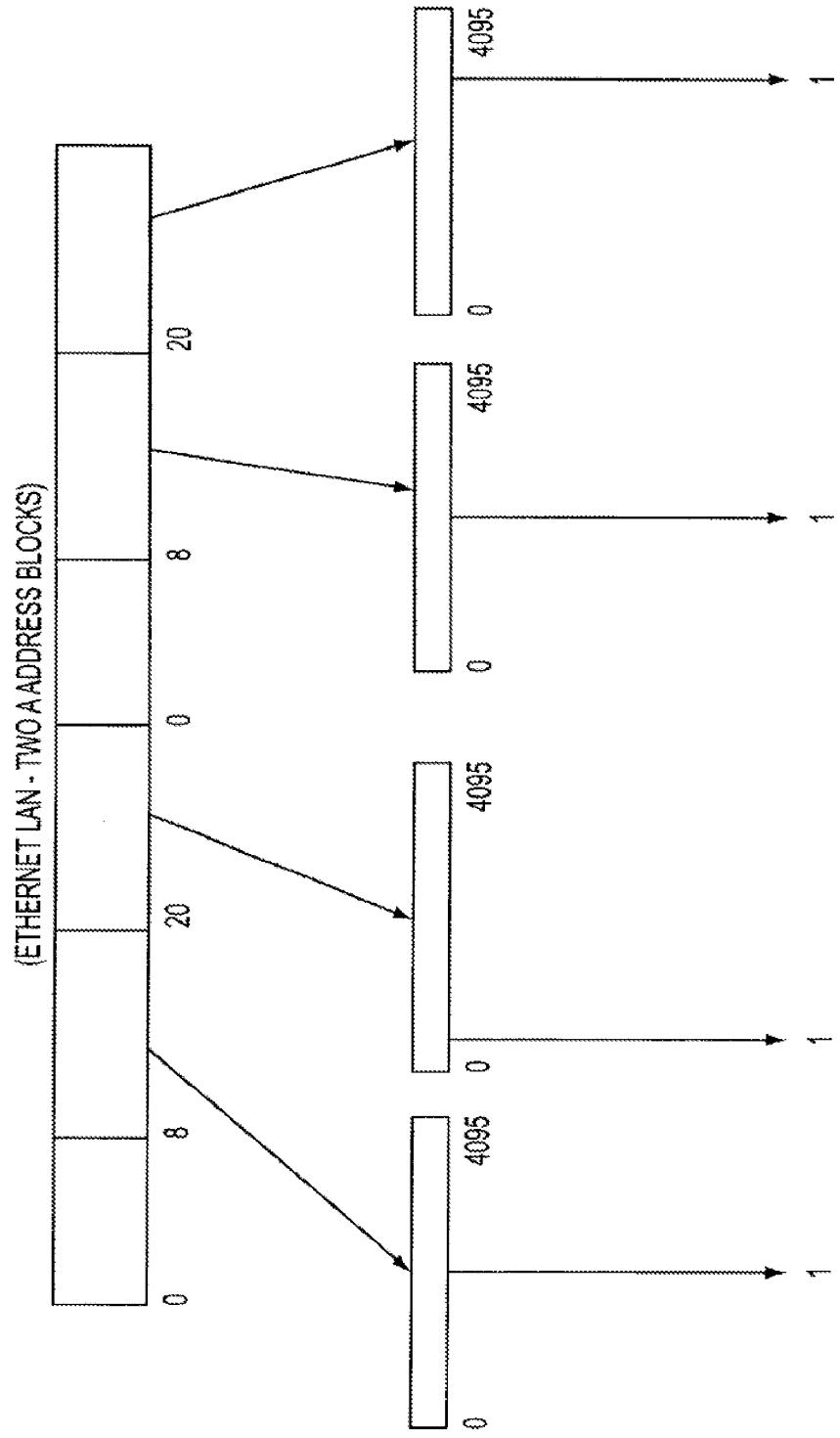

(ETHERNET LAN - TWO A ADDRESS BLOCKS)
20
8
0
20
8
0
4095
0
4095
0
4095
0
4095
0
1
1
1
1

FIG. 16

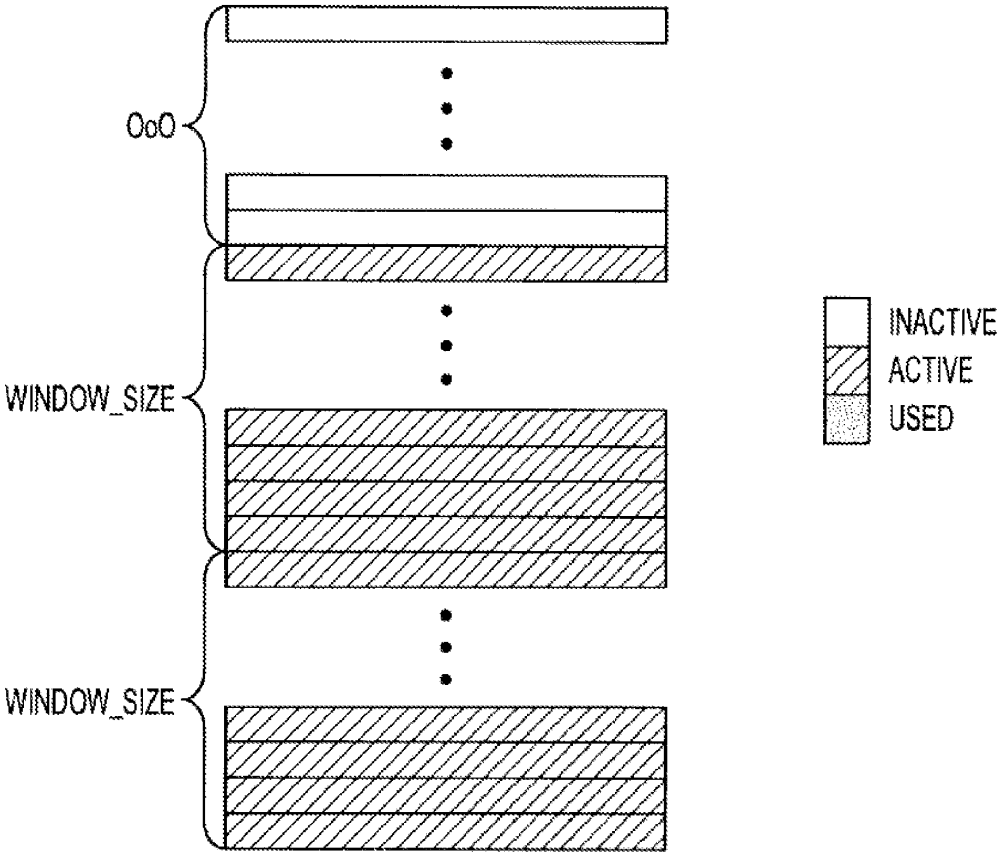


FIG. 17

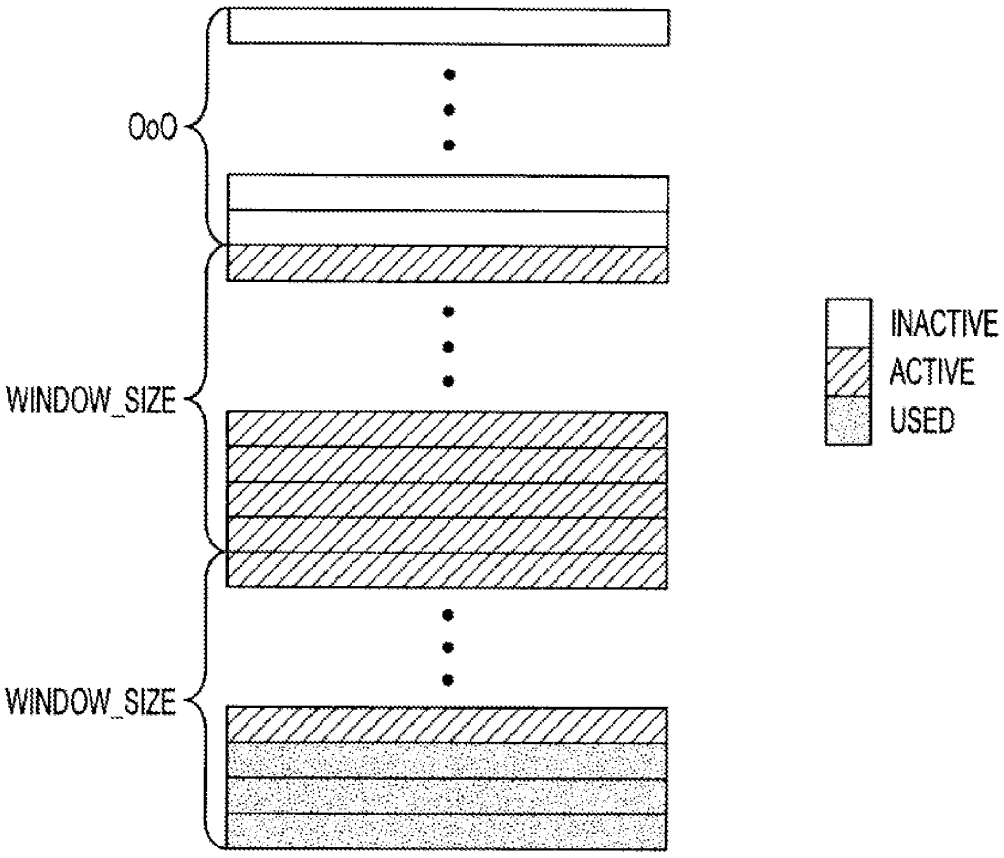


FIG. 18

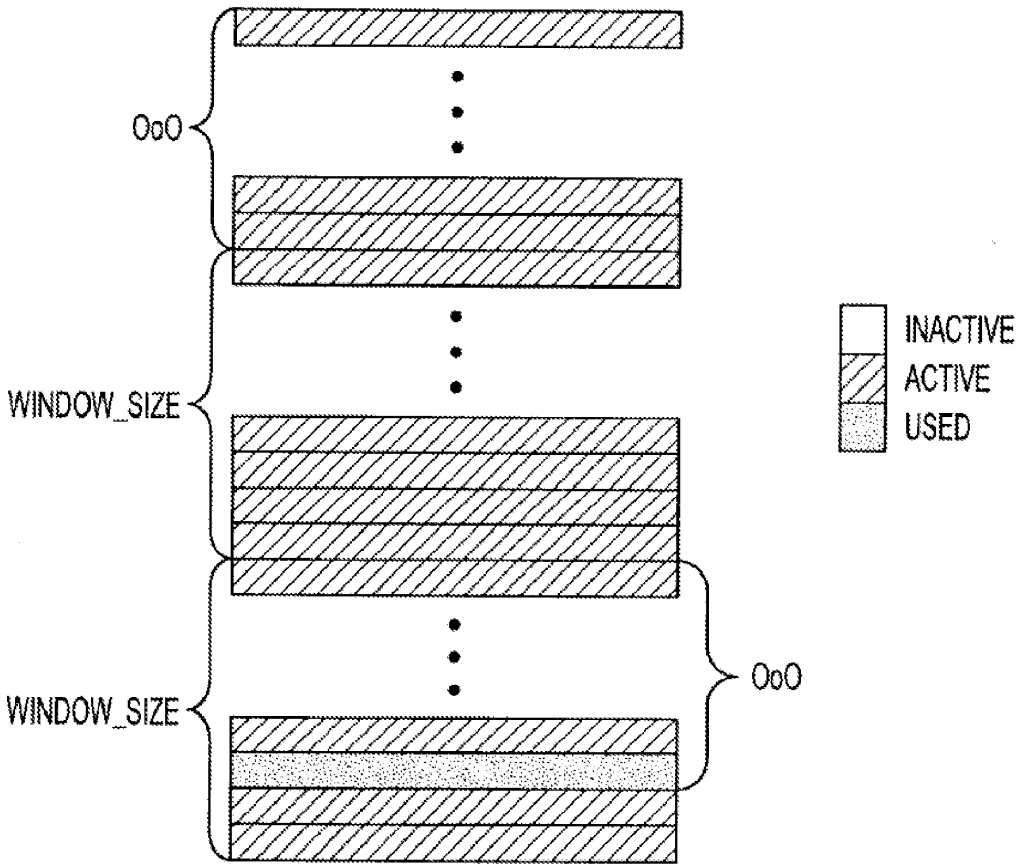


FIG. 19

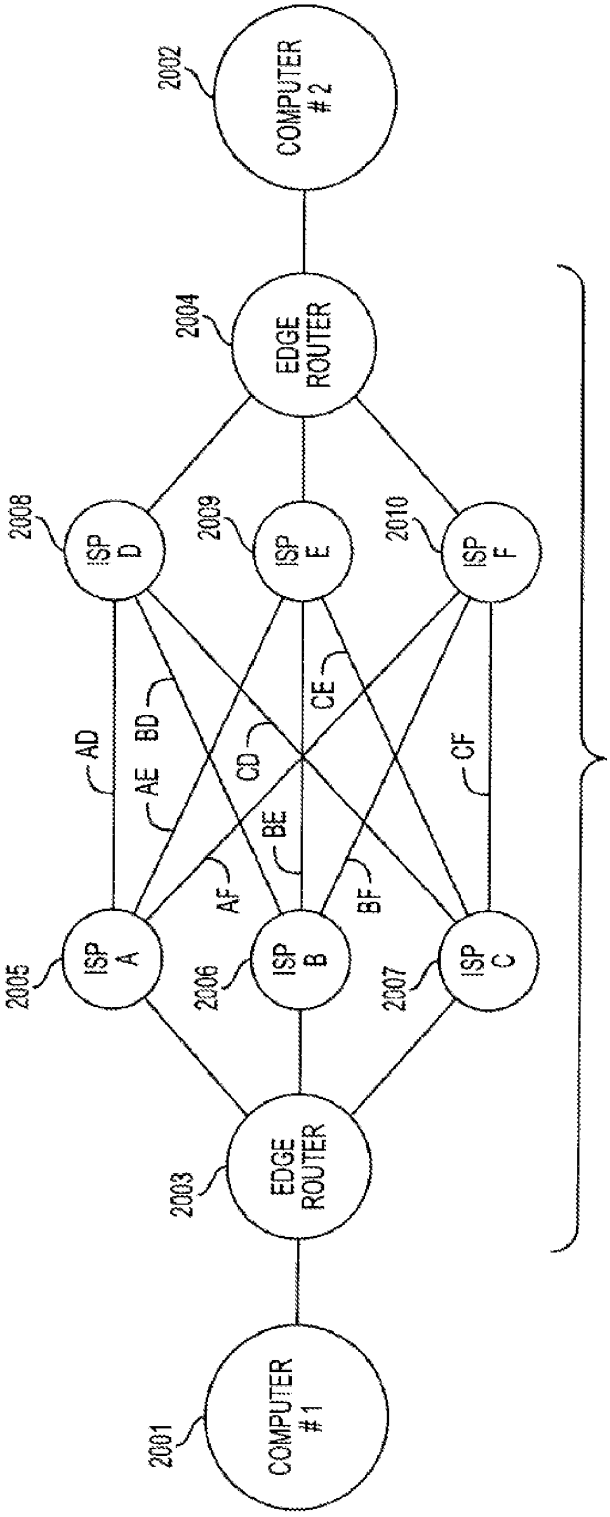

2002
COMPUTER #2
2004
EDGE ROUTER
2008
ISP D
2009
ISP E
2010
ISP F
AD
AE
BD
CD
CE
CF
AF
BE
BF
2005
ISP A
2006
ISP B
2007
ISP C
2003
EDGE ROUTER
2001
COMPUTER #1
2011

FIG. 20

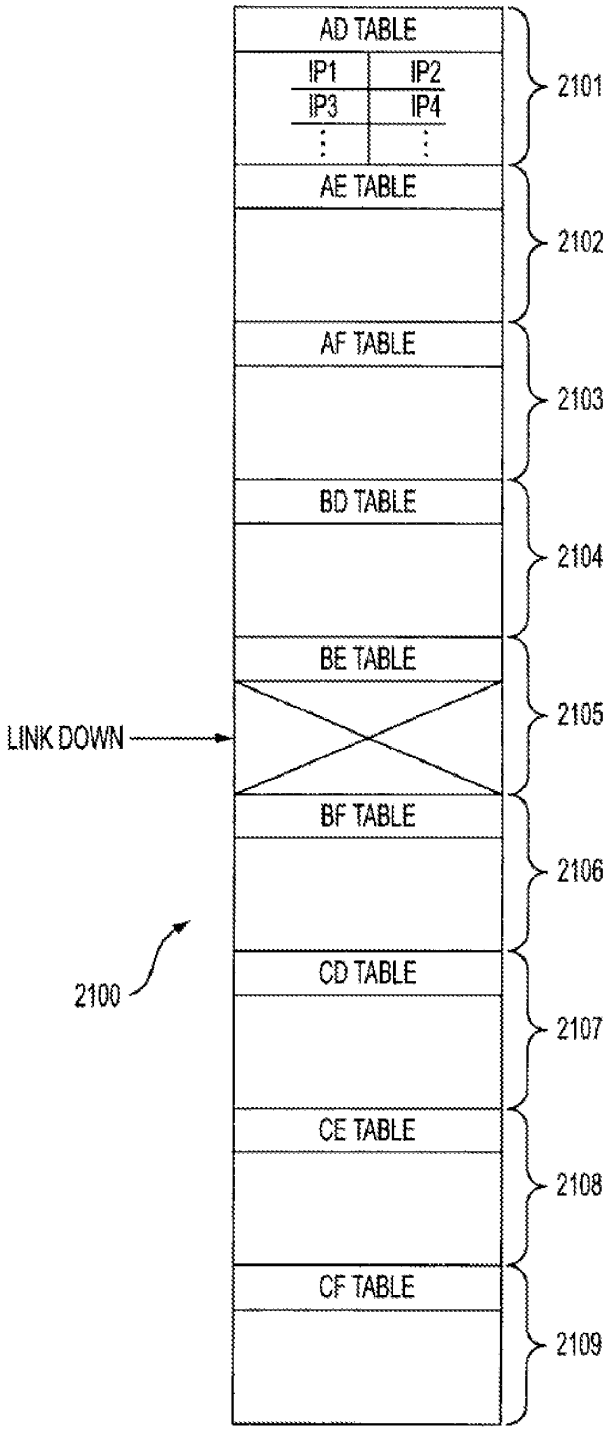

AD TABLE
IP1
IP2
IP3
IP4
2101
AE TABLE
2102
AF TABLE
2103
BD TABLE
2104
BE TABLE
LINK DOWN
2105
BF TABLE
2106
2100
CD TABLE
2107
CE TABLE
2108
CF TABLE
2109

FIG. 21

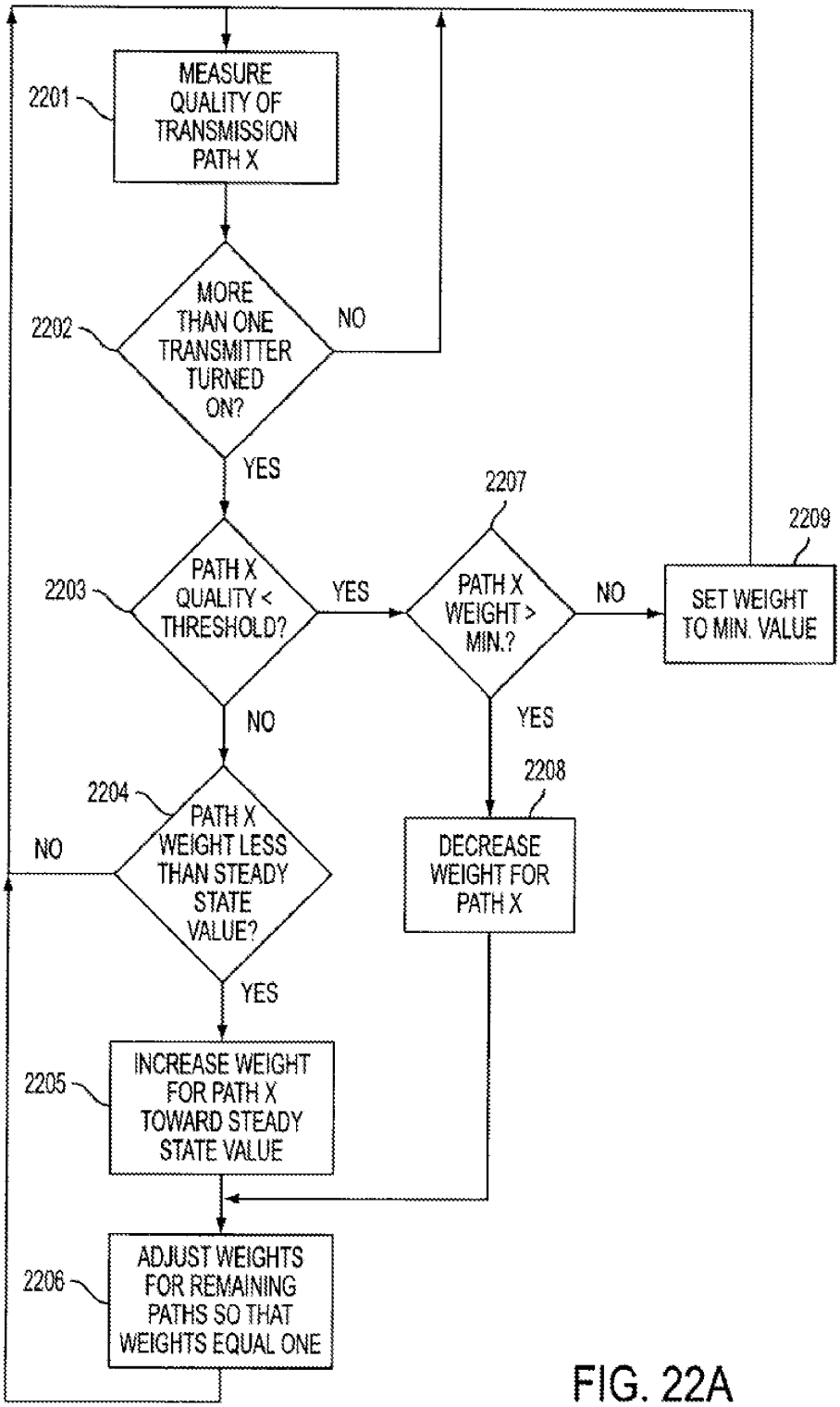

2201
MEASURE QUALITY OF TRANSMISSION PATH X
2202
MORE THAN ONE TRANSMITTER TURNED ON?
NO
YES
2203
PATH X QUALITY < THRESHOLD?
YES
2207
PATH X WEIGHT > MIN.?
NO
2209
SET WEIGHT TO MIN. VALUE
NO
YES
2204
PATH X WEIGHT LESS THAN STEADY STATE VALUE?
NO
YES
2208
DECREASE WEIGHT FOR PATH X
2205
INCREASE WEIGHT FOR PATH X TOWARD STEADY STATE VALUE
2206
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE

FIG. 22A

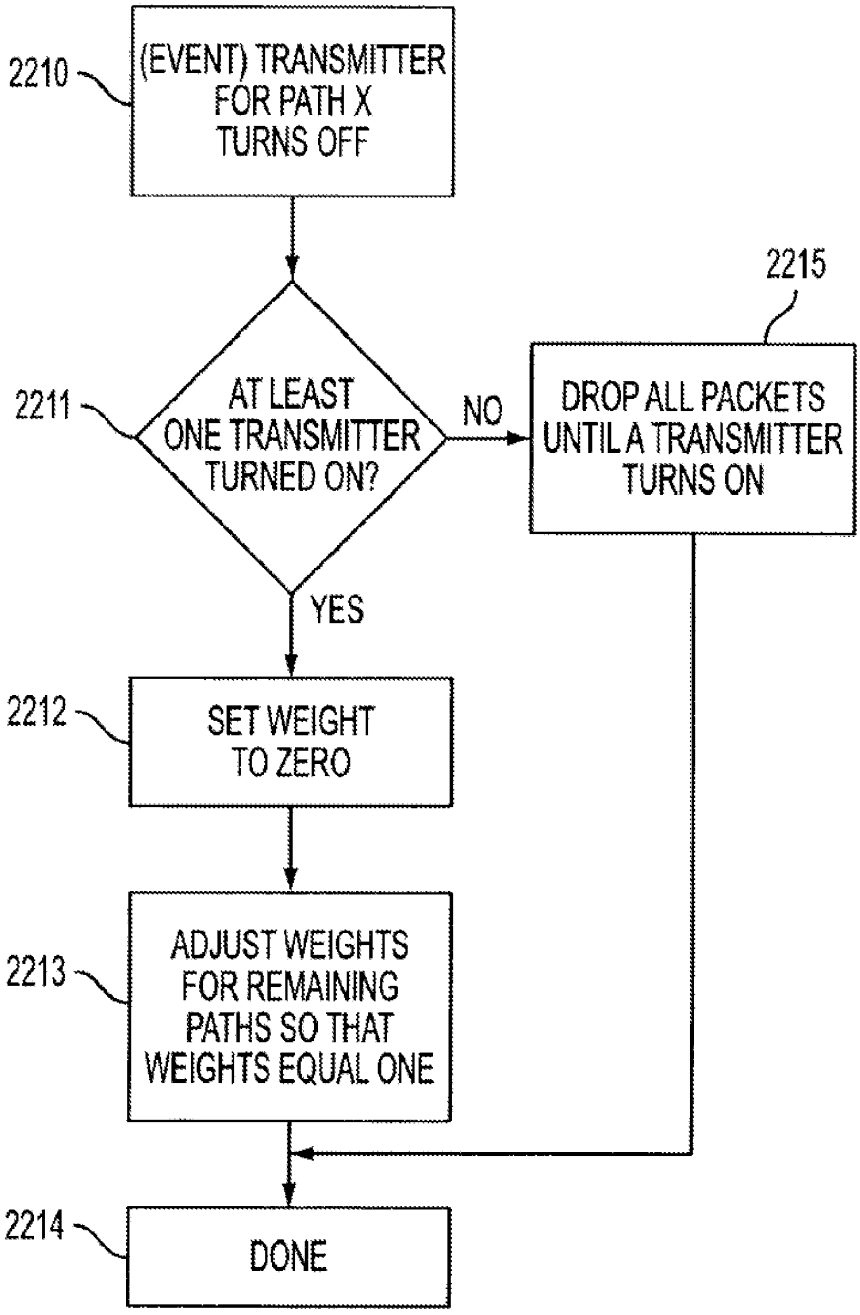

2210
(EVENT) TRANSMITTER FOR PATH X TURNS OFF
2211
AT LEAST ONE TRANSMITTER TURNED ON?
NO
2215
DROP ALL PACKETS UNTIL A TRANSMITTER TURNS ON
YES
2212
SET WEIGHT TO ZERO
2213
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2214
DONE

FIG. 22B

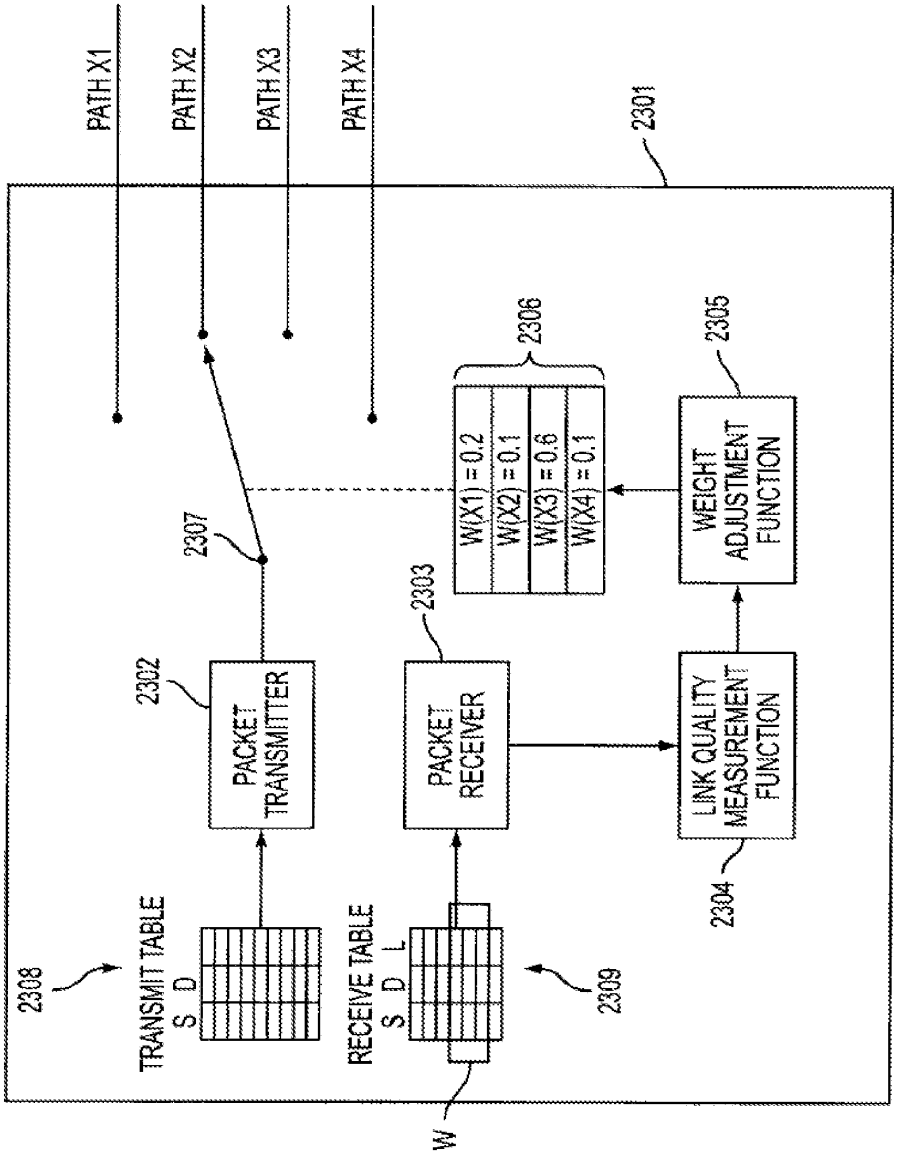

PATH X1
PATH X2
PATH X3
PATH X4
2301
2306
2305
W(X1) = 0.2
W(X2) = 0.1
W(X3) = 0.6
W(X4) = 0.1
WEIGHT ADJUSTMENT FUNCTION
2307
2303
2302
PACKET TRANSMITTER
PACKET RECEIVER
LINK QUALITY MEASUREMENT FUNCTION
2304
2308
TRANSMIT TABLE
S D
RECEIVE TABLE
S D L
2309
W

FIG. 23

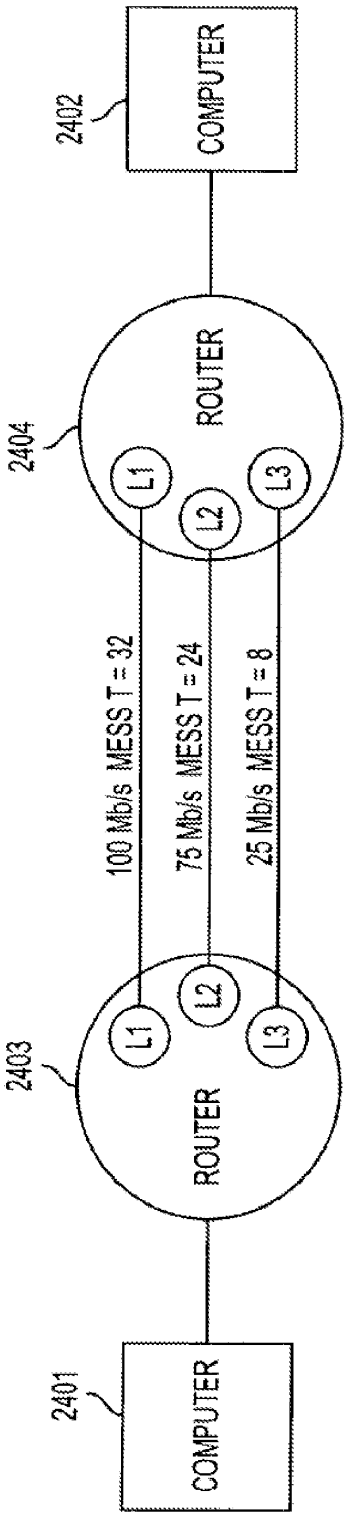

2402
COMPUTER
2404
ROUTER
L1
L2
L3
100 Mb/s MESS T = 32
75 Mb/s MESS T = 24
25 Mb/s MESS T = 8
2403
ROUTER
L1
L2
L3
2401
COMPUTER

FIG. 24

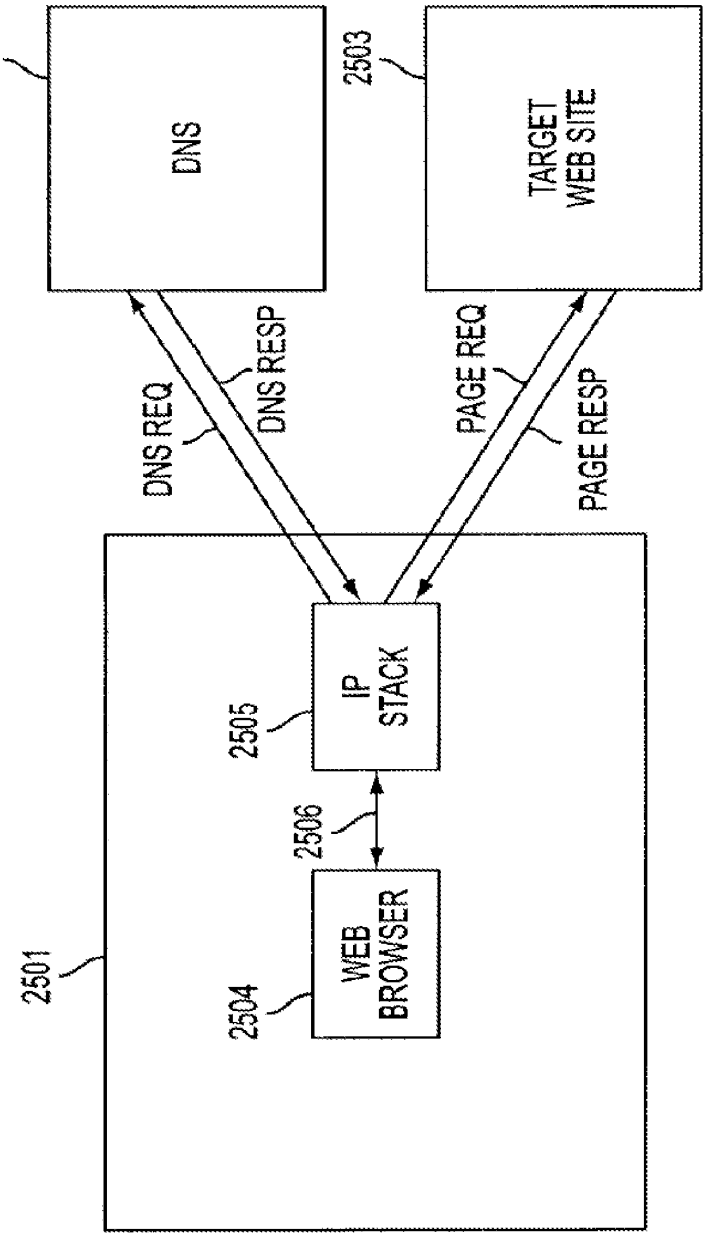


FIG. 25
(PRIOR ART)

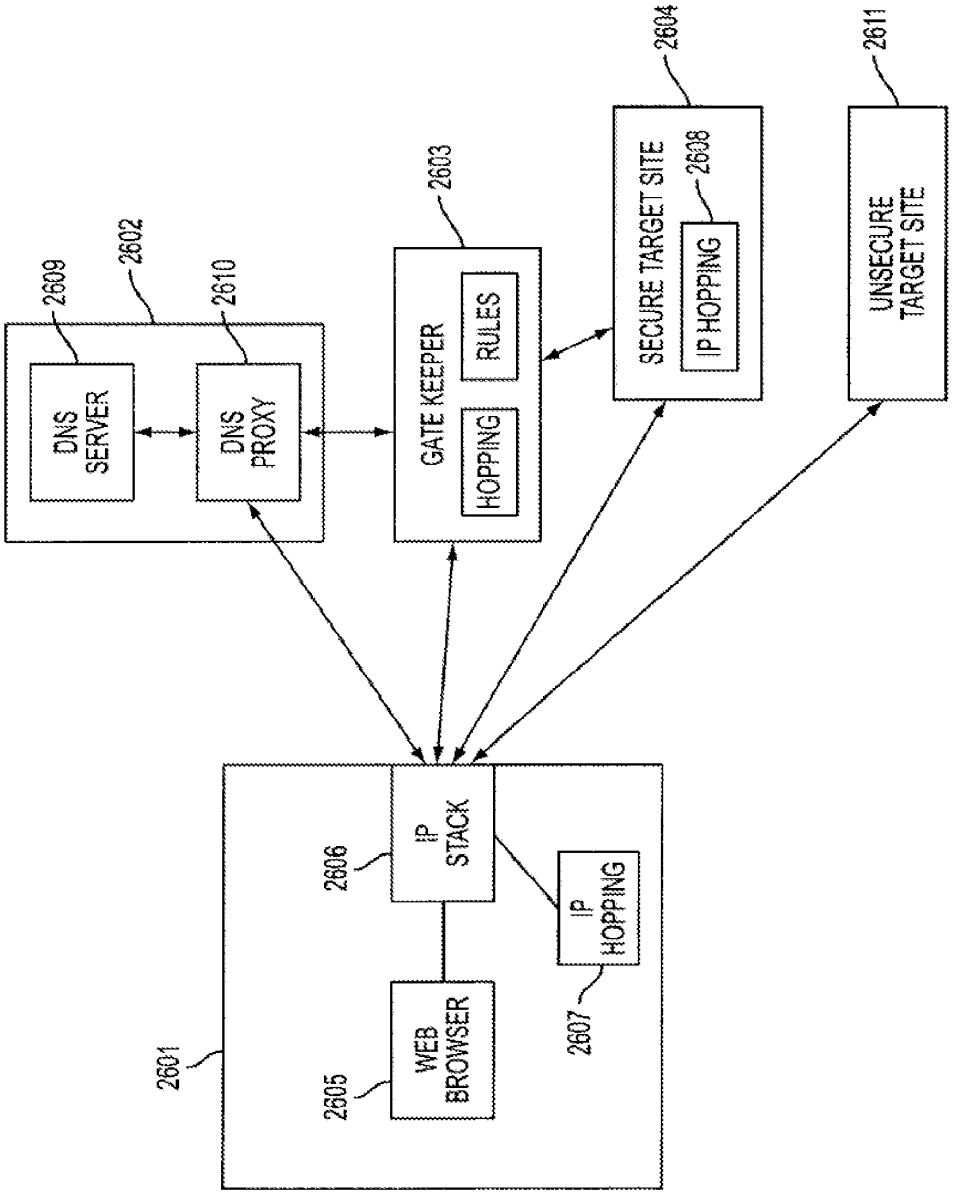

2609
2602
2610
2603
2604
2611
DNS SERVER
DNS PROXY
GATE KEEPER
RULES
HOPPING
SECURE TARGET SITE
IP HOPPING
UNSECURE TARGET SITE
2606
IP STACK
IP HOPPING
2607
2605
WEB BROWSER
2601

FIG. 26

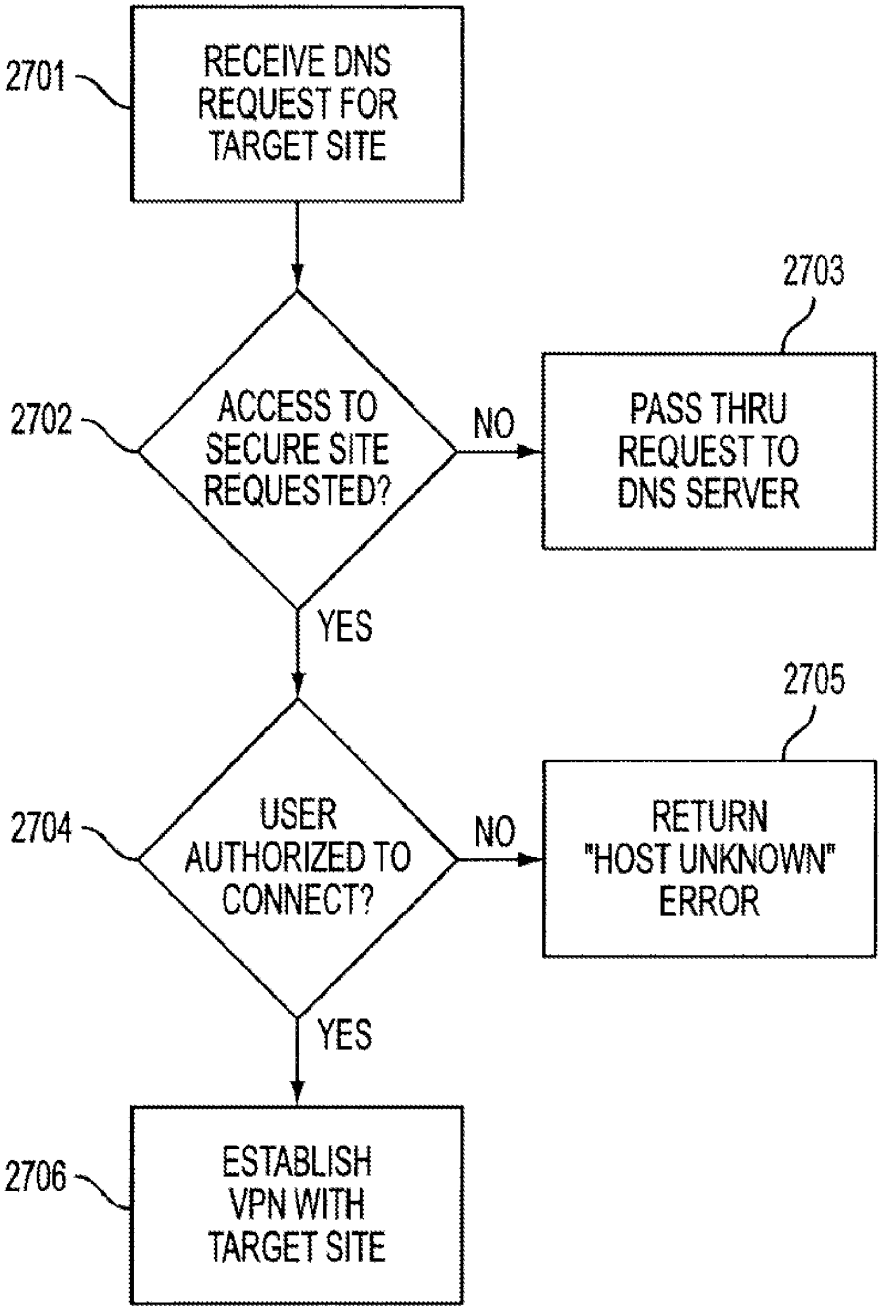

2701
RECEIVE DNS REQUEST FOR TARGET SITE
2702
ACCESS TO SECURE SITE REQUESTED?
NO
2703
PASS THRU REQUEST TO DNS SERVER
YES
2704
USER AUTHORIZED TO CONNECT?
NO
2705
RETURN "HOST UNKNOWN" ERROR
YES
2706
ESTABLISH VPN WITH TARGET SITE

FIG. 27

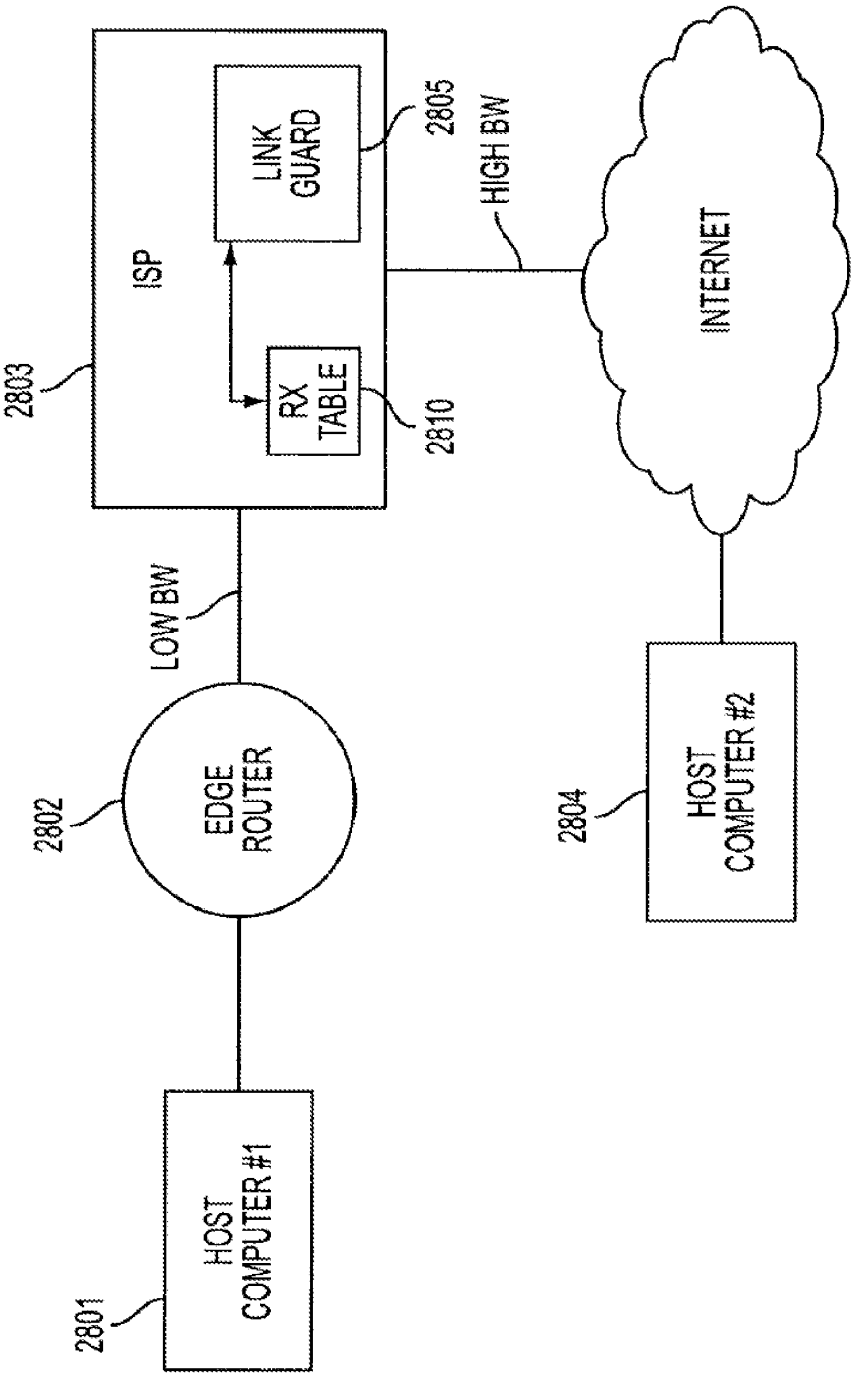

ISP
LINK GUARD
RX TABLE
2803
2805
2810
HIGH BW
INTERNET
LOW BW
EDGE ROUTER
2802
HOST COMPUTER #1
2801
HOST COMPUTER #2
2804

FIG. 28

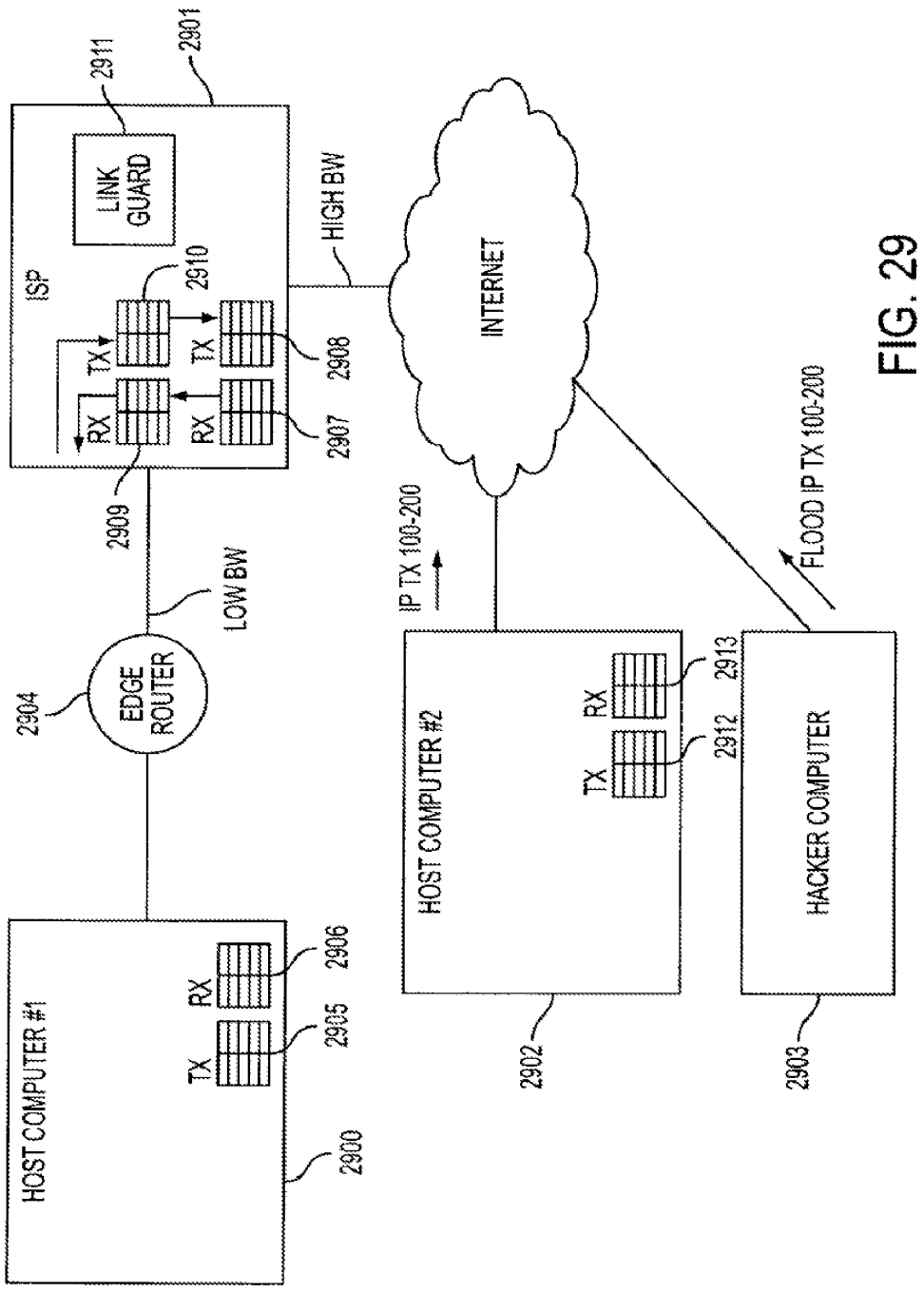

2911
2901
LINK GUARD
ISP
2910
HIGH BW
INTERNET
TX
RX
TX
RX
2908
2907
2909
LOW BW
IP TX 100-200
FLOOD IP TX 100-200
2904
EDGE ROUTER
HOST COMPUTER #2
RX
TX
2913
2912
HACKER COMPUTER
HOST COMPUTER #1
RX
TX
2906
2905
2900
2902
2903

FIG. 29

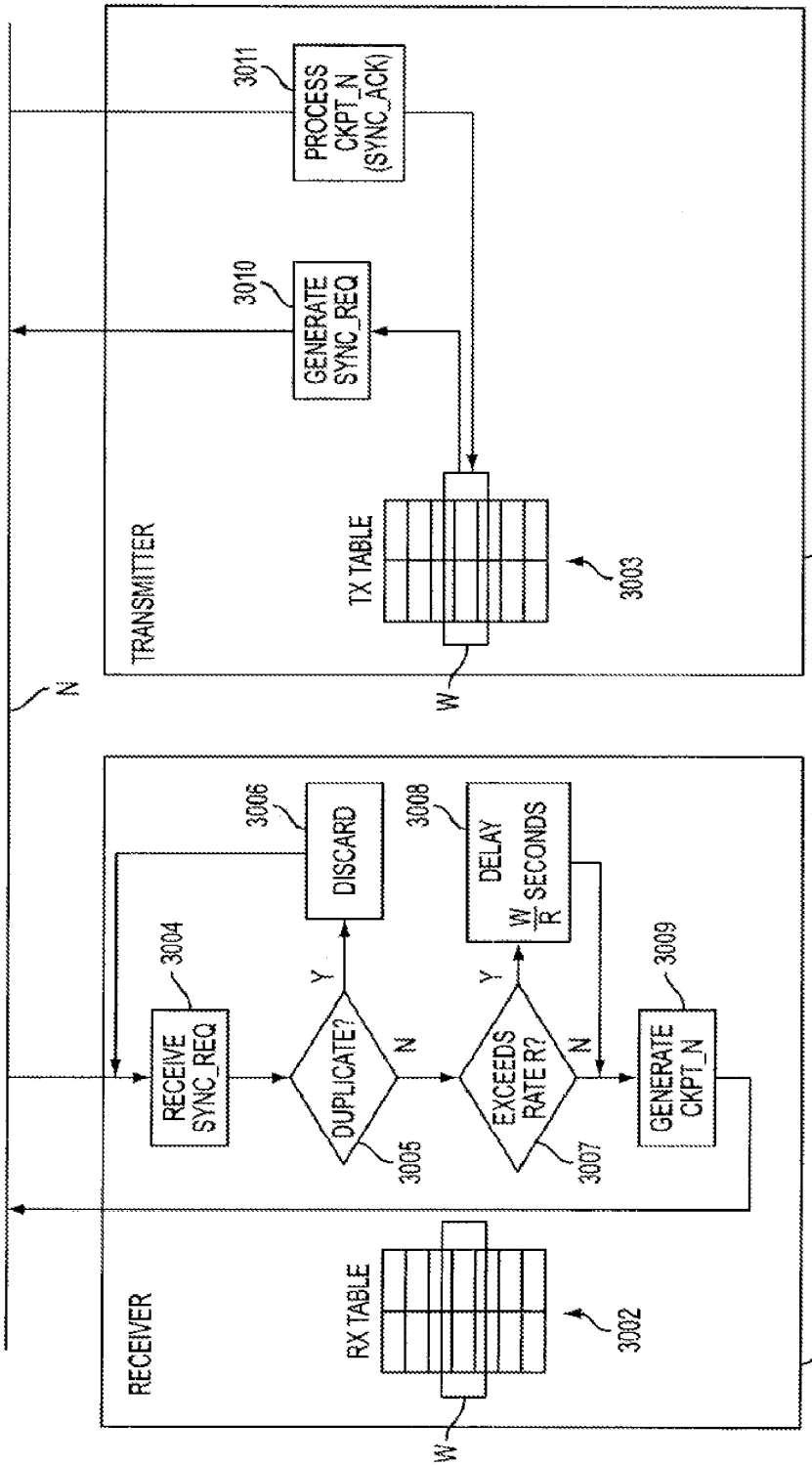

TRANSMITTER
3001
PROCESS CKPT_N (SYNC_ACK)
3011
GENERATE SYNC_REQ
3010
TX TABLE
3003
W
N
RECEIVER
3000
RX TABLE
3002
W
RECEIVE SYNC_REQ
3004
DUPLICATE?
3005
Y
N
DISCARD
3006
EXCEEDS RATE R?
3007
Y
N
DELAY W/R SECONDS
3008
GENERATE CKPT_N
3009

FIG. 30

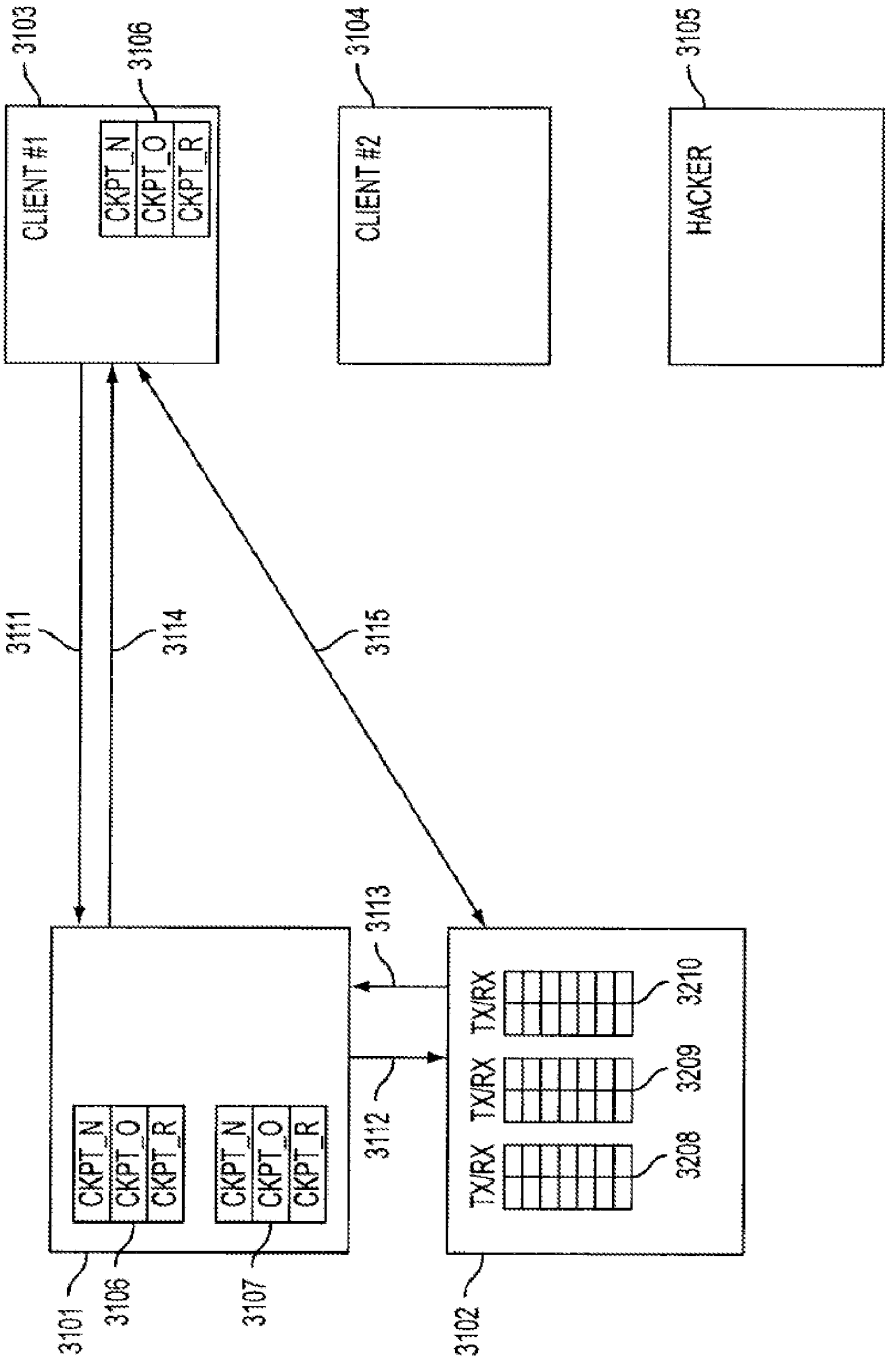

3103
3106
CLIENT #1
CKPT_N
CKPT_O
CKPT_R
3104
CLIENT #2
3105
HACKER
3111
3114
3115
3113
3112
TX/RX
TX/RX
TX/RX
3210
3209
3208
3101
3106
3107
3102

FIG. 31

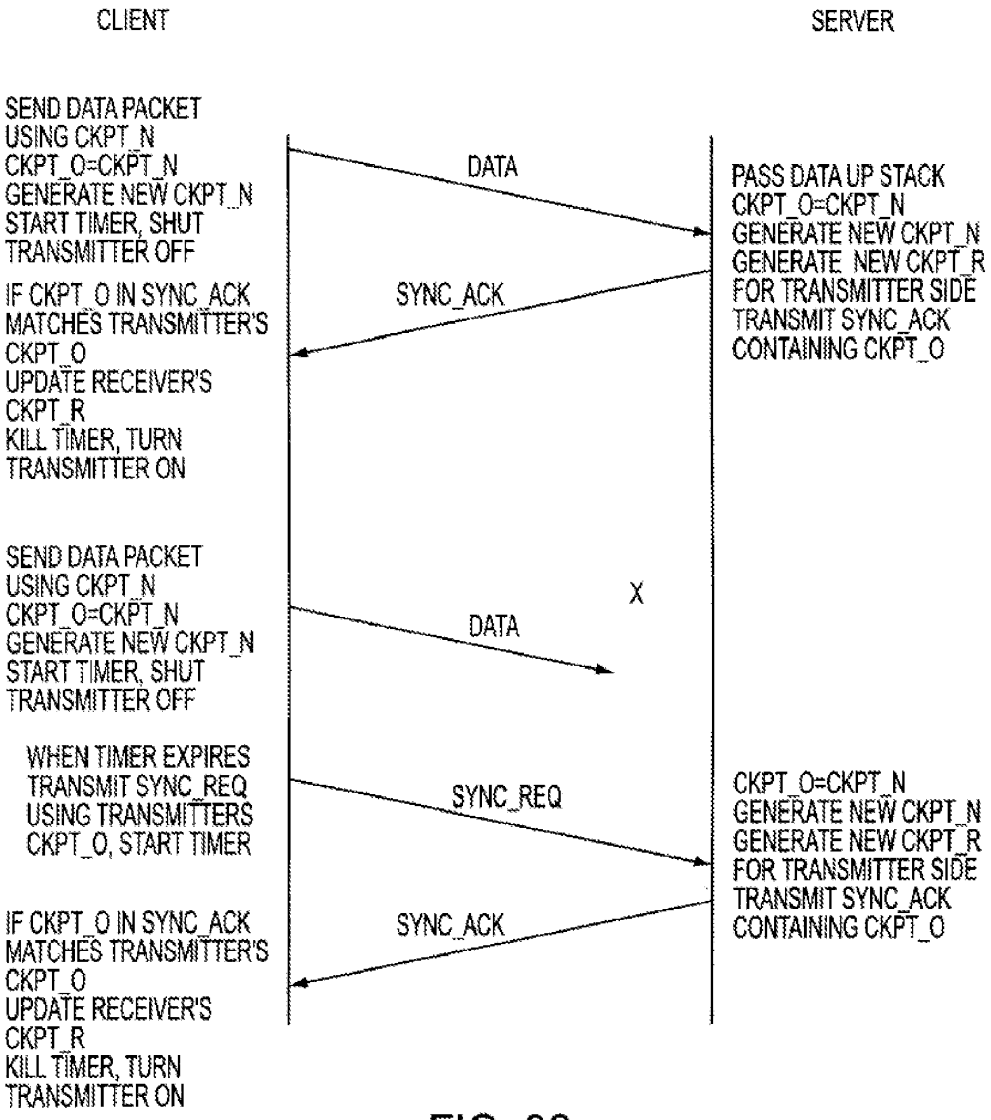

CLIENT
SERVER
SEND DATA PACKET
USING CKPT_N
CKPT_O=CKPT_N
GENERATE NEW CKPT_N
START TIMER, SHUT
TRANSMITTER OFF
DATA
PASS DATA UP STACK
CKPT_O=CKPT_N
GENERATE NEW CKPT_N
GENERATE NEW CKPT_R
FOR TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING CKPT_O
IF CKPT_O IN SYNC_ACK
MATCHES TRANSMITTER'S
CKPT_O
UPDATE RECEIVER'S
CKPT_R
KILL TIMER, TURN
TRANSMITTER ON
SYNC_ACK
SEND DATA PACKET
USING CKPT_N
CKPT_O=CKPT_N
GENERATE NEW CKPT_N
START TIMER, SHUT
TRANSMITTER OFF
X
DATA
WHEN TIMER EXPIRES
TRANSMIT SYNC_REQ
USING TRANSMITTERS
CKPT_O, START TIMER
SYNC_REQ
CKPT_O=CKPT_N
GENERATE NEW CKPT_N
GENERATE NEW CKPT_R
FOR TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING CKPT_O
IF CKPT_O IN SYNC_ACK
MATCHES TRANSMITTER'S
CKPT_O
UPDATE RECEIVER'S
CKPT_R
KILL TIMER, TURN
TRANSMITTER ON
SYNC_ACK

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

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 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 TPT 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

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 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 a an embodiment of the invention.

FIG. **3***a* is an illustration of a process of forming a tunneled IP packet according to an embodiment of the invention.

FIG. **3***b* 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. **12**A shows a system that employs hopped hardware addresses, hopped IP addresses, and hopped discriminator fields.

FIG. **12**B 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_{D}. 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. **3***a*, to construct a series of TARP packets, a data stream **300** of IP packets **207***a*, **207***b*, **207***c*, 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 **207***a*–**207***c* 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 **207***a* 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 **207***a*–**207***c*, 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.

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 **207***a*–**207***c* 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. **3***b*, 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. **3***b*. After encryption, the encrypted block is broken into separate payloads and segments which are interleaved as in the embodiment of FIG. **3***a*. The resulting interleaved packets A, B, and C, are then packaged as TARP packets with TARP headers as in the Example of FIG. **3***a*. The remaining process is as shown in, and discussed with reference to, FIG. **3***a*.

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 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 (fishbowled) 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 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.

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_T headers are converted into normal IP_C headers. The window sequence numbers are integrated in the IP_C headers.

S50. The packets are then handed up to the IP layer processes.

3. 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.

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 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. **12**A and **12**B 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 **1208**X 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 **1222**X 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 **1221**X, and node **1201** maintains a matching receive table **1209** using the same algorithm **1209**X. 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. **12**A (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 **1210**X to generate source and destination hardware addresses that are inserted into frame headers (e.g., fields **1101**A 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. **12**B 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 is 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 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 . . . X_i starting with seed X_0 using a recurrence

$$X_i=(aX_{i-1}+b) \bmod c, \tag{1}$$

where a, b and c define a particular LCR. Another expression for X_i,

$$X_i=((a^i(X_0+b)-b)(a-1)) \bmod c \tag{2}$$

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:

$$X_i=(a^i(X_0(a-1)+b)-b)(a-1) \bmod c \tag{3}$$

It can be shown that:

$$(a^i(X_0(a-1)+b)-b)(a-1) \bmod c=((a^i \bmod((a-1)c))(X_0(a-1)+b)-b)(a-1)) \bmod c \tag{4}$$

$(X_0(a-1)+b)$ can be stored as $(X_0(a-1)+b) \bmod c$, b as b mod 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^n, 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

$$X_{j+1}^n=X_{n(j+1)}=((a^n \bmod((a-1)c))(X_j^n(a-1)+b)-b)(a-1) \bmod c. \tag{5}$$

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:

$$X_n=(31X_{n-1}+4) \bmod 15. \quad (6)$$

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 $a''=31^3=29791$, $c'=(a-1)=15*30=450$ and $a'' \bmod((a-1)c)=31^3 \bmod(15*30)=29791 \bmod(450)=91$. Equation (5) becomes:

$$((91(X_i30+4)-4)/30) \bmod 15 \quad (7).$$

Table 1 shows the jump ahead calculations from (7). The calculations start at 5 and jump ahead 3.

TABLE 1

I	X_i	(X_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^{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 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×WINDOW_SIZE+OoO active addresses ($1 \leq OoO \leq WINDOW\_SIZE$ and $WINDOW\_SIZE \geq 1$). 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×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×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 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 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. **22**A 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. **22**A. 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. **22**A 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. **22**B 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 asses 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:

$$P'=\alpha\times MIN+(1-\alpha)\times P \quad (1)$$

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 β 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 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, α=0.75 and β=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 L 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 **L33**'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**. 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** requesting that a virtual private network be created between user computer **2601** and secure target site **2604**. In one embodiment, gatekeeper **2603** creates "hop-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 though 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×N×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 Ti 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×N×W/R seconds after the last SYNC_REQ has been received and accepted, 2×M×N×W/R seconds after next to the last SYNC_REQ has been received and accepted, C×M×N×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 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

CKPT_N, it generates an 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.

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 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 Page 1 of 1
DATED : December 31, 2002
INVENTOR(S) : Edmund Colby Munger et al.

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



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**

(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)

(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**

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.
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

(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)

OTHER PUBLICATIONS

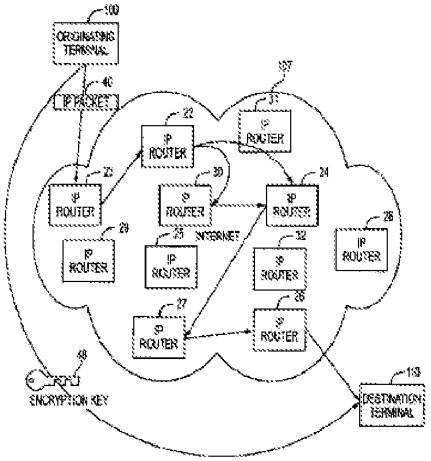
Alan O. Frier et al., "The SSL Protocol Version 3.0", Nov. 18, 1996, printed from http://www.netscape.com/eng/ss13/draft302.txt on Feb. 4, 2002, 56 pages.

(Continued)

Primary Examiner—Andrew L Nalven

(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.



U.S. PATENT DOCUMENTS

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.
5,384,848	A	1/1995	Kikuchi
5,511,122	A	4/1996	Atkinson
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
5,629,984	A	5/1997	McManis
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.
5,764,906	A	6/1998	Edelstein et al.
5,771,239	A	6/1998	Meroney et al.
5,774,660	A	6/1998	Brendel et al.
5,787,172	A	7/1998	Arnold
5,796,942	A	8/1998	Esbensen
5,805,801	A	9/1998	Holloway et al.
5,805,803	A	9/1998	Birrell et al.
5,822,434	A	10/1998	Caronni et al.
5,842,040	A	11/1998	Hughes et al.
5,845,091	A	12/1998	Dunne et al.
5,864,666	A	1/1999	Shrader
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.
5,905,859	A	5/1999	Holloway et al.
5,918,019	A	6/1999	Valencia
5,950,195	A	9/1999	Stockwell et al.
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 et al
6,055,574	A	4/2000	Smorodinsky et al.
6,061,346	A	5/2000	Nordman
6,061,736	A	5/2000	Reciberger et al.
6,079,020	A	6/2000	Liu
6,081,900	A	6/2000	Subramaniam et al.
6,092,200	A	7/2000	Muniyappa et al.
6,101,182	A	8/2000	Sistanizadeh et al.
6,119,171	A	9/2000	Alkhatib
6,119,234	A	9/2000	Aziz et al.
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
6,173,399	B1	1/2001	Gilbrech
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.
6,199,112	B1	3/2001	Wilson
6,202,081	B1	3/2001	Naudus
6,222,842	B1	4/2001	Sasyan 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.
6,256,671	B1	7/2001	Strentzsch et al.
6,262,987	B1	7/2001	Mogul
6,263,445	B1	7/2001	Blumenau
6,286,047	B1	9/2001	Ramanathan et al.
6,298,341	B1	10/2001	Mann 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.
6,314,463	B1	11/2001	Abbott 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.
6,333,372	B1	12/2001	McMillin et al.
6,338,082	B1	1/2002	Schneider
6,353,614	B1	3/2002	Borella 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.
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.
6,618,761	B2	9/2003	Munger et al.
6,671,702	B2	12/2003	Kruglikov et al.
6,687,551	B2	2/2004	Steindl
6,687,746	B1	2/2004	Shuster et al.
6,701,437	B1	3/2004	Hoke et al.
6,714,970	B1	3/2004	Fiveash et al.
6,717,949	B1	4/2004	Boden et al.
6,752,166	B2	6/2004	Latil et al.
6,757,746	B1	6/2004	Parekh et al.
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.
6,937,597	B1	8/2005	Rosenberg et al.
7,010,604	B1	3/2006	Munger et al.
7,039,713	B1	5/2006	Van Gunter et al.
7,072,964	B1	7/2006	Whittle et al.
7,133,930	B2	11/2006	Munger et al.
7,167,904	B1	1/2007	Devarajan et al.
7,188,175	B1	3/2007	McKeeth
7,188,180	B2	3/2007	Larson et al.
7,197,563	B2	3/2007	Sheymov et al.
7,353,841	B2	4/2008	Koso 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.
2002/0004898	A1	1/2002	Droge
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.
2005/0055306	A1	3/2005	Miller 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

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
WO	WO 9827783 A	6/1998

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 Abstact), 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.freeswan.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.freeswan.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.freeswan.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 (dataed 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 conferece on Communications architectures & protocols, pp. 84–91, ACM Press, NY, NY 1986.

Shree Murthy et al., "Congestion–Oriented Shortest Multi–path Routing", Proceedings of IEEE INFOCOM, 1996, pp. 1028–1036.

W. Stallings, "Cryptography And Network Security", 2nd, Edition, Chapter 13, IP Security, Jan. 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, Craig 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).

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 H, 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.

Automative Industry Action Group, "ANX Release 1 Document Publication," AIAG (1997). (AIAG, ANX).

Automative 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 NewOpportunities 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 8 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).

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 hap //www.microsoft.com/presspass/features/1998/10-19nt5.mspxpfinue). (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 (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). (DNSSEC-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 3.1/2.6Administrator'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). (Bhattcharya 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).

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, "Buttet–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–insuit).

Microsoft Corp. Windows NT Servier 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 (Cariplo I).

Marc Levy, COM Internet Services (Apr. 23, 1999), available at 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 (Horstmann).

Microsoft Corp., DCOM: A Business Overview (Apr. 1997), available at 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 (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 12 in PDC DVD-ROM (DCOM Technical Overview II).

125. Scott Suhy & Glenn Wood, DNS and Microsoft Windows NT 4.0 (1996) available at http://msdn2.microsoft.com/en-us/library/ms810277(printer).aspx (Suhy).

126. Aaron Skonnard, *Essential Wininet* 313-423 (Addison Wesley Longman 1998) (Essential Wininet).

Microsoft Corp. Installing, Configuring, and Using PPTP with Microsoft Clients and Servers, (1998) available at 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.mspx (Internet Connection Services I).

Microsoft Corp. Internet Connection Services for RAS, Commercial Edition, available athttp://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/bcgstrtc.mspx (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.mspx (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.mspx (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 (Remote Access).

Microsoft Corp., Understanding PPTP (Windows NT 4.0), available at http://www.microsoft.com/technet/archive/winntas/plan/pptpudst.mspx (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.mspx (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.mspx (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/rrasch05.mspx (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 SSH2.0 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).

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. Matehrs and Shawn P. Genoway, *Windows NT Thing Client Solutions: Implemetning 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 Releaset 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, *IIS 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 Inegrated 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 35 Art VPN Technology.
M. Shane Stigler & Mark A. Linsenbardt, *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.
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/19980210013830/interdyn.com/Accessware.html.
Extended System Press Release, Sep. 2, 1997; *Extended VPN Uses The Internet to Create Virtual Private Networks*, www.extendedsystems.com.

Socks Version 5, Executive Summary, http://web.archive.org/web/19997062003l945/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.
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.
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.
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.
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.
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.
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.
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.
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.
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.
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.
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 Microfoft 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.

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:

(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.

* * * * *

Exhibit 1014

Plaintiff's Opening Claim Construction Brief, *PacSec3, LLC v. NetScout Systems, Inc.*, No. 6:20-cv-00914 (W.D. Tex.) (filed April 12, 2021) (Albright)

Customer No.: 000027683

Haynes and Boone, LLP
IP Section
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Telephone [214] 651.5000
Fax [214] 200.0853

IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
WACO DIVISION

PACSEC3, LLC,	)	
Plaintiff,	)	
	)	**Civil Action No. 6:20-CV-00914-ADA**
v.	)	
	)	
NETSCOUT SYSTEMS, INC.,	)	**JURY TRIAL DEMANDED**
Defendant.	)	

PLAINTIFF'S OPENING CLAIM CONSTRUCTION BRIEF

TABLE OF CONTENTS

I. INTRODUCTION 1

A. Patents 1

1. The '190 Patent 1
2. The '564 patent 1
3. The '497 patent 1

B. The Accused NetScout Systems 2

II. LEGAL PRINCIPLES 2

a. General Claim Construction Principles 2
b. Requirements under 35 U.S.C. §112(f) 3
c. Level of Ordinary Skill in the Art 5

III. DISPUTED CLAIM TERMS 5

1. means for classifying data packets received at said firewall 5
2. means for associating a maximum acceptable transmission rate with each class of data packet received at said firewall 6
3. means for said firewall to find information for packets it receives regarding the path by which said packets came to Said firewall 6
4. whereby, said firewall can use said information to allocate the transmission rate for each class in a desired way 10
5. path 11
6. router 11
7. host computer 12
8. means for limiting the transmission rate from the firewall to the maximum acceptable transmission rate for each class of data packet 12
9. means for said firewall to measure the amount of service requested by each packet 15
10. whereby, packet flooding and other over usage type distributed denial of service attacks cannot be effectively launched through said non-redundant connection 16
11. whereby, said firewall will limit the transmission rate for data packets of each class from locations within one of said networks to provide proportionally fair forwarding service to other locations within said network that communicates through said non-redundant connection 16
12. means for said firewall to measure the amount of service consumed in order to send each identified response data packet 17
13. means for storing and recalling past measurements of amounts of service provided for each type of service 18
14. whereby, said firewall will limit the transmission rate for data packets that are requests for each type of service to limit usage of each service over extended periods of time 18
15. means for classifying data packets received at a/said host computer/router into wanted data packets and unwanted data packets/...by path 19
16. means for associating a maximum acceptable processing rate with each class of data packet received at said computer 19

17. means for said computer to find information for packets it receives regarding the path by which said packets came to said computer via packet marks provided by routers leading to said host computer 25
18. means in said computer for using said information to allocate the processing rate available for unwanted data packets to be less than or equal to said maximum acceptable processing rate 25
19. means for said router to find information for packets it receives regarding the path by which said packets came to said router via packet marks provided by routers leading to said host computer 26
20. means in said router for said router to use said information to allocate the transmission rate for unwanted data packets to be less than equal to said maximum acceptable transmission rate.....27
21. means for determining a path by which data packets arrive at a host computer via packet marks provided by routers leading to said host computer 27
22. means for assigning a maximum acceptable processing rate to each class of data packet.....27
23. means for allocating a processing rate equal to or less than said maximum acceptable processing rate to said unwanted data packets 28

TABLE OF AUTHORITIES

Cases

Altiris, Inc. v. Symantec Corp., 318 F.3d 1363, 65 USPQ2d 1865 (Fed. Cir. 2003) 3
Apex Inc. v. Raritan Computer, Inc., 325 F.3d 1364, 66 USPQ2d 1444 (Fed. Cir. 2003) 5
Budde v. Harley-Davidson, Inc., 250 F.3d 1369 (Fed. Cir. 2001) 5
Ergo Licensing, LLC v. CareFusion 303, Inc., 673 F.3d 1361 (Fed. Cir. 2012) 4
Finisar Corp. v. DirecTV Grp., Inc., 523 F.3d 1323 (Fed. Cir. 2008) 4
Greenberg v. Ethicon Endo-Surgery, Inc., 91 F.3d 1580, 39 USPQ2d 1783 (Fed. Cir. 1996) 4
Honeywell Int'l, Inc. v. Nikon Corp., 589 F. Supp. 2d 433 (D. Del. 2008) 10, 17, 18, 19
Ihance, Inc. v. Eloqua Ltd., Civ. Act. No. 2:11cv2572012, U.S. Dist. LEXIS 62514, (E.D. Va. May 2, 2012) 10, 16, 17, 19
Johnson Worldwide Assocs. v. Zebco Corp., 175 F.3d 985 (Fed. Cir. 1999) 2
Lighting World, Inc. v. Birchwood Lighting, Inc., 382 F.3d 1354 (Fed. Cir. 2004) 3
Markman v. Westview Instruments, Inc., 52 F.3d 967 (Fed. Cir. 1995) (en banc), *aff'd*, 517 U.S. 370 (1996) 2
Mass. Inst. of Tech. v. Abacus Software, 462 F.3d 1344 (Fed. Cir. 2006) 4
MeetrixIP, LLC v. Citrix Sys., Inc., No. 1:16-CV-1033-LY, 2017 WL 5986191, (W.D. Tex. Dec. 1, 2017) 2
Minton v. Nat'l Ass'n of Sec. Dealers, Inc., 336 F.3d 1373 (Fed. Cir. 2003) 11
PerdiemCo, LLC v. IndusTrack LLC, No. 2:15-cv-727-JRG-RSP, 2016 U.S. Dist. LEXIS 87927 (E.D. Tex. July 7, 201 11
Phillips v. AWH Corp., 415 F.3d 1303 (Fed. Cir. 2005) (en banc) 3
Pisony v. Commando Construction, Inc., W-17-CV-00055-ADA, 2019 WL 928406 (W.D. Tex. Jan. 23, 2019) 2
TecSec, Inc. v. IBM, 731 F.3d 1336, 1348-9 (Fed. Cir. 2013) 3
Tex. Instruments Inc. v. U.S. Int'l Trade Comm'n, 988 F.2d 1165 (Fed.Cir.1993) 10, 16, 17, 19
Thorner v. Sony Computer Entm't Am. LLC, 669 F.3d 1362 (Fed. Cir. 2012) 3
TriMed, Inc. v. Stryker Corp., 514 F.3d 1256, 85 USPQ2d 1787 (Fed. Cir. 2008) 3
Watts v. XLSys., L.P., No. 1:06-cv-653-LY, 2008 WL 5731945 (W.D. Tex. July 1, 2008) 2
Williamson v. Citrix Online, LLC, 792 F.3d 1339 (Fed. Cir. 2015) 3

I. INTRODUCTION

Plaintiff PacSec3, LLC ("PacSec") asserts US 6,789,190 ("the '190 patent") (Claim 1); US 7,047,564 ("the '564 patent") (Claims 1-6); and, US 7,523,497 ("the '497 patent") (Claims 1, 4, 7, 10, 13, 16) (collectively, the "Asserted Patents") against NetScout, Inc. ("NetScout")'s Distributed Denial of Service ("DDoS") attack protection products including Peakflow SP/Peakflow SP TMS (and other Peakflow versions).

II. BACKGROUND

A. Patents

1. The '190 Patent

Embodiments of the '190 patent relate to systems and methods for preventing "packet flooding" or DDoS attacks where an attacker uses up all available bandwidth with a victim, typically supplying useless data and to prevent other related denial of service attacks. The defense is distributed among cooperating sites and routers. [1] The claimed invention solved prior art problems associated with association of data packets for which bandwidth should be fairly allocated and control over which packets are allowed to arrive.[2] In essence, the '190 patent claims a technical solution to these problems through cooperating sites and routers.[3] PacSec3 is asserting Claim 1 of the '190 patent.

2. The '564 patent

Embodiments of the '564 patent relate to systems and methods to reduce or limit DDoS attacks by identifying and classifying data packets arriving at a "Reverse Firewall" for

[1] *See* Doc. No. 1-1, '190 patent at Abstract.
[2] *See* Doc. No. 1-1, '190 patent at Column 1, line 65 to Column 2, line 4 ("1:65-2:4").
[3] *See* Doc. No. 1-1, '190 patent at 2:5-16.

transmission to the external network.[4] The '564 patent claims a technical solution comprising a firewall that includes hardware and software providing a non-redundant connection between networks and serves to control packet transmission between the networks.[5] PacSec3 is asserting Claims 1-6 of the '564 patent.

3. The '497 patent

Embodiments of the '497 patent relate to systems and methods to reduce or limit DDoS attacks by identifying and classifying data packets arriving at a cooperating router for transmission to the external network.[6] The '497 patent claims a technical solution comprising a host computer, router, communication lines and transmitted data packets.[7] PacSec3 is asserting Claims 1, 4, 7, 10, 13, and 16 of the '497 patent.

B. The Accused NetScout Systems

PacSec3 accuses NetScout of directly infringing the Asserted Patents through Defendant's Arbor DDoS system.[8]

II. LEGAL PRINCIPLES

a. General Claim Construction Principles

Determining the proper meaning of patent claims is a question of law that exclusively belongs to the Court.[9] During claim construction, a court first looks at the words of the claims

[4] *See* Doc. No. 1-2, '564 patent at Abstract.
[5] *See* Doc. No. 1-2, '564 patent at 3:32-42.
[6] *See* Doc. No. 1-3, '497 patent at Abstract.
[7] *See* Doc. No. 1-3, '497 patent at 3:30-42.
[8] https://www.netscout.com/arbor-ddos#section--1.
[9] *Markman v. Westview Instruments, Inc.*, 52 F.3d 967, 970-71 (Fed. Cir. 1995) (en banc), *aff'd*, 517 U.S. 370 (1996).

themselves to define the scope of the patented invention.[10] In determining the meaning of the claims, "there is a 'heavy presumption in favor of the ordinary meaning of claim language.'"[11] Ordinary meaning is defined as the "meaning that term would haveto a person of ordinary skill in the art in question at the time of invention."[12]

b. Requirements under 35 U.S.C. §112(f).

The use of the term "means" triggers a rebuttable presumption that *§ 112*, ¶ 6 [now *§ 112(f)*] applies.[13] The presumption that 35 USC 112(f) applies is overcome when the limitation further includes the structure, material or acts necessary to perform the recited function. "Sufficient structure exists when the claim language specifies the exact structure that performs the function in question without need to resort to other portions of the specification or extrinsic evidence for an adequate understanding of the structure."[14] The standard is whether the words of the claim are understood by persons of ordinary skill in the art (POSITA) to have a sufficiently definite meaning as the name for structure.[15]

Another manner in which this presumption can be overcome is if "the claim recites sufficient structure for performing the described functions in their entirety."[16] To determine if the

---

[10] *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005) (en banc).
[11] *Watts v. XLSys., L.P.*, No. 1:06-cv-653-LY, 2008 WL 5731945, at *7 (W.D. Tex. July 1, 2008) (quoting *Johnson Worldwide Assocs. v. Zebco Corp.*, 175 F.3d 985, 989 (Fed. Cir. 1999)); *see also MeetrixIP, LLC v. Citrix Sys., Inc.*, No. 1:16-CV-1033-LY, 2017 WL 5986191, at *2 (W.D. Tex. Dec. 1, 2017) (citing *Thorner v. Sony Computer Entm't Am. LLC*, 669 F.3d 1362, 1365 (Fed. Cir. 2012))("The Federal Circuit has reaffirmed that a departure from the ordinary and customary meaning isthe exception, not the rule.").
[12] *Phillips*, 415 F.3d at 1313; *see also Pisony v. Commando Construction, Inc.*, W-17-CV-00055-ADA, 2019 WL 928406,at *1 (W.D. Tex. Jan. 23, 2019). "[T]he person of ordinary skill in the art is deemed to read the claim term not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the specification." *Phillips*, 415 F.3d at 1313.
[13] *TecSec, Inc. v. IBM*, 731 F.3d 1336, 1348-9 (Fed. Cir. 2013) *citing TriMed, Inc. v. Stryker Corp., 514 F.3d 1256, 1259 (Fed. Cir. 2008)*.
[14] See *TriMed, Inc. v. Stryker Corp.,* 514 F.3d 1256, 1259-60, 85 USPQ2d 1787, 1789 (Fed. Cir. 2008); ; see also *Altiris, Inc. v. Symantec Corp.,* 318 F.3d 1363, 1376, 65 USPQ2d 1865, 1874 (Fed. Cir. 2003).
[15] *Williamson v. Citrix Online, LLC,* 792 F.3d 1339, 1349 (Fed. Cir. 2015).
[16] *Id.*

claim recites sufficient structure, "it is sufficient if the claim term is used in common parlance or by persons of skill in the pertinent art to designate structure, even if the term covers a broad class of structures and even if the term identifies the structures by their function."[17]

If persons of ordinary skill in the art reading the specification understand the term to have a sufficiently definite meaning as the name for the structure that performs the function, even when the term covers a broad class of structures or identifies the structures by their function (e.g., "filters," "brakes," "clamp," "screwdriver," and "locks") 35 USC 112(f) will not apply.[18] Many devices take their names from the functions they perform. The term is not required to denote a specific structure or a precise physical structure to avoid the application of 35 USC 112(f).[19] The following are examples of structural terms that have been found **not** to invoke 35 USC 112(f): "circuit," "detent mechanism," "digital detector," "reciprocating member," "connector assembly," "perforation," "sealingly connected joints," and "eyeglass hanger member." Likewise, the Federal Circuit found the recitation of "aesthetic correction circuitry" sufficient to avoid a means plus function analysis or treatment because the term "circuit," combined with a description of the function of the circuit, connoted sufficient structure to one of ordinary skill in the art.[20]

For claims employing a computer implemented means-plus-function limitation where sufficient structure is determined not to have been disclosed, to avoid indefiniteness, the specification must disclose a special purpose computer as corresponding structure—*i.e.*, a

---

[17] *Williamson v. Citrix Online, LLC,* 792 F.3d at 1349, *citing Lighting World, Inc. v. Birchwood Lighting, Inc., 382 F.3d 1354, 1359-60 (Fed. Cir. 2004).*

[18] *Apex Inc. v. Raritan Computer, Inc.,* 325 F.3d 1364, 1372-73, 66 USPQ2d 1444, 1451-52 (Fed. Cir. 2003)

[19] *Greenberg v. Ethicon Endo-Surgery, Inc.,* 91 F.3d 1580, 1583, 39 USPQ2d 1783, 1786 (Fed. Cir. 1996) ("See *Watts,* 232 F.3d at 880, 56 USPQ2d at 1838; *Inventio AG v. Thyssenkrupp Elevator Americas Corp.,* 649 F.3d 1350, 99 USPQ2d 1112 (Fed. Cir. 2011) (holding that the claim terms "modernizing device" and "computing unit" when read in light of the specification connoted sufficient, definite structure to one of skill in the art to preclude application of 35 [USC112(f)]).

[20] *Mass. Inst. of Tech. v. Abacus Software*, 462 F.3d 1344, 1359-1360 (Fed. Cir. 2006).

computer programmed to perform a disclosed algorithm.[21] An algorithm may be disclosed in "any understandable terms including as a mathematical formula, in prose, or as a flow chart, or in any other manner that provides sufficient structure."[22] However, "[s]imply reciting 'software' without providing some detail about the means to accomplish the function is not enough."[23] The party alleging that the specification fails to disclose sufficient corresponding structure must make that showing by clear and convincing evidence.[24]

c. Level of Ordinary Skill in the Art

From my review of the Patents-in-Suit, taking into account my education and experience, I am of the opinion that a person of ordinary skill in the art ("POSITA") at the time of the filing of the Asserted Patents or Patents-in-Suit would have at least a Master of Science ("MS") Degree in Computer Science or Computer Engineering, or equivalent work experience in the field of computer networks, along with knowledge of the general structure of networked communication systems, its hardware and software components and underlying communications technologies. In addition, a POSITA would be familiar with the latest communications standards.[25]

III. DISPUTED CLAIM TERMS

1. means for classifying data packets received at said firewall[26]

Plaintiff's proposed construction	Defendant's proposed construction	By:[27]
Function: identifying classes of the data packets that are received at said firewall	Indefinite	J

[21] *Ergo Licensing, LLC v. CareFusion 303, Inc.*, 673 F.3d 1361, 1364-65 (Fed. Cir. 2012).
[22] *Finisar Corp. v. DirecTV Grp., Inc.*, 523 F.3d 1323, 1340 (Fed. Cir. 2008) (internal citation omitted).
[23] *TecSec, Inc.*, 731 F.3d at 1348-9.
[24] *Id.*, *citing Budde v. Harley-Davidson, Inc.*, 250 F.3d 1369, 1380-81 (Fed. Cir. 2001).
[25] See Declaration of Krishnamurthy Narayanaswamy, Ph.D. ("Narayanaswamy Decl.") at ¶13.
[26] Claim 1 of the '190 patent; Claim 1 of the '564 patent.
[27] J=Joint; P=Plaintiff; D=Defendant

Structure: 1:57 to 2:4; 3:1 to 3:5; Fig 1: 4:54 to 4:59 and 5:37 to 5:51; Fig 4: 5:3 to 5:6 and 6:13 to 6:24; Fig 7: 5:18 to 5:22 and 6:55 to 7:3		

2. means for associating a maximum acceptable transmission rate with each class of data packet received at said firewall[28]

Plaintiff's proposed construction	**Defendant's proposed construction**	**By:**
Function: associating a maximum threshold rate of transmission for each identified packet class through said firewall Structure: 1:65 to 2:4; 3:6 to 3:12; Fig 1: 4:59 to 4:54 and 5:37 to 5:51; Fig 4: 5:3 to 5:6 and 6:13 to 6:24; Fig 7: 5:18 to 5:22 and 6:55 to 7:3	Indefinite	J

3. means for said firewall to find information for packets it receives regarding the path by which said packets came to Said firewall[29]

Plaintiff's proposed construction	**Defendant's proposed construction**	**By:**
Function: providing the firewall with the path through which the packets were forwarded to the firewall Structure: 2:4 to 2:17; 3:6 to 3:12; 4:14 to 4:18; Fig 1: 4:59 to 4:54 and 5:37 to 5:51; Fig 4: 5:3 to 5:6 and 6:13 to 6:24; Fig 7: 5:18 to 5:22 and 6:55 to 7:3	Indefinite	J

Plaintiff groups these three terms together as the disclosure of structure and argument is largely coextensive. However, while Plaintiff contends these terms do not require the 112(f) analysis, Plaintiff's identified function and structure are provided in the boxed chart with argument and explanation following.

These claim terms are not subject to the requirements of 112(f) because from the perspective of one of ordinary skill in the art, the preamble of claim 1 of the '190 patent, followed by the subsequent claim language provide the structure of a router or firewall for each of "means for classifying...", means for associating..." and "means for said firewall..."[30] Both a router and

[28] Claim 1 of the '190 patent; Claim 1 of the '564 patent.
[29] Claim 1 of the '190 patent.
[30] See Narayanaswamy Decl. at ¶15.

a firewall have a very well-defined meaning in the art. A firewall is a device which is placed between two networks by the owner of one of those networks to safeguard that network from various network attacks.[31] These capabilities include identifying sets of data packets by certain properties (referred to as classification) and allocating a transmission rate to those packets identified by class. In the current invention, this method is extended to include information about the path taken by the packets as they traverse the computer network before they arrive at the firewall. A router is a device which forwards data packets to locations within a computer network. The data packets are forwarded along communication lines or paths between the appropriate parts of the computer network. It is very common for a router and/or a firewall to be part of securing a computer network for inspecting the contents of packets, and/or limiting the rates of transmission of specific sets of packets identified by properties (i.e., classification) including blocking that set of packets entirely.

One of ordinary skill in the art would understand that a router and a firewall provides the following capabilities:

1. classifying data packets received at the firewall;

2. associating a maximum acceptable transmission rate with each class of data packet received at said firewall; and,

3. receiving data packets from a path by which said packets came to the firewall.[32]

Plaintiff's proposed claim construction is further based on the literal language of the claim and the specification. First, the claim sets forth in the preamble that a packet flooding defense system of the invention includes a router.[33] Second, the "at least one firewall" comprises "hardware and software serving to control packet transmission" which one of ordinary skill in the art would know is a firewall that can be configured by software to provide the capabilities described.[34] This is sufficient structure for each of "means for classifying...", means for

[31] *See* Narayanaswamy Decl. at ¶15.
[32] *See* id. at ¶15.
[33] *See* Doc. No. 1-1 at 7:40-44.
[34] *See* id. at 7:45-47.

associating..." and "means for said firewall..."

In an alternative, adequate disclosure of the structure of a firewall and router is also disclosed in the specification. The claimed invention describes how the forwarding path for each packet is associated with that packet, thereby making it possible for a receiving computer to know that information. The software, or configuration, at the firewall is explained through the algorithms disclosed in Figure 1 by flowchart and diagram where a Class A and Class B (38) of a maximum acceptable processing rate (34) are assigned by the firewall to each data packet (30). The description provides that the computer can use the information, the maximum acceptable processing rate, to allocate the processing rate for each class (38) in a desired way among the places from which packets (30) are transmitted.[35] Figure 2, Figure 3, Figure 4, Figure 5, and Figure 6 illustrate additional embodiments, or algorithms, in which path information associated with each packet can be combined with other properties to classify the packet and assign it a transmission rate. Figure 7, Figure 8 and Figure 9 illustrate how a firewall protecting a network communicates the rate limit for packets routed to it or routed to it by class functions. Moreover, the marking of data packets by a firewall or router is within the ordinary parlance of one of ordinary skill in the art and as is disclosed in the highly related '497 patent.[36] Prior art discussed in the specification of the '190 patent (US5455865 and US6044402 for example) rely on the well-understood and accepted functionality of routers to maintain flow information between source and destination addresses by marking.[37] Marking of data packets that traverse a router is well-known in the art. Marking of data packets to encode path information is unique to this invention, and anyone familiar with firewalls would understand how to use a firewall capability to encode paths. The specification describes how a firewall and a router can add to packet information about the interface points where the packet entered the firewall or router.

The '190 patent provides very detailed description of packet marking. At 2:5-17, the '190

---

[36] *See* Doc. No. 1-3 at 3:64-66 ("which is done via packet marks provided by router...."); *See* Narayanaswamy Decl. at ¶17.
[37] *See* Narayanaswamy Decl. at ¶17.

patent provided that the claimed invention provides a distributed defense:

> among cooperating sites and routers. A set of transitively connected cooperating machines is called a "cooperating neighborhood". The quality of the defense is related to the size of the cooperating neighborhood, a larger neighborhood providing better defense. Within the neighborhood it is possible to trace the forwarding path of packets. The association of packets with the "users" is approximated by associating packets with "places" in the cooperating neighborhood from which those packets are forwarded. That is, service will be allocated in a fair (or otherwise reasonable) manner among these places. A "place" in this sense is typically a particular interface from which a packet arrived at a cooperating router.

At 2:28-35, the '190 patent provides:

> Routers will supply data about the forwarding path of the packets that arrive at a site. The site can use this data to allocate service as described above among the packets that arrive. This would solve the problem of unfair service if the packets that arrived were a fair sample of those that were sent to the site. This may not be the case, however, if routers are unable to forward all the packets they receive.

At 2:38-43, the '190 patent provides:

> However another potential cause for this problem is a flooding attack against a router. That problem is solved by letting routers allocate their services in a similar way to that described above for sites. That is, they allocate the limited resource of forwarding bandwidth along any given output in a reasonable way among some set of places in the cooperating neighborhood

At 2:44-50, the '190 patent provides:

> The final step in the defense is that cooperating routers will limit the rate at which they forward packets to places that so request. This may not be essential in the allocation of service, but it is useful for limiting the bandwidth used by "unwanted" packets. The rate-limiting request is to be made when a site detects a high rate of unwanted packets coming from one place.

Figures 1, 2, 3, 4, 5, 6, 7 and 8 each illustrate illustrative methods of determining, by packet marking, how the path a packet takes is identified. These Figures, in cooperation with the written description, further provide various algorithms, by flowchart, used by the claimed invention to perform the claimed functions.[38]

[38] *See* Narayanaswamy Decl. at ¶19.

It is common parlance for one of ordinary skill in the art to use a router for

1. using a router or a firewall to mark data packets;[39]
2. associating a maximum acceptable transmission rate with each class of data packet;[40] and,
3. receiving data packets along identifiable paths;[41]

Prior art cited in this patent specification including US5455865 and US5353353 use such capabilities to implement other network attack defenses. In this patent specification the same capabilities are utilized to specify path information. The 190 patent's specification describes how a router can use the path information to limit the rate of transmission of packets with that path (i.e., that class of packet).[42]

4. whereby, said firewall can use said information to allocate the transmission rate for each class in a desired way[43]

Plaintiff's proposed construction	Defendant's proposed construction	By:
No construction necessary or "whereby, the firewall is capable of using the information to allocate the transmission rate for each class in a desired way"	whereby, said firewall uses said information to allocate thetransmission rate for each class in a desired way	D

Construction of long claim terms is disfavored.[44] Additionally, this "whereby" clause does not provide a limitation on the claim and need not be construed.[45] "A whereby clause in a method

[39] *See* Narayanaswamy Decl. at ¶21.
[40] *See* id. at ¶23.
[41] *See* id. at ¶25.
[42] *See* id. at ¶s 21, 23, 25.
[43] Claim 1 of the '190 patent.
[44] *Ihance, Inc. v. Eloqua Ltd.*, Civ. Act. No. 2:11cv2572012, U.S. Dist. LEXIS 62514, at *15, *16 (E.D. Va. May 2, 2012) (describing a 42-word term as "lengthy" and declining to construe the term); *Honeywell Int'l, Inc. v. Nikon Corp.*, 589 F. Supp. 2d 433, 440–444 (D. Del. 2008) (describing a 24-word term as "rather lengthy," then focusing on 2 words of the claim term, and ultimately declining to construe the 24-word term).
[45] *Tex. Instruments Inc. v. U.S. Int'l Trade Comm'n*, 988 F.2d 1165, 1172 (Fed.Cir.1993) (concluding, in process claims, that the whereby clauses described "the result of arranging the components of the claims in the manner recited in the claims" and thus were not limitations).

claim is not given weight when it simply expresses the intended result of a process step positively recited."[46] Thus, if a whereby clause adds nothing to the claims and simply characterizes the intended result of the executing steps, it need not be construed by the court.[47] Here, no meaningful limitation is added by this "whereby" clause and no construction is necessary.

5. path[48]

Plaintiff's proposed construction	Defendant's proposed construction	By:
Plain and ordinary meaning	each router	D

Path is a well-understood word.[49] There is nothing ambiguous about it and there is no evidence that the patent applicant clearly redefined (in the application or prosecution history). Thus, it should simply be construed as "plain and ordinary meaning."[50] Rejection of Defendant's proposed construction and adoption of a "plain and ordinary meaning" construction resolves the claim-construction dispute without need for further construction of the term.[51]

The defendant's proposed construction would render limitations in the claims of the '497 patent superfluous as Claim 1 of the '497 patent adds a limitation of "all routers." Defendant's construction would make the term "said each router comprising all routers." Further, a path and a router are separate terms and should not be construed as one term.

6. router[52]

Plaintiff's proposed construction	Defendant's proposed construction	By:
Plain and ordinary meaning	a device that determines the optimal path	D

[46] *Minton v. Nat'l Ass'n of Sec. Dealers, Inc.*, 336 F.3d 1373, 1381 (Fed. Cir. 2003) (*citing Tex. Instruments Inc.*, 988 F.2d at 1172).
[47] *See Minton*, 336 F.3d at 1381.
[48] 'Claim 1 of the '190 patent.
[49] *See* Narayanaswamy Decl. at ¶73.
[50] *See Arthrex*, 2016 U.S. Dist. LEXIS 105750, at **106, 108, 127 173, 174 (Payne, Mag).
[51] *PerdiemCo, LLC v. IndusTrack LLC*, No. 2:15-cv-727-JRG-RSP, 2016 U.S. Dist. LEXIS 87927, *64 (E.D. Tex. July 7, 2016 (stating, "The Court therefore expressly rejects Defendants' indefiniteness arguments. No further construction is necessary. [Citations and paragraph break omitted].
[52] Claim 1 of the '190 patent; Claims 1, 4, 7, 10, 13, and 16 of the '497 patent.