2005328336 22 Aug 2011

parallel firewall where different portions of a firewall rule set are distributed among different firewall nodes. Figure 19 is a flow chart illustrating exemplary steps of a process for filtering packets using a rule set distributed among firewall nodes in a function-parallel manner. Referring to Figure 19, in step **1900**, packet filtering rules of a rule set are distributed among nodes in a function-parallel manner so that different nodes implement different portions of the rule set. In step **1902**, packets to be filtered are replicated to each of the nodes. In step **1904**, the packets to be filtered are replicated to each of the nodes. In a hierarchical design, step **1904** may be implemented at each level in the hierarchy. If a packet passes or fails at one level, the packet may be forwarded into a network or dropped, depending on the distribution of the rule set. In step **1904**, the rules are applied to filter the packets.

The steps illustrated in Figure 19 and in all of the examples described herein may be applied to firewall systems that filter packets based on packet header information, intrusion detection systems that filter packets based on packet payload information, or intrusion protection systems that filter packets based on a combination of packet header and packet payload information.

In the claims which follow and in the preceding description of the invention, except where the context requires otherwise due to express language or necessary implication, the word "comprise" or variations such as "comprises" or "comprising" is used in an inclusive sense, i.e. to specify the presence of the stated features but not to preclude the presence or addition of further features in various embodiments of the invention.

It is to be understood that, if any prior art publication is referred to herein, such reference does not constitute an admission that the publication forms a part of the common general knowledge in the art, in Australia or any other country.

2005328336 22 Aug 2011

The following references correspond to the citations in this document. The disclosure of each of these references is hereby incorporated herein by reference it its entirety.

5

References

[1] A. V. Aho, J.E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1987.

[2] M. Al-Suwaiyel and E. Horowitz. Algorithms for Trie Compaction. *ACM Transactions on Database Systems*, 9(2):243–263, June 1984.

[3] S. M. Bellovin and W. Cheswick. Network Firewalls. *IEEE Communications Magazine*, pages 50–57, Sept. 1994.

[4] C. Benecke. A Parallel Packet Screen for High Speed Networks. In *Proceedings of the 15th Annual Computer Security Applications Conference*, 1999.

[5] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley and Sons, Inc., 1998.

[6] D. Comer. Analysis of a Heuristic for Full Trie Minimization. *ACM Transactions on Database Systems*, 6(3):513–537, Sept. 1981.

[7] U. Ellermann and C. Benecke. Firewalls for ATM Networks. In *Proceedings of INFOSEC'COM*, 1998.

[8] D. Eppstein and S. Muthukrishnan. Internet Packet Filter Management and Rectangle Geometry. In *Proceedings of the Symposium on Discrete Algorithms*, pages 827–835, 2001.

[9] E. Fulp, Z. Fu, D. S. Reeves, S. F. Wu, and X. Zhang. Preventing Denial of Service Attacks on Quality of Service. In *Proceedings of the 2001 DARPA Information Survivability Conference and Exposition II*, 2001.

[10] R. Funke, A. Grote, and H.-U. Heiss. Performance Evaluation of Firewalls in Gigabit-Networks. In *Proceedings of the Symposium on Performance Evaluation of Computer and Telecommunication Systems*, 1999.

[11] X. Gan, T. Schroeder, S. Goddard, and B. Ramamurthy. LSMAC vs. SLNAT: Scalable ClusterbasedWeb Servers. *Journal of Networks, Software Tools, and Applications*, 3(3):175–185, 2000.

[12] S. Goddard, R. Kieckhafer, and Y. Zhang. An Unavailability Analysis of Firewall Sandwich Configurations. In *Proceedings of the 6th IEEE Symposium on High Assurance Systems Engineering*, 2001.

[13] A. Hari, S. Suri, and G. Parulkar. Detecting and Resolving Packet Filter Conflicts. In *Proceedings of IEEE INFOCOM*, pages 1203–1212, 2000.

[14] O. Paul and M. Laurent. A Full Bandwidth ATM Firewall. In *Proceedings of the 6th European Symposium on Research in Computer Security ESORICS'2000*, 2000.

[15] L. Qui, G. Varghese, and S. Suri. Fast Firewall Implementations for Software and Hardware-Based Routers. In *Proceedings of ACM SIGMETRICS*, June 2001.

[16] G. V. Rooij. Real Stateful TCP Packet Filtering in IP Filter. In *Proceedings of the 10th USENIX Security Symposium*, 2001.

[17] M. Schwartz. *Telecommunication Networks: Protocols, Modeling, and Analysis*. Addison-Wesley, 1987.

IPR2023-01353 Page 00830

[18] S. Suri and G. Varghese. Packet Filtering in High Speed Networks. In *Proceedings of the Symposium on Discrete Algorithms*, pages 969 – 970, 1999.

[19] P. Warkhede, S. Suri, and G. Varghese. Fast Packet Classification for Two-Dimensional Conflict-Free Filters. In *Proceedings of IEEE INFOCOM*, pages 1434–1443, 2001.

[20] J. Xu and M. Singhal. Design and Evaluation of a High-Performance ATM Firewall Switch and Its Applications. *IEEE Journal on Selected Areas in Communications*, 17(6):1190–1200, June 1999.

[21] R. L. Ziegler. *Linux Firewalls*. New Riders, second edition, 2002.

[22] E. D. Zwicky, S. Cooper, and D. B. Chapman. *Building Internet Firewalls*. O'Reilly, 2000.

[23] E. Al-Shaer and H. Hamed. Modeling and Management of Firewall Policies. *IEEE Transactions on Network and Service Management*, 1(1), 2004.

[24] S. M. Bellovin and W. Cheswick. Network Firewalls. *IEEE Communications Magazine*, pages 50–57, Sept. 1994.

[25] D. E. Culler and J. P. Singh. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufman, 1999.

[26] E. W. Fulp. Firewall Architectures for High Speed Networks. Technical Report 20026, Wake Forest University Computer Science Department, 2002.

[27] E. W. Fulp. Optimization of Firewall Policies Using Directed Acyclical Graphs. Submitted for publication, 2004.

[28] R. Rivest. On Self-Organizing Sequential Search Heuristics. *Communications of the ACM*, 19(2), 1976.

[29] W. E. Smith. Various Optimizers for Single-Stage Production. *Naval Research Logistics Quarterly*, 3:59 – 66, 1956.

It will be understood that various details of the present subject matter may be changed without departing from the scope of the present subject matter. Furthermore, the foregoing description is for the purpose of illustration only, and not for the purpose of limitation.

IPR2023-01353 Page 00831

2005328336 29 Jun 2007

CLAIMS

What is claimed is:

1. A function-parallel firewall system comprising:
   (a) a first firewall node for filtering received packets using a first portion of a rule set including a plurality of rules, the first portion including less than all of the rules in the rule set; and
   (b) at least one second firewall node for filtering packets using a second portion of the rule set, the second portion including at least one rule in the rule set that is not present in the first portion, wherein the first and second portions together include all of the rules in the rule set, wherein the first and second firewall nodes are configured to implement a gateless design wherein the rules are distributed among the first and second firewall nodes such that for a given packet, one of the first and second firewall nodes will accept the packet and the other of the first and second firewall nodes will deny the packet.
2. The function-parallel firewall system of claim 1 wherein the first and second portions respectively include rules present in first and second levels of a firewall hierarchy.
3. The function-parallel firewall system of claim 2 wherein the first and second firewall nodes are connected to each other such that the rules in the first level are applied to received packets before the rules in the second level.
4. The function-parallel firewall system of claim 3 wherein the first firewall node is adapted to forward packets that pass one of the rules in the first portion to an internal network in a manner that bypasses the second firewall node.
5. The function-parallel firewall system of claim 3 wherein the first firewall node is adapted to discard packets that fail one of the rules in the first portion.
6. The function-parallel firewall system of claim 3 wherein the rules in the second portion are more processor-intensive than the rules in the first portion.

7. The function-parallel firewall system of claim 1 wherein the first and second firewall nodes implement rules of the same hierarchical level.

8. The function-parallel firewall system of claim 1 wherein the first and second firewall nodes are adapted to simultaneously process copies of the same packets.

9. The function-parallel firewall system of claim 1 wherein the rule set includes intrusion detection rules for filtering packets based on packet payloads.

10. The function-parallel firewall system of claim 1 wherein the rule set includes intrusion protection rules for filtering packets based on a combination of packet headers and packet payloads.

11. The function-parallel firewall system of claim 1 wherein the first and second firewall nodes are components of a first firewall subsystem for filtering packets entering a network and wherein the system further comprises a second function-parallel firewall sub-system including a second set of firewall nodes implementing a second rule set for filtering packets departing from the network, wherein the second function-parallel firewall subsystem includes at least third and fourth firewall nodes, the third and fourth firewall nodes implementing first and second portions of the second rule set, the first portion of the second rule set including at least one rule that is not present in the second portion of the second rule set, wherein the first and second function-parallel firewall subsystems share state information regarding connections established through the first and second sets of firewall nodes.

12. The function-parallel firewall system of claim 1 wherein the first and second firewall nodes are components of a firewall grid with logically configurable interconnections between the firewall nodes.

13. A method for controlling access to a network based on a set of packet filtering rules distributed in a function-parallel manner, the method comprising:

(a) distributing packet filtering rules of a rule set among a plurality of different nodes in a function-parallel manner so that at least some of the different nodes implement different portions of the rule set;

(b) replicating packets to each of the nodes; and

(c) applying the rules to filter the packets using a gateless design wherein the rules are distributed among first and second firewall nodes such that for a given packet, one of the first and second firewall nodes will accept the packet and the other of the first and second firewall nodes will deny the packet.

14. The method of claim 13 wherein the packet filtering rules comprise intrusion detection rules for filtering packets based on packet payload information.

15. The method of claim 13 wherein the packet filtering rules comprise intrusion protection rules for filtering packets based on a combination of packet header and packet payload information.

16. The method of claim 13 comprising, receiving a first packet at a first node, determining whether the packet matches a first rule implemented by the first node, and, in response to determining that a match occurs, notifying the other nodes of the match.

17. The method of claim 16 comprising, at the other nodes, ceasing processing of the first packet.

18. A computer program product comprising computer executable instructions embodied in a computer readable medium for performing steps comprising:

(a) a first firewall node for filtering received packets using a first portion of a rule set including a plurality of rules, the first portion including less than all of the rules in the rule set; and

(b) at least one second firewall node for filtering packets using a second portion of the rule set, the second portion including at least one rule in the rule set that is not present in the first portion, wherein the first and second portions together include all of the rules in the rule set, wherein the first and second firewall nodes are configured to implement a gateless design

2005328336 22 Aug 2011

wherein the rules are distributed among the first and second firewall nodes such that for any given packet, one of the first and second firewall nodes will accept the packet and the other of the first and second firewall nodes will deny the packet

19. The method of claim 13, or computer program product of claim 18, wherein distributing the rules in the rule set among a plurality of firewall nodes comprises:

(a) assigning rules in the rule set to nodes and branches in a trie data structure, wherein each node in the trie data structure corresponds to a data field and each branch represents a value for each data field to be compared to values in corresponding fields in received packets; and

(b) pruning the trie data structure in a manner that preserves ordering of the rules in the rule set.

20. The method or product of claim 19 wherein pruning the trie data structure includes pushing a rule to its descendants and deleting the original rule.

21. The method or product of claim 19 comprising reducing the depth of the trie using level reordering.

22. The method or product of claim 19 wherein assigning rules in the rule set to nodes and branches in the trie data structure includes assigning rules requiring longer processing time to lower levels in the trie.

23. The method of claim 13 or computer program product of claim 18, wherein distributing the rules in the rule set among a plurality of firewall nodes comprises:

(a) assigning packet filtering rules to nodes in a directed acyclical graph (DAG);

(b) representing relationships between rules by edges in the DAG; and

(c) distributing the rules among the plurality of firewall nodes using the DAG such that different firewall nodes implement different rules and such that relationships between the rules specified in the DAG are

2005328336 22 Aug 2011

preserved.

24. The method or product of claim 23, wherein distributing the rules among the plurality of firewall nodes using the DAG includes distributing the rules such that intersecting accept rules are not present in more than one firewall node.

25. The method or product of claim 23, wherein distributing the rules among the plurality of firewall nodes using the DAG includes distributing the rules such that the edges to not span multiple firewall nodes.

26. The method or product of claim 23, wherein distributing the rules among the plurality of firewall nodes using the DAG includes replicating deny rules in each of the firewall nodes.

27. The method or product of claim 26, comprising eliminating a deny rule implemented by a first firewall node in response to determining that the deny rule for the first firewall node includes no inbound edges and outbound edges from the first deny rule terminate only in a superset deny rule.

28. A function-parallel firewall system, a method for controlling access to a network, or a computer program product, substantially as herein described with reference to Figures 2 to 19.

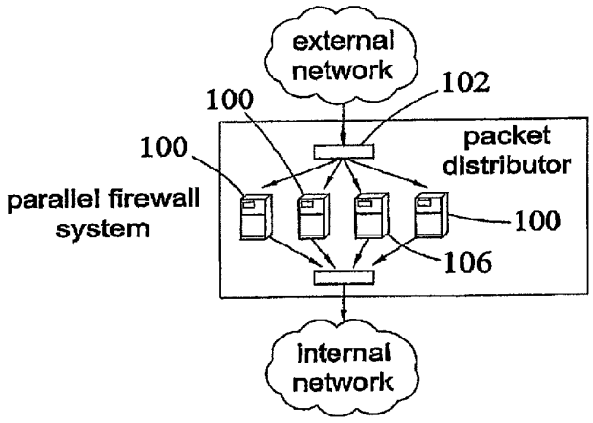


FIG. 1
(PRIOR ART)

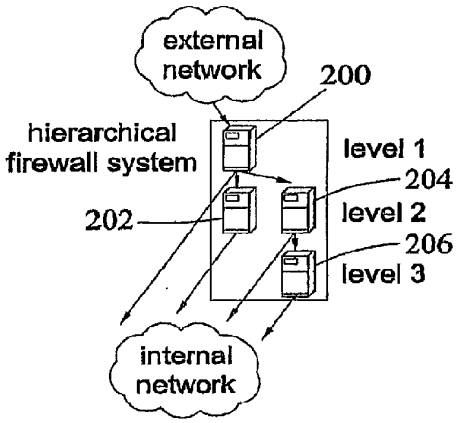

external network
200
hierarchical firewall system
level 1
204
level 2
202
206
level 3
internal network

FIG. 2

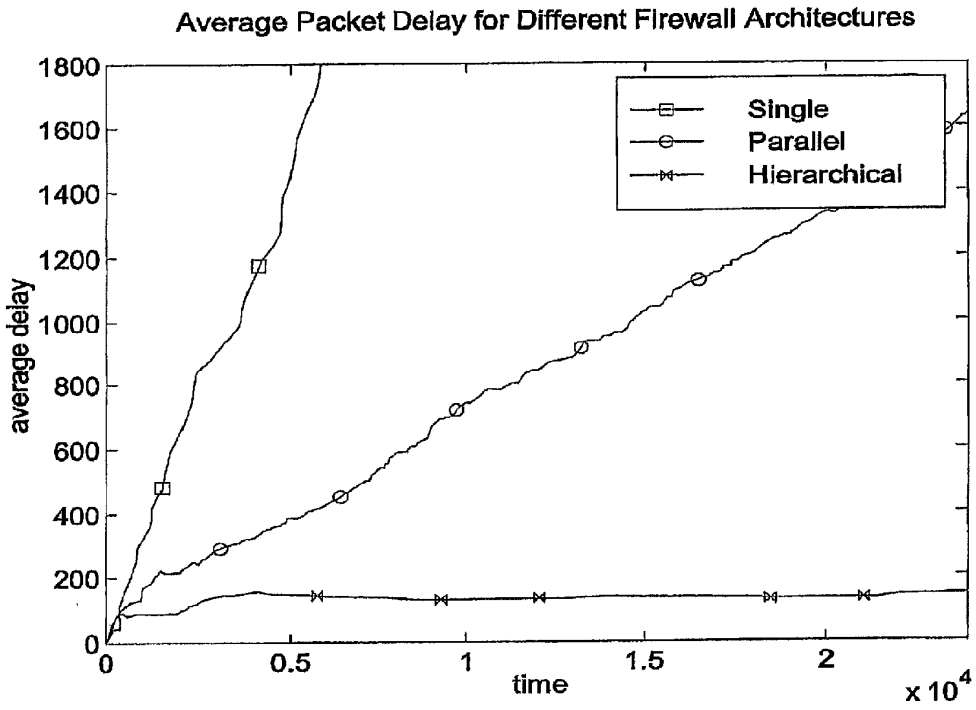

Average Packet Delay for Different Firewall Architectures
average delay
time
x 10^4
Single
Parallel
Hierarchical
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.5
1
1.5
2

FIG. 3A

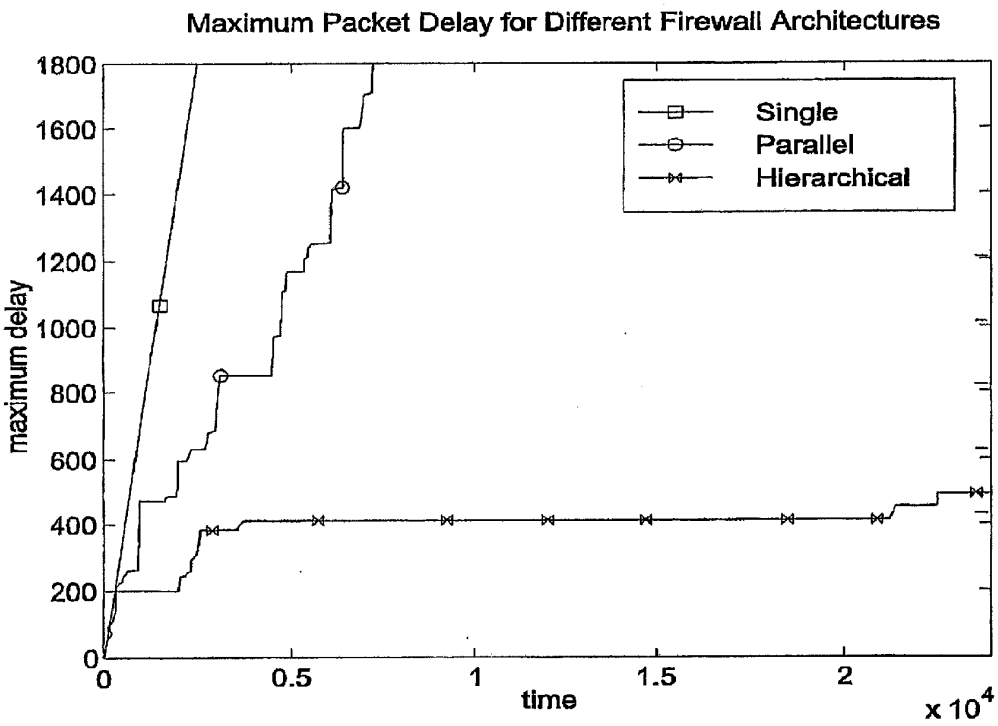

Maximum Packet Delay for Different Firewall Architectures
maximum delay
time
x 10^4
Single
Parallel
Hierarchical
0
200
400
600
800
1000
1200
1400
1600
1800
0
0.5
1
1.5
2

FIG. 3B

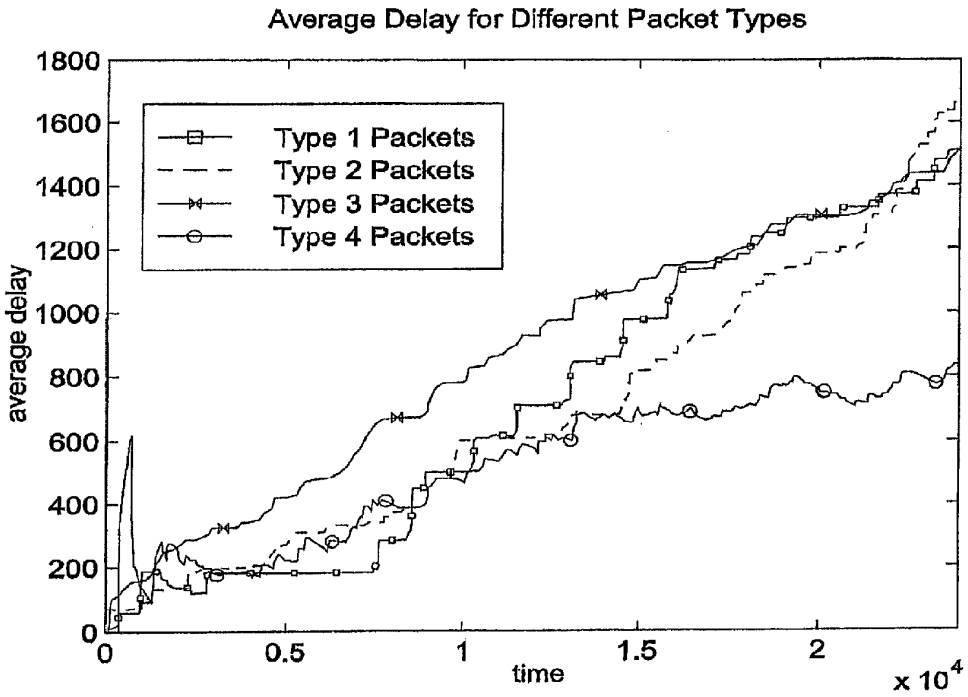

Average Delay for Different Packet Types
Type 1 Packets
Type 2 Packets
Type 3 Packets
Type 4 Packets
average delay
time
x 10^4
0
200
400
600
800
1000
1200
1400
1600
1800
0.5
1
1.5
2

FIG. 4A

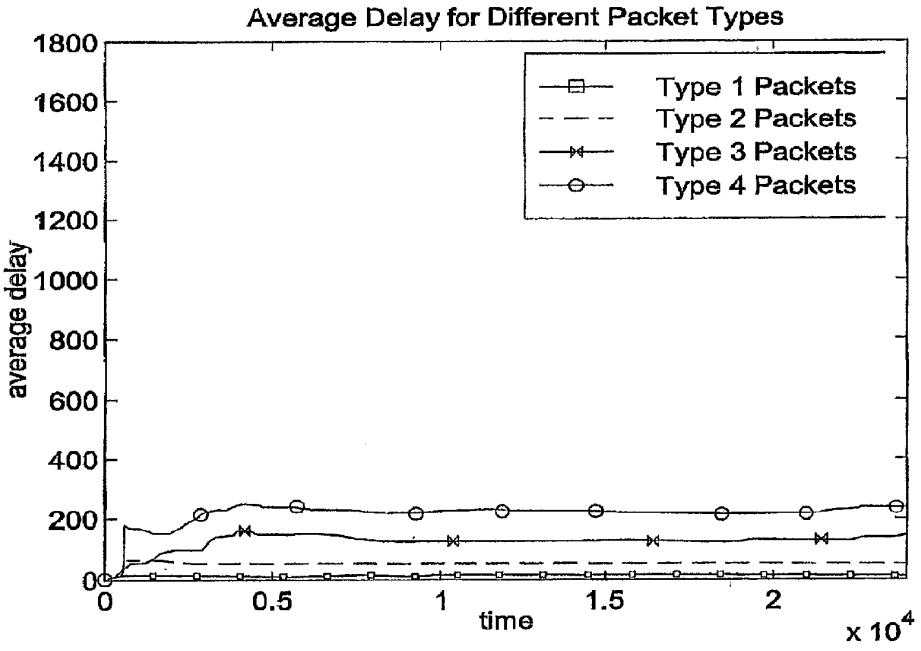

Average Delay for Different Packet Types
Type 1 Packets
Type 2 Packets
Type 3 Packets
Type 4 Packets
average delay
time
x 10^4
0
200
400
600
800
1000
1200
1400
1600
1800
0.5
1
1.5
2

FIG. 4B

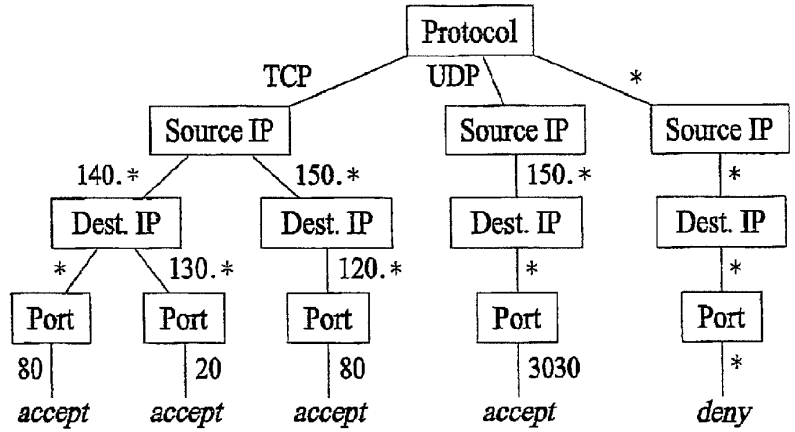

Protocol
TCP
UDP
*
Source IP
Source IP
Source IP
140.*
150.*
150.*
*
Dest. IP
Dest. IP
Dest. IP
Dest. IP
*
130.*
120.*
*
*
Port
Port
Port
Port
Port
80
20
80
3030
*
accept
accept
accept
accept
deny

FIG. 5

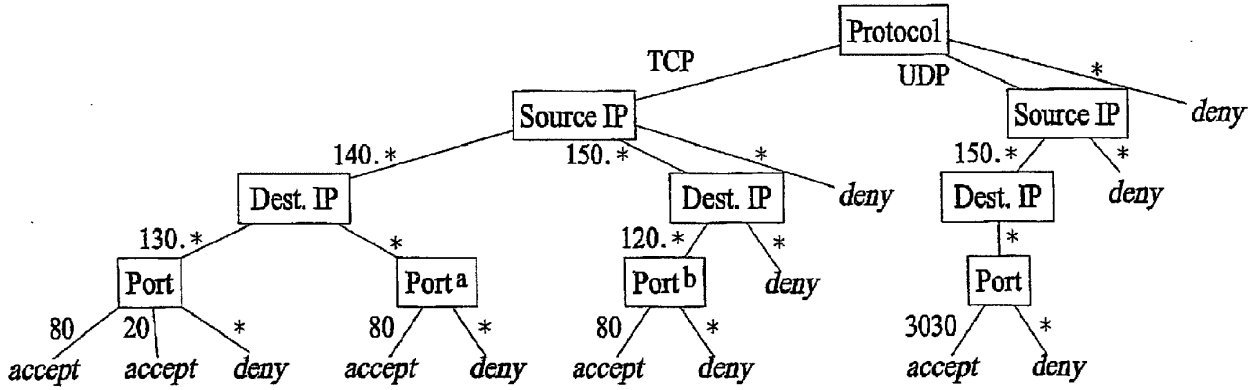

Protocol
TCP
UDP
*
Source IP
Source IP
deny
140.*
150.*
*
150.*
*
Dest. IP
Dest. IP
deny
Dest. IP
deny
130.*
*
120.*
*
*
Port
Port a
Port b
deny
Port
80
20
*
80
*
80
*
3030
*
accept
accept
deny
accept
deny
accept
deny
accept
deny

FIG. 6

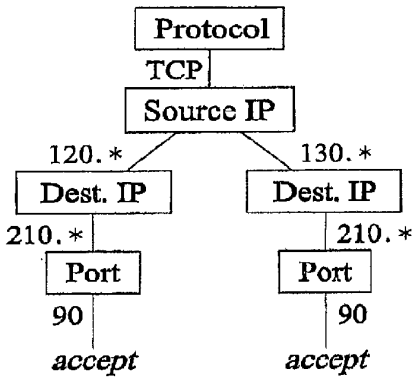

Protocol
TCP
Source IP
120.*
130.*
Dest. IP
Dest. IP
210.*
210.*
Port
Port
90
90
accept
accept

FIG. 7A

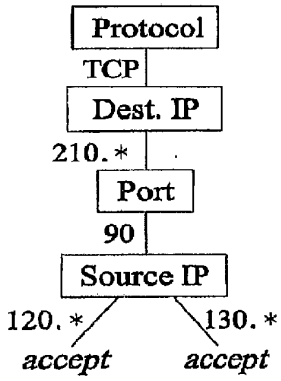

Protocol
TCP
Dest. IP
210.*
Port
90
Source IP
120.*
130.*
accept
accept

FIG. 7B

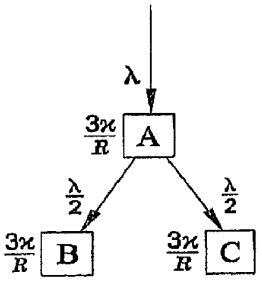

λ
3κ/R
A
λ/2
λ/2
3κ/R
B
3κ/R
C

FIG. 8A

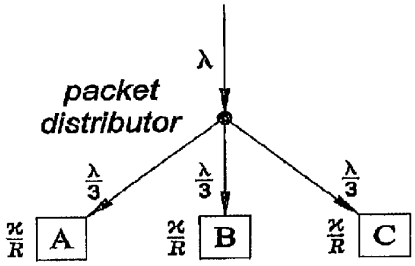

λ
packet
distributor
λ/3
λ/3
λ/3
κ/R
A
κ/R
B
κ/R
C

FIG.8B

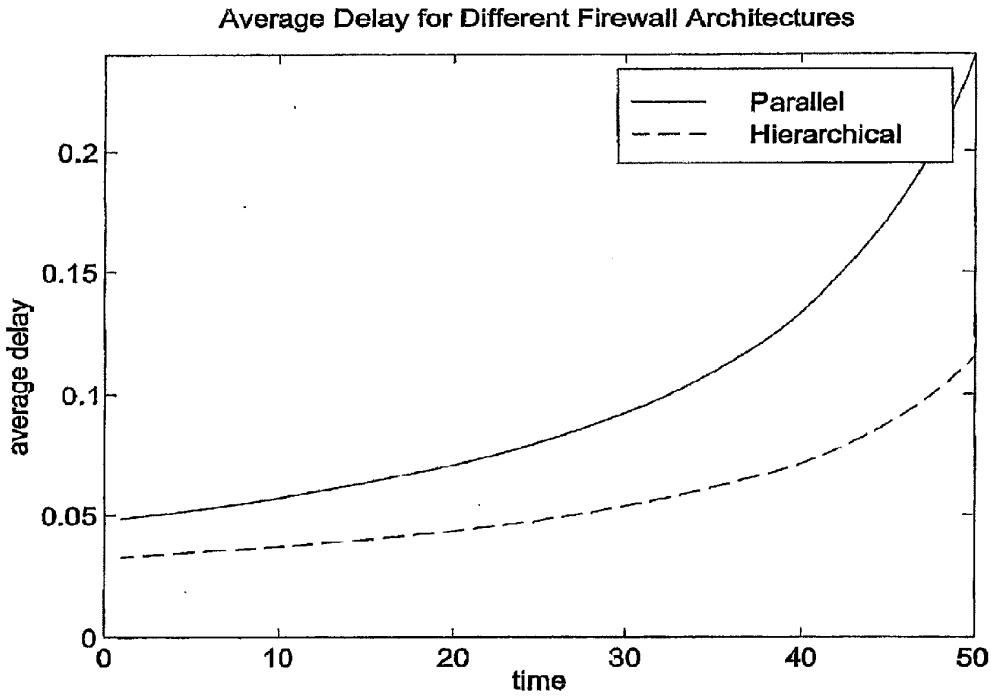

Average Delay for Different Firewall Architectures
Parallel
Hierarchical
average delay
0
0.05
0.1
0.15
0.2
0
10
20
30
40
50
time

FIG. 9

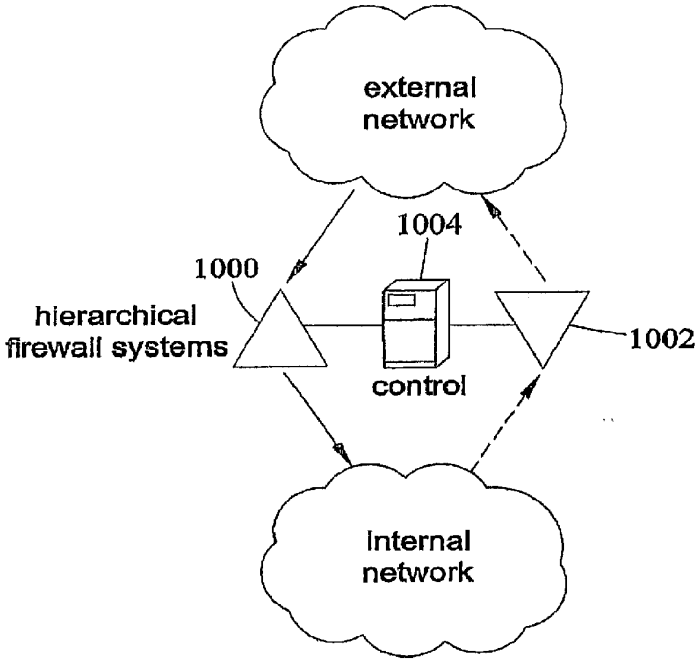

external network
1004
1000
hierarchical firewall systems
1002
control
internal network

FIG. 10

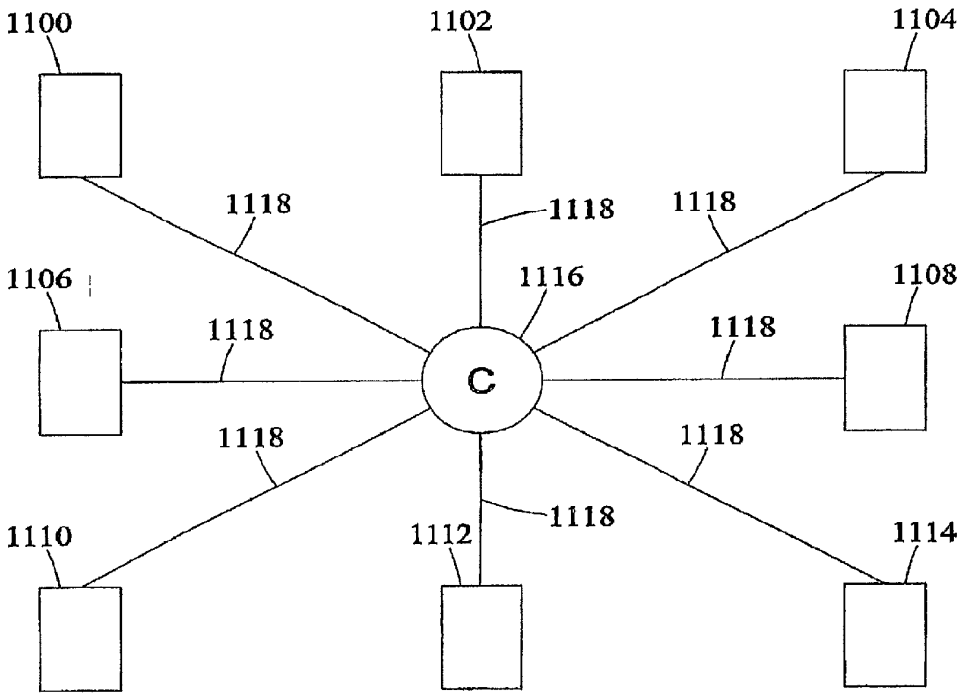

1100
1102
1104
1118
1118
1118
1106
1116
1108
1118
1118
C
1118
1118
1118
1110
1112
1114

FIG. 11A

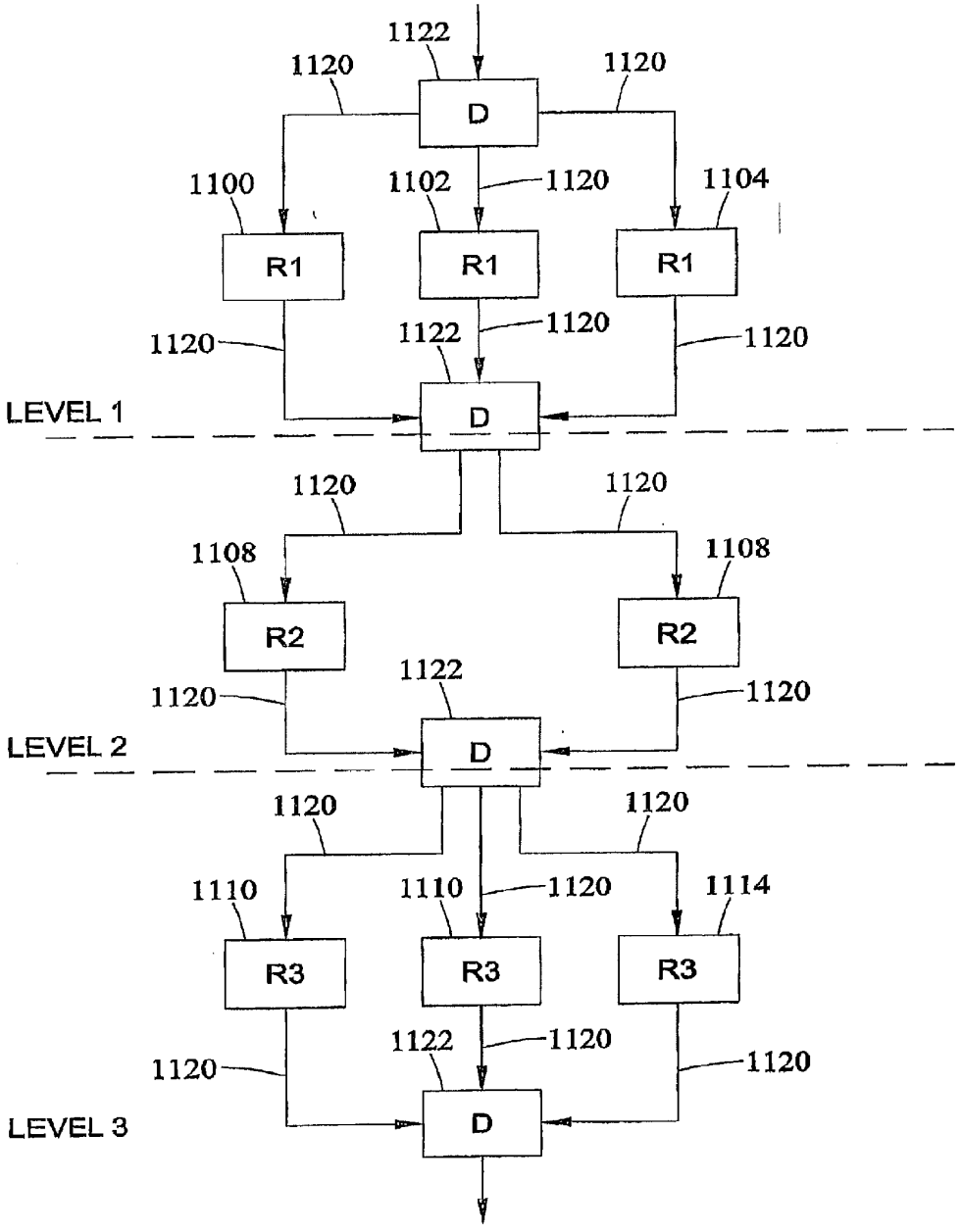

1122
1120
1120
D
1100
1102
1120
1104
R1
R1
R1
1120
1122
1120
1120
D
LEVEL 1
1120
1120
1108
1108
R2
R2
1122
1120
1120
D
LEVEL 2
1120
1120
1110
1110
1120
1114
R3
R3
R3
1122
1120
1120
1120
D
LEVEL 3

FIG. 11B

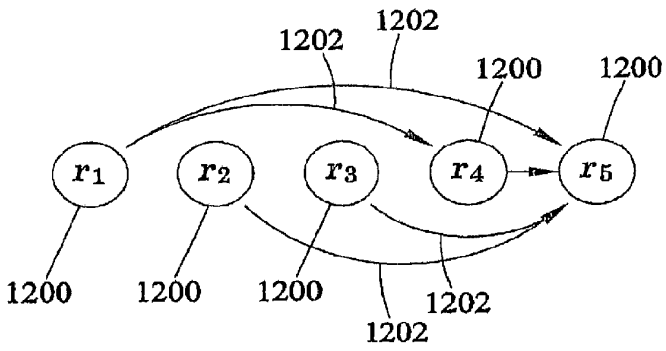


FIG. 12

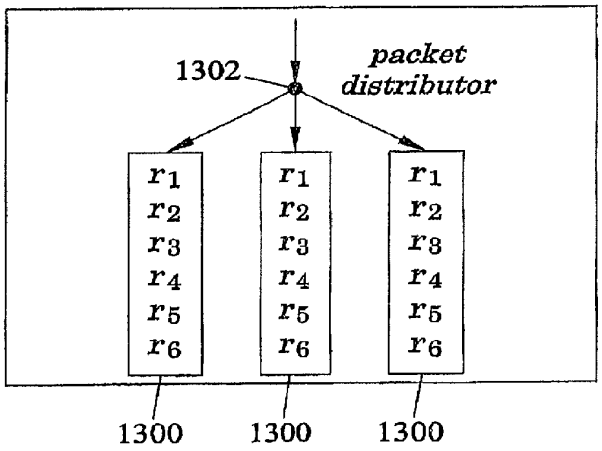


FIG. 13A

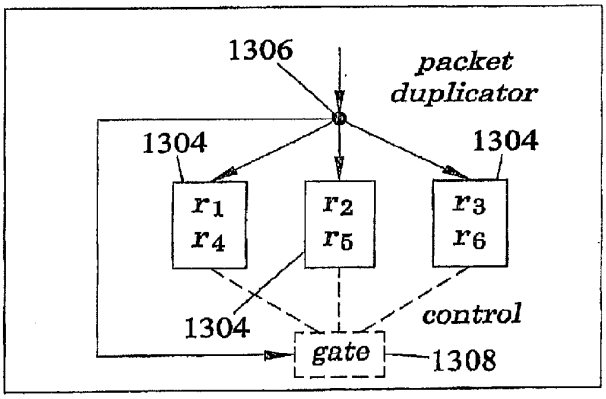


FIG. 13B

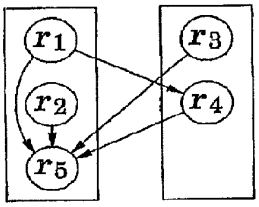


FIG. 14A

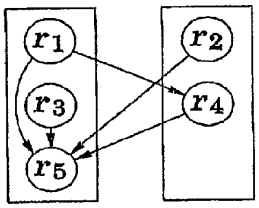


FIG. 14B

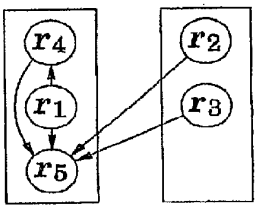


FIG. 14C

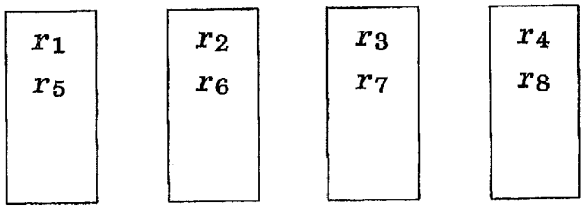


FIG. 15A

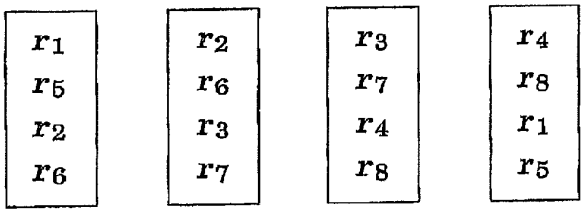


FIG. 15B

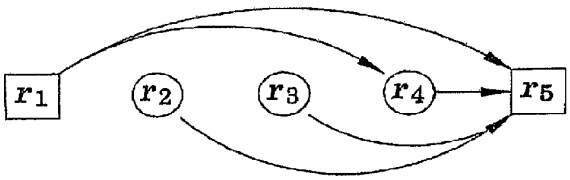


FIG. 16A

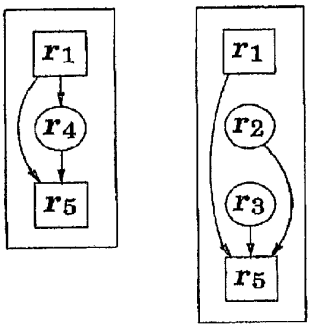


FIG. 16B

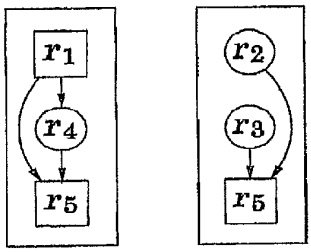


FIG. 16C

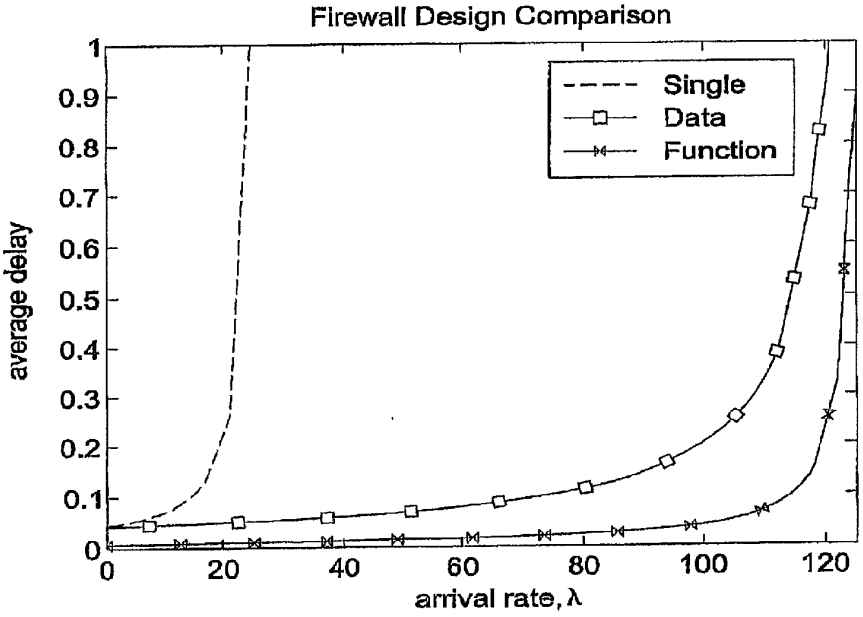

Firewall Design Comparison
Single
Data
Function
average delay
arrival rate, λ
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
20
40
60
80
100
120

FIG. 17A

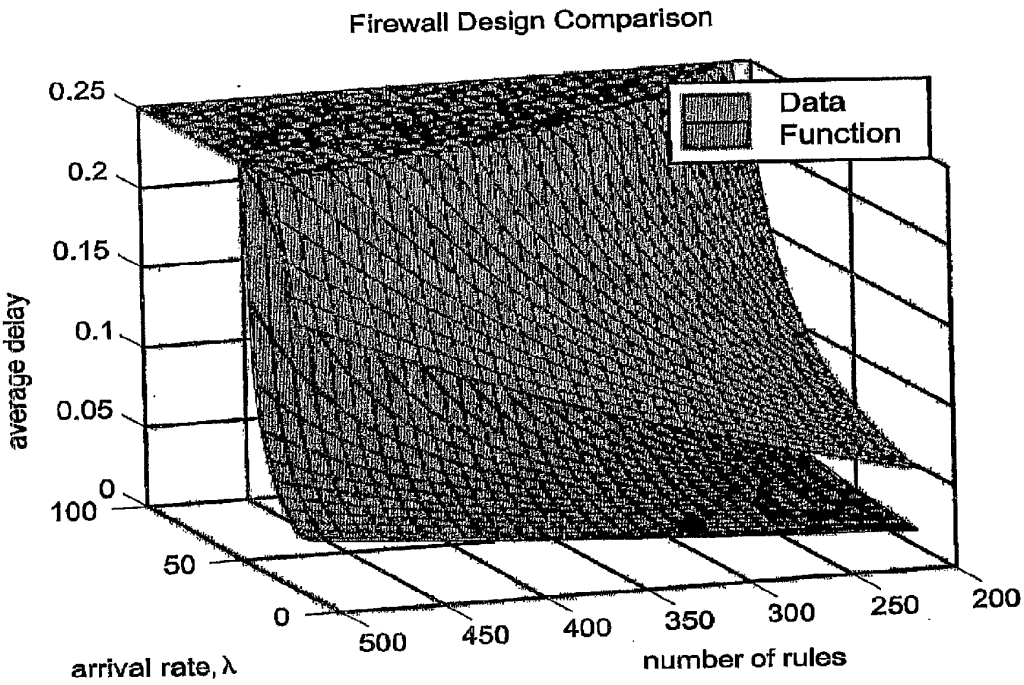

Firewall Design Comparison
Data
Function
0.25
0.2
0.15
0.1
0.05
0
average delay
100
50
0
arrival rate, λ
500
450
400
350
300
250
200
number of rules

FIG. 17B

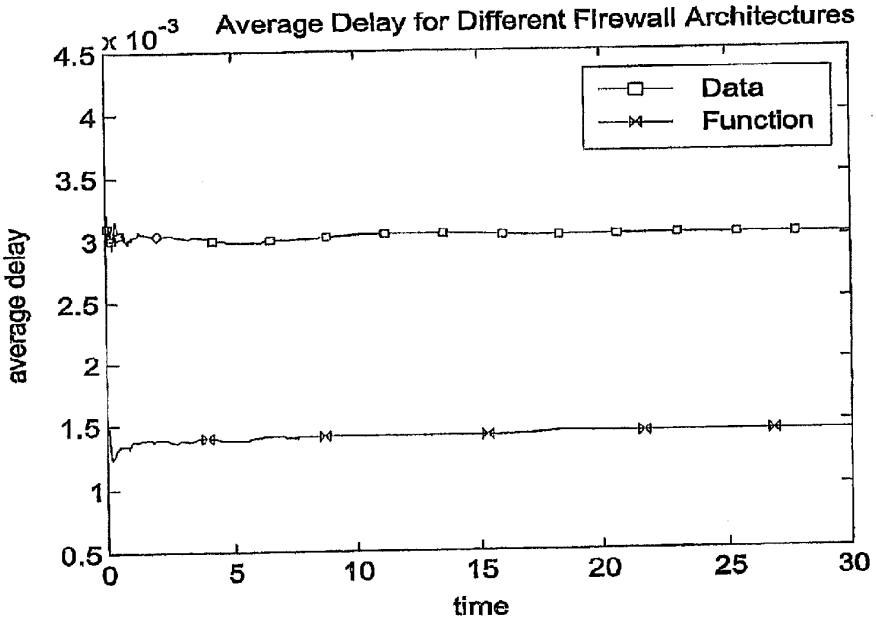

Average Delay for Different Firewall Architectures
x 10^-3
4.5
4
3.5
3
2.5
2
1.5
1
0.5
average delay
0
5
10
15
20
25
30
time
Data
Function

FIG. 18A

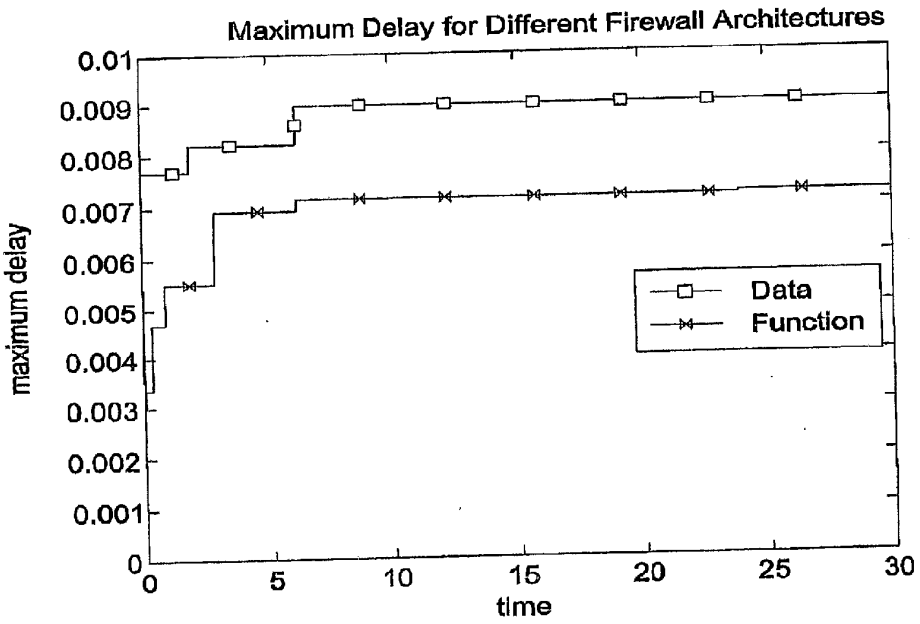

Maximum Delay for Different Firewall Architectures
maximum delay
time
Data
Function
0.01
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
0
5
10
15
20
25
30

FIG. 18B

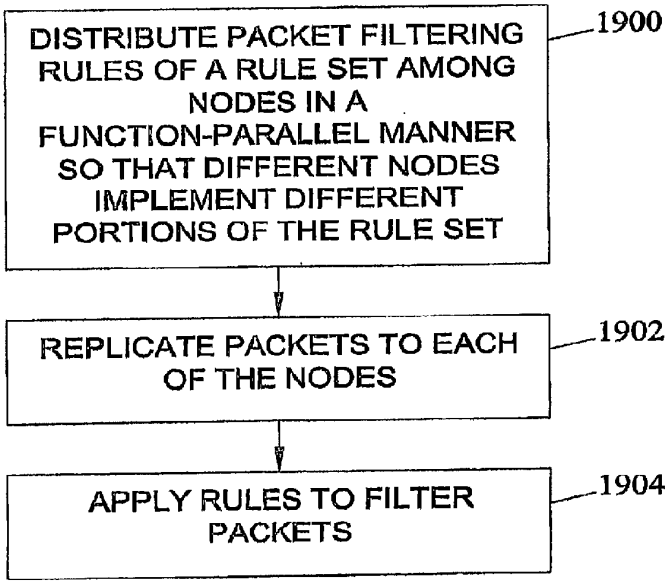

DISTRIBUTE PACKET FILTERING RULES OF A RULE SET AMONG NODES IN A FUNCTION-PARALLEL MANNER SO THAT DIFFERENT NODES IMPLEMENT DIFFERENT PORTIONS OF THE RULE SET
1900
REPLICATE PACKETS TO EACH OF THE NODES
1902
APPLY RULES TO FILTER PACKETS
1904

FIG. 19

Espacenet

Bibliographic data: WO2011038420 (A2) — 2011-03-31

METHODS, SYSTEMS, AND COMPUTER READABLE MEDIA FOR ADAPTIVE PACKET FILTERING

Inventor(s): AHN DAVID K [US] ± (AHN, DAVID, K)

Applicant(s): GREAT WALL SYSTEMS [US]; AHN DAVID K [US] ± (GREAT WALL SYSTEMS, ; AHN, DAVID, K)

Classification: **- international:*H04L12/22; H04L12/26; H04L12/56; H04L29/06***
- cooperative: G06F21/562; H04L63/0227; H04L63/0263; H04L63/1416

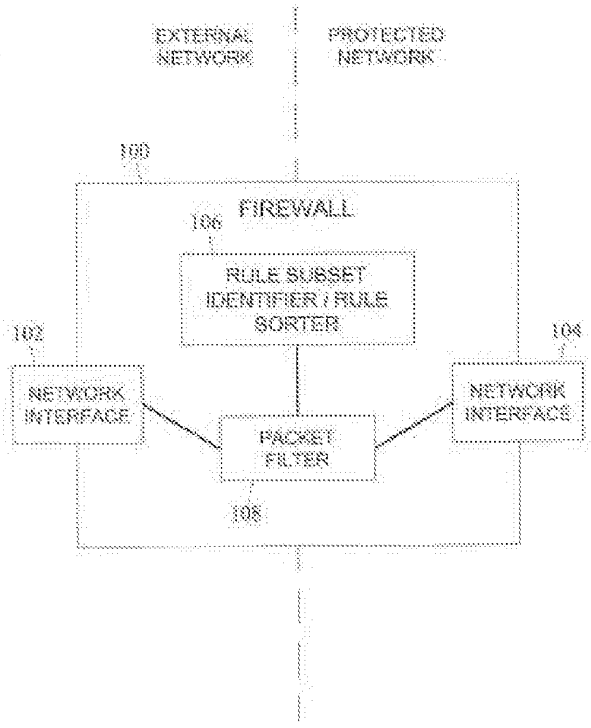
Application number: WO2010US54520 20101028

Priority number (s): US20090237974P 20090828 ; US20100871806 20100830

Also published as: WO2011038420 (A3) US2011055916 (A1) US8495725 (B2) US2014007216 (A1) EP2471218 (A2) more

Abstract of WO2011038420 (A2)

The subject matter described herein includes methods, systems, and computer readable media for adaptive packet filtering. One method includes identifying at least one subset of rules and an ordered set of firewall packet filtering rules that defines a firewall policy such that the subset contains disjoint rules. Disjoint rules are defined as rules whose order can be changed without changing integrity of the firewall policy. Rules in the subset are sorted to statistically decrease the number of comparisons that will be applied to each packet that a firewall encounters.; Packets are filtered at the firewall using the sorted rules in the subset by comparing each packet to each of the sorted rules in the subset until the packet is allowed or denied and ceasing the comparing for the packet in response to the packet being allowed or denied and



thereby achieving sub-linear searching for packets filtered using the sorted rules in the subset.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 March 2011 (31.03.2011)

PCT

(10) International Publication Number
WO 2011/038420 A2

(51) International Patent Classification:
H04L 12/22 (2006.01) *H04L 12/56* (2006.01)
H04L 12/26 (2006.01) *H04L 29/06* (2006.01)

(21) International Application Number:
PCT/US2010/054520

(22) International Filing Date:
28 October 2010 (28.10.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/237,974 28 August 2009 (28.08.2009) US
12/871,806 30 August 2010 (30.08.2010) US

(71) Applicant *(for all designated States except US)*: **GREAT WALL SYSTEMS** [US/US]; 635 W 4th St., Ste 200, Winston-Salem, NC 27101-2740 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **AHN, David, K.** [US/US]; 740 N. Pine Valley Road, Winston-salem, NC 27106-5508 (US).

(74) Agent: **HUNT, Gregory, A.**; Jenkins, Wilson, Taylor & Hunt, P.A., Suite 1200, University Tower, 3100 Tower Boulevard, Durham, NC 27707 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK,

[Continued on next page]

(54) Title: METHODS, SYSTEMS, AND COMPUTER READABLE MEDIA FOR ADAPTIVE PACKET FILTERING

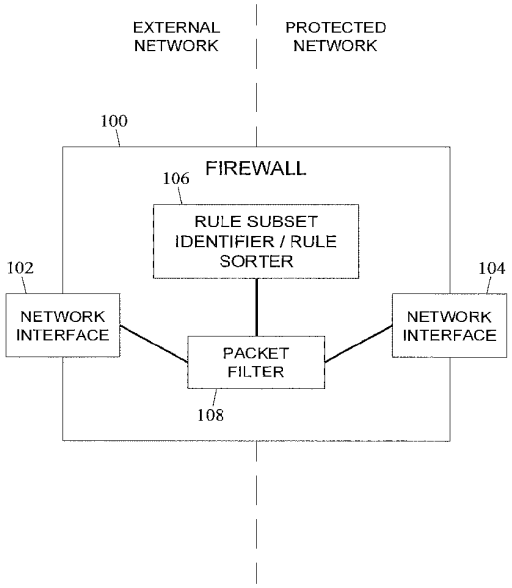


FIG. 1

(57) Abstract: The subject matter described herein includes methods, systems, and computer readable media for adaptive packet filtering. One method includes identifying at least one subset of rules and an ordered set of firewall packet filtering rules that defines a firewall policy such that the subset contains disjoint rules. Disjoint rules are defined as rules whose order can be changed without changing integrity of the firewall policy. Rules in the subset are sorted to statistically decrease the number of comparisons that will be applied to each packet that a firewall encounters. Packets are filtered at the firewall using the sorted rules in the subset by comparing each packet to each of the sorted rules in the subset until the packet is allowed or denied and ceasing the comparing for the packet in response to the packet being allowed or denied and thereby achieving sub-linear searching for packets filtered using the sorted rules in the subset.

WO 2011/038420 A2

SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

— *with information concerning request for restoration of the right of priority in respect of one or more priority claims (Rules 26bis.3 and 48.2(b)(vii))*

DESCRIPTION

METHODS, SYSTEMS, AND COMPUTER READABLE MEDIA FOR ADAPTIVE PACKET FILTERING

PRIORITY CLAIM

This application claims the benefit of U.S. Provisional Patent Application Serial No. 61/237,974, filed August 28, 2009 and U.S. Patent Application Serial No. 12/871,806, filed August 30, 2010; the disclosures of which are incorporated herein by reference in their entireties.

TECHNICAL FIELD

The subject matter described herein relates to network firewall filtering. More particularly, the subject matter described herein relates to methods, systems, and computer readable media for adaptive packet filtering.

BACKGROUND

A firewall generally processes a packet against a list of ordered rules to find the first rule match. The list of ordered rules represents an aggregate security policy, and arbitrarily changing the order of the rules can result in a violation of the aggregate security policy. The Wake Forest University (WFU) techniques described in U.S. patent application publication nos. 2006/0248580 and 2006/0195896 provide the methods to optimally reorder the list while preserving the aggregate security policy, thereby improving the performance of the firewall. The WFU techniques also include methods to break apart rules into functionally independent lists containing (groups of) dependent rules such that a function parallel firewall can simultaneously process one packet against multiple lists, which can substantially improve the performance of the firewall. However, these improvements provided by WFU techniques can be dwarfed by the performance degradation as the number of rules in the list becomes very large.

A key reason for the lack of scalability of most firewall implementations is due to the common use of linear search algorithms for comparing packets against a list of rules. In the worst case, a packet is matched at the last N^{th} rule in the list, so it must also be compared against all *N-1* prior rules for a total of *N*

IPR2023-01353 Page 00864

comparisons. This poses a computational resource problem when the size of *N* is very large on a single processing node (including when such nodes are arranged in a data-, function-, hierarchical- or hybrid-parallel system), where the time required for processing each packet can quickly increase latency and reduce throughput to unacceptable levels. In fact, the WFU techniques provide good results in part because the reordering of, or the reduction in size of, rules on each processing node allows for a larger percentage of the total rules to reside in each processor's cache(s), which then substantially increases their performance relative to when only a small portion of those rules are cached.

The problem of searching firewall rule sets is well understood and highly researched, and there are some published techniques for sub-linear (substantially faster than linear) techniques applicable to firewall rules. However, these sub-linear techniques generally involve changing the underlying representation of rules. Examples of such an approach might be to use a graph, trie- or tree-like structure instead of a list to represent a set of rules, which would allow a match to be determined using tree search algorithms by traversing down the graph, trie or tree (see E. Fulp, Trie-Based Policy Representations for Network Firewalls, Proceedings of the IEEE International Symposium on Computer Communications, 2005 and Al-Shaer et al., Modeling and Management of Firewall Policies, IEEE Transactions on Network and Service Management, 2004) . These approaches have potential but can add complexity or limitations that may reduce their practical usefulness in a commercial high performance firewall product.

SUMMARY

Adaptive packet filtering (APF), a set of techniques for processing firewall rules and packets, is described herein. APF offers improved processing performance compared to the WFU techniques in most cases, and can be combined with the WFU techniques or other parallel, pipelining and optimization techniques to achieve even greater performance.

The subject matter described herein includes methods, systems, and computer readable media for adaptive packet filtering. One method includes identifying at least one subset of rules in an ordered set of firewall packet

IPR2023-01353 Page 00865

filtering rules that defines a firewall policy such that the subset contains disjoint rules. Disjoint rules are defined as rules whose order can be changed without changing the integrity of the firewall policy. Rules in the subset are sorted to statistically decrease the number of comparisons that will be applied to each packet that a firewall encounters. Packets are filtered at the firewall using the sorted rules in the subset by using binary search, interpolated search, informed search, or hash lookup search algorithms to compare each packet to the sorted rules in the subset until the packet is allowed or denied and ceasing the comparing for the packet in response to the packet being allowed or denied and thereby achieving sub-linear searching for packets filtered using the sorted rules in the subset.

The subject matter described herein for adaptive packet filtering can be implemented in a non-transitory computer readable medium having stored thereon executable instructions that when executed by the processor of a computer control the computer to perform steps. Exemplary computer readable media suitable for implementing the subject matter described herein include chip memory devices, disk memory devices, programmable logic devices and application specific integrated circuits. In addition, a computer readable medium that implements a subject matter described herein may be located on a single device or computing platform or may be distributed across multiple devices or computing platforms.

Further, the subject matter described herein for adaptive packet filtering can be implemented on a particular machine, such as a network firewall including one or more network interfaces for receiving packets and packet filtering hardware and software for optimizing rules as described herein and for filtering packets using the optimized arrangement of rules.

BRIEF DESCRIPTION OF THE DRAWINGS

Preferred embodiments of the subject matter described herein will now be described with reference to the accompanying drawings of which:

Figure 1 is a block diagram of a system for adaptive packet filtering according to an embodiment of the subject matter described herein;

Figure 2 is a block diagram of application of the present subject matter to a pipelined processing approach according to an embodiment of the subject matter described herein;

Figure 3 is a block diagram illustrating application of the present subject matter to a combination of pipelined and data parallel processing approaches according to an embodiment of the subject matter described herein; and

Figure 4 is a block diagram illustrating application of the present subject matter to a short-circuiting pipelined processing approach according to an embodiment of the subject matter described herein; and

Figure 5 is a block diagram illustrating application of the present subject matter to a combination of pipelined and function parallel processing approaches according to an embodiment of the subject matter described herein.

DETAILED DESCRIPTION

Methods, systems, and computer readable media for adaptive packet filtering are disclosed. Figure 1 is a block diagram illustrating an exemplary system for adaptive packet filtering according to an embodiment of the subject matter described herein. Referring to Figure 1, a firewall **100** may function at the boundary between an external network and a protected network. Firewall **100** may include one or more network interfaces **102** for receiving packets from the external network. Firewall **100** may also include one or more network interfaces **104** for transmitting allowed packets to the protective network. In one implementation, firewall **100** may filter Internet protocol (IP) packets based on a combination of source and destination addresses in the IP headers of the packets. However, the subject matter described herein is not limited to filtering any particular protocol. Any packet network protocol with parameters for which firewall filtering rules can be defined is intended to be within the scope of the subject matter described herein.

As used herein, the term "firewall" includes any network security device or system of devices that inspects network traffic data that originates, terminates, or traverses the device system in any capacity and compares that

traffic data (headers, payload, raw bits, etc.) to a set of one or more rules, signatures, or conditions, either inline (i.e., in real time) or offline (i.e., capture and replay of the traffic data). The term "firewall" is also intended to include an intrusion detection device that analyzes network traffic in real time or historically to detect the presence of intrusion events in a network. The term "firewall" is also intended to include a deep packet inspection device that analyzes network traffic in real time or historically to detect the presence of certain packet content in a network.

Firewall **100** includes a firewall rule subset identifier/rule sorter **106** for identifying at least one subset of rules in an ordered set of firewall packet filtering rules that defines a firewall policy such that the subset contains disjoint rules, where disjoint rules are defined as rules whose order can be changed without changing the firewall policy. Firewall rule subset identifier/rule sorter **106** may sort the rules in the subset or subsets to statistically decrease the number of comparison that will be applied to each packet that the firewall encounters. Exemplary methods for grouping and sorting rules will be described below.

Although in the example illustrated in Figure 1 rule subset identifier/rule sorter **106** is illustrated as a component of firewall **100**, the subject matter described herein is not limited to such an implementation. Rule subset identifier/rule sorter **106** can be implemented on any computing platform capable of sorting firewall rules using the methods described herein, and the sorted rule set can be provided to firewall **100** through any suitable means, such as communication over a network. In one implementation, rule subset identifier/rule sorter **106** may be implemented on a management platform separate from firewall **100**.

Firewall **100** further includes a packet filter **108** for filtering packets at the firewall using the rules in the subset by using binary search, interpolated search, informed search, hash lookup search algorithms, or other sub-linear algorithms to compare each packet to each of the sorted rules in the subset until the packet is allowed or denied and ceasing the comparing for the packet in response to the packet being allowed or denied and thereby achieving

sub-linear searching for the packets filtered using the sorted rules in the subset.

Once the subsets of disjoint rules have been identified by rule subset identifier/rule sorter **106**, the rule subsets can be distributed across plural firewall processors in order to improve packet filtering efficiency. Figure 2 is a block diagram illustrating an exemplary pipelined approach where rules and different subsets are distributed across plural firewall processors for processing packets in a pipelined manner. Referring to Figure 2, firewalls **200** and **202** each include separate processors **204** and **206** for executing packet filters **108**. In this example, rule subset identifier/rule sorter **106** identifies two rule subsets, subset A **208** and subset B **210**. The rules within each subset **208** and **210** are disjoint and sorted to statistically decrease the number of comparisons that will be applied to each packet using the methods described herein. However, the rules in subset B **210** are dependent on the rules in rule subset A **208**. Accordingly, rule subset identifier/rule sorter **106** distributes the rule across firewall processors **204** such that the rules in rule subset A **208** are applied before the rules in rule subset B **210**. Because the rules in different subsets are distributed across plural processors in a pipeline manner, packet filtering efficiency is improved over a single-processor approach because the different processors can simultaneously apply rules to different packets. In the example illustrated in Figure 2, packets that pass the filtering of rule subset A **208** are processed by processor **206**, which applies rule subset B **210**, at the same time that processor **204** applies rule subset A to new incoming packets.

In yet another embodiment, rule subset identifier/rule sorter **106** may distribute the grouped, sorted rules across firewall processors such that a combination of pipelined and data parallel processing techniques are used. Figure 3 illustrates an example where firewalls **300**, **302**, **304**, and **306** each include separate processors **308**, **310**, **312**, and **314** for applying their respective packet filters. In the illustrated example, rule subset identifier/rule sorter **106** distributes rule subset A **316** to firewall **300**, rule subset B **318** to firewalls **302** and **304**, and rule subset C **320** to firewall **306**. The rules within each subset A, B and C are disjoint. The rules in subset B are dependent

IPR2023-01353 Page 00869

upon the rules in subset A. The rules in subset C are dependent upon the rules in subsets A and B.

In operation, packets entering firewall **300** are filtered using rule subset A **316**. The packets that are allowed by rule subset A **316** are divided between firewalls **302** and **304** such that the application of the rules in rule subset B **318** to different packets is performed in parallel. This is referred to as a data parallel approach. The packets that pass the filtering by rule subset B **318** are passed to firewall **306** for application of the rules in rule subset C **320**. Accordingly, Figure 3 illustrates an example where the rule subsets that are identified and sorted by rule subset identifier/rule sorter **106** are distributed across the firewall processors for a combination of pipelined and data parallel processing.

In yet another embodiment, the rules subsets that are identified and in which the rules are sorted using rule subset identifier/rule sorter **106** may be distributed across firewall processors in a short-circuiting pipelined manner. Figure 4 is an example of short-circuiting pipelined filtering using rule subsets that are identified and sorted by rule subset identifier/rule sorter **106**. Referring to Figure 4, a first firewall **400** and a second firewall **402,** respectively including processors **404** and **406,** filter packets using packet filters **108**. In the illustrated example, packet filter **108** uses rule subset A **408** and packet filter **108** uses rule subset B **410**. Rule subsets A and B **408** and **410** may respectively implement different levels of a firewall hierarchy such that packets that pass the filtering by rule subset A **408** are allowed into the protected network. Packets that are identified by rule subset A **408** is requiring further filtering are distributed to rule subset B **410** for that filtering. Thus, rule subset identifier/rule sorter **106** can also be used with short-circuiting pipelined firewall techniques without departing from the scope of the subject matter described herein.

In yet another embodiment, rule subset identifier/rule sorter **106** may distribute the grouped, sorted rules across firewall processors such that a combination of pipelined and function parallel processing techniques are used. Figure 5 illustrates an example where firewalls **500**, **502**, **504**, and **506** each include separate processors **508**, **510**, **512**, and **514** for applying their

respective packet filters. In the illustrated example, rule subset identifier/rule sorter **106** distributes rule subset A **516** to firewall **500**, rule subset B **518** to firewalls **502**, and rule subset C **520** to firewall **504**, and rule subset D **522** to firewall **506**. The rules within each subset A, B, C and D are disjoint. The rules in subset B and C are dependent upon the rules in subset A. The rules in subset D are dependent upon the rules in subsets A, B and C.

In operation, packets entering firewall **500** are filtered using rule subset A **516**. The packets that are allowed by rule subset A **516** are copied to both firewalls **502** and **504** such that the application of the rules in rule subsets B **518** and C **520** to the packets is performed in parallel. This is referred to as a function parallel approach. The packets that pass the filtering by rule subsets B **518** and C **520** are passed to firewall **506** for application of the rules in rule subset D **522**. Accordingly, Figure 5 illustrates an example where the rule subsets that are identified and sorted by rule subset identifier/rule sorter **106** are distributed across the firewall processors for a combination of pipelined and function parallel processing.

Technique

APF analyzes and orders the list of firewall rules in-place to contain functionally dependent groups, where each group contains a subset of rules that are disjoint, dependent or both, without substantially changing the underlying representation of rules and while preserving the aggregate security policy. APF then uses varying criteria to sort each group containing disjoint rules, then uses sub-linear search algorithms when comparing packets against the rules within that group. APF uses linear search algorithms when comparing packets within a group containing dependent rules or when otherwise appropriate. A detailed computational complexity analysis of APF would need to be completed. However, on average, it is hypothesized that only $O(log(N))$ comparisons would be needed to process a rule list of size N. In the theoretical best case when all rules are disjoint, this translates to about 20 comparisons (instead of 1,000,000) for a list of N = 1,000,000 rules and about 30 comparisons for a list of N = 1,000,000,000 rules. In the worst case when all rules are dependent, APF performs the same as linear search firewall

cores. In practice, APF should process a packet against a very large list of rules (*N* = millions) in the same amount of time that other techniques can process against a very small list (*N* = hundreds, or thousands). APF does not inherently use parallel techniques; therefore, it can be combined with WFU techniques or other parallel/pipelining techniques to increase performance.

The following table shows preliminary results comparing a single linear search firewall) with a single APF core as the number of rules is increased.

	linear firewall core		APF core	
N = # rules in the core	PPS = packets per second at max throughout with 0 loss	Latency in microseconds at max throughput with 0 loss	PPS = packets per second at max throughout with 0 loss	Latency in microseconds at max throughput with 0 loss
1	844,595	12.3	811,688	12.3
10	730,994	11.9	766,871	12.3
100	314,861	13.9	718,391	12.0
1,000	29,357	44.6	683,060	12.0
10,000	930	1,112.4	464,684	12.0
100,000	Fail	Fail	292,740	17.4
1,000,000			252,525	14.3
10,000,000			132,556	18.6

Detailed Technique

This section describes an exemplary algorithm for implementing the subject matter described herein.

A firewall *rule* is defined as an *n*-tuple criteria and an associated action for matching packets. For example, a 5-tuple rule that matches Internet Protocol version 4 (IPv4) packets might consist of 5 IPv4 header fields (source address, source port, destination address, destination port and protocol) and an action (allow, deny), and might specify the rule ***R1*** as:

Rule	Source Addr	Source Port	Dest Addr	Dest Port	Protocol	Action
R1	192.168.1.1	12345	10.1.1.1	80	TCP	DENY

A firewall *rule set* is defined as an ordered list of *n* rules *R1,R2,R3,...,Rn* where the *i* in *Ri* is the index of the rule in the list. Packets that traverse the firewall are checked against each rule in the rule set until the

first matching rule is found and its associated action is applied. An example rule set is ***S1*** which contains:

Rule	Source Addr	Source Port	Dest Addr	Dest Port	Protocol	Action
R1	192.168.1.1	12345	10.1.1.1	80	TCP	DENY
R2	192.168.2.2	ANY	10.2.2.2	25	TCP	ALLOW
R3	192.168.3.3	ANY	10.3.3.3	53	UDP	ALLOW
R4	ANY	ANY	10.1.1.1	ANY	ANY	ALLOW
R5	ANY	ANY	ANY	ANY	ANY	DENY

An example TCP packet from source 192.168.4.4 port 54321 to destination 10.1.1.1 port 80 would be checked against but not match *R1*, *R2* and *R3*; would be checked against and match *R4* and be allowed; and, would not be checked against *R5* because *R4* was the first matching rule.

A firewall *security policy* is defined as the set of all possible packets that can traverse the firewall along with their specified outcomes as defined by the rule set. Changing the rules in a rule set usually results in a change of its security policy.

Within a rule set, a firewall rule is *dependent* on another rule if swapping the order of the two rules results in a change in the security policy of the rule set. Otherwise, the two rules are *disjoint* if swapping the order does **not** result in a change the security policy. For example, in rule set ***S1*** above, rules ***R1*** and ***R4*** are dependent because placing ***R4*** ahead of ***R1*** would render ***R1*** ineffective, thereby changing the security policy. Rules ***R1*** and ***R2*** are disjoint because placing ***R2*** ahead of ***R1*** does not change the security policy.

A *permutation* of a rule set is defined as a new rule set which contains the same rules as the original rule set, but which lists a different ordering of the rules from the original rule set without changing the original security policy. For example, in the rule set ***S1*** above, swapping the order of the disjoint rules ***R1*** and ***R2*** would result in a permutation rule set ***S1'***:

Rule	Source Addr	Source Port	Dest Addr	Dest Port	Protocol	Action
R2	192.168.2.2	ANY	10.2.2.2	25	TCP	ALLOW
R1	192.168.1.1	12345	10.1.1.1	80	TCP	DENY
R3	192.168.3.3	ANY	10.3.3.3	53	UDP	ALLOW

R4	ANY	ANY	10.1.1.1	ANY	ANY	ALLOW
R5	ANY	ANY	ANY	ANY	ANY	DENY

Two rules are *spatially disjoint* if they are disjoint and their corresponding tuples are either identical or do not overlap. For example, in the rule set ***S1*** above, rules ***R1*** and ***R2*** are disjoint but not spatially disjoint because the source ports 12345 and ANY overlap. However, rules ***R2*** and ***R3*** are both disjoint and spatially disjoint because the source ports ANY and ANY are identical, and the other 4 tuples do not overlap. (Other examples follow.)

A *transform function* is an algorithm that can be applied to a rule to create a *sortable key* for that rule, which can then be used to sort the rules by their keys using a *key comparison function*. For example, the transform function ***Tfn*** could concatenate the tuples of a rule into a bit array that is interpreted as a large integer, and a corresponding comparison function ***Cfn*** could be a simple integer comparison function. (Other examples follow.)

A *rule subset* is defined as an ordered grouping of one or more rules within a rule set. For example, in rule set ***S1*** above, the rule subsets might be:

Subset	Rules
T1	*R1, R2, R3*
T2	*R4*
T3	*R5*

A *rule group* is defined as a rule subset with a *group type* (dependent, disjoint), *transform function*, *comparison function*, and a *search algorithm hint* (linear, sub-linear). The *group type* can be *dependent* if the group contains dependent rules, or can be *disjoint* if the group strictly contains disjoint rules. For example, the rule set ***S1*** above might contain the following disjoint rule group:

Group	Group Type	Rules	Transform Fn	Comparison Fn	Hint
G1	Disjoint	*R1, R2, R3*	*Tfn*	*Cfn*	Sub-linear

A rule set may be partitioned into a list of ordered rule groups such that the security policy of the rule set is not changed when each rule group is decomposed in the listed order. This partitioning is accomplished by applying

a rule subset identification method to a given rule set. An example of such a method is:

1. For a given rule set *S* containing *n* rules *R1,R2,...,Rn*:
    a. Create a new empty disjoint rule group *Gj* (initially G1) in *S*.
    b. Place the first ungrouped rule *Ri* (initially R1) into *Gj*.
    c. For each remaining ungrouped rule *Ri* in *S*:
        i. If *Ri* is disjoint from rules in *Gj* and placing *Ri* into *Gj* does not modify the security policy of *S*, then place *Ri* into *Gj*.
        ii. Otherwise, leave *Ri* ungrouped.
    d. If *S* contains ungrouped rules, then go to step 1.a.
2. The rule set *S* now contains *m* disjoint rule groups *G1,G2,G3,...,Gm* which group together the *n* (possibly reordered) rules *R1,R2,...,Rn*.

Applying the above method to rule set ***S1*** might result in its partitioning into the following list of disjoint ordered rule groups:

Group	Group Type	Rules	Transform Fn	Comparison Fn	Hint
G1	Disjoint	*R1, R2, R3*			Linear
G2	Disjoint	*R4*			Linear
G3	Disjoint	*R5*			Linear

Decomposing the disjoint rule groups would result in ***G1,G2,G3*** = [***R1,R2,R3***], [***R4***], [***R5***] = ***R1,R2,R3,R4,R5*** = ***S1***.

A partitioned rule set containing disjoint rule groups may then be sorted by applying a transform function to each rule within each disjoint group to derive a sortable key for each rule. Then, the rules may be reordered within their disjoint groups using their sortable keys. The resulting sorted groups may be searched using sub-linear searching algorithms. An example of the sorting method is:

1. For each disjoint group *Gj* in rule set *S*:
    a. For each rule *Ri* in *Gj*:
        i. Apply Transform *Tfn* to *Ri* to derive a sortable key *Ki*.
    b. Sort the rules in *Gj* using the comparison function *Cfn* on the sortable keys *K*.

3. The rule set *S* now contains *m* sorted disjoint rule groups *G1,G2,G3,...,Gm*.

Applying the above method to the partitioned rule set ***S1*** might result in a new permutation rule set ***S1'*** that contains the following list of sorted rule groups where the ordering of rules within ***G1*** might change from ***R1,R2,R3*** to ***R3,R1,R2***:

Group	Group Type	Rules	Transform Fn	Comparison Fn	Hint
G1	Disjoint	*R3, R1, R2*	*Tfn*	*Cfn*	Sub-linear
G2	Disjoint	*R4*	*Tfn*	*Cfn*	Sub-linear
G3	Disjoint	*R5*	*Tfn*	*Cfn*	Sub-linear

The permutated rule set containing disjoint rule groups may be consolidated to reduce the number of groups that contain a rule count at or below a certain threshold, such as 1 rule, by merging two or more consecutive disjoint groups into a larger dependent group that may be searched using linear searching algorithms. An example of the consolidation method is:

1. For each group *Gj* in rule set S:
    a. If the sum of the number of rules in *Gj* and its subsequent group *Gj+1* is less than or equal to a specified threshold (e.g. 1), then the two groups are merged, their rules are concatenated, and the group type is set to *dependent*.
2. The rule set *S* now contains *m* or fewer rule groups of both dependent and disjoint types.

Applying the above method to the permuted rule set ***S1'*** might merge the disjoint group ***G3*** into ***G2***:

Group	Group Type	Rules	Transform Fn	Comparison Fn	Hint
G1	Disjoint	*R3, R1, R2*	*Tfn*	*Cfn*	Sub-linear
G2	Dependent	*R4, R5*	(none)	(none)	Linear

The APF packet filtering method matches packets against a given rule set by sequentially iterating over each of the ordered rule groups, then performing the specified sub-linear or linear search within each group. An example of a rule filtering method is:

1. For each packet that traverses the firewall:
    b. For each group *Gj* in rule set *S*:
        i. If *Gj* is a dependent group, then perform linear search within that group until there is a first rule match.
        ii. If *Gj* is a disjoint group:
            1. Apply transform *T* to the packet to derive a lookup key *K*.
            2. Use the comparison function *C* and sub-linear search within the group until there is a first rule match on key *K*.
        iii. If there is a first rule match, then process the packet according to its specified action. Otherwise, continue to the next group *Gj+1*.

Additional Notes

- The primary purpose of the partitioning of rules is to create rule groups that, in aggregate, enable the fastest possible searching of each packet against the rules in the rule set. In most cases, the optimal partitioning should be the grouping of maximal subsets of spatially disjoint rules. However, in some cases where these disjoint groups are small (e.g. less than 10 rules), consolidating the disjoint groups into a single larger group containing rules that are ordered using other criteria (such as hit probabilities or hardware cache friendliness) and employing linear or interpolated search algorithms may improve performance. A critical concept is the flexibility to organize the rules within the rule set in different ways that enable the use of the most efficient and applicable search algorithm that is available that accounts for the hardware capabilities, which is the motivation behind the term "adaptive" in Adaptive Packet Filtering.
- Rule sets may be partitioned, sorted and consolidated in-place.
- When partitioning a rule set into rule groups, each disjoint group should generally contain the maximal subsets of disjoint rules in order to reduce the number of disjoint groups in the rule set.

IPR2023-01353 Page 00877

- When partitioning a rule set into rule groups and/or sorting those groups, the transform and comparison functions may be different for each rule group.
- When partitioning a rule set into rule groups and/or sorting those groups, the algorithms may account for the hit probabilities of each rule and the aggregate hit probabilities of each group.
- When filtering packets, the search performed within any given rule group may employ the fastest available search algorithm applicable to that group even if it may be different from the specified search algorithm hint.
- When filtering packets, a sub-linear binary search within a disjoint rule group may account for hit probabilities at each pivot so that each recursion could maximize the probability of a rule match.
- When filtering packets, a constant-time search within a disjoint rule group is possible by defining a hashing function as the transform function such that the hash values for all rules within a group are unique within that group.

Examples

This section provides examples of the following items described in the algorithm in the previous section.

1) Example of the rule breakup

2) Example of transform function

3) Definition of rule representation

4) Difference between sub-linear and linear

5) Explanation of disjoint rules

1) Examples of the rule breakup

Define

```
S1 = (R1, R2, R3, R4, R5)
```

and

```
R1 = from 1.2.3.4 to 3.4.5.6 deny
R2 = from 2.3.4.5 to 4.5.6.7 allow
```

```
R3 = from 3.4.5.6 to 5.6.7.8 allow
R4 = from *.*.*.* to 3.4.5.6 allow
R5 = from *.*.*.* to *.*.*.* deny
```

Then S1' contains 3 groups of disjoint rules:

```
S1' = (G1, G2, G3)
G1 = (R1, R2, R3)
G2 = (R4)
G3 = (R5)
```

Or, S1' can contain 2 groups of disjoint and dependent rules:

```
S1' = (G1, G2)
G1 = (R1, R2, R3), disjoint, sub-linear
G2 = (R4, R5), dependent, linear
```

or

Define

```
S2 = (R1, R2, R3, R4, R5, R6, R7, R8, R9)
```

and

```
R1 = from 1.2.3.4 to 3.4.5.6 deny
R2 = from 2.3.4.5 to 4.5.6.7 allow
R3 = from 1.2.3.4 to 4.5.6.7 deny
R4 = from 1.*.*.* to *.*.*.* allow
R5 = from 3.4.5.6 to 5.6.7.8 deny
R6 = from 2.*.*.* to *.*.*.* deny
R7 = from *.*.*.* to 3.4.5.6 allow
R8 - from *.*.*.* to 5.6.7.8 allow
R9 = from *.*.*.* to *.*.*.* deny
```

Then S2' contains 4 groups of "spatially disjoint" rules:

```
S2' = (G1, G2, G3, G4)
G1 = (R1, R2, R3, R5)
G2 = (R4, R6)
G3 = (R7, R8)
G4 = (R9)
```

Or, S2' can contain 2 groups of disjoint and dependent rules:

```
S2' = (G1, G2)
G1 = (R1, R2, R3, R5), disjoint, sub-linear
G2 = (R4, R6, R7, R8, R9), dependent, linear
```

---

2) Example of transform function

A rule R usually consists of an N-tuple, most basically a 3-tuple such as "from 1.2.3.4 to 3.4.5.6 deny"

* tuple a = source IP address, e.g. "1.2.3.4"
* tuple b = destination IP address, e.g. "3.4.5.6"
* tuple c = action, e.g. "deny"

Each of these tuples have underlying scalar integer/bit vector representation, so in the above example:
* source IP address = "1.2.3.4" = 32 bit integer 0016909060
* destination IP address = "3.4.5.6" = 32 bit integer 0050595078
* action = "deny" = 8 bit integer 000

One possible transform function is a transform to scalar key which concatenates the digits of each of the tuples into a large integer value:
* T(a, b, c) = abc
* T(1.2.3.4, 3.4.5.6, deny) = 0016909060 0050595078 000 = 001690906000050595078000

Another possible transform function is a transform to scalar key which concatenates the bits of each of the tuples into a large integer/bit vector:
* a = 1.2.3.4 = 00000001000000100000001100000100 (32 bits)
* b = 3.4.5.6 = 00000011000001000000010100000110 (32 bits)
* c = 0 = 00000000 (8 bits)

T(a, b, c) = abc = 000000010000001000000011000001000000001100000100000001010000011000000000

Another possible transform function is an identity function (i.e. transformation function that does not do anything), then defining a multi-dimensional comparison function for sorting purposes. An example of this is a comparison function that is radix-based for each tuple, which would essentially result in a rule set that is radix sorted by each tuple.

Note that the transform function must convert the rule into a sortable key, which does not necessarily have to be a scalar key (i.e. it can be a multi-dimensional key that uses a multi-dimensional comparison function for sorting).

---

3) Definition of rule representation.
Rule representation is the way a rule and a rule set are conceptually represented in software. The most common representation of a rule is as an N-tuple object or structure that simply holds all the tuples together:

```
struct rule {
      unsigned char ip_proto;
      unsigned int ip_src_addr;
      unsigned int ip_dst_addr;
      unsigned short ip_src_port;
```

```
    unsigned short ip_dst_port;
    ...
    unsigned char action;
};
```

The most common representation of a rule set is an array or linked list that hold the rules in a fixed order, and allows for iteration forwards and backwards in the array or list.

Example of array:

memory location	0	1	2	3	4	5	6	7	8
value	R1	R2	R3	R4	R5	R6	R7	R8	R9

Example of linked list:

memory location	0	1	2	3	4	5	6	7	8
value	R1	R9	R2	R8	R3	R5	R4	R6	R7
linked list prev	-	0	2	4	6	5	7	8	3
rule	R1	R2	R3	R4	R5	R6	R7	R8	R9
next	2	4	6	5	7	8	3	1	-

An alternate rule representation is to hold the rule and rule set in a trie or other graph structure. An example of this is described in "Balancing Trie-Based Policy Representations for Network Firewalls."Stephen J. Tarsa and Errin W. Fulp. Proceedings of the IEEE International Symposium on Computer Communications, 2006

Another alternate rule set representation is a hierarchical one as described in OPTWALL, described in Acharya et al., "OPTWALL: A Hierarchical Traffic-Ware Firewall," available at:

http://www.isoc.org/isoc/conferences/ndss/07/papers/OPTWALL.pdf,

where rule sets are broken down into mutually exclusive rule subsets which are arranged in a hierarchical order. Despite some similarities in terminology, OPTWALL and APF are different. For example, APF does not change the underlying rule or rule set representation. It simply reorders the rules in place and keeps track of the beginning and ending rules in each subset T externally from the rule or rule set. For example, say that a given rule is a standard structure and the rule set is an array form so that the rule set contains the following in the computer's memory:

Rule set S:

Location	0	1	2	3	4	5	6	7	8
value	R1	R2	R3	R4	R5	R6	R7	R8	R9

Suppose that the above rule set S can be divided into subsets containing disjoint rules:

```
T1 = R2, R9, R1, R3
```

```
T2 = R5, R7
T3 = R2, R4, R8
```

so that

```
S' = T1, T2, T3 = R2, R9, R1, R3, R5, R7, R2, R4, R8
```

Then the order of the rules in memory can be changed in place:

Rule set S':

```
location  0  1  2  3  4  5  6  7  8
value     R2 R9 R1 R3 R5 R7 R2 R4 R8
          |---------| |---| |------|
               T1       T2     T3
```

The memory ranges for T1 (0-3), T2 (4-5) and T3 (6-8) are stored outside of the rule and rule set data structures.

---

4) Difference between linear and sub-linear.

A linear algorithm is one whose computational time increases linearly as the size of the set is increased. The best example of this is when looking up a word in a dictionary. If the dictionary is unsorted, then the order of the words would be arbitrary. Therefore, when looking up the word "zebra," one could start from the beginning and search until the end to find it. If the dictionary contains 1,000 entries, you would need to examine all 1,000 words in the worst case.

A sub-linear algorithm is one whose computational time increases sub-linearly as the size of the set is increased. In the above example, if the dictionary were sorted alphabetically, then one could still use a linear search by starting from the beginning and searching until the end to find "zebra." However, one could also use a sub-linear binary search algorithm by looking in the middle of the dictionary, then seeing if the middle entry comes alphabetically before or after (or is equal), then recursively selecting the middle of the appropriate half again and again to find the word "zebra." Since at each recursion 1/2 of the remaining words are eliminated, it would take about $\log_2(1{,}000)$ or about 10 examinations to find the entry in the worst case.

Another example of a sub-linear algorithm is hashing. Suppose that the above dictionary contains only 5 letter words. If we define a hash function that sums the alphabet order of each letter in the word (z = 26, e = 05, b = 02, r = 18, a = 01), then hash("zebra") = 26 + 05 + 02 + 18 + 01 = 52. The computer could have an array containing all the words in the dictionary where each word's position in the array is the hash value of the word (subject to collisions). In the above example, the array's 52nd position would have the word "zebra," so it would take only 1 comparison to determine a match without collision. This hash technique can be selectively used in APF.

---

5) Explanation of disjoint rules.

A rule R1 is "disjoint" from another rule R2 if their positions in the rule set S can be exchanged without altering the overall security policy. An example of this is rule set S containing:

```
R1 = from 1.*.*.* to 2.3.4.5 deny
R2 = from 1.2.3.4 to 1.2.3.4 allow
```

has the same security policy as rule set S' containing:

```
R2 = from 1.2.3.4 to 1.2.3.4 allow
R1 = from 1.*.*.* to 2.3.4.5 deny
```

because P = P'; therefore, R1 and R2 are disjoint.

The technique set forth above does not explain the concept of "spatially disjoint" rules. This is important if the transform function T cannot account for overlapping tuple values, which can be very common in practical settings.

A rule R1 is "spatially disjoint" from another rule R2 if they are "disjoint" and their corresponding tuples do not unevenly overlap (must be exactly equal, or do not overlap at all). In the above example, R1 and R2 are disjoint but not spatially disjoint because the first tuple of R1 (1.*.*.*) and R2 (1.2.3.4) are not equal but do overlap, i.e. the value of 1.2.3.4 would match the first tuple of both R1 and R2. An example of spatially disjoint rules are:

```
R3 = from 1.2.3.4 to 3.4.5.6 deny
R4 = from 2.3.4.5 to 3.4.5.6 allow
```

Here, the first tuple of R1 (1.2.3.4) and the first tuple of R2 (2.3.4.5) do not overlap, the second tuple of R1 (3.4.5.6) and R2 (3.4.5.6) are equal, and the >> third tuple of R1 (deny) and R2 (allow) do not overlap.

The importance of "spatially disjoint" rules is dependent upon the definition of a transform function T, so it may be possible to define T such that rules need not be "spatially disjoint" so long as rules are "disjoint."

The disclosure of each of the publications referenced herein is hereby incorporated by reference in its entirety.

It will be understood that various details of the presently disclosed subject matter may be changed without departing from the scope of the presently disclosed subject matter. Furthermore, the foregoing description is for the purpose of illustration only, and not for the purpose of limitation.

CLAIMS

What is claimed is:

1. A method for adaptive packet filtering, the method comprising:

identifying at least one subset of rules in an ordered set of firewall packet filtering rules that defines a firewall policy such that the at least one subset contains disjoint rules, where disjoint rules are defined as rules whose order can be changed without changing the integrity of the firewall policy;

sorting the rules in the at least one subset to statistically decrease the number of comparisons that will be applied to each packet that a firewall encounters; and

filtering packets at the firewall using the sorted rules in the at least one subset by comparing each packet to each of the sorted rules in the at least one subset until the packet is allowed or denied and ceasing the comparing for the packet in response to the packet being allowed or denied and thereby achieving sub-linear searching for the packets filtered using the sorted rules in at least one subset.

2. The method of claim 1 comprising, for the disjoint rules within the at least one subset, identifying spatially disjoint rules and grouping the spatially disjoint rules together within the at least one subset and wherein sorting the rules includes sorting the rules within each of the spatially disjoint groupings.

3. The method of claim 1 comprising identifying at least another subset of rules in the ordered set of firewall rules that are non-disjoint and refraining from sorting the rules in the at least another subset and wherein filtering the packets includes filtering the packets using the sorted, disjoint rules and the non-sorted, non-disjoint rules, thereby achieving sub-linear searching for the sorted, disjoint rules and achieving linear searching for the non-sorted, non-disjoint rules.

4. The method of claim 1 wherein sorting the rules includes applying a transform function to each rule in the at least one subset to compute a sortable key for each of the rules and sorting the rules according to the keys.

5. The method of claim 1 wherein sorting the rules includes sorting the rules according to hit probabilities.

6. The method of claim 1 wherein sorting the rules includes sorting the rules in a binary search tree.

7. The method of claim 1 wherein filtering packets by comparing each packet to the sorted rules includes comparing each packet to the sorted rules using a binary search algorithm.

8. The method of claim 1 wherein filtering packets by comparing each packet to the sorted rules includes comparing each packet to the sorted rules using an interpolated search algorithm.

9. The method of claim 1 wherein filtering packets by comparing each packet to the sorted rules includes comparing each packet to the sorted rules using an informed search algorithm.

10. The method of claim 1 wherein filtering packets by comparing each packet to the sorted rules includes comparing each packet to the sorted rules using a hash lookup algorithm.

11. The method of claim 1 wherein each rule includes an IP source address and an IP destination address.

12. The method of claim 1 wherein the firewall includes a plurality of processors, the at least one subset comprises a plurality of subsets and further comprising distributing the subsets across the plurality of processors to implement at least one of pipelined processing, data parallel processing, and function parallel processing.

13. The method of claim 12 wherein the firewall implements a combination of pipelined and function parallel processing.

14. The method of claim 12 wherein the firewall implements a combination of pipelined and data parallel processing.

15. The method of claim 12 wherein the firewall implements a combination of pipelined, function parallel, and data parallel processing.

16. A system for adaptive packet filtering, the system comprising:

a firewall rule subset identifier/rule sorter for identifying at least one subset of rules in an ordered set of firewall packet filtering rules that defines a firewall policy such that the at least one subset contains

disjoint rules, where disjoint rules are defined as rules whose order can be changed without changing integrity of the firewall policy, and for sorting the rules in the at least one subset to statistically decrease the number of comparisons that will be applied to each packet that a firewall encounters; and

a packet filter for filtering packets at the firewall using the sorted rules in the at least one subset by comparing each packet to each of the sorted rules in the at least one subset until the packet is allowed or denied and ceasing the comparing for the packet in response to the packet being allowed or denied and thereby achieving sub-linear searching for the packets filtered using the sorted rules in at least one subset.

17. The system of claim 16 comprising, for the disjoint rules within the at least one subset, the firewall rule subset identifier/rule sorter identifies spatially disjoint rules and groups the spatially disjoint rules together within the at least one subset and wherein the firewall rule subset identifier/rule sorter the rules within each of the spatially disjoint groupings.

18. The system of claim 16 wherein the firewall rule subset identifier/rule sorter identifies at least another subset of rules in the ordered set of firewall rules that are non-disjoint and refrains from sorting the rules in the at least another subset and wherein the firewall packet filter filters the packets using the sorted, disjoint rules and the non-sorted, non-disjoint rules, thereby achieving sub-linear searching for the sorted, disjoint rules and achieving linear searching for the non-sorted, non-disjoint rules.

19. The system of claim 16 wherein the firewall rule subset identifier/rule sorter applies a transform function to each rule in the at least one subset to compute a sortable key for each of the rules and sorting the rules according to the keys.

20. The system of claim 16 wherein the firewall rule subset identifier/rule sorter sorts the rules includes sorting the rules according to hit probabilities.

21. The system of claim 16 wherein the firewall rule subset identifier/rule sorter sorts the rules includes sorting the rules in a binary search tree.

22. The system of claim 16 wherein the packet filter compares each packet to the sorted rules using a binary search algorithm.

23. The system of claim 16 wherein the packet filter compares each packet to the sorted rules using an interpolated search algorithm.

24. The system of claim 16 wherein the packet filter compares each packet to the sorted rules using an informed search algorithm.

25. The system of claim 16 wherein the packet filter compares each packet to the sorted rules using a hash lookup algorithm.

26. The system of claim 16 wherein each rule includes an IP source address and an IP destination address.

27. The system of claim 16 wherein the firewall includes a plurality of processors, the at least one subset comprises a plurality of subsets and further comprising distributing the subsets across the plurality of processors to implement at least one of pipelined processing, data parallel processing, and function parallel processing.

28. The system of claim 27 wherein the firewall implements a combination of pipelined and function parallel processing.

29. The system of claim 27 wherein the firewall implements a combination of pipelined and data parallel processing.

30. The system of claim 27 wherein the firewall implements a combination of pipelined, data parallel, and function parallel processing.

31. A non-transitory computer readable medium having stored thereon on executable instructions that when implemented by a processor of a computer controls the computer to perform steps comprising:

identifying at least one subset of rules in an ordered set of firewall packet filtering rules that defines a firewall policy such that the at least one subset contains disjoint rules, where disjoint rules are defined as rules whose order can be changed without changing integrity of the firewall policy;

sorting the rules in the at least one subset to statistically decrease the number of comparisons that will be applied to each packet that a firewall encounters; and

filtering packets at the firewall using the sorted rules in the at least one subset by comparing each packet to each of the sorted rules in the at least one subset until the packet is allowed or denied and ceasing the comparing for the packet in response to the packet being allowed or denied and thereby achieving sub-linear searching for the packets filtered using the sorted rules in at least one subset.

IPR2023-01353 Page 00888

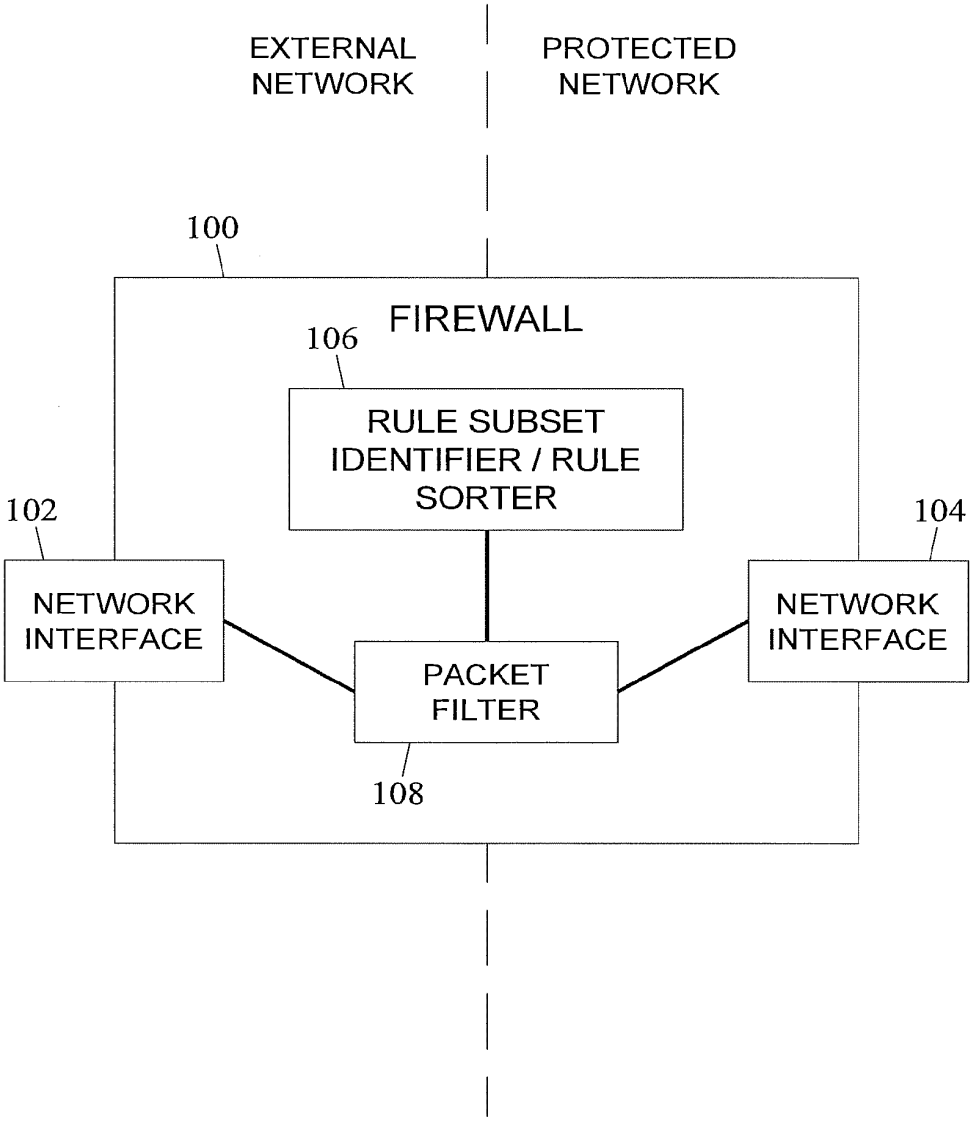

EXTERNAL NETWORK
PROTECTED NETWORK
100
FIREWALL
106
RULE SUBSET IDENTIFIER / RULE SORTER
102
104
NETWORK INTERFACE
NETWORK INTERFACE
PACKET FILTER
108

FIG. 1

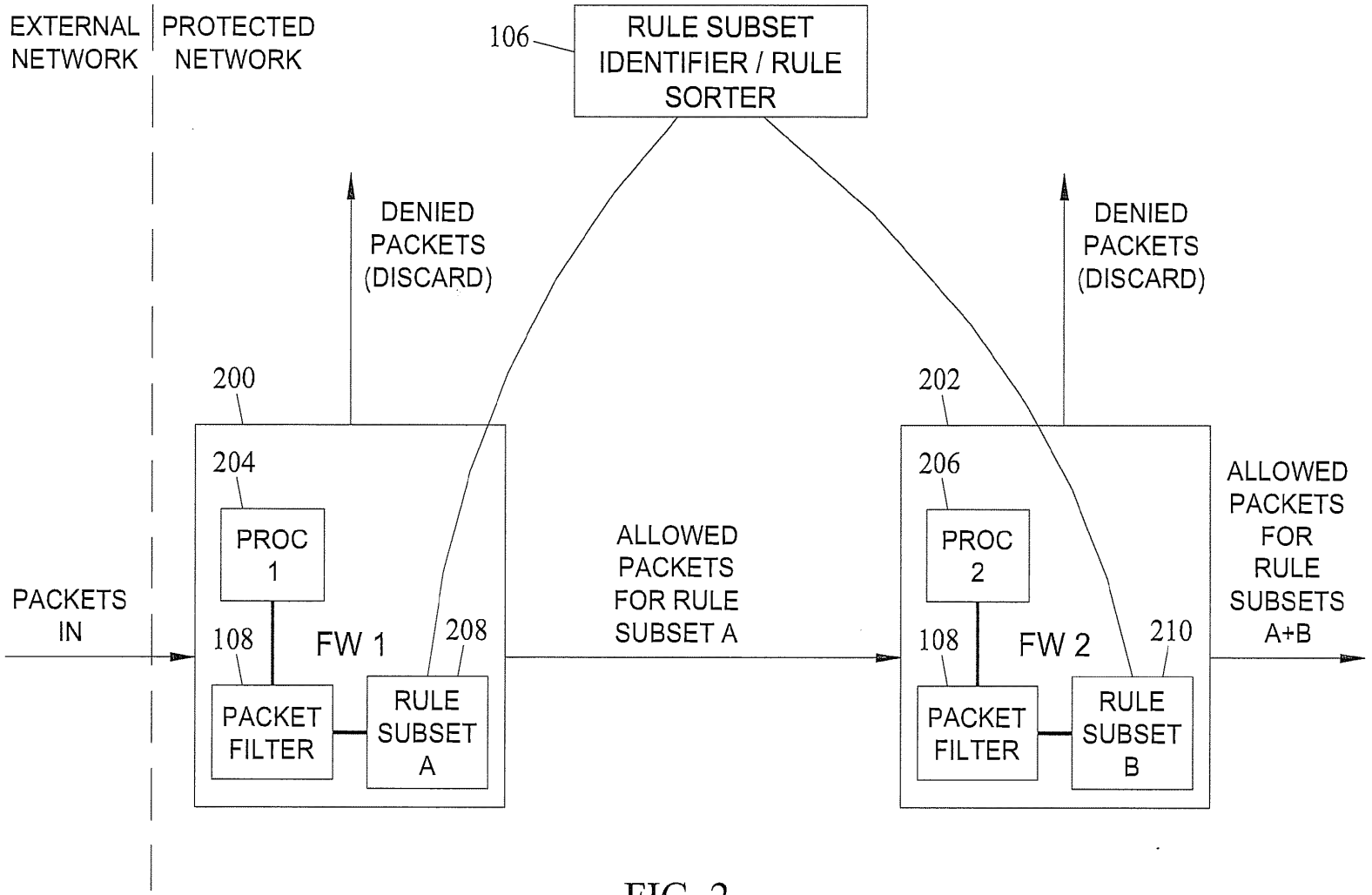

EXTERNAL NETWORK
PROTECTED NETWORK
106
RULE SUBSET IDENTIFIER / RULE SORTER
DENIED PACKETS (DISCARD)
DENIED PACKETS (DISCARD)
200
202
204
206
PROC 1
PROC 2
PACKETS IN
108
FW 1
208
ALLOWED PACKETS FOR RULE SUBSET A
108
FW 2
210
ALLOWED PACKETS FOR RULE SUBSETS A+B
PACKET FILTER
RULE SUBSET A
PACKET FILTER
RULE SUBSET B

FIG. 2

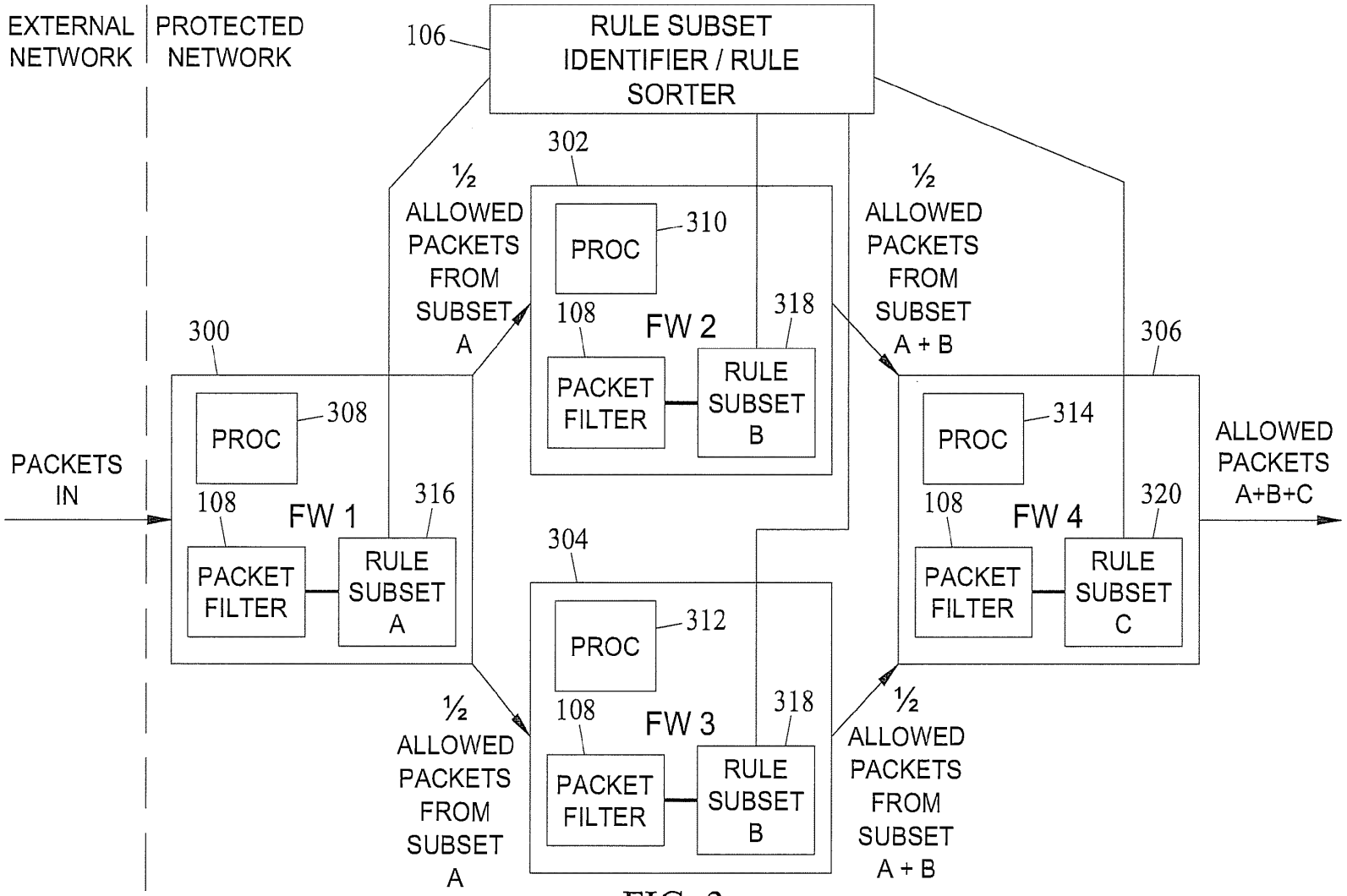

EXTERNAL NETWORK
PROTECTED NETWORK
106
RULE SUBSET IDENTIFIER / RULE SORTER
302
½ ALLOWED PACKETS FROM SUBSET A
PROC
310
108
FW 2
318
PACKET FILTER
RULE SUBSET B
½ ALLOWED PACKETS FROM SUBSET A + B
300
306
PROC
308
314
PACKETS IN
ALLOWED PACKETS A+B+C
108
FW 1
316
108
FW 4
320
PACKET FILTER
RULE SUBSET A
PACKET FILTER
RULE SUBSET C
304
PROC
312
½ ALLOWED PACKETS FROM SUBSET A
108
FW 3
318
½ ALLOWED PACKETS FROM SUBSET A + B
PACKET FILTER
RULE SUBSET B

FIG. 3

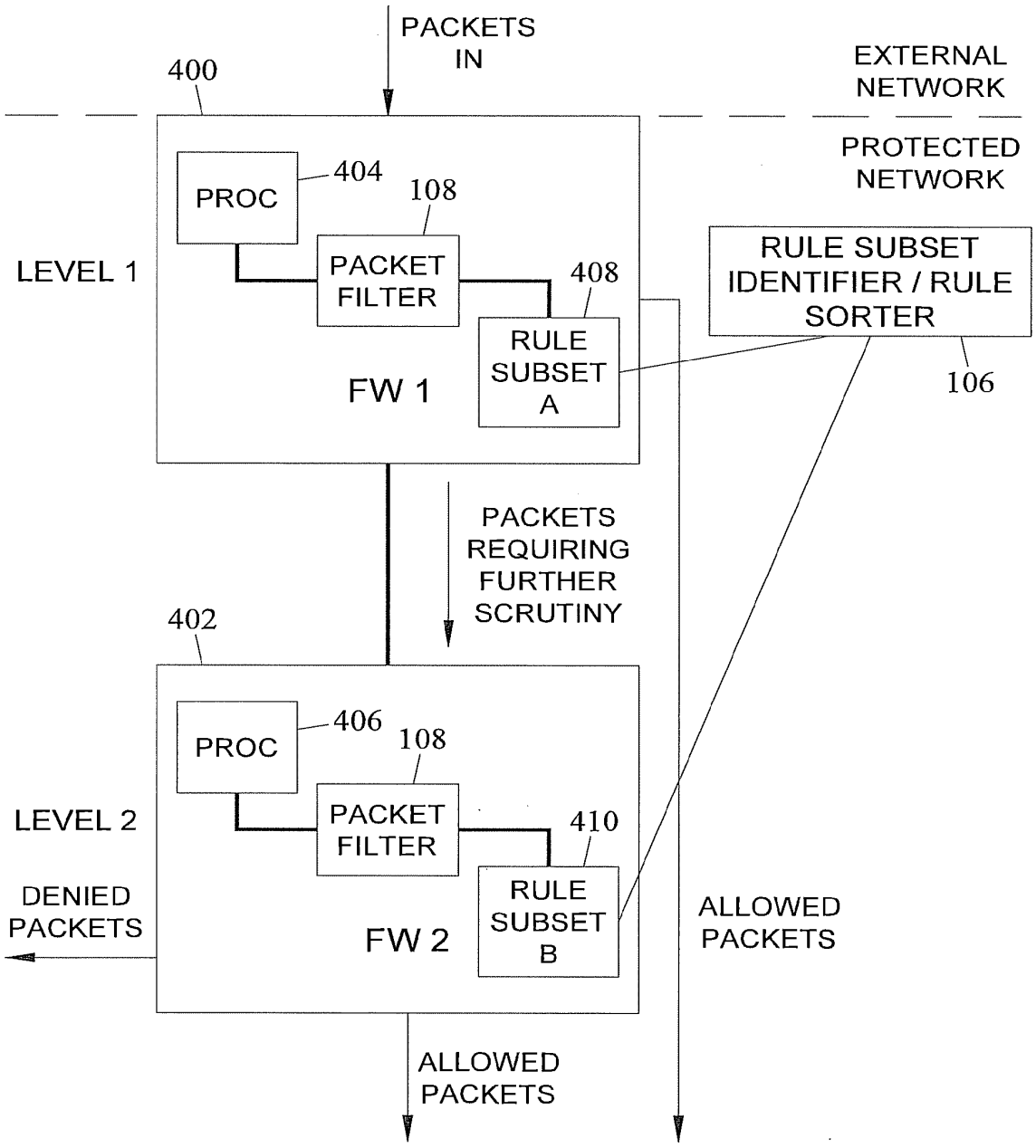

PACKETS IN
400
EXTERNAL NETWORK
PROTECTED NETWORK
PROC
404
108
PACKET FILTER
408
RULE SUBSET A
FW 1
LEVEL 1
RULE SUBSET IDENTIFIER / RULE SORTER
106
PACKETS REQUIRING FURTHER SCRUTINY
402
PROC
406
108
PACKET FILTER
410
RULE SUBSET B
FW 2
LEVEL 2
DENIED PACKETS
ALLOWED PACKETS
ALLOWED PACKETS

FIG. 4

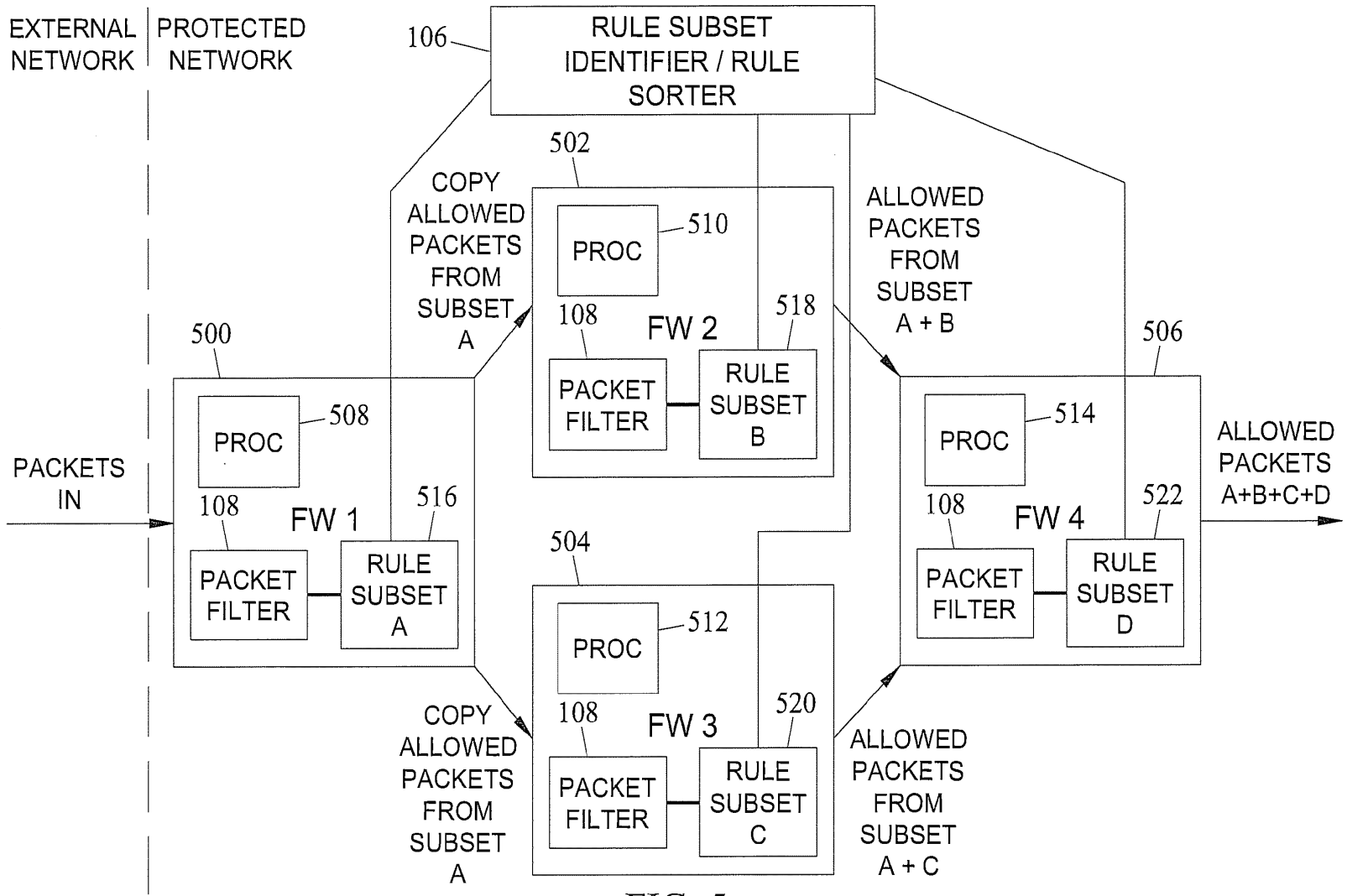

EXTERNAL NETWORK
PROTECTED NETWORK
106
RULE SUBSET IDENTIFIER / RULE SORTER
502
COPY ALLOWED PACKETS FROM SUBSET A
PROC
510
108
FW 2
518
PACKET FILTER
RULE SUBSET B
ALLOWED PACKETS FROM SUBSET A + B
500
506
PROC
508
514
PACKETS IN
ALLOWED PACKETS A+B+C+D
108
FW 1
516
108
FW 4
522
PACKET FILTER
RULE SUBSET A
504
PACKET FILTER
RULE SUBSET D
PROC
512
COPY ALLOWED PACKETS FROM SUBSET A
108
FW 3
520
ALLOWED PACKETS FROM SUBSET A + C
PACKET FILTER
RULE SUBSET C

FIG. 5

Espacenet

Bibliographic data: EP1484884 (A2) — 2004-12-08

Multi-layered firewall architecture

Inventor(s): SWANDER BRIAN D [US]; MAYFIELD PAUL G [US] ± (SWANDER, BRIAN D, ; MAYFIELD, PAUL G)

Applicant(s): MICROSOFT CORP [US] ± (MICROSOFT CORPORATION)

Classification: - international: ***G06F1/00; G06F11/00; G06F12/14; G06F13/00; G06F15/00; G06F17/00; H04L12/22; H04L12/56; H04L12/66; H04L29/06; H04L9/00; G06F;*** (IPC1-7): H04L29/06

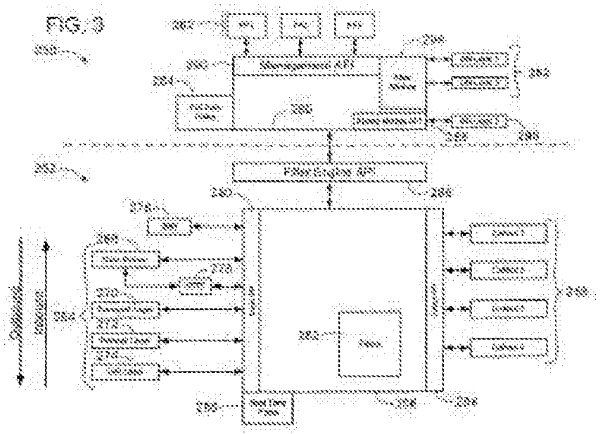
- cooperative: **H04L63/0218**; **H04L63/0227**; **H04L63/0236**; **H04L63/0254**; H04L63/164

Application number: EP20040009147 20040416

Priority number(s): US20030456766 20030606

Also published as: EP1484884 (A3) US2005022010 (A1) US7509673 (B2) JP2004362581 (A) JP4572089 (B2) more

Abstract of EP1484884 (A2)



A method and system are provided for implementing a firewall architecture in a network device. The firewall architecture includes a plurality of network layers, a first firewall engine, and one or more callout modules. The layers send packets and packet information to the first firewall engine, maintain and pass packet context to subsequent layers, and process the packets. The first firewall engine compares the packet information to one or more installed filters and returns an action to the layers indicating how to treat the packet. The callouts provide additional functionality such as intrusion detection, logging, and parental control features.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 484 884 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
08.12.2004 Bulletin 2004/50

(51) Int Cl.[7]: **H04L 29/06**

(21) Application number: **04009147.2**

(22) Date of filing: **16.04.2004**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PL PT RO SE SI SK TR
Designated Extension States:
AL HR LT LV MK

(30) Priority: **06.06.2003 US 456766**

(71) Applicant: **MICROSOFT CORPORATION**
Redmond, Washington 98052-6399 (US)

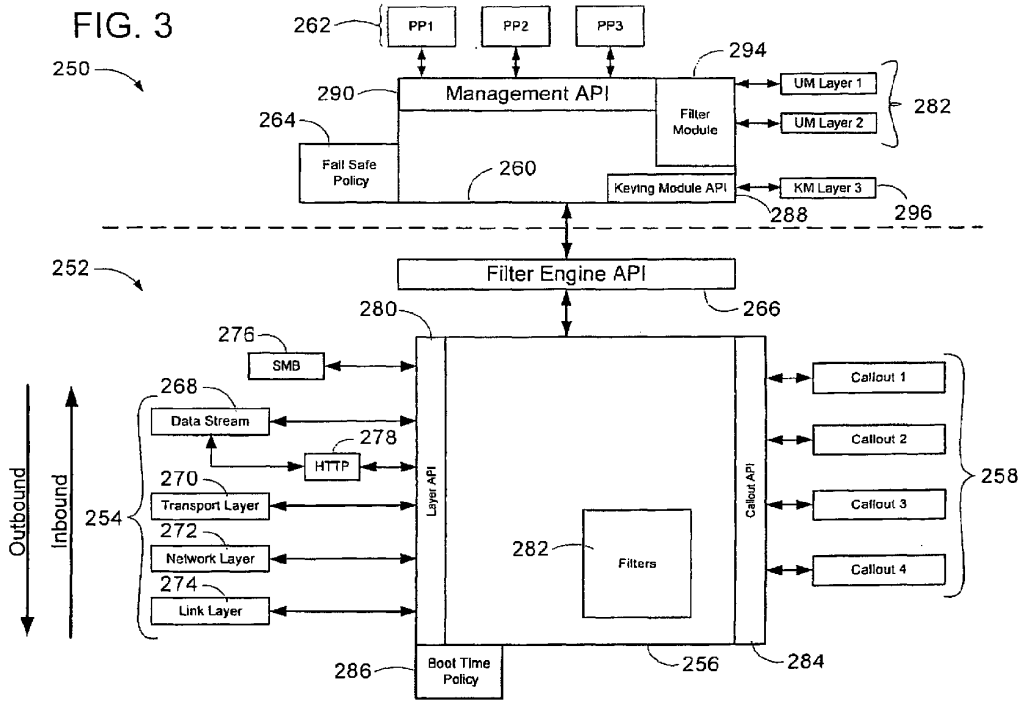
(72) Inventors:
- **Swander, Brian D.**
  Bellevue Washington 98004 (US)
- **Mayfield, Paul G.**
  Sammamish 98075 (US)

(74) Representative: **Grünecker, Kinkeldey,**
Stockmair & Schwanhäusser Anwaltssozietät
Maximilianstrasse 58
80538 München (DE)

(54) Multi-layered firewall architecture

(57) A method and system are provided for implementing a firewall architecture in a network device. The firewall architecture includes a plurality of network layers, a first firewall engine, and one or more callout modules. The layers send packets and packet information to the first firewall engine, maintain and pass packet context to subsequent layers, and process the packets. The first firewall engine compares the packet information to one or more installed filters and returns an action to the layers indicating how to treat the packet. The callouts provide additional functionality such as intrusion detection, logging, and parental control features.

FIG. 3



EP 1 484 884 A2

IPR2023-01353 Page 00895

Description

RELATED APPLICATIONS

[0001] The present application contains subject matter related to that of patent application "Multi-Layer Based Method for Implementing Network Firewalls," attorney docket number 221039; "A Method and Framework for Integrating a Plurality of Network Policies," attorney docket number 221041; and "Method for Managing Network Filter Based Policy," attorney docket number 221037; filed on the same day as the present application, the disclosures of which are hereby expressly incorporated by reference.

TECHNICAL FIELD

[0002] This invention generally relates to computer systems and to network security. More particularly, this invention relates to a firewall architecture implemented in one or more network devices.

BACKGROUND OF THE INVENTION

[0003] Network protocols are designed to facilitate communication between network devices through an open exchange of data. While the open exchange of data greatly enhances the use of network devices to accomplish tasks, it also creates problems because network protocols are not designed for, and generally do not provide, network security. Computers coupled to both public and private networks, such as Local Area Networks (LANs), Wide Area Networks (WANs), intranets, and the Internet are susceptible to malicious attacks perpetrated by other network devices coupled either directly or indirectly to the network. Such malicious attacks include theft of data, Denial of Service (DOS) attacks, the proliferation of computer viruses, and the like. Other related issues arise when coupling computers to networks such as controlling access to undesirable or inappropriate web sites by children.

[0004] A firewall is a tool used to protect individual users, network devices, and networks in general, from malicious attacks, while also adding the ability to control the exchange of data over the network through implementation of a policy. The firewall implements the policy by examining network packets and determining, based on the examination, whether the packets should be permitted, or conversely blocked, from further traversing the network.

[0005] The policy implemented via the firewall is defined by one or more filters. Each filter includes filter parameters and an associated action. The filter parameters are used to identify network packets that are subject to the firewall policy and include information such as hardware addresses, e.g. Media Access Control (MAC) addresses, network addresses, e.g. Internet Protocol (IP) addresses, protocol type, e.g. Transport Control Protocol (TCP), port numbers, and the like. The action defines how packets with parameters that match the filter parameters should be treated. As a specific example, the filter includes as its parameters a Uniform Resource Locator (URL) address, e.g. "http://www.foo.com." The filter further associates the action of block, i.e. drop the packet, with that URL address. Whenever the firewall examines a packet and through that examination identifies the URL address "http://www.foo.com" as embedded in the packet, the firewall drops the packet thereby preventing it from traversing the network.

[0006] Network devices exchange data by sending and receiving packets through a network stack comprising a layered network architecture. While different network architecture models exist, most include at least an application layer, a transport layer, a network layer, and a link layer. Network packets traverse each layer sequentially and, as each layer is traversed, the packet is subject to processing. For outbound packets, the application layer processes data according to application protocols such as Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), and Simple Mail Transfer Protocol (SMTP) to name a few. Other layers, such as the network layer and the transport layer packetize the data by embedding it in TCP and IP headers. The layers perform reciprocal processing for inbound packets by, for example, parsing headers, unpacketizing data etc. The layered "stack" architecture and processing function performed by the layers results in a dynamic packet structure whereby packet content including the packet parameters change as the packet traverses the network protocol stack.

[0007] Firewalls examine packets at an inspection point located within the layered network stack. At one extreme, the inspection point is at the application layer. For example, the firewall is deployed as a Layered Service Provider (LSP). Packets at the application layer include the underlying data that will be transmitted to another network device or that has been received from another network device. Examining the packet at the application layer permits the firewall to identify application layer parameters, such as a URL address, and compare the application layer parameters to the filter parameters. However, other packet parameters such as IP addresses, port numbers, MAC addresses, and the like are not available because they have either not been added to outbound packets or have been parsed away from inbound packets.

[0008] At the other extreme, the firewall inspection point is implemented at lower levels of the network stack as an intermediate driver interposed between the link layer and the network layer. Packets at the lower levels of the network

IPR2023-01353 Page 00896

stack include a maximum number of parameters, e.g. interface numbers, MAC addresses, IP addresses, protocol type, ports, and payload data. Although the packets include such parameters, it does not follow that the parameters are readily identifiable. After the firewall receives the packet, the firewall needs to parse and interpret the relevant packet parameter for comparison with the filter parameters. Thus, both the layers in the network stack and the firewall perform redundant packet parsing and interpretation functions.

SUMMARY OF THE INVENTION

[0009] The present invention is directed towards a firewall architecture that permits application of firewall filters to network packets at all layers within a network stack. The architecture includes a set layer processors capable of processing layer parameters from a network packet. The layer parameters are parameters associated with the layer processor and include parameters that the layer processor parses from, adds to, or otherwise derives from the network packet. The layer processors are further capable of issuing classification requests that include the layer parameters.

[0010] The architecture also includes a first firewall engine comprising a layer interface, a set of installed filters, and a lookup component. The layer interface receives the layer parameters sent as part of the classification request. The first firewall engine lookup component uses the layer parameters to identify one or more matching filters from the set of installed filters. Each matching filter includes an action that instructs the layer processor whether to permit the packet to further traverse the network or whether to block the packet. The first firewall engine returns at least one action to the layer processor via the layer interface.

[0011] In an embodiment of the invention, the layer processors maintain packet context. The layer processor receives the packet context from a previous layer processor and sends the packet context to the first firewall engine. The lookup component in the first firewall engine uses the packet context in combination with the layer parameters to identify the one or more matching filters. The layer processor also modifies the packet context by adding the layer parameters and then sending the modified packet context to a next layer processor.

[0012] In another embodiment of the invention, one or more callouts are included as part of the firewall architecture. Each of the one or more callout modules provide added functionality (beyond permit and block) such as packet logging functions, Internet Protocol Security (IPSec) verification, and execution of parental controls. The callout module is executed on the packet when it is identified as the action in one of the matching filters.

[0013] In another embodiment of the invention, a second firewall engine is provided. The second firewall engine adds new filters into the set of installed filters via a management API. The second firewall engine also includes an instance of the first firewall engine, referred to as a filter module that replicates the services of the first firewall engine within the second firewall engine.

[0014] Additional features and advantages of the invention will be made apparent from the following detailed description of illustrative embodiments, which proceeds with reference to the accompanying figures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] While the appended claims set forth the features of the present invention with particularity, the invention, together with its objects and advantages, may be best understood from the following detailed description taken in conjunction with the accompanying drawings of which:

[0016] Figure 1 is a block diagram generally illustrating an exemplary computer system on which the present invention resides;

[0017] Figure 2 is a block diagram generally illustrating an exemplary network environment whereby the present invention is employed;

[0018] Figure 3 is a block diagram generally illustrating the firewall architecture of the present invention;

[0019] Figure 4 is a block diagram illustrating an exemplary filter used with the present invention;

[0020] Figure 5 is a block diagram illustrating an exemplary data structure for packet context used with the present invention;

[0021] Figure 6 is a block diagram illustrating an exemplary set of application programming interfaces used with the present invention;

[0022] Figure 7 is a block diagram illustrating an exemplary application programming interface used with the present invention;

[0023] Figure 8 is a block diagram illustrating functions performed by network layers according to the present invention; and

[0024] Figure 9 is a block diagram illustrating an exemplary set of callouts used with the present invention.

IPR2023-01353 Page 00897

DETAILED DESCRIPTION OF THE INVENTION

[0025] A firewall architecture that permits filtering at multiple layer processes, referred to herein as layers, is disclosed. The architecture includes a user mode process and a kernel mode process executing in an operating system. Alternatively, the firewall architecture is executed outside of the operating system in one or more program modules or within a single operating system mode.

[0026] The kernel mode process includes a plurality of layers comprising a protocol stack, a kernel firewall engine, and one or more callouts. The protocol stack includes an application layer, a transport layer, a network layer, and link layer. Additional layers are added or deleted from the architecture as desired. The layers each form a requesting layer capable of receiving a network packet and corresponding packet context data from a previous layer or process. The requesting layer then issues a classification request to the kernel firewall engine via a layer API. The classification request includes the packet received by the requesting layer, the packet context, and a set of layer parameters associated with the requesting layer. The kernel firewall engine processes the request and returns an action. The action, by way of example, instructs the requesting layer how to treat the packet (e.g. permit or block). If the action is permit, the requesting layer processes the packet according to a layer protocol, modifies the packet context to include the layer parameters, and passes the packet and packet context to a next layer. If the action is block, the requesting layer drops the packet and does not pass the packet to the next layer. The requesting layer may perform additional functions as a result of the block action, such as tearing down a TCP connection where a TCP packet is dropped.

[0027] The kernel firewall engine includes the layer API, a set of installed filters, and a callout API. Each of the set of installed filters includes a set of filter conditions and an associated action. The kernel firewall engine processes the classification request sent from the requesting layer by identifying one or more matching filters. The matching filters have filter conditions that match the layer parameters and packet context. Once the matching filters are identified, they are applied in order of filter priority. If the action of the filter being applied is permit or block, the action is returned to the requesting layer. If the action is callout, the classification request issued by the requesting layer, along with matching filter identification, is passed to one of the callout modules. The callout module performs its programmed function, and returns an action to the kernel firewall engine.

[0028] The kernel firewall engine applies matching filters in order of priority to the packet until at least a terminating action is designated. Once the terminating action is designated, it is returned to the requesting layer and no additional matching filters are applied. If no matching filters are identified for a packet, the requesting layer is notified that no matching filters were found and the requesting layer then decides how to treat the packet. Typically, where no matching filter is identified, the requesting layer treats the packet as if a permit action was returned.

[0029] Exemplary user mode processes include a user mode firewall engine and one or more policy providers. The policy providers obtain policy from any suitable source such as volatile or non-volatile memory or is user entered through a graphical user interface or similar tool. The policy is a source of information for rendering a new filter, including the set of filter conditions and associated actions.

[0030] The user mode also includes an instance of the kernel firewall engine within the user firewall engine thereby permitting user mode layers to be created. The user mode layers then use the user mode instance of the kernel firewall engine to install filters, and identify filters that match a set of parameters allowing the application of filtering within the user mode.

[0031] In an embodiment of the invention, a callout interface from the firewall engine to a set of callout modules enables virtually unlimited extension of the firewall capabilities. By way of example, an HTTP context callout provides a parental control feature by identifying acceptable and unacceptable URL addresses. An IPSec callout verifies that packets that are supposed to use IPSec have been properly subject to IPSec processing. A logging callout logs packets meeting established criteria thereby facilitating later examination of packets. An intrusion detection callout identifies suspicious packets based on known algorithms.

[0032] Turning to the drawings, wherein like reference numerals refer to like elements, the invention is illustrated as being implemented in a suitable computing environment. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

[0033] Figure 1 illustrates an example of a suitable computing system environment 100 on which the invention may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

IPR2023-01353 Page 00898

[0034] The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0035] The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

[0036] With reference to Figure 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Associate (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

[0037] Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer readable media.

[0038] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, Figure 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

[0039] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Figure 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0040] The drives and their associated computer storage media discussed above and illustrated in Figure 1, provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In Figure 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating

IPR2023-01353 Page 00899

system 144, application programs 145, other program modules 146, and program data 147 are given different numbers hereto illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 110 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through a output peripheral interface 195.

[0041] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the personal computer 110, although only a memory storage device 181 has been illustrated in Figure 1. The logical connections depicted in Figure 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0042] When used in a LAN networking environment, the personal computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the personal computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, Figure 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0043] In the description that follows, the invention will be described with reference to acts and symbolic representations of operations that are performed by one or more computer, unless indicated otherwise, As such, it will be understood that such acts and operations, which are at times referred to as being computer-executed, include the manipulation by the processing unit of the computer of electrical signals representing data in a structured form. This manipulation transforms the data or maintains it at locations in the memory system of the computer, which reconfigures or otherwise alters the operation of the computer in a manner well understood by those skilled in the art. The data structures where data is maintained are physical locations of the memory that have particular properties defined by the format of the data. However, while the invention is being described in the foregoing context, it is not meant to be limiting as those of skill in the art will appreciate that various of the acts and operation described hereinafter may also be implemented in hardware.

[0044] A network environment wherein the firewall architecture of the present invention is used will now be described with reference to FIG. 2. The network is exemplary in nature as the firewall architecture of the present invention is implemented in any network device that is coupled to any network configuration. The network environment includes a private network 200 and a public network 202. The private network 200 and public network 202 are any suitable type such as Local Area Networks (LANs), Wide Area Networks (WANs), intranets, the Internet, or any combination thereof.

[0045] The network environment includes a plurality of network devices 204, 206, 208, 210 and 212. Network devices 204, 206 are coupled to the private network 200. Network devices 210, 212 are coupled to the public network 202. The network device 208 is coupled to, and provides an interface between, both the private network 200 and the public network 202. The network devices are coupled to the public and private networks using any suitable technology such as Ethernet, 1394, or 802 11 (b). The network devices are further implemented as any suitable computing devices such as personal computers, servers, handheld devices, printers, switches, routers, bridges, repeaters or the like.

[0046] The network device 208 includes a firewall 214 and one or more filters 216. The firewall 214 is a program module or a set of program modules, implemented according to the architecture of the present invention, that examine network packets exchanged between the network devices 204, 206, 208 coupled to the private network 200 and the network devices 210, 212 coupled to the public network 202. In an embodiment of the invention, the firewall 214 also examines locally destined network packets sent from, and destined to, network devices within the private network 200.

[0047] The firewall 214 is implemented in network device 208 to protect and control network traffic exchanged between the private network 200 and the public network 202, referred to as an edge firewall. Alternatively, the firewall 214 is implemented in, and protects, a single network device such as illustrated in the network device 210, referred to as a host firewall. The firewall is also capable of being implemented as a centrally administered set of hosts and/or edge firewalls in a synchronized manner, referred to as a distributed firewall. Placement of the network device or devices implementing the firewall 214 is preferably selected such that the firewall 214 examines all network traffic

IPR2023-01353 Page 00900

destined for the network devices it is supposed to protect.

[0048] The filters 216 are implemented as part of the firewall 214. Alternatively, the filters 216 are implemented as part of separate data structure accessible by the firewall 214. The firewall 214 and filters 216 execute a firewall policy designed to protect network devices 204, 206, 208 from malicious attacks originating from network devices 210, 212 coupled to the public network. The firewall 214 also provides added functionality such as facilitating parental controls, intrusion detection, logging of network packets, and other added filter based functionality.

[0049] Each filter 216 includes a set of filter conditions and one or more associated actions. The filter conditions include parameters and information that can be parsed or otherwise obtained from network packets such as interface numbers, hardware addresses, network addresses, protocol type, port numbers, and payload data. The one or more associated actions define how the network device implementing the firewall should treat packets that match the filter conditions. Typical actions include permit, i.e. allow the packet to continue network traversal, and block, i.e. preclude further network traversal by dropping packet.

[0050] The firewall 214 examines network packets traversing the network when received at network device 208 and, by comparing the packet parameters to the filter conditions, identifies one or more matching filters. A matching filter results when the filter conditions match the packet parameters. The packet parameters, like filter conditions, include information that is parsed or otherwise obtained from packets. When the firewall identifies a matching filter, the one or more actions associated with the filter conditions are executed.

[0051] FIG. 3 shows an overall view an embodiment of the firewall architecture of the present invention. The architecture provides the ability to subject packets to filters at all layers of a network stack. The firewall provides central management capabilities that allow filters to be added and deleted and for filter conflicts to be identified and resolved. The firewall architecture is extensible in that filter layers are added and deleted as needed and is expanded to include specialized functionality beyond permit and block actions. Although the invention is described with specific reference to firewalls and firewall filters, it is also used to facilitate and manage other filters and policies. As specific examples, the present invention is suitable for facilitating and managing filters used with Quality of Service (QOS), Internet Protocol Security (IPSec) suite, as well as other encrypting, authenticating, and key management protocols.

[0052] The firewall architecture includes a user mode process 250 and a kernel mode process 252. The user mode process 250 and kernel mode process 252 execute as part of an operating system in a network device. Those skilled in the art will appreciate that the user mode process 250 and the kernel mode process 252 of the operating system include additional components that for simplicity are not shown. Alternatively, the firewall architecture is executed in whole or in part outside of the operating system as one or more program modules or application programs or within a single operating system process.

[0053] The kernel mode process 252 includes a network stack 254, a first firewall engine referred to herein as a kernel firewall engine 256, and optional callouts 258. Collectively, the kernel mode process 252 implements an established firewall policy by identifying matching filters for network packets, processing packets according to known protocols, and executing other actions on the packet as designated by matching filters.

[0054] The network stack 254 comprises a plurality of layer processes, referred to herein as layers, including a data stream layer 268, a transport layer 270, a network layer 272, and a link layer 274. The firewall architecture of the present invention is extensible and additional layers are dynamically added and deleted as needed. An example of an added layer includes file access layer 276, implemented according to a Server Master Block (SMB) protocol. The layers may work in cooperation with other program modules such as a Hypertext Transfer Protocol (HTTP) parser module 278.

[0055] The layers in the network stack 254 process inbound and outbound network packets. Outbound network packets are packets being transmitted from the network device implementing the firewall architecture onto the network. Inbound packets are packets received at the network device implementing the firewall architecture. As indicated by the respective arrows shown in Fig. 3, inbound packets traverse the network stack 254 from bottom to top and outbound packets traverse the network stack 254 from top to bottom.

[0056] Network packets traverse, and are processed by, the network layers sequentially. In accordance with known techniques, each layer in the network stack 254 is capable of receiving packets from a previous layer or process, processing the packet according to a specification or protocol, and sending the processed packet to a next layer or process. In accordance with the invention, each layer in the network stack 254 also maintains packet context, passes the packet context to the next layer, issues a classification request to the kernel firewall engine 256, and takes action on the packet according to the firewall policy..

[0057] The packet context is a data structure that follows the packet from layer to layer. Each layer maintains the context by adding into the context data structure a set of parameters that the layer is designed to process, e.g. the information that the layer is designed to parse from inbound packets, add to outbound packets or otherwise derive from contents of packets. An exemplary data structure used for the packet context is described with reference to Fig. 5.

[0058] One of the operations carried out by the layers of the network stack 254 is to invoke the kernel firewall engine by issuing the classification request. The classification request is a call by a layer in the network stack 254 requesting that any filters that match the packet be identified and any associated action, e.g. permit or block, be returned to the

IPR2023-01353 Page 00901

layer. The layer issuing the classification request is referred to herein as a requesting layer. Each layer also takes the action on the packet that is returned by the kernel firewall engine 256.

[0059] The kernel firewall engine 256 includes a layer API 280, a set of installed filters 282, and a callout API 284. The kernel firewall engine 256 performs various functions including (1) maintaining the set of installed filters 282 defining the firewall policy, (2) receiving classification requests from the layers in the network stack 254, (3) identifying one or more matching filters based on the classification request, and (4) instructing the requesting layer of any action to take on the packet based on the matching filters.

[0060] Each filter of the set of installed filters includes a set of filter conditions and one or more associated actions. As described with reference to FIG. 2, the filter conditions identify the network packets that are subject to the associated filter action. The actions specified in the set of installed filters 282 include permit and block. Additional functionality is added via optional callouts 258. An exemplary form of the filters is described with reference to Fig. 4.

[0061] The layer API 280 provides an interface between the layers in the network stack 254 and the kernel firewall engine 256. Through the layer API 280, the requesting layer issues the classification request to the kernel firewall engine 256. The classification request includes the packet as received by requesting layer, the packet context as received by the requesting layer, and layer parameters. The layer parameters are packet parameters processed by, e.g. added or parsed, by the requesting layer. As a specific example, source and destination Internet Protocol (IP) addresses are layer parameters sent by the network layer 272 when implementing the IP protocol. Layer parameters may also include information beyond the packet parameters that are added to or parsed from the packet. As a specific example, the layer parameters include a local address type. The local address type is determined by the IP layer and sent as part of the classification request. Local address types include unicast, broadcast, multicast, anycast and the like. A specific implementation of the layer API 280 is described with reference to FIG. 6.

[0062] Optionally, the callouts 258 are used to implement added functionality beyond the permit and block filter actions. A callout is executed when the kernel firewall engine 256 identifies a matching filter for the packet that includes, as the associated action, a callout to one of the callout modules. The kernel firewall engine sends the classification request as issued by the requesting layer, i.e. full packet, layer parameters, and packet context, to the callout module, along with an identification of the matching filter, via the callout API 284. In an embodiment of the invention, the firewall architecture includes a base set of callouts 258. Additional callouts, like layers, are added as needed to the extensible firewall architecture. A specific implementation of the callout API 284 is described with reference to Fig. 6.

[0063] The user mode process 250 includes a second firewall engine referred to as a user firewall engine 260 and one or more policy providers 262 identified as "PP1," "PP2," and "PP3." The policy providers 262 are processes that add firewall policy, i.e. installed filters 282, into the firewall architecture. Any process is used to accomplish this task. An example is a legacy IPSec policy service (LIPS). The legacy IPSec policy service adds filters defining network traffic that is supposed to use IPSec protocols such as Encapsulating Security Protocol (ESP) and Authentication Header Protocol (AH). As a specific example, the legacy IPSec policy adds a firewall policy indicating that all unsolicited inbound packets must be encrypted according to the ESP protocol. The policy further provides that any unsolicited inbound packet in clear text, i.e. an unencrypted packet, should be blocked. The policy providers 262 obtain the policy from any suitable source such as data in volatile or nonvolatile memory, or a Graphical User Interface (GUI) that permits an administrator or system users to directly enter policy. The user firewall engine 260 converts the policy into a new filter, i.e. defines the policy in terms of filter conditions and associated actions, and adds the new filter into the set of installed filters 282.

[0064] The user firewall engine 260 also performs filter arbitration and conflict resolution functions. When the policy provider 262 provides new policy to the user mode firewall engine 260, the user firewall engine determines whether the new filter resulting from the new policy conflicts with any of the installed filters 282. An example of a method of identifying and resolving conflicts that is suitable for use in the framework of the present invention is described in United States Patent Application entitled "Method for Managing Network Filter Based Policy," attorney docket number 221037.

[0065] The architecture further includes a filter engine API 266 that forms an interface between the user mode firewall engine 260 and the kernel firewall engine 256. The filter engine API 266 provides a mechanism for the user firewall engine 260 to add new filters to or remove existing filters from the set of installed filters 282, and to examine installed filters 282 so that filter conflicts can be detected and resolved. A management API 290, accessible by policy providers 262, also provides a mechanism to add and remove filters from the architecture.

[0066] The user mode firewall engine 260 also includes a filter module 294. The filter module 294 is an instance of the kernel firewall engine 256 in the user mode 250. The instance of the filter module 294 in the user mode firewall engine 260 permits the user firewall engine 260 to replicate the services of the kernel firewall engine 256 for one or more user mode layer processes referred to herein as user mode layers 282. The user mode layers 282 are added in the same way that kernel mode layers are created. Because the filter module 294 is the user mode instance of the kernel firewall engine 256, it will be understood that any functionality described herein for the kernel firewall engine 256 also applies to the filter module 294.

[0067] A keying module API 288 provides an interface between the user policy engine 260 and keying module 296.

IPR2023-01353 Page 00902

The keying module API 288 is used to signal the keying module of the need to establish an SA.

[0068] The firewall architecture includes a boot time policy 286 and a failsafe policy 264. The boot time policy 286 and the failsafe policy 264 are implemented to provide a basic set of installed filters 282 while the system is in a transitional state such as when the network device is initialized, e.g. turned on, reset, or restarted. During the transitional state, there is a latent period during which the user mode 250 is not established and it is not possible to install filters via policy providers 262 and the user firewall engine 260. As a result, the network device is vulnerable to malicious attacks from, for example, unsolicited inbound packets. The boot time policy 286 resides in the kernel mode 252 and is loaded into the kernel firewall engine 256 prior to the time the user mode 250 is established. The boot time policy 296 provides the basic installed filter 282 configurable to provide any suitable filter conditions and associated actions for protecting the system during the transitional state. An example of the boot time policy 286 is one that blocks all inbound and outbound network packets.

[0069] Once the user mode 250 is established, the transitional state continues while the policy providers 262 are identifying policy and attempting to add installed filters 282 into the kernel firewall engine 256. When the user mode 250 is established, the user firewall engine 260 first loads the fail-safe policy 264 into the kernel firewall engine 256 before any policy is accepted by the policy providers 262. The boot time policy 286 is then disabled. Like the boot time policy 286, the fail-safe policy 264 is a basic filter designed to protect the system from attacks during the transitional state when the policy providers and user firewall engine are still in the process of installing filters. Once the policy providers 262 successfully install their respective filters into the network device, the fail-safe policy 264 is disabled.

[0070] Referring to FIG. 4, the set of installed filters 282 will now be described. Each filter 310 of the set of installed filters 282 has a plurality of fields including a filter Id 312, a weight 314, one or more actions 316, a policy context 328, and a set of filter conditions 318. The filter Id 312 provides a unique identification to the filter. The filter Id 312 is used, for example, as a means for the kernel firewall engine 256 to return matching filter information to the user firewall engine 260 and the callouts 258. In an embodiment of the invention, the filter is assigned to one of the layers in the network stack 254. The filter Id 312 is used by the kernel firewall engine 256 to track which filter is assigned to which layer.

[0071] The weight field 314 includes a value that identifies priority of the filter 310. The higher the value in the weight field 314, the higher the priority of the filter. The filter priority determines the order whereby matching filters are applied to the packet by the kernel firewall engine 256.

[0072] In an embodiment of the invention, the filter with the highest priority, i.e. highest weight value, is applied first, then the next highest priority filter is applied and so on, until a matching filter with a terminating action is encountered. Terminating actions are described in more detail below. Once the matching filter with the terminating action is applied, the kernel firewall engine 256 stops applying matching filters. Thus, the action 316 specified by lower priority matching filters are not taken on the packet after the terminating action is applied. Alternatively, the firewall engine 256 identifies a single matching filter and returns a set of actions from the single matching filter.

[0073] The set of filter conditions 318 determine whether a packet matches the filter 310. Each filter condition 318 includes a type 320, data 322, and a layer Id:field Id 324. The type 320 defines the length and number of variables included in the corresponding data 322. The architecture provides for predefined known variable types such as Byte, Short, Long, 8 Bytes, String, Internet Protocol version 4 (IPv4) Address, Internet Protocol version 6 (IPv6) Address, IPv4 Address plus Mask, IPv6 Address plus Mask, and Address Range.

[0074] The data field 322 includes data matching the type. For example, if the type is IPv4 Address, an acceptable value for the data field 322 is a 32-bit number in range of 00.00.00.00 to 255.255.255.255 as expressed in dotted decimal notation. In some instances, the type 320 provides for multiple values in the data field 322. The Address Range, IPv4 Address plus Mask, and IPv6 Address plus mask types allow two IP address values, defining a beginning and ending range of IP addresses. For maximum flexibility, the architecture also permits user-defined types. Alternatively, additional types are manually added to the system architecture.

[0075] The layer Id:field Id 324 is used to identify an originating layer and a parameter from the originating layer, respectively. The originating layer and the parameter from the originating layer define packet parameters, i.e. layer parameters and packet context that the data 322 is compared against when identifying matching filters. The originating layer identifies a layer in the network stack. The parameter from the originating layer identifies a specific parameter associated with the originating layer. A specific example is illustrated by filter condition 326. The type is IPv4 thereby indicating the data 322 is a 32-bit IP address. The layer Id is "IP" representing that the 32 bit number is an IP, i.e. network, layer parameter. The field Id is "Src IP Addr" which in the example represents a source IP address. The IP address provided in the data field 322 is "123.3.2.1" indicating that any packet with that source IP address meets the filter condition thereby matching the filter. The filter 310 may include multiple filter conditions 318, in which case a packet matches the filter 310 only if all filter conditions are met.

[0076] The action 316 in the filter 310 is permit, block, or callout. If the action 324 in the filter 310 is permit or block and the packet matches the filter 310, the permit or block action is returned to the requesting layer by the kernel firewall engine 256. If the action 316 is callout, the kernel firewall engine 256 issues its own classification request that includes

IPR2023-01353 Page 00903

the full packet, layer parameters, context and identification of the matching filter to the designated callout module 258. The callout module 258 performs its programmed function on the packet, e.g. intrusion detection. The callout may return an action (either permit, block) to the kernel firewall engine, which in turn relays the action to the requesting layer. The callout is also capable of maintaining packet context that is likewise returned to the requesting layer via the kernel firewall engine 256. The action may also be a value, such as a null value, indicating that no firewall policy exists.

[0077] Actions are designated terminating or non-terminating. If the action is terminating, once that action is identified for a packet, the action is returned to the requesting layer and no additional filters are applied to the packet. If the action is non-terminating, additional filter lookups continue until a terminating action for the packet is identified. As a default, permit and block are designated terminating actions.

[0078] The policy context 328 is used to store policy other than firewall policy, such as security policy or QOS policy. The policy context is any suitable data structure. For example, the policy context is a 64 bit number that is interpreted by a process that added the policy context. The policy context and/or action may be a null value.

[0079] FIG. 5 illustrates an example of a data structure 330 used for the packet context that is maintained by, and passed to, the layers in the network stack 254 and the callout modules 258. The packet context 330 follows the inbound or outbound network packet as it traverses the layers and includes one or more entries, labeled 336-340. Each entry includes a layer Id:field Id 332 and a corresponding value 334.

[0080] The layer Id:field Id 332 has the same meaning as the layer Id:field Id 324 provided as part of the filter conditions 318 in the filters 310 (FIG. 4). Namely, the layer Id:field Id 322 identifies the originating layer and the layer parameter from the originating layer for the data in the value field 334. The value field 334 includes a specific layer parameter.

[0081] As a specific example, entry 336 includes the layer Id:field Id 332 "NDIS: Src. MAC Addr." "NDIS" represents a Network Driver Interface Specification implementation of the Link layer 274 (Fig. 1). "Src MAC addr." represents a source MAC address. Thus, the layer:field Id 332 indicates that the data in the value field 334 is a source MAC address that was processed by the NDIS (Link) layer. The value field 334 includes the actual source MAC address which in the example is "00.08.74.4F.22.E5" as expressed in hexadecimal notation.

[0082] As a second example, entry 338 has a layer Id:field Id 332 of "NDIS:IF No." This again identifies the layer as NDIS, but in this case identifies the parameter as "IF No" representing an interface number as the specific NDIS parameter. The value field 334 includes the actual interface number, which in this case is 2.

[0083] As a third example, entry 340 has a layer Id:field Id 332 of "IP:Dst IP Addr." The "IP" represents the network layer using the IP protocol and the "Dst IP Addr" represents a destination IP address as the IP layer parameter. The value field 334 includes the actual destination IP address of "123.3.2.1."

[0084] Having described the underlying firewall architecture of the present invention, attention is drawn to the functional interfaces of the system and exemplary methods that are performed using the underlying firewall architecture described herein. The functional interfaces are implemented as a plurality of application programming interfaces (APIs) The APIs include the layer API 280, the callout API 284, the filter engine API 266, and the keying module API 288 as shown in FIG. 6 and FIG. 7.

[0085] The layer API 280 facilitates data exchange between each of the layers in the network stack 254 and the kernel firewall engine 256.. As shown, the layer API 280 includes a Classify method 402, an Add Layer method 404, and a Delete Layer method 406. The layer API 280 may also execute within the filter module 294 to facilitate communication between the user mode layers and the filter module 294.

[0086] The Classify method 402 is used by the requesting layer to send layer parameters, the packet as received by the requesting, and the packet context to the kernel firewall engine 256. The kernel firewall engine 256 compares the (1) layer parameters from the requesting layer and (2) packet context entries to the filter conditions 318 in each filter 310 assigned to the requesting layer to identify matching filters. The following is an exemplary implementation of the Classify method. It will be understood that the following methods are described as receiving or returning data values. According to known programming techniques, the methods may use pointers to data values instead of actual data values.

```
NTSTATUS
WFPClassify
(
```


IPR2023-01353 Page 00904

```
    IN ULONG                          LayerId,
    IN WFP_INCOMING_VALUES*           pInFixedValues,
    IN WFP_INCOMING_CONTEXT_VALUE*    pInContext,
    PVOID                             pPacket,
    OUT WFP_ACTION_TYPE*              pActionType,
    OUT UINT64*                       pOutContext
);
```

[0087] Where the following characterizes the recited parameters.

[0088] **LayerId** identifies the network layer issuing the classification request, i.e. the requesting layer. Referring to FIG. 3, the Layer Id identifies the layer as the data stream layer 268, the transport layer 270, the network layer 272 or the link layer 274. Other layers are valid if added to the system. For example, if the SMB layer 276 is added, it has its own unique identification. The firewall architecture of the present invention further permits multiple protocol implementations at a layer in the network stack 254. For example the stack has two transport layers 270, a first transport layer uses the TCP protocol and a second transport layer uses the UDP protocol. User mode layers 282 may also be added thereby forming a valid layer.

[0089] **pInFixedValues** includes a subset of the layer parameters processed by the requesting layer. The pInFixedValues, along with the packet context entries, are compared to the filter conditions to determine if the packet matches the filter. An example of default layer parameters included in the pInFixedValues for each layer are identified in Table A below. It will be understood that the following values are exemplary only as an advantage of the architecture is that it allows the layers to use any values accessible to that layer.

Table A

Layer	Default Layer Parameters
Link Layer	Source and Destination MAC Addresses; Interface Number
Network Layer	Source and Destination IP Addresses; Protocol Type; Local Address Type
Transport Layer	Source and Destination Port Numbers;
Application	Deciphered application layer protocol payload

[0090] **pInContext** includes the context data structure 330 (FIG. 5) as received by the requesting layer. The packet context is used by the kernel firewall engine 256 in conjunction with the layer parameters to identify matching packets.

[0091] **pPacket** includes the entire packet as received by the requesting layer. The pPacket is not used by the kernel firewall engine 256 to identify matching filters. As previously described, the kernel firewall engine 256 uses the pInFixedValues and pInContext to identify matching filters. The pPacket is included in the Classify method so that the kernel firewall engine 256 can send it to one or more callout modules 258 identified as the action 316 in matching filters.

[0092] **pActionType** includes the action 316 that is returned to the requesting layer. The action 316 returned is permit or block as identified in the matching filter, or callout modules executed by the matching filter.

[0093] **pOutContext** includes the policy context data. As previously described, the policy context is used to facilitate network policies associated with IPSec, QOS and any other non-firewall filter based policy.

[0094] The Add Layer 404 and Delete Layer 406 methods are used to add and remove a layer from the firewall architecture, respectively. The following is an exemplary form of the Add Layer 404 method.

```
NTSTATUS
AddExtensionLayer(OUT PULONG pLayerId);
```

[0095] Where the following characterizes the recited parameters.

[0096] **pLayerId** is a unique layer identification value returned to the layer being added, i.e. the layer executing the Add Layer method.

[0097] The following is an exemplary form of the Delete Layer 406 method.

```
NTSTATUS
RemoveExtensionLayer(ULONG LayerId);
```

[0098] Where the following characterizes the recited parameters.

[0099] **LayerId** identifies the layer being removed; i.e. the layer executing the Delete Layer method.

IPR2023-01353 Page 00905

[0100] The Callout API 284 facilitates data exchange between the kernel firewall engine 256 and the callouts 258. The Callout API 284 includes a Classify method 410, a Notify method 412, a Register Callout method 414, and a Deregister Callout method 416. The Classify method 410 of the Callout API 284 is similar to the Classify method 402 of the Layer API 280 except that it also includes matching filter data. The following is an exemplary form of the Classify method 410 used to execute a callout.

```
typedef NTSTATUS (*WFP_CALLOUT_CLASSIFY_FN)
(
        IN const WFP_INCOMING_VALUES*          fixedValues,
        IN WFP_INCOMING_CONTEXT_VALUE*         wfpContext,
        IN VOID*                               packet,
        IN WFP_FILTER*                         matchedFilter,
        OUT WFP_ACTION_TYPE*                   action,
        OUT UINT64*                            outContext
);
```

[0101] Where the following characterizes the recited parameters.

[0102] **fixedValues** includes the layer parameters sent from the requesting layer. The fixedValues is the same data provided by the requesting layer in pInFixedValues data sent as part of the Classify method 402 in the layer API 280.

[0103] wfpContext includes the context data structure 330 (FIG. 5). This data is the same as sent by the requesting layer in the pInContext sent as part of the Classify method 402 in the layer API 280.

[0104] **packet** includes the entire packet as received by the requesting layer. This data is the same as sent by the requesting layer in the pPacket sent as part of the Classify method 402 in the layer API 280.

[0105] **matchedFilter** identifies the filter requesting the callout. Typically, the matching filter is identified by the filter Id 312 of the matching filter 310 initiating the Classify method 410 of the callout API 284.

[0106] **pActionType** includes the action that is returned to the kernel firewall engine 256 from the callout 258. If the pActionType is permit or block, it is returned to the requesting layer as the pActionType returned by the Layer API 280. The callout can also return a continue action that instructs the kernel firewall engine 256 to continue applying matching filters to the packet.

[0107] **pOutContext** includes the policy context data, such as security or QOS policy data.

[0108] The Notify method 412 is used to notify a callout when a filter 310 is added or deleted to the set of installed filters 282 that identifies as one of its actions 316, the callout module 258. The notify provides the callout with an opportunity to take any required action such as allocating or de-allocating buffers that will be used by the callout 258 when it is executed by the kernel firewall engine 256. The following is an exemplary form of the Notify method.

```
typedef NTSTATUS (*WFP_CALLOUT_NOTIFY_FN)
(
        IN WFP_NOTIFY_ENUM    notify,
        IN WFP_FILTER*        filter
);
```

[0109] Where the following characterizes the recited parameters.

[0110] **notify** includes a numerical value that indicates whether the filter is being added or deleted. For example, a value of 1 indicates the filter is being added and a value of 2 indicates that the filter is being deleted.

[0111] **filter** identifies the filter being added or deleted by unique value. This may be accomplished by providing the filter Id 312 included as part of the filter 310.

[0112] The callout Registration method 414 and callout Deregistration 416 method are used to add and remove callout modules, respectively. An exemplary form the registration method 414 is as follows:

IPR2023-01353 Page 00906

```
NTSTATUS WfpRegisterCallout
(
        IN const GUID*                  calloutId,
        IN const WFP_CALLOUT*           callout,
        IN const SECURITY_DESCRIPTOR*   sd
);
```

[0113] Where the following characterizes the recited parameters.

[0114] **callout Id** provides a unique identification for the registering callout module.

[0115] **callout** provides any callout specific information such as a driver service name, device name, and pointers to the callout classify and notify functions.

[0116] **sd** provides a security descriptor for the callout. The security descriptor provides an access control list (ACL) that specifies which processes can modify and delete the callout.

[0117] An exemplary form of the callout deregistration method 416 is as follows:

```
NTSTATUS WfpDeregisterCallout
(
        IN const GUID*      calloutId
);
```

[0118] Where the following characterizes the recited parameters.

[0119] **callout Id** is the unique Id of the callout to be removed.

[0120] The filter engine API 266 facilitates data exchange between the user mode firewall engine 260 and the kernel mode firewall 256 engine. As shown, the management API 266 includes an Add Filter method 418, a Delete Filter method 420, and an Enum Layer method 422. The methods of the filter engine API 266 are also provided as part of the management API 290.

[0121] The Add Filter 418 and Delete Filter 418 methods are used to add a new filter to the set of installed filters 282 and to delete an existing filter from the set of installed filters 282, respectively. The following is an exemplary form of the Add Filter method 418.

```
NTSTATUS
AddFilterToLayer
(
        ULONG               LayerId,
        WFP_FILTER*         pFilter
);
```

[0122] Where the following characterizes the recited parameters.

[0123] **LayerId** identifies the layer that the filter is assigned to.

[0124] **pFilter** is the filter 310 being added to the set of installed filters 282.

[0125] The following is an exemplary form of the Delete Filter method 420.

```
NTSTATUS
DeleteFilterFromLayer
(
        ULONG       LayerId,
        ULONG       FilterId
);
```


IPR2023-01353 Page 00907

[0126] Where the following characterizes the recited parameters.

[0127] **LayerId** identifies the layer that the filter is assigned to.

[0128] **pFilter** is the filter being deleted from the set of installed filters.

[0129] The Enum Layer method 422 provides a mechanism for the user firewall engine 260 to identify all filters matching a set of criteria. This allows the filter engine API to identify conflicting filters for filter arbitration and conflict resolution. The following is an exemplary form of the Enum Layer method 422.

```
LayerStartEnum
(
        PWFP_ENUM_TEMPLATE      pEnumTemplate,
        OUT PULONG              pMatchCount,
        OUT PWFP_ENUM_HANDLE    pEnumHandle
)
```

[0130] Wherein the following characterizes the recited parameters.

[0131] **pEnumTemplate** includes a data structure defining the filters that should be returned. Fore example, it include parameters that the filter conditions must match for the filter to be returned.

[0132] **pMatchCount** includes the number of filter matches based on the specified pEnumTemplate.

[0133] **pEnumHanlde** includes a reference to the matched filter entries.

[0134] The keying module API 288 provides an interface between keying modules and the user firewall engine 260. The keying module API 288 includes an IPSec SA Acquire method 430, an Expire Notify method 432, and an IPSec SA Acquire Complete method 434, a Keying Module Register method 436, a Keying Module Deregister method 438, an IPSec Inbound Get SPI method 440, an Add Inbound SA method 444, an Add Outbound SA method 446, an Inbound SA Expire 448 method, and a Keying Module Initiate method 350. The keying module API 288 may also include the methods previously described for the layer API, such as a Classify method 402, an Add Layer method 404 and a Delete Layer method 406.

[0135] The keying module API is used to facilitate use of known security protocols, such as defined by IPSec, used by an initiating computer and a responding computer. IPSec includes protocols such as AH and ESP. The ESP protocol is an authenticating and encrypting protocol that uses cryptographic mechanisms to provide integrity, source authentication, and confidentiality of data. The AH protocol is an authentication protocol that uses a hash signature in the packet header to validate the integrity of the packet data and authenticity of the sender.

[0136] An IKE protocol provides a method for an initiating computer and a responding computer to negotiate security settings used with the AH and ESP protocols. The negotiated security settings form a data structure called a security association (SA). The SA defines parameters such as an authentication algorithm, encryption algorithm, keys, and the lifetime of keys, used by ESP or AH to protect the contents of an IP packet. Because ESP and AH require an established SA, an IKE negotiation is executed before the ESP or AH protocols are used by the initiating and responding computer. A given SA is identified by a value known as a Security Parameter Index (SPI).

[0137] Each of the initiating and responding computers include an IPSec driver that determines, based on an IPSec policy, whether data sent between the initiating and responding computers requires encryption or authentication. The IPSec policy is a set of filters that defines how the network device uses IPSec and includes filter lists, authentication methods, and other information In an embodiment of the invention, the IPSec policy is defined by filters included in the set of installed filters in the kernel firewall engine or the filter module 294. The policy that applies to a packet is identified by invoking the keying module layer which uses the Classify method to identify matching filters.

[0138] The IPSec SA Acquire method 430 is called by the user firewall engine 260 (via a client proxy) to pass a driver acquire or an external initiate request a user mode layer 282 implemented as a keying module layer. The keying module layer returns this call and does the negotiation asynchronously. Once the keying module layer has completed the negotiation, the keying module layer calls the IPSec SA Acquire Complete method 343 to notify the user policy engine 260 that the negotiation is complete. The following is an exemplary form of the IPSec SA Acquire method.

IPR2023-01353 Page 00908

```
typedef WIN32_ERR (*PROCESS_IPSEC_SA_ACQUIRE0)
(
        IN FWP_IPSEC_ACQUIRE_CONTEXT0   ipsecContext,
        IN const FWP_IPSEC_SA_ACQUIRE0* acquire,
        IN FWP_IPSEC_SPI                inboundSAspi
);
```

[0139] Wherein the following characterizes the recited parameters.

[0140] **ipsecContext** is a handle to link the acquire with the SA being added.

[0141] **acquire** includes the necessary information for negotiating the SA according to known protocols such as IKE.

[0142] **inboundSAspi** includes a SPI that is used for an inbound SA.

[0143] The Expire Notify method 432 is called to pass an expire-notify to the keying module layer that added the inbound SA. The following is an exemplary form the Expire Notify method.

```
typedef VOID (*PROCESS_IPSEC_SA_EXPIRE0)
(
        IN const FWP_IPSEC_SA_EXPIRE_NOTIFY0* expireNotify
);
```

[0144] Wherein the following characterizes the recited parameters.

[0145] **expireNotify** contains information identifying the expiring SA. For example, in the case of an outbound SA, the SPI is provided.

[0146] The IPSec SA Acquire Complete 434 method is called by a keying module layer to close the user firewall engine's context after it has finished negotiation and added all the SAs, or after it has come across an error. After this method is executed, the keying module layer does not reuse the ipsecContext for any other API method. The following is an exemplary form the IPSec SA Acquire Complete method.

```
WIN32_ERR
FwpIPSecSAAcquireComplete0
(
        IN FWPM_ENGINE_HANDLE                   engineHandle,
        IN FWP_IPSEC_ACQUIRE_CONTEXT0           ipsecContext,
        IN const FWP_IPSEC_NEGOTIATION_STATUS0* status
);
```

[0147] Wherein the following characterizes the recited parameters.

[0148] **engincHandle** provides a handle to the user firewall engine 260.

[0149] **ipsecContext** is the context passed by user firewall engine with the IPSec Acquire method.

[0150] **status** provides status and other details of the SA negotiation. The status is returned by the user firewall engine 260 if the acquire was externally initiated via FwpKeyingModuleInitiate0.

[0151] The Keying Module Register method 436 is called by a keying module layer to register with the user firewall engine 260 and pass its function pointers. The following is an exemplary form of the Keying Module Register method.

IPR2023-01353 Page 00909

```
WIN32_ERR
FwpKeyingModuleRegister0
(
        IN FWPM_ENGINE_HANDLE                   engineHandle,
        IN const GUID*                          keyingModuleID,
        IN const FWP_KEYING_MODULE_INFO0*       keymodInfo
);
```

[0152] Wherein the following characterizes the recited parameters.

[0153] **engineHandle** provides the handle to the user firewall engine 260.

[0154] **keyingModuleID** is a unique ID for the keying module layer.

[0155] **keymodInfo** includes registration information about the keying module layer such as pointers to process IPSec SA Acquire and process IPSec SA Expire functions.

[0156] The Keying Module Deregister method 438 is called by the keying module layer to deregister the keying module from the user firewall engine 260. The following is an exemplary form of the Keying Module Deregister method.

```
WIN32_ERR
FwpKeyingModuleDeregister0
(
        IN FWPM_ENGINE_HANDLE       engineHandle,
        IN const GUID*              keyingModuleID
);
```

[0157] Wherein the following characterizes the recited parameters.

[0158] **engineHandle** is the handle to the user firewall engine 260.

[0159] **keyingModuleID** is the unique ID of the keying module layer.

[0160] The IPSec Inbound Get SPI method 440 is called by the keying module layer to obtain the SPI for a new inbound SA. The IPSec Inbound Get SPI method 440 is usually used when the keying module layer executes in a responding network device. The following is an exemplary form the IPSec Inbound Get SPI method.

```
WIN32_ERR
FwpIPSecSAInboundGetSpi0
(
        IN FWPM_ENGINE_HANDLE               engineHandle,
        IN const FWP_IPSEC_TRAFFIC0*        ipsecTrafficDescription,
        IN const FWP_IPSEC_UDP_ENCAP0*      udpEncapInfo,
        OUT FWP_IPSEC_SPI*                  inboundSpi
);
```

[0161] Wherein the following characterizes the recited parameters.

[0162] **engineHandle** is the handle to the user firewall engine 260.

[0163] **ipsecTrafficDescription** is a 5-tuple description for creating an inbound larval SA. The 5-tuple includes source and destination IP addresses, source and destination ports, and transport layer protocol type.

[0164] **udpEncapInfo** is UDP encapsulation data for creating the larval SA. UDP encapsulation is a known method of embedding a packet formatted according to a security protocol into an unencrypted UDP packet.

[0165] **inboundSpi** is the SPI for the inbound SA.

[0166] The Add Inbound SA method 444 is called by the keying module layer to add an inbound SA, i.e. update the larval SA. The user firewall engine 260 uses the SPI in the SA to map this call to its internal state, and ioctl the SA down to the IPSec driver. The following is an exemplary form of the Add Inbound SA method.

IPR2023-01353 Page 00910

```
WIN32_ERR
FwpIPSecSAInboundAdd0
(
        IN FWPM_ENGINE_HANDLE              engineHandle,
        IN const FWP_IPSEC_SA_STRUCT0*     inboundSA
);
```

[0167] Wherein the following characterizes the recited parameters.

[0168] **engineHandle** in the handle to the user firewall engine 260.

[0169] **inboundSA** includes the inbound SA.

[0170] The Add Outbound SA 446 method is called by a keying module layer to add an outbound SA. The user firewall engine uses an inbound SPI parameter to map this call to its internal state, and ioctl the SA down to the IPSec driver.. The following is an exemplary form the Add Outbound SA method.

```
WIN32_ERR
FwpIPSecSAOutboundAdd0
(
        IN FWPM_ENGINE_HANDLE              engineHandle,

        IN FWP_IPSEC_SPI                   inboundSpi,
        IN const FWP_IPSEC_SA_STRUCT0*     outboundSA
);
```

[0171] Wherein the following characterizes the recited parameters.

[0172] **engineHandle** is the handle to the user firewall engine 260.

[0173] **inboundSpi** is the SPI for the inbound SA, with which an outbound SA is paired.

[0174] **outboundSA** includes the outbound SA.

[0175] The Inbound SA Expire method 448 is called by the keying module layer to expire the inbound SA that was previously added. The following is an exemplary form the Inbound SA Expire method 448.

```
WIN32_ERR
FwpIPSecSAInboundExpire0
(
        IN FWPM_ENGINE_HANDLE              engineHandle,
        IN const FWP_IPSEC_SA_EXPIRE0*     expire
);
```

[0176] Wherein the following characterizes the recited parameters.

[0177] **engineHandle** is the handle to the user firewall engine 260.

[0178] **expire** includes data for the SA to be expired.

[0179] The Keying Module Initiate method 450 is called by known external application like RAS, Winsock API, and the like to initiate the keying module layer and setup SAs before an application starts sending its network traffic. The user firewall engine 260 asynchronously pends the RPC call, gets the SPI from an IPSec driver, and passes the acquire to the appropriate keying module. Once the keying module layer calls FwpIPSecSAAcquireComplete0, the user firewall engine completes the asynchronous RPC with the negotiation status. The following is an exemplary form of the Keying Module Initiate method.

IPR2023-01353 Page 00911

```
WIN32_ERR
FwpKeyingModuleInitiate0
(
        IN FWPM_ENGINE_HANDLE                   engineHandle,
        IN const FWP_IPSEC_SA_ACQUIRE0*         acquire,
        IN HANDLE                               waitEvent,
        OUT FWP_IPSEC_NEGOTIATION_STATUS0*      negotiationStatus
);
```

[0180] Wherein the following characterizes the recited parameters.

[0181] **engineHandle** is the handle to the user firewall engine 260.

[0182] **acquire** includes data necessary for negotiating an SA.

[0183] **waitEvent** is a handle to an event that is triggered when negotiation status is available. If a client, i.e. calling external application, is not interested in waiting for the negotiation to complete, it can set this parameter to NULL. Internally the client proxy optionally passes this event to RPC and request it to set the event once the asynchronous RPC call completes.

[0184] **negotiationStatus** includes an outcome of the negotiation. The negotiationStatus is NULL if waitEvent is NULL. Otherwise negotiationStatus remains valid until the waitEvent is triggered.

[0185] FIG. 8 illustrates the methods used by the layers in the network stack 254 according to the present invention. The method illustrated in FIG. 8 may also be used by one or more user mode layers 282 in concert with the filter module 294 and the user firewall engine 260.

[0186] Each layer is capable of a plurality of functions including processing network packets, issuing classification requests to the kernel firewall engine 256, and managing packet context. In an embodiment of the invention, the functions are executed at each of the layers in a shim 460, 462, 464, 466 installed in the network stack 254. Alternatively, the functionality is directly built into the individual layers without the need for the shims.

[0187] The network stack 254 includes the data stream layer 268, the transport layer 270, the network layer 272, and the link layer 274. For the purposes of illustrating the present invention, the link layer 274 is implemented as a NDIS driver, the network layer 272 is implemented as an IP layer, the transport layer 270 is implemented as a TCP layer, and the data stream layer 268 is implemented as a HTTP layer. It will be understood that layers may be implemented according to any protocol. For example the transport layer also accommodates a User Datagram Protocol (UDP). The application layer supports File Transfer Protocol (FTP), Remote Procedure Call (RPC), Simple Mail Transfer Protocol (SMTP), Server Master Block (SMB) etc. As previously described, additional layers can be added to the architecture and layers may be deleted. For example, layers are added and deleted using the Add Layer and Delete Layer methods described with reference to FIG 6.

[0188] A network packet, labeled 468(a)-(d), illustrates the network packet as it traverses and is processed by the layers in the network stack 254. If the packet 468(a)-(d) is an inbound packet, it traverses the network stack from bottom to top. If the packet 468(a)-(d) is an outbound packet, it traverses the network stack from top to bottom. While such processing is well known, it is briefly described for the purpose of illustrating the present invention.

[0189] Assuming an application executing in a network device, such as a web browser, initiates a request for contents of a web page located on another network device, the application issues the request to the data stream layer 268. In the example, the data stream 264 formats the request according to the HTTP protocol and sends the request to the transport layer in packet 468(a). The transport layer 270 receives packet 468(a). The transport layer 270, which implements the TCP protocol, places the data in one or more packets, and each packet is provided with a TCP header. The TCP header includes information such as the source and destination ports, protocol type, i.e. TCP, sequence numbers, flags, and checksums. The transport layer then sends the packet, labeled 468(b) to the network layer.

[0190] The network layer implements the IP protocol and encapsulates the data within an IP header that includes the source and destination IP addresses, flags, checksums and other known information. The IP header also indicates whether the packet is fragmented. A packet is fragmented when the size of the IP packet exceeds a Maximum Transmission Unit (MTU) size for the network technology used to transmit the packet. For example, Ethernet technology specifies that the MTU is 1500 bytes. If the IP packet length exceeds the MTU, it is fragmented into two or more IP packets, each with its own IP header, all of equal or smaller length as compared to the MTU.

[0191] In an embodiment of the invention, the network layer is divided into first and second layers. The first layer, referred to as a fragment layer, processes IP packet fragments. The second layer, referred to as a fully assembled layer, processes full IP packets, e.g. before outbound IP packets are fragmented and after inbound IP packets are reassembled into a single IP packet. Following network layer processing and possible fragmentation, the packet 468

IPR2023-01353 Page 00912

(c) is sent to the link layer 274. The link layer 274 further packetizes the data by providing a MAC header, with the source and destination MAC addresses, as well as other information. The packet is then sent to a network interface card (NIC) where it is physically transmitted onto the network.

[0192] Inbound packets are processed in a reciprocal fashion. Packet 468(d) is received by the NIC and sent to the link layer 274. The MAC header is removed and the packet 468(c) is sent to the network layer where the IP packet fragments are reassembled, if necessary, and the IP header is parsed. The network layer then sends the packet 468 (b) to the transport layer where the TCP header is removed and if the data stream was sent in a plurality of TCP packets, the data stream is reassembled. Finally, the data stream 468(a) is sent to the data stream layer 268 where the data is deciphered by the application protocol, in this case the HTTP protocol.

[0193] For each outbound packet, the layers of the network stack maintain packet context 470(a)-(c). For each inbound packet, the layers of the network stack maintain packet context 472(a)-(c). The packet context follows each packet as it traverses the network layers. Packet context is also passed to, and may be modified by, the callouts 258 (FIG. 1).

[0194] The packet context is updated as the packets are processed at each layer. Each layer adds its layer parameters to the packet context thereby providing this information to subsequent layers or processes. As shown, the link layer 274 adds source and destination MAC addresses and an interface number for an inbound packet as illustrated by context 472(a) That context is received by the network layer 272, which adds source and destination IP addresses as illustrated by context 472(b). The transport layer 266 receives the context and adds the port numbers as illustrated by context 472(a).

[0195] A similar process occurs for the context 470(a)-(c) associated with outbound packets. The data stream layer 268 adds information such as a URL address from the packet payload as illustrated by context 470(a), the transport layer 270 further adds source and destination port numbers as illustrated by context 470(b) and the network layer adds source and destination IP addresses as illustrated by context 470(c).

[0196] It will be understand that each layer can add any context information available to that layer. Typically, this includes any information that the layer is designed to process, i.e. add or parse or otherwise derive from packets.

[0197] When the packet and its corresponding context are received at each layer, the layer functions as a requesting layer by identifying the layer parameters and sending a classification request, labeled 474. The classification request 474 includes, the layer parameters 476, the packet context 478 received from the prior layer, and the full packet 480. An exemplary method used to issue the classification request is the Classify method 402 described with reference to the layer API 280 (Fig.7).

[0198] In response to each classify request, the kernel mode firewall engine 256 compares the layer parameters 476 and packet context 478 to the filter conditions 318 (FIG. 4) of the filters assigned to the requesting layer. The kernel firewall engine 256 sends a response, labeled 482, with the action 484 from the matching filters 310 with the highest weight 314 to the requesting layer. The kernel firewall engine 256 also returns the policy context 482. If no matching filters are identified by the kernel firewall engine 256, the kernel firewall engine notifies the requesting layer that no matching filters exist. The kernel firewall engine 256 continues to identify matching filters until a matching filter designates a terminating action, i.e. permit or block, or until all filters assigned to the requesting layer are checked, whichever comes first. Alternatively, the kernel firewall engine 256 identifies all matches and returns the actions to the requesting layer in a single response.

[0199] As generally illustrated, identifying layer parameters is done as part of the normal layer processing performed by the layers of the network stack 254. Additional packet parsing is not required thereby minimizing impact on system performance. Moreover, because the layers cooperate in maintaining packet context, it is possible to for the firewall engine 256 to compare filter conditions with packet parameters at layers that normally do not have access to those packet parameters. For example, the network layer 268 receives context for inbound packets from the link layer 274 that includes source and destination MAC addresses. Because the network layer 272 issues classification requests with network layer parameters, e.g. source and destination IP addresses, as well as the packet context, the kernel firewall engine 256 can filter on both IP and MAC addresses at the network layer 272 even though the MAC addresses are usually not available.

[0200] Referring to Fig. 9 and exemplary set of callout modules 258 included with the firewall architecture of the present invention will now be described. The callout modules 258 include HTTP context callout 500, intrusion detection callout 502, IPSec callout 504, and logging callout 506. The callout API 284 forms the interface between the kernel firewall engine 256 and the callouts 258.

[0201] The HTTP context callout 500 maintains a cache 508 of acceptable, or alternatively, unacceptable URL addresses. The HTTP context callout 508 periodically accesses a server 510 interfaced to the public network that maintains URL addresses and classifies them as acceptable or unacceptable. When the kernel firewall engine 256 executes the HTTP context callout, the callout examines the packet, deciphers the URL address if necessary, and determines if it is acceptable based on information in the cache 508. The HTTP callout then returns permit as the action 316 if the URL address is acceptable and block if the URL address is unacceptable to the kernel mode firewall engine 256, which

IPR2023-01353 Page 00913

in turn returns the action 316 to the requesting layer via layer API 280. The HTTP context callout is useful in implementing a parental control function.

[0202] The intrusion detection callout 502 uses available algorithms and techniques to examine the packet to identify indicia of viruses or otherwise suspicious packets. If a suspicious packet is detected, an action 316 of block is returned. An example of a suspicious packet is a packet where all flags in IP and TCP headers are set to a value of one. The packet is suspicious because it is never valid and could indicate an attack signature. Alternatively, the intrusion detection callout 502 modifies packet context to flag the suspicious nature of the packet thereby deferring a decision on whether to block the packet to subsequent layers in the network stack.

[0203] The IPSec callout 504 is designed to determine to whether a proper security protocol was applied to the packet. The IPSec callout 504 communicates with an IPSec process and determines based on IPSec policy whether the packet was supposed to be subject to IPSec processing. If so, the IPSec callout 504 verifies, based on packet context, whether the packet was in fact subject to the IPSec processing. If the packet was supposed to be processed according to IPSec, but was not, e.g. the packet was in clear text, an action of block is returned. If the packet was subject to IPSec processing, the IPSec callout verifies that a proper Security Association (SA) was applied. If the proper SA was not applied, the callout returns an action 316 of block, otherwise an action 316 of permit is returned. Alternatively, the IPSec callout modifies the packet context deferring a decision on whether to block or permit the permit to a subsequent layer.

[0204] The logging callout 506 is used to save information pertaining to the packet, such as the full packet for later use. Such later use might be, for example, diagnostics for certain network traffic that is unexpectedly not traversing the network because it is being blocked or for tracking malicious attacks on the system.

[0205] All of the references cited herein, including are hereby incorporated in their entireties by reference.

[0206] In view of the many possible embodiments to which the principles of this invention may be applied, it should be recognized that the embodiment described herein with respect to the drawing figures is meant to be illustrative only and should not be taken as limiting the scope of invention. For example, those of skill in the art will recognize that the elements of the illustrated embodiment shown in software may be implemented in hardware and vice versa or that the illustrated embodiment can be modified in arrangement and detail without departing from the spirit of the invention. Therefore, the invention as described herein contemplates all such embodiments as may come within the scope of the following claims and equivalents thereof.

Claims

1. A firewall framework implemented within a computer system for providing multi-layering filtering of packet, comprising:

   a set of layer processors, each layer processor being capable of processing layer parameters for the packet associated with the layer processor and each layer processor being further capable of issuing a classification request that includes the layer parameters; and
   a first firewall engine including:

   a layer interface for receiving first layer parameters from a requesting layer processor and for returning an action to the requesting layer, the requesting layer processor being one of the set of layer processors;
   a set of installed filters; and
   a lookup component for identifying at least one matching filter from the set of installed filters and identifying from the matching filter the action to be returned by the layer interface.

2. The firewall framework of claim 1, wherein the requesting layer receives packet context including second layer parameters from a previous layer processor and the classification request includes the packet context, wherein the lookup component further uses the packet context for identifying the at least one matching filter.

3. The firewall framework of claim 2, wherein the firewall engine further comprises:

   a callout interface for sending the first layer parameters, the packet context, and the packet to a callout, wherein the callout analyzes the packet and returns the action to the firewall engine.

4. The firewall framework of claim 2, wherein the requesting layer passes the packet context to a next layer.

5. The firewall framework of claim 1, further comprising:

IPR2023-01353 Page 00914

a policy provider for establishing a new filter; and
a second firewall engine for adding the new filter to the set of installed filers.

6. The firewall framework of claim 1, wherein each of the installed filters comprises:

a filter condition including a type-data pair, the type defining the size of the data and the data including a filter parameter, and
the action.

7. The firewall framework of claim 6, wherein the filter parameter includes a range of values.

8. The firewall framework of claim 6, wherein the filter comprises a weight factor defining a priority of the filter.

9. The firewall framework of claim 1, wherein the set of layer processors further comprises:

a link layer having layer parameters including an interface number and source and destination MAC addresses;
an network layer having layer parameters including a source and destination IP address;
a transport layer having layer parameters including source and destination ports; and
an application layer having layer parameters including a stream of data.

10. The firewall framework of claim 9, wherein the network layer is further divided into a fragment layer and a fully assembled layer, the fragment layer processing IP packet fragments and the fully assembled layer processing complete IP packets, each of the fragment layer and the fully assembled layer issuing the classify request.

11. The firewall framework of claim 1, wherein the first firewall engine executes in an operating system kernel mode.

12. The firewall framework of claim 1, wherein the first firewall engine executes in an operating system user mode.

13. A method of communicating between a first layer process and a firewall process in an operating system, comprising the steps of:

issuing, by the first layer process, a classify call having a plurality of parameters comprising, a protocol packet, at least one layer parameter. and a packet context from a second layer process;
receiving, by the firewall process, the classify call; and
issuing, by the firewall process, an action identified from a filter matching the at least one layer parameter.

14. The method of claim 13, wherein the first layer process is a new process, further comprising:

issuing by the first layer process, an add layer call.

15. The method of claim 13, wherein the first layer process is an existing process, further comprising:

issuing by the first layer process, a delete layer call.

16. A method of communicating between a firewall process and a callout process in an operating system, comprising the steps of:

issuing, by the firewall process, a classify call having a plurality of parameters comprising, a protocol packet, at least one layer parameter, a packet context, and a matching filter identification;
receiving, by the callout process, the classify call; and
issuing, by the callout process, an action identified from the plurality of parameters in the classify call.

17. The method of claim 16, wherein the callout process is a parental control module maintaining a cache of acceptable resource location, further comprising:

examining the packet to identify a resource location and comparing the resource location to the acceptable locations.

IPR2023-01353 Page 00915

18. The method of claim 17, wherein the action issued by the callout process is block and the protocol packet is prevented from network traversal.

19. The method of claim 17, wherein the action issued by the callout process is permit and the protocol packet is permitted to further traverse the network.

20. The method of claim 16, wherein the callout process is a logging module, further comprising storing in a memory the protocol packet.

21. The method of claim 16, wherein the callout process is a security module, further comprising:

determining, by the security module, that the protocol packet is required to conform to a security protocol;
verifying that the protocol packet conforms to the security protocol.

22. The method of claim 21, wherein the security protocol is an IPSec security protocol.

23. A computer-readable medium for executing computer-readable instructions for facilitating a firewall framework implemented within a computer system for providing multi-layering filtering of a packet, comprising:

a set of layer processors, each layer processor being capable of processing layer parameters for the packet associated with the layer processor and each layer processor being further capable of issuing a classification request that includes the layer parameters; and
a first firewall engine including:

a layer interface for receiving first layer parameters from a requesting layer processor and for returning an action to the requesting layer, the requesting layer processor being one of the set of layer processors;
a set of installed filters; and
a lookup component for identifying at least one matching filter from the set of installed filters and identifying from the matching filter the action to be returned by the layer interface.

24. The computer-readable medium of claim 23, wherein the requesting layer receives packet context including second layer parameters from a previous layer processor and the classification request includes the packet context, wherein the lookup component further uses the packet context for identifying the at least one matching filter.

25. The computer-readable medium of claim 24, wherein the firewall engine further comprises:

a callout interface for sending the first layer parameters, the packet context, and the packet to a callout, wherein the callout analyzes the packet and returns the action to the firewall engine.

26. The computer-readable medium of claim 24, wherein the requesting layer passes the packet context to a next layer.

27. The computer-readable medium of claim 23, further comprising:

a policy provider for establishing a new filter; and
a second firewall engine for adding the new filter to the set of installed filers.

28. A computer-readable medium for executing computer-readable instructions for communicating between a first layer process and a firewall process in an operating system, comprising the steps of:

issuing, by the first layer process, a classify call having a plurality of parameters comprising, a protocol packet, at least one layer parameter, and a packet context from a second layer process;
receiving, by the firewall process, the classify call; and
issuing, by the firewall process, an action identified from a filter matching the at least one layer parameter.

29. The computer-readable medium of claim 28, wherein the first layer process is a new process, further comprising:

issuing by the first layer process, an add layer call.

IPR2023-01353 Page 00916

30. The computer-readable medium of claim 28, wherein the first layer process is an existing process, further comprising:

issuing by the first layer process, a delete layer call.

31. A computer-readable medium for executing computer-executable instructions for communicating between a firewall process and a callout process in an operating system, comprising the steps of:

issuing, by the firewall process, a classify call having a plurality of parameters comprising, a protocol packet, at least one layer parameter, a packet context, and a matching filter identification;
receiving, by the callout process, the classify call; and
issuing, by the callout process, an action identified from the plurality of parameters in the classify call.

32. The computer-readable medium of claim 31, wherein the callout process is a logging module, further comprising storing in a memory the protocol packet.

33. The computer-readable medium of claim 31, wherein the callout process is an security module, further comprising:

determining, by the security module, that the protocol packet is required to conform to a security protocol;
verifying that the protocol packet conforms to the security protocol.

34. A functional interface for allowing a requesting layer to obtain policy for a packet, the requesting layer being one of a plurality layers, comprising:

a classify method comprising:

the packet received by the requesting layer;
a set of parameters associated with the packet, the set of parameters including data processed by the requesting layer,
a packet context received by the requesting layer from another layer of the plurality of layers; and
an action to be returned to the requesting layer identifying a first policy to be applied to the packet.

35. The functional interface of claim 34, further comprising:

an add layer method for allowing a new layer to be added to the plurality of layers.

36. The functional interface of claim 34, further comprising:

a delete layer method for removing an existing layer from the plurality of layers.

37. The functional interface of claim 34, further comprising:

a policy context defining a second policy to be applied to the packet.

38. The functional interface of claim 37, wherein the first policy is a firewall policy and the second policy is a non-firewall policy.

IPR2023-01353 Page 00917

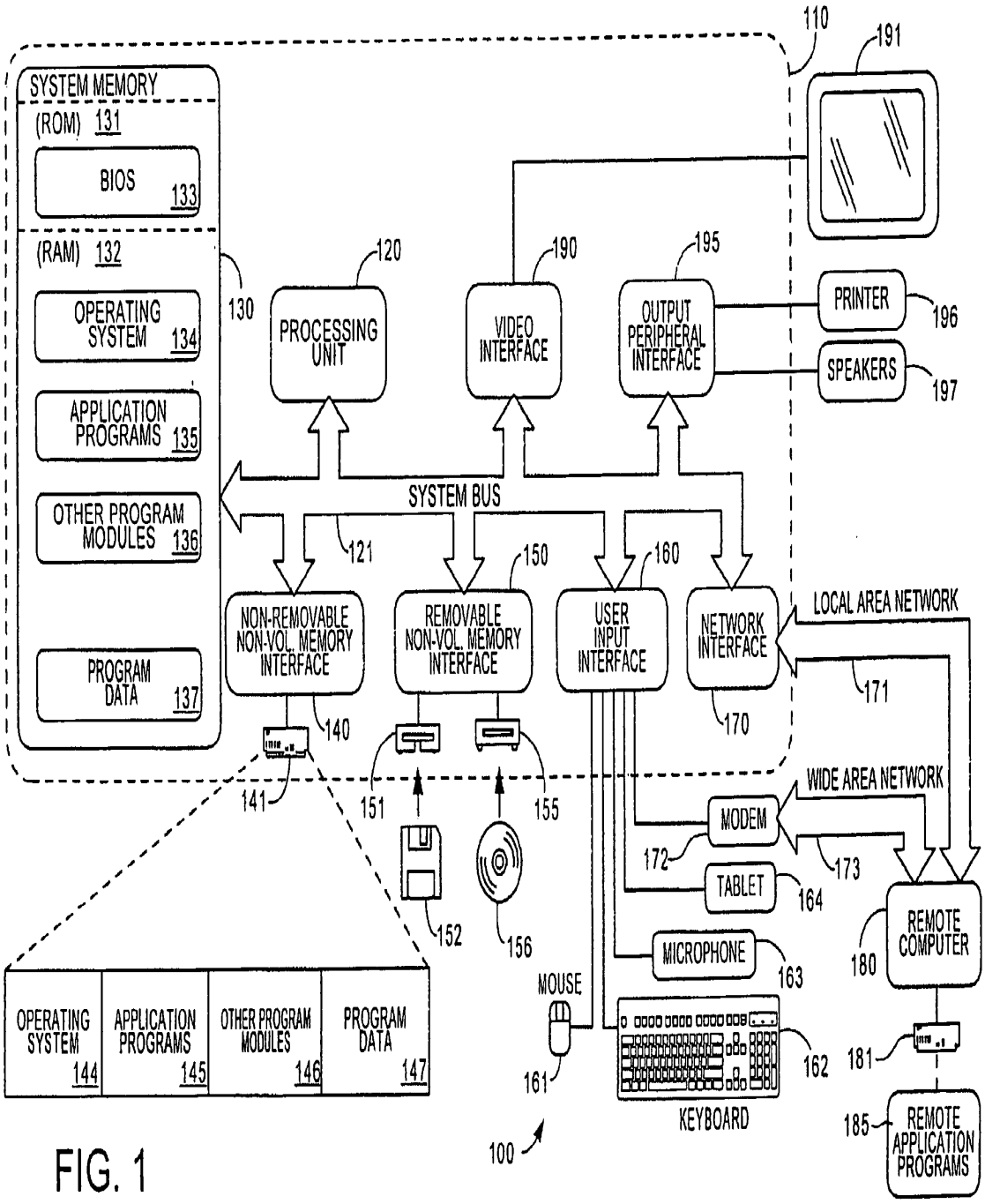


FIG. 1

EP 1 484 884 A2

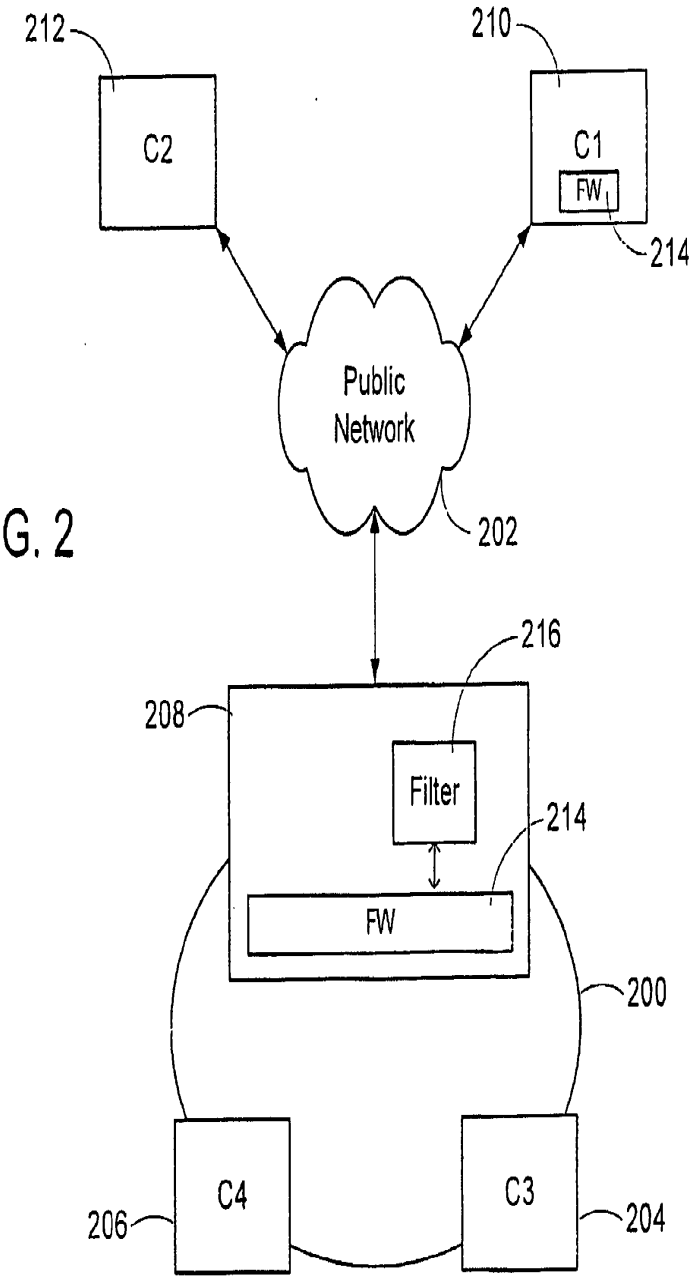

212
210
C2
C1
FW
214
Public
Network
202
216
208
Filter
214
FW
200
C4
C3
206
204

FIG. 2

FIG. 3

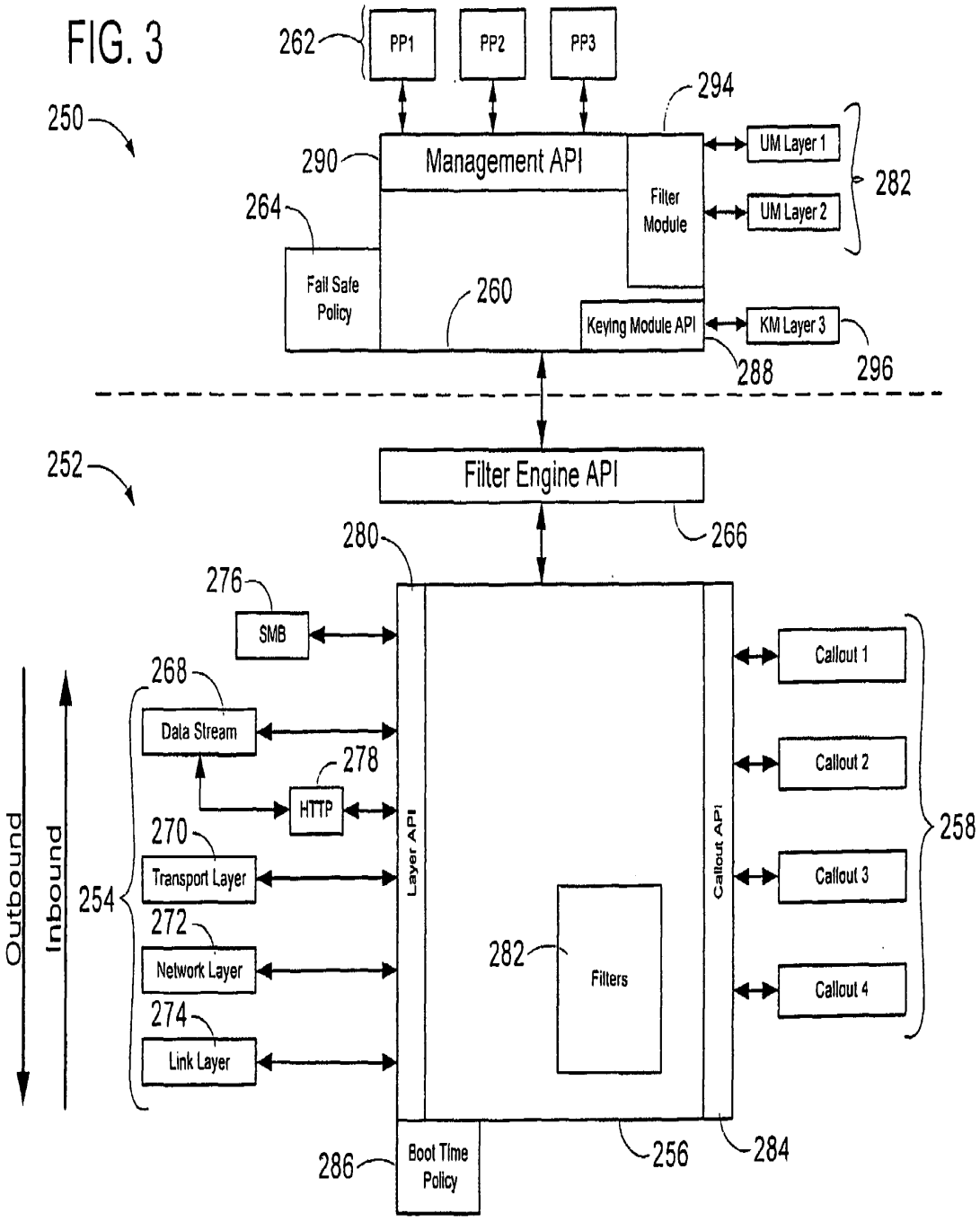

250
262
PP1
PP2
PP3
294
290
Management API
Filter Module
UM Layer 1
UM Layer 2
282
264
Fail Safe Policy
260
Keying Module API
KM Layer 3
288
296
252
Filter Engine API
266
280
276
SMB
268
Data Stream
278
HTTP
270
Transport Layer
272
Network Layer
274
Link Layer
254
Outbound
Inbound
Layer API
Callout API
Callout 1
Callout 2
Callout 3
Callout 4
258
282
Filters
286
Boot Time Policy
256
284

FIG. 4

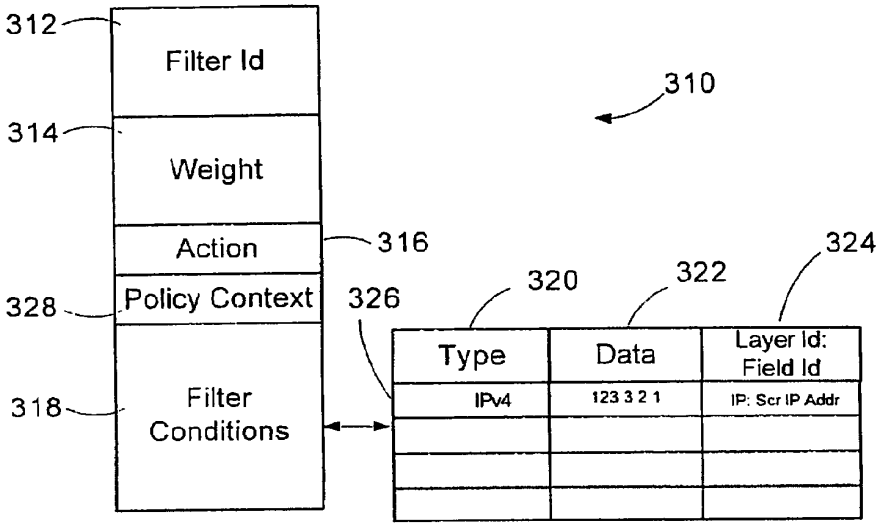
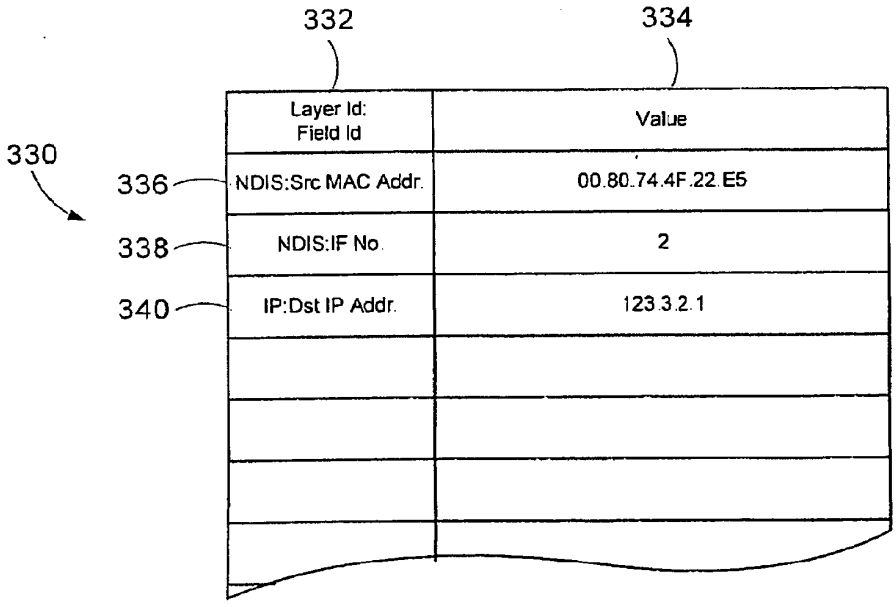


FIG. 5



IPR2023-01353 Page 00921

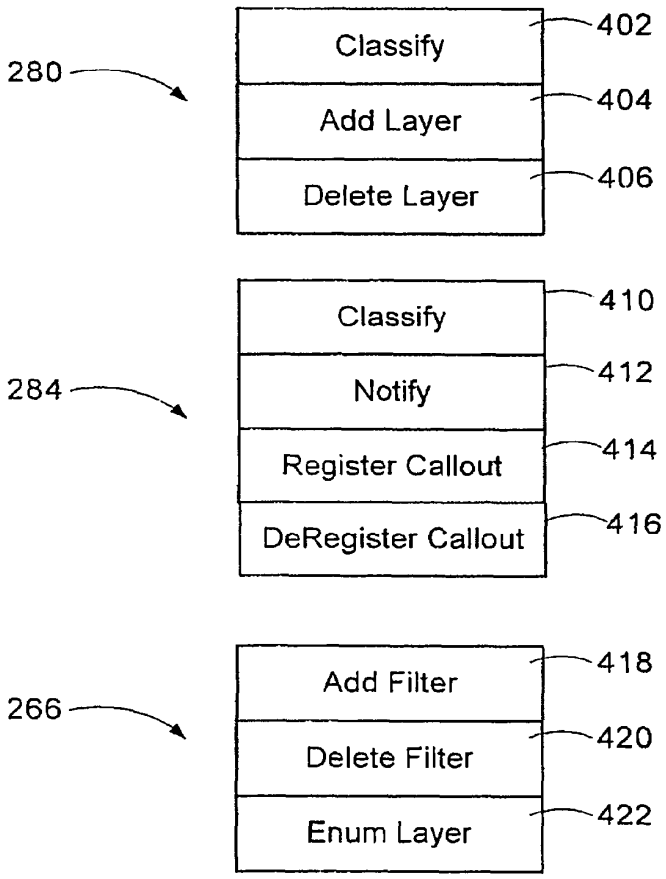


FIG. 6

IPR2023-01353 Page 00922

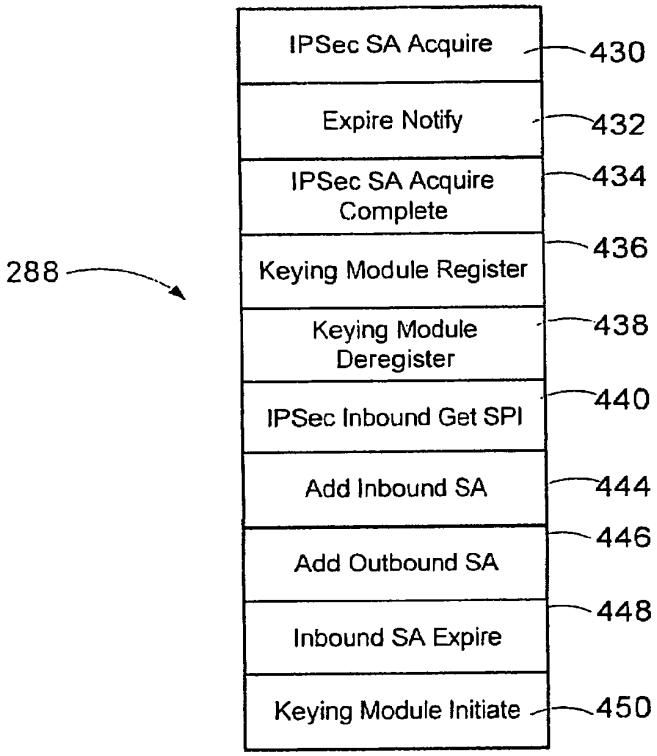
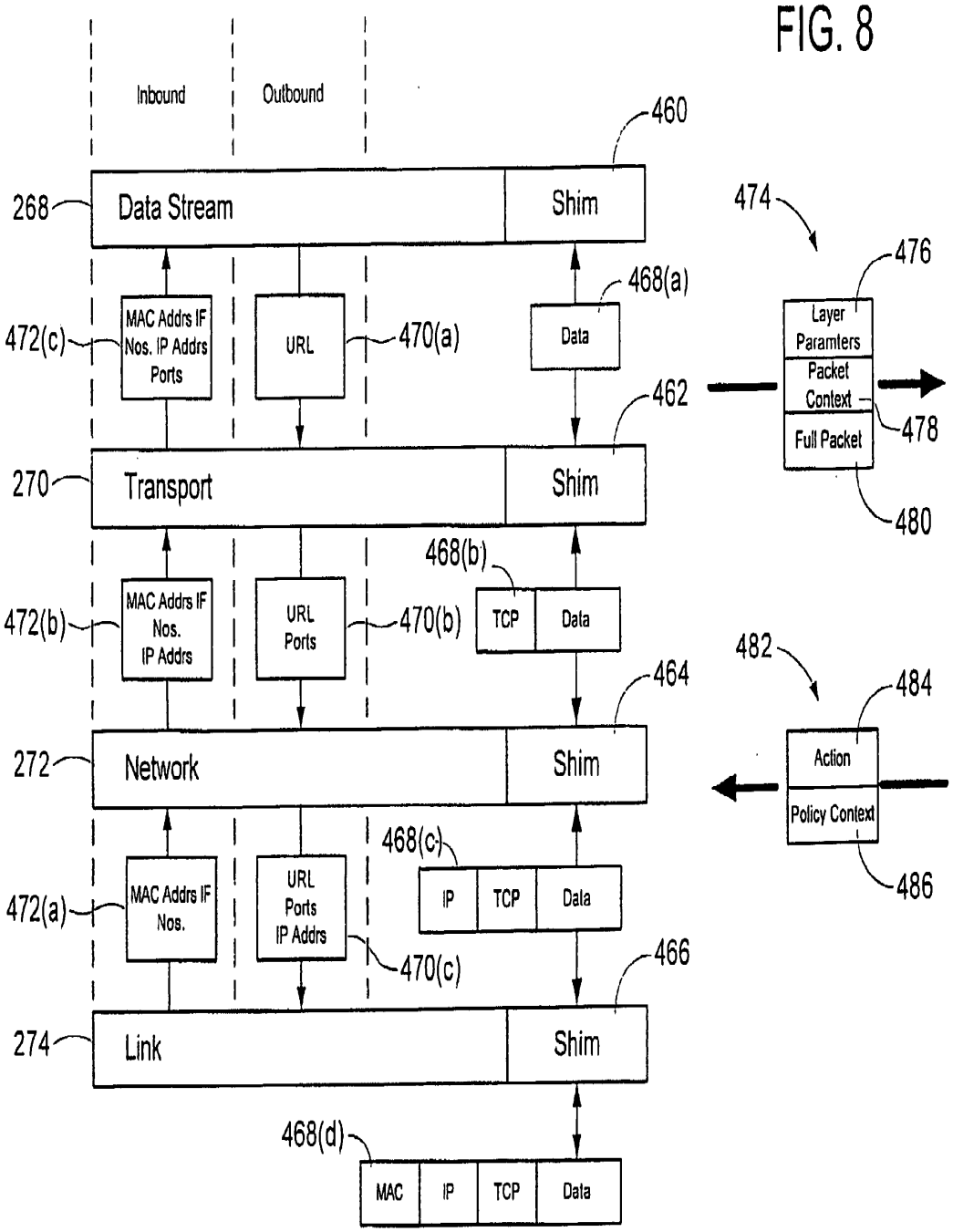


FIG. 7

IPR2023-01353 Page 00923

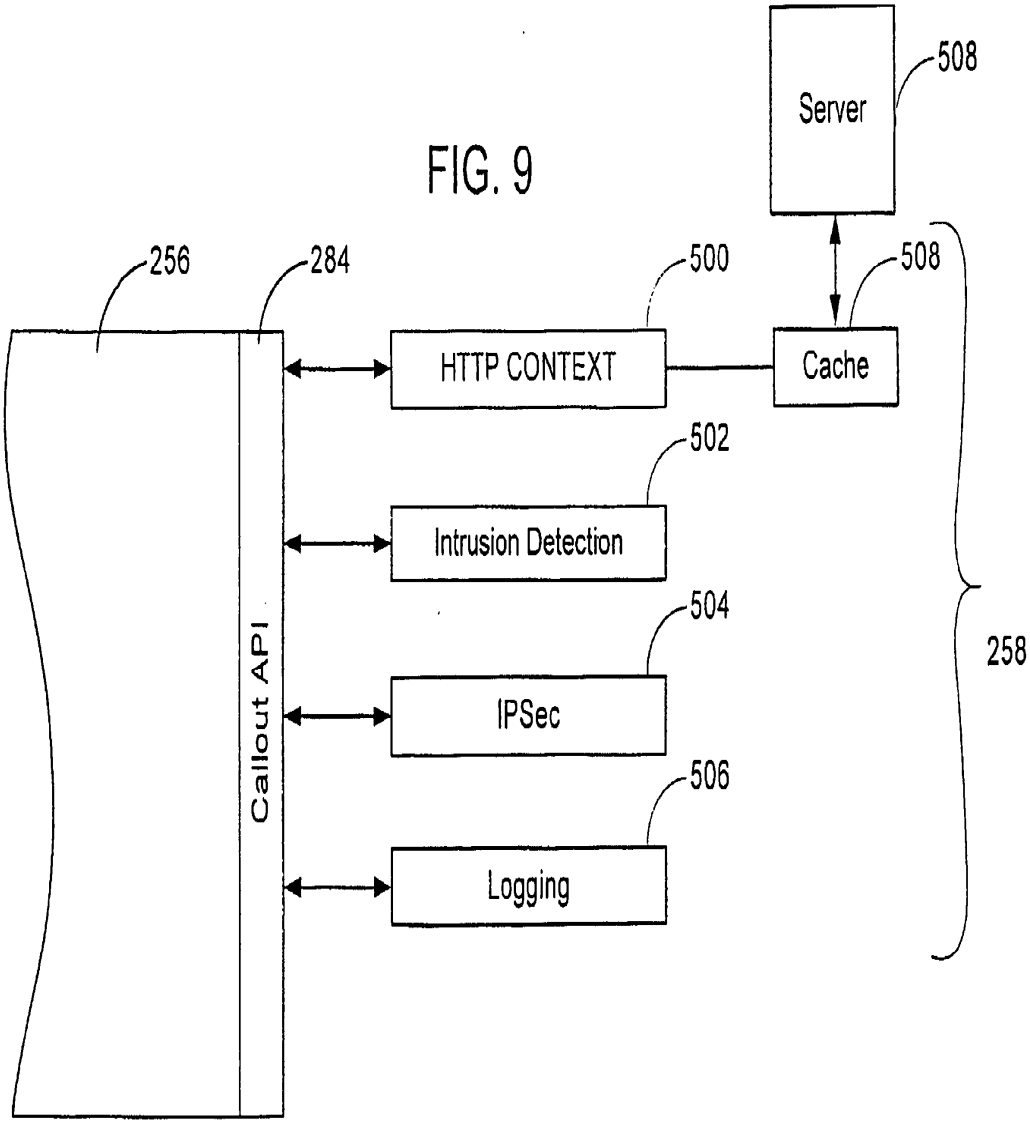
FIG. 8



30

EP 1 484 884 A2

FIG. 9


Server
508
256
284
500
508
HTTP CONTEXT
Cache
502
Intrusion Detection
504
IPSec
506
Logging
Callout API
258

Australian Government
IP Australia

LETTERS PATENT

STANDARD PATENT

2006230171

I, Robyn Foster, the Commissioner of Patents, grant a Standard Patent with the following particulars:

Name and Address of Patentee(s):

Wake Forest University
Office of Tachnology Asset Management, Wake Forest University Health Sciences, 200 East First Street, Suite 101, Winston Salem, North Carolina, 27101, United States of America

Name of Actual Inventor(s):

Fulp, Errin W. and Tarsa, Stephen J..

Title of Invention:

Methods, systems, and computer program products for network firewall policy optimization

Term of Letters Patent:

Twenty years from 28 March 2006

Priority Details :

Number	*Date*	*Filed with*
60/665,664	28 March 2005	US

COMMONWEALTH OF AUSTRALIA
AUSTRALIA
PATENT OFFICE

Dated this 4th day of October 2012

R E Foster

Robyn Foster
Commissioner of Patents

PATENTS ACT 1990

(12) STANDARD PATENT (11) Application No. AU 2006230171 B2
(19) AUSTRALIAN PATENT OFFICE

(54) Title
Methods, systems, and computer program products for network firewall policy optimization

(51) International Patent Classification(s)
H04L 9/00 (2006.01) ***G06F 15/18*** (2006.01)
G06F 9/00 (2006.01) ***G06F 17/00*** (2006.01)
G06F 15/16 (2006.01)

(21) Application No: **2006230171** (22) Date of Filing: **2006.03.28**

(87) WIPO No: **WO06/105093**

(30) Priority Data

(31) Number	(32) Date	(33) Country
60/665,664	**2005.03.28**	**US**

(43) Publication Date: **2006.10.05**
(44) Accepted Journal Date: **2012.06.21**

(71) Applicant(s)
Wake Forest University

(72) Inventor(s)
Tarsa, Stephen J.;Fulp, Errin W.

(74) Agent / Attorney
Griffith Hack, GPO Box 3125, Brisbane, QLD, 4001

(56) Related Art
WO 2006/093557 A2
US 2005/0125697 A1
EP 1006701 A2

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
5 October 2006 (05.10.2006)

PCT

(10) International Publication Number
WO 2006/105093 A2

(51) International Patent Classification:
G06F 15/16 (2006.01)

(21) International Application Number:
PCT/US2006/011291

(22) International Filing Date: 28 March 2006 (28.03.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/665,664 28 March 2005 (28.03.2005) US

(71) Applicant *(for all designated States except US)*: **WAKE FOREST UNIVERSITY** [US/US]; OFFICE OF TECHNOLOGY ASSET MANAGEMENT, WAKE FOREST UNIVERSITY HEALTH SCIECNES, 200 East First Street, Suite 101, Winston Salem, North Carolina 27101 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **FULP, Errin, W.** [US/US]; 308 N. Hawthorne Road, Winston-Salem, North Carolina 27104 (US). **TARSA, Stephen, J.** [US/US]; 135 King Phillips Path, Duxbury, Massachusetts 02332-3503 (US).

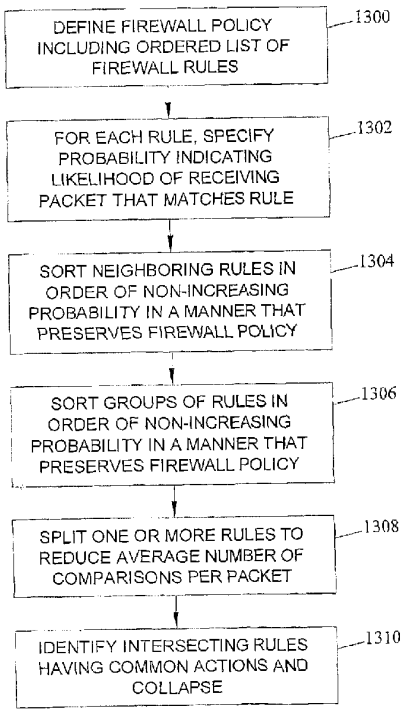
(74) Agent: **HUNT, Gregory, A.**; JENKINS, WILSON, TAYLOR & HUNT, P.A., Suite 1200 University Tower, 3100 Tower Boulevard, Durham, North Carolina 27707 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),

[Continued on next page]

(54) Title: METHODS, SYSTEMS, AND COMPUTER PROGRAM PRODUCTS FOR NETWORK FIREWALL POLICY OPTIMIZATION



(57) Abstract: Methods, systems, and computer program products for firewall policy optimization are disclosed. According to one method, a firewall policy including an ordered list of firewall rules is defined. For each rule, a probability indicating a likelihood of receiving a packet matching the rule is determined. The rules are sorted in order of non-increasing probability in a manner that preserves the firewall policy.

WO 2006/105093 A2

European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

2006230171 06 Jun 2012

TITLE

METHODS, SYSTEMS, AND COMPUTER PROGRAM PRODUCTS FOR NETWORK FIREWALL POLICY OPTIMIZATION

RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Patent Application Serial No. 60/665,664, filed March 28, 2005; the disclosure of which is incorporated herein by reference in its entirety.

GOVERNMENT INTEREST

This invention was made with Government support under Grant No. DE-FG02-03ER25581 awarded by U.S. Department of Energy, Mathematical and Computational Information Sciences Division. The U.S. Government has certain rights in the invention.

TECHNICAL FIELD

The subject matter described herein relates to optimizing polices for network firewalls. More particularly, the subject matter described herein relates to methods, systems, and computer program products for optimizing performance of network firewalls.

BACKGROUND ART

Network firewalls remain the forefront defense for most computer systems. Guided by a security policy, these devices provide access control, auditing, and traffic control [3, 30, 31]. A security policy is a list of ordered rules, as seen in Table 1, that defines the action to perform on matching packets. For example, an accept action passes the packet into or from the secure network, while deny causes the packet to be discarded. In many implementations, the rule set is stored internally as a linked list. A packet is sequentially compared to the rules, starting with the first, until a match is found; otherwise, a default action is performed [30, 31]. This is referred to as a first-match policy and is used in many firewall systems including the Linux firewall implementation iptables [25].

COMS ID No: ARCS-372276 Received by IP Australia: Time (H:m) 11:49 Date (Y-M-d) 2012-06-06

No.	Proto.	Source IP	Source Port	Destination IP	Destination Port	Action	Prob.
1	UDP	1.1.*	*	*	80	deny	0.01
2	TCP	2.*	*	1.*	90	accept	0.02
3	UDP	*	*	1.*	*	accept	0.10
4	TCP	2.*	*	1.*	20	accept	0.17
5	UDP	1.*	*	*	*	accept	0.20
6	*	*	*	*	*	deny	0.50

Table 1: Example Security Policy Consisting of Multiple Ordered Rules

Traditional firewall implementations consist of a single, dedicated machine, similar to a router, that sequentially applies rules to each arriving packet. However, packet filtering represents a significantly higher processing load than routing decisions [24, 29, 31]. For example, a firewall that interconnects two 100 Mbps networks would have to process over 300,000 packets per second [30]. Successfully handling these traffic loads becomes more difficult as rule sets become more complex [4, 22, 31]. Furthermore, firewalls must be capable of processing even more packets as interface speeds increase. In a high-speed environment (e.g. Gigabit Ethernet), a single firewall can easily become a bottleneck and is susceptible to DoS attacks [4, 9, 13, 14]. An attacker could simply inundate the firewall with traffic, delaying or preventing legitimate packets from being processed.

One approach to increase firewall performance focuses on improving hardware design. Current research is investigating different distributed firewall designs to reduce processing delay [4, 9, 22], and possibly provide service differentiation [11]. Another approach focuses on improving performance via better firewall software [6, 7, 12, 16, 17, 24]. Similar to approaches that address the longest matching prefix problem for packet classification [8, 10, 25, 28], solutions typically represent the firewall rule set in different fashions (e.g. tree structures) to improve performance. While both approaches, together or separate, show great promise, each requires radical changes to the firewall system, and therefore are not amenable to current or legacy systems.

IPR2023-01353 Page 00931

Accordingly, there exists a need for improved methods, systems, and computer program products for network firewall policy optimization.

SUMMARY

The subject matter described herein includes methods, systems, and computer program products for network firewall policy optimization. According to one method, a firewall policy including an ordered list of firewall rules is defined. For each rule, a probability indicating a likelihood of receiving a packet matching the rule is defined. Neighboring rules are sorted in order of non-increasing probability in a manner that preserves the firewall policy.

As used herein, the term "firewall" refers to a logical entity that is adapted to filter packets at the ingress and/or egress portions of a network based on a policy. The term "firewall" is intended to include systems that block packets from entering or leaving a network and systems that perform intrusion detection and intrusion protection functions for packets entering or leaving a network. Thus, the methods and systems described herein for firewall policy optimization can be used to optimize policies for intrusion detection and intrusion protection firewalls.

The subject matter described herein includes a method to improve firewall performance and lower packet delay that can be applied to both legacy and current systems. In one implementation, a firewall rule set may be re-ordered to minimize the average number of rule comparisons to determine the action, while maintaining the integrity of the original policy. Integrity is preserved if the reordered and original rules always arrive at the same result. To maintain integrity, a firewall rule set may be modeled as a Directed Acyclical Graph (DAG), where vertices are firewall rules and edges indicate precedence relationships. Given this representation, any linear arrangement of the DAG is proven to maintain the original policy integrity. Unfortunately, determining the optimal rule order from all the possible linear arrangements is proven to be *NP*-complete, since it is equivalent to sequencing jobs with precedence constraints for a single machine [15]. Although determining the optimal order is *NP*-complete, a heuristic will be described below to order firewall rules that reduces the average number of comparisons while maintaining integrity. Simulation

IPR2023-01353 Page 00932

results show the proposed reordering method yields rule orders that are comparable to optimal; thus, provides a simple means to significantly improve firewall performance and lower packet delay.

The subject matter described herein for network firewall policy optimization may be implemented using any combination of hardware, software, or firmware. For example, the subject matter described herein may be implemented using a computer program product comprising computer executable instructions embodied in a computer readable medium. Exemplary computer readable media suitable for implementing the subject matter described herein include disk memory devices, chip memory devices, programmable logic devices, application specific integrated circuits, and downloadable electrical signals. In addition, a computer program product that implements the subject matter described herein may be implemented using a single device or computing platform or may be distributed across multiple devices or computing platforms.

BRIEF DESCRIPTION OF THE DRAWINGS

Preferred embodiments of the subject matter described herein will now be explained with reference to the accompanying drawings of which:

Figure 1A is a block diagram of a rule list Directed Acyclical Graph (DAG) for the firewall rules in Table 1;

Figure 1B is a linear arrangement of firewall rules corresponding to the original rule order;

Figure 2A is a policy DAG representing a reordered version of the firewall rule set in Figure 1 after sorting;

Figure 2B is a policy DAG for the rule set illustrated in Table 1 illustrating an optimal rule order;

Figure 3A is a graph illustrating average numbers of packet comparisons versus intersection percentage for different orderings of a firewall policy;

Figure 3B is a graph illustrating the percent difference in average numbers of packet comparisons between sorted and optimal sortings of a firewall policy;

IPR2023-01353 Page 00933

Figure 4A is a graph of percent difference comparisons between original and sorted versions of a firewall policy;

Figure 4B is a graph illustrating the performance impact of considering rule intersection and actions in performing firewall rule sorting;

Figure 5A is a table illustrating a firewall policy before sorting;

Figure 5B is a table illustrating a sorted version of the firewall policy of Figure 5A;

Figure 6A is a policy trie representation of the firewall policy of Figure 5B;

Figure 6B is a table representing the firewall policy of Figure 5A after TCP and UDP sub-trie rotation;

Figure 7A is a policy trie illustrating the splitting of rule r_5 into rules r_{5a} and r_{5b};

Figure 7B is a table illustrating the firewall policy of Figure 5A after splitting rule r_5 into rules r_{5a} and r_{5b};

Figure 8A is a graph illustrating firewall performance with and without rule splitting;

Figure 8B is a graph illustrating locations and probabilities where splitting is better;

Figure 9A is a policy DAG of a firewall rule set;

Figure 9B is a policy DAG illustrating compression of rules r_1 and r_5;

Figure 10A is a policy DAG of a firewall rule list;

Figure 10B is an original policy trie before sorting;

Figure 10C is a policy trie illustrating the firewall rules after rotating sub-tries T_1 and T_2;

Figure 11A is a policy DAG illustrating a firewall policy;

Figure 11B is a policy trie illustrating firewall rules of the policy before sorting;

Figure 11C is a policy trie illustrating the firewall policy after exchanging nodes r_1 and r_2 and nodes r_7 and r_8;

Figure 11D is a policy trie illustrating the firewall policy after exchanging sub-tries T_3 and T_4;

Figure 11E is a policy trie illustrating the firewall policy after exchanging sub-tries T_1 and T_2;

IPR2023-01353 Page 00934

Figure 12A is a policy DAG illustrating an original order for a firewall policy;

Figure 12B is a policy DAG illustrating a sorted version of the firewall policy of Figure 12A after exchanging neighbors;

Figure 12C is a policy DAG illustrating a sorted version of the firewall policy after exchanging lists L_1 including rules r_1 and r_3 and lists L_2 including rules r_2 and r_4;

Figure 13 is a flow chart illustrating exemplary steps for network firewall policy optimization according to an embodiment of the subject matter described herein;

Figure 14 is a block diagram illustrating exemplary components of a system for firewall policy optimization according to an embodiment of the subject matter described herein; and

Figure 15 is a block diagram illustrating a plurality of firewall nodes for implementing a firewall policy according to an embodiment of the subject matter described herein.

DETAILED DESCRIPTION OF THE INVENTION

Modeling Firewall Security Policies

As described above, a firewall rule set, also known as a firewall policy, is traditionally an ordered list of firewall rules. Firewall policy models have been the subject of recent research [1, 2, 16]; however, the primary purpose is anomaly detection and policy verification. In contrast, the policy model described in this herein is designed for firewall performance optimization and integrity. Firewall performance refers to reducing the average number of comparisons required to determine an action, while integrity refers to maintaining the original policy intent. Although improving the worst-case performance is important, it is not possible without changing the list-based representation [16, 24].

Firewall Rule and Policy Models

In the examples described herein, a rule r is modeled as an ordered tuple of sets, $r = (r[1], r[2], ..., r[k])$. Order is necessary among the tuples since

IPR2023-01353 Page 00935

comparing rules and packets requires the comparison of corresponding tuples. Each tuple $r[l]$ is a set that can be fully specified, specify a range, or contain wildcards '*' in standard prefix format. For the Internet, security rules are commonly represented as a 5-tuple consisting of: protocol type, IP source address, source port number, IP destination address, and destination port number [30, 31]. Given this model, the ordered tuples can be supersets and subsets of each other, which forms the basis of precedence relationships. In addition to the prefixes, each filter rule has an action, which is to accept or deny. However, the action will not be considered when comparing packets and rules. Similar to a rule, a packet (IP datagram) d can be viewed as an ordered k-tuple $d = (d[1], d[2], ..., d[k])$; however, ranges and wildcards are not possible for any packet tuple.

Using the previous rule definition, a standard security policy can be modeled as an ordered set (list) of n rules, denoted as $R = \{r_1, r_2, ..., r_n\}$. A packet d is sequentially compared against each rule r_i starting with the first, until a match is found ($d \Rightarrow r_i$) then the associated action is performed. A match is found between a packet and rule when every tuple of the packet is a subset of the corresponding tuple in the rule.

Definition Packet d matches r_i if

$$d \Rightarrow r_i \quad \text{iff} \quad d[l] \subseteq r_i[l], \quad l = 1, ..., k$$

The rule list R is comprehensive if for every possible legal packet d a match is found using R. Furthermore, two rule lists R and R' are equivalent if for every possible legal packet d the same action is performed by the two rule lists. If R and R' are different (e.g. a reorder) and the lists are equivalent, then the policy integrity is maintained.

As previously mentioned, a rule list has an implied precedence relationship where certain rules must appear before others if the integrity of the policy is to be maintained. For example consider the rule list in Table 1. Rule r_1 must appear before rules r_3 and r_5, likewise rule r_6 must be the last rule in the policy. If for example, rule r_3 was moved to the beginning of the policy, then it will shadow [2] the original rule r_1. However, there is no precedence relationship between rules r_2 and r_4 given in Table 1. Therefore, the relative ordering of these two rules will not impact the policy integrity and can be

IPR2023-01353 Page 00936

changed to improve performance. The present example assumes that the original policy is free from any anomalies. Likewise, when a policy is reordered to improve performance it should not introduce any anomalies, which will occur if precedence relationships are not maintained. As a result, a model is needed to effectively represent precedence relationships.

Modeling Rule List Precedence Relationships

The precedence relationship between rules in a policy will be modeled as a Directed Acyclical Graph (DAG) [23, 18]. Such graphs have been successfully used to represent the relative order of individual tasks that must take place to complete a job (referred to as a task graph model). Since certain rules must appear before others to maintain policy integrity, this structure is well suited for modeling the precedence of firewall rules. Let $G = (R, E)$ be a rule list DAG for a rule list R, where vertices are rules and edges E are the precedence relationships (constraint). A precedence relationship, or edge, exists between rules r_i and r_j, if $i < j$, the actions for each rule are different, and the rules intersect.

Definition The intersection of rule r_i and r_j, denoted as $r_i \cap r_j$ is

$$r_i \cap r_j = (r_i[l] \cap r_j[l]), \qquad l = 1, \ldots, k$$

Therefore, the intersection of two rules results in an ordered set of tuples that collectively describes the packets that match both rules. The rules r_i and r_j intersect if every tuple of the resulting operation is non-empty. In contrast, the rules r_i and r_j do not intersect, denoted as $r_i \not\cap r_j$, if at least one tuple is the empty set. Note the intersection operation is symmetric; therefore, if r_i intersects r_j, then r_j will intersect r_i. The same is true for rules that do not intersect.

For example consider the rules given in Table 1, the intersection of r_1 and r_3 yields (UDP, 1.1.*, *, 1.*, 80). Again, the rule actions are not considered in the intersection or match operation. Since these two rules intersect, a packet can match both rules for example d = (UDP, 1.1.1.1, 80, 1.1.1.1, 80). Furthermore, the actions of the two rules are different. Therefore, the relative order must be maintained between these two rules and an edge drawn from r_1 to r_3 must be present in the rule list DAG, as seen in Figures 1A and 1B. More

IPR2023-01353 Page 00937

particularly, Figure 1A illustrates a rule list DAG for the rules in Table 1. Figure 1B illustrates a linear arrangement of the rules in Figure 1A. In Figures 1A and 1B, the vertices represent rules, the circles represents an accept rules, and the squares represents deny rules. Edges that connect the vertices represent precedence requirements. As can be seen from Figure 1B, because of the edge between rules r_1 and r_3, precedence between rules r_1 and r_3 must be maintained. In contrast consider the intersection of rules r_3 and r_5. These two rules intersect, indicating packets belonging to the set (UDP, 1.*, *, 1.*, *) would match both rules. However, it does not matter which of the two rules a packet matches first, since the action is the same for both rules. Therefore, an edge does not exist between rules r_3 and r_5 in the diagram. Similarly, rules r_2 and r_4 do not intersect due to the fifth tuple (destination port). A packet cannot match both rules indicating the relative order can change; therefore, an edge will not exist between them.

The match operation can be used to identify precedence relationships, but it cannot do so in every case. Consider a partial-match example [1], where r_a = (UDP, *, 80, 10.*, 90, accept) and r_b = (UDP, 10.*, 80, *, 90, deny). The intersection of r_a and r_b is (UDP, 10.*, 80, 10.*, 90); therefore a packet, such as d = (UDP, 10.10.10.10, 80, 10.10.10.10, 90), can match both rules. If r_a appears before r_b then the packet d is accepted, but if r_b occurs before r_a then d is rejected. As a result, the order of r_a and r_b in the original policy must be maintained. However, the match operation is unable to identify the precedence in this example. A partial match exists in between rules r_3 and r_5 in Table 1, but as previously discussed an edge does not exist between the rules since the actions are the same.

Using the rule list DAG representation a linear arrangement is sought that improves the firewall performance. As depicted in Figure 1B, a linear arrangement (permutation or topological sort) is a list of DAG vertices where all the successors of a vertex appear in sequence after that vertex [23]. Therefore it follows that a linear arrangement of a rule list DAG represents a rule order, if the vertices are read from left to right. Furthermore, it is proven in the following theorem that any linear arrangement of a rule list DAG maintains integrity.

IPR2023-01353 Page 00938

Theorem Any linear arrangement of a rule list DAG maintains integrity.

Proof Assume a rule list DAG *G* is constructed from the security policy *R* that is free of anomalies. Consider any two rules r_i and r_j in the policy, where $i < j$. If an edge between r_i and r_j in *G* does not exist, then a linear arrangement of *G* can interchange the order of the two rules. An edge will not exist if the rules do not intersect; however, a reorder will not affect integrity since a packet cannot match both rules. Shadowing is not introduced due to the reorder since the intersection operation is symmetric. An edge will not exist if the two rules intersect but have the same action; however, a reorder will not affect integrity since the same action will occur regardless of which rule is matched first. If an edge does exist between the rules, then their relative order will be maintained in every linear arrangement of *G*; thus maintaining precedence and integrity.

Rule List Optimization

As described above, it is important to inspect packets as quickly as possible given increasing network speeds and QoS requirements. Using the rule list DAG to maintain policy integrity, a linear arrangement is sought that minimizes the average number of comparisons required. However, this will require information not present in the firewall rule list. Certain firewall rules have a higher probability of matching a packet than others. As a result, it is possible to develop a policy profile over time that indicates frequency of rule matches (similar to cache hit ratio). Let $P = \{p_1, p_2, ..., p_n\}$ be the policy profile, where p_i is the probability that a packet will match rule *i* (first match in the policy). Furthermore, assume a packet will always find a match, $\sum_{i=1}^{n} p_i = 1$; therefore *R* is comprehensive. Using this information, the average number of rule comparisons required is

$$E[n] = \sum_{i=1}^{n} i \cdot p_i \qquad (1)$$

For example, the average number of comparisons required for the rule set in Table 1 is 5.03.

IPR2023-01353 Page 00939

Given a rule list DAG $G = (R, E)$ and policy profile $P = \{p_1, p_2, ..., p_n\}$ a linear arrangement π of G is sought that minimizes Equation 1. In the absence of precedence relationships, the average number of comparisons is minimized if the rules are sorted in non-increasing order according to the probabilities [26], which is also referred to as Smith's algorithm [27]. Precedence constraints cause the problem to be more realistic; however, such constraints also make determining the optimal permutation more problematic.

Determining the optimal rule list permutation can be viewed as job scheduling for a single machine with precedence constraints [15, 21]. The notation for such scheduling problems is $\alpha|\beta|\gamma|\delta$, where α is the number of machines, β is the precedence (or absence of) which can be represented as a DAG, γ is a restriction on processing time, and δ is the optimality criterion [15]. Determining the optimal rule order is similar to the $1|\beta|1|\sum \omega C_i$ scheduling problem, or optimality criterion, where ω is a weight associated with a job (for example, importance) and C_i is the completion time. As previously noted, the $1|||\sum \omega C_i$ problem can be solved in linear time the using Smith's algorithm [27], which orders jobs according to non-decreasing $\frac{t_i}{\omega}$ ratio, where t_i is the processing time of job i. In this case set $t_i = 1$ and $\omega = p_i \ \forall i$. However, Lawler [19] and Lenstra et al. [21] proved $1|\beta|1|\sum \omega C_i$ to be *NP*-complete via the linear arrangement problem, which implies determining the optimal firewall rule order is also *NP*-complete. Note, determining the number of possible permutations has been proven to be *NP*-hard [5].

Theorem $1|\beta|1|\sum \omega C_i$ α *Determining the optimal order of a firewall rule list*

Proof Consider the $1|\beta|1|\sum \omega C_i$ problem. Each of n jobs J_i, $i \in I$, has to be processed without preemption on a single machine that can handle at most one job at a time. For each $i \in I$, let ω be the associated weight. Furthermore, let $G = (V, E)$ be a DAG that represents the precedence order of the jobs J_i. Assume the processing time of each job equals 1 time unit, the weights to be $0 \leq \omega \leq 1$ such that $\sum \omega = 1$, and β, which is G, to be a rule list DAG. In this case, the

IPR2023-01353 Page 00940

optimization criterion $\sum \omega \cdot C_i$ is the same as $\sum p_i \cdot i$, which is given in equation 1. Clearly, the optimal firewall rule ordering problem has a solution if and only if $1|\beta|1|\sum \omega C_i$ has a solution. Therefore, determining the optimal permutation of firewall rules is *NP*-complete.

Exemplary Rule Sorting Algorithm

Although determining the optimal rule permutation was proven to be *NP*-complete, reducing the average number of comparisons required to process a packet remains an important objective. As previously discussed, a sorting algorithm must maintain the precedence relationships among rules. Of course an exhaustive search is possible if the number of rules is small (generate and test all possible topological orders); however as proven in the previous section, this is not feasible with a realistic rule list.

One exemplary algorithm starts with the original the rule set, then sorts neighboring rules based on nonincreasing probabilities. However, an exchange of neighbors should never occur if the rules intersect and have different actions. This test preserves any precedence relationships in the policy. For example, the following sorting algorithm uses such a comparison to determine if neighboring rules should be exchanged. Note a_i denotes the action associated with the ith rule.

```
done = false
while(!done)
    done = true
    for(i = 1; i <n; i++)
        if(pi < pi+1 AND (ri ∩ ri+1 OR ai == ai+1))then
            exchange rules, actions, and probabilities
            done = false
        endif
    endfor
endwhile
```

Other sorting algorithms are possible if the completion time of the sort is an issue; however, the method presented is easy to implement and only requires a simple neighbor comparison. Assume the match probabilities for the rule list given in Table 1. Applying the sorting algorithm to this rule list results in the ordering depicted in Figure 2A, which has 10% fewer comparisons on

average. However when using the algorithm, it is possible that one rule can prevent another rule from being reordered. For example, rule r_1 prevents rule r_5 from being placed closer to the beginning of the rule set. However, if both rules r_1 and r_5 are placed closer to the beginning of the policy while maintaining their relative order, the average number of comparison will be further reduced, 15% fewer. This is the optimal order for the 6 rule set, which is depicted in Figure 2B. Although this simple sorting algorithm is unable to move groups of rules, it can still improve the performance of the firewall system. Its effectiveness is measured experimentally in the next section.

Experimental Results

In this section, the average number of rule comparisons required after sorting the rules is compared with average number of comparisons required using the original order and the optimal order. Note the optimal rule ordering was determined via an exhaustive search. As a result, it was not feasible to determine the optimal ordering once the number of rules equaled 15 given the number of permutations to consider. Two different variations of the rule sorting method described in the preceding section were implemented. The difference between the methods was the comparison used to determine if neighboring rules should be exchanged. The first sorting algorithm exchanged neighboring rules if they were out of order and did not intersect. The if-condition was as follows.

if ($p_i < p_{i+1}$ **AND** $r_i \not\cap r_{i+1}$) **then**

Therefore, rule actions were not considered and the method will be referred to as *non-action sort*. The second sorting method did consider the rule action (as described in the preceding section) and will be referred to as *action sort*. The comparison between the two sorting algorithms will indicate the importance of considering rule actions when ordering rules.

In the first experiment, lists of 10 firewall rules were generated with random precedence relationships. The match probability of each rule was given by a Zipf distribution [20], which assigns probabilities according to the rank of an item. For this simulation, the last rule had the highest probability

IPR2023-01353 Page 00942

which is consistent with most policies (last rules are more general). As a result, the original order yields the worst average number of comparisons. The intersection percentage measured the percentage of rules that intersect in the policy. This metric gives a value for the dependency level in the policy; however note, rule action is not considered when calculating this metric. Rules were equally likely to have an accept or deny action and the results (average number of rule comparisons) for a particular intersection percentage were averaged over 1000 simulations.

Results of the first experiment are given in Figures 3A and 3B. More particularly, Figure 3A is a graph illustrating the average number of packet comparisons versus intersection percentage for the original rule sorting, the non-action sort, the action sort, and the optimal sort. Figure 3B is a graph illustrating the percent difference in the average number of comparisons required between the sorted and optimal firewall policy configurations. The average number of comparisons required was lower for the sorted and the optimal lists when the intersection percentage was low, as seen in Figure 3A. This was expected since there is a large number of possible rule permutations (few edges in the DAG). When the intersection percentage approached 100%, the values converged to the number required for the original list. This is due to the limited number of possible rule orders, only one order in the extreme case. The percent difference between the sorted and optimal order is shown in Figure 3B. At zero intersection percentage the sorted and optimal orders are equal, since any ordering is possible. Similarly at an intersection percentage of 100, the sorted and optimal orders are equal since only one order is possible. Between these two extremes, the action sorting algorithm remains close to the optimal value. In contrast, the non-action sort had a maximum difference of 33%.

The benefit of rule sorting on larger policies is also of interest; however as previously described, it was not possible to determine the optimal ordering once the number of rules approaches 15. Therefore, the second experiment used larger rule sets, but only compared the original order and the sorted orders. The number of rules ranged from 10 to 1000, while the matching probabilities and intersection percentages were the same as the previous

IPR2023-01353 Page 00943

experiment. The results of this experiment are depicted in Figure 4A. The sorted rule sets always performed equal to or better than the original order, while the action sort consistently performed better than the non-action sort. As noted in the previous experiment, the percent difference is very large given few intersections (e.g. 80% decrease for 1000 rules with 0% dependency), but approaches zero as all the rules intersect. As the number of rules increases, sorting is increasingly beneficial, although only at low intersection percentages. The benefit of sorting drops more significantly as the intersection percentage increases with larger rule sets. This is primarily due to the low matching probabilities of each rule, which requires a complete reordering to have a significant impact on the average number of comparisons. As a result, large rule sets can benefit from sorting if the intersection percentage is low.

The impact of considering rule actions when sorting is illustrated in Figure 4B. In this experiment lists of 1000 rules were generated, where the intersection percentages ranged from 0% to 100% and the percentage of rules with the same action varied from 50% to 100%. The performance of the action sort to the original list was then compared. As the percentage of rules with the same action increased, the percent difference (reduction) in the average number of comparisons also increased. This is due to the increased number of permutations possible when the rule actions are increasingly the same (fewer edges in the rule list DAG to consider). This is depicted in Figure 4B where a policy with a 100% intersection percentage can significantly reduce the number of comparisons (80% reduction) if all the rules have the same action. This performance increase is not possible with the non-action sort. Therefore, considering the rule action increases the number of possible rule orders, thereby providing more possibilities to improve firewall performance.

Conclusions

Network firewalls enforce a security policy by comparing arriving packets to a list of rules. An action, such as accept or deny, is then performed on the packet based on the matching rule. Unfortunately packet filtering can impose significant delays on traffic due to the complexity and size of rule sets.

IPR2023-01353 Page 00944

Therefore, improving firewall performance is important, given network Quality of Service (QoS) requirements and increasing network traffic loads.

The sections above describe rule ordering methods to improve the performance of network firewalls. Assuming each rule has a probability of a packet matching, firewall rules should be sorted such that the matching probabilities are non-increasing. This reduces the average number of comparisons and the delay across the firewall. However, a simple sort is not possible given precedence relations across rules. It is common in a security policy that two rules may match the same packet yet have different actions. It is this precedence relationship between rules that must be maintained to preserve integrity. The method described above uses Directed Acyclical Graphs (DAG) to represent the precedence order rules must maintain. Given this representation, a topological sort can be used to determine the optimal order (minimum average number of comparisons); however, the examples above prove this problem to be *NP*-complete (similar to job scheduling for a single nonpreemptive machine with precedence constraints). As an alternative, a simple sorting method was introduced that maintained the precedence order of the rules. Simulation results indicate this method can significantly reduce the average number of comparisons required and is comparable to optimal ordering.

Several areas exist for future research in optimizing firewall rule lists. The sorting method proposed above is based on a simple algorithm. Although it can offer an improvement over the original rule order, algorithms that can move groups of rules could provide a larger reduction in the average number of comparisons. The effect of stateful firewalls should also be addressed in future research. Security can be enhanced with connection state and packet audit information. For example, a table can be used to record the state of each connection, which is useful for preventing certain types of attacks (e.g., TCP SYN flood) [30, 31]. The impact of such rules on the firewall needs to be investigated and whether sorting can be done on-line to reflect temporal changes. In addition, more research is needed to determine more accurate probability distributions for packet matching and dependency percentages.

IPR2023-01353 Page 00945

Given this information, better algorithms can be designed and more realistic simulations can be performed.

Trie-Based Network Firewall Optimization Methods and Rule Splitting Overview

As described above, while assuring firewall policy integrity is critical, performance is equally important given increasing network speeds and traffic volumes. Firewall processing delay can be reduced via hardware as well as policy optimization, where rules are manipulated to reduce the number of comparisons. For a given firewall installation, it can be determined that certain security rules have a higher probability of matching a packet than others. As a result, it is possible to develop a *policy profile* over time that indicates frequency of rule matches (similar to cache hit ratio). Given this information, trie-based methods are described herein to minimize the average number of comparisons required per packet. Although most firewall systems still utilize an ordered list representation, the proposed enhancements still are applicable to current firewall systems since a policy trie can be converted into an ordered list using an inorder traversal.

Rule Sorting

As described above, rule sorting of neighboring rules based on ascending probabilities in a manner that considers rules intersection and rule actions can be used to improve firewall performance by reducing the number of packet comparisons. The methods described above use Directed Acyclic Goal Graphs (DAGs) to represent firewalls. The methods described in this section will use the rule sorting algorithms in combination with policy trie representations to perform sorting of groups of rules in a manner that improves firewall performance.

In this example, it is assumed that a security policy contains a list of n ordered rules, $\{r_1, r_2, ..., r_n\}$. In addition, let p_i represent the probability that a packet will match rule i (first match in the policy). Therefore, the policy is comprehensive, which means every packet will have a match, if $\sum_{i=1}^{n} p_i = 1$.

IPR2023-01353 Page 00946

Using this information, the average number of rule comparisons required can be calculated as

$$E[n] = \sum_{i=1}^{n} i \cdot p_i$$

The average number of comparisons is minimized if the rules are sorted in non-ascending order according to the probabilities. However, this can only be achieved if the sorting algorithm considers the probabilities and whether one rule is a superset of another. For example, sorting algorithms must use the following comparison to determine if neighboring rules should be interchanged Note, a_i denotes the action associated with the ith rule.

if ($p_i < p_{i+1}$ **AND** $r_i \cap r_{i+1}$ **OR** $a_i == a_{i+1}$)) **then**

Sorting rules in this fashion can have positive impact on the average number of comparisons required. Figure 5A is a table illustrating a firewall policy prior to rule sorting. The expected number of comparisons for the policy of Figure 5A is 4.26 comparisons per packet. Figure 5B is a table illustrating the firewall policy of Figure 5A after sorting using the above-described algorithm. In Figure 5B, the expected number of comparisons per packet has been reduced to 3.94. Thus, using the method described above, the number of comparisons per packet can be reduced by 8%.

However, it can be seen from Figures 5A and 5B that one rule can prevent another rule from being reordered. For example in Figure 5B, rule r_1 prevents rule r_4 from being placed closer to the beginning of the rule set. However, if both rules r_1 and r_4 are placed closer to the beginning of the policy while maintaining their relative order, the average number of comparison will be reduced.

To solve this problem, rule sets can be sorted a using policy tries. First, the rule list is converted into an equivalent policy trie. Each sub-trie will have an associated probability $\overline{p}$ that is average probability (hit-ratio) of the rules comprising the sub-trie. Sub-tries can be rotated around their parent node to increase performance, using the method described earlier. Since the policy trie (or equivalent rule set) is tested from left to right, rotation should occur if the

IPR2023-01353 Page 00947

probability of the right sub-trie is greater than the left sub-trie. As a result, the average number of comparisons required will be reduced and the policy integrity is maintained. Figure 6A is a trie representation of the rules in Figure 5B. Figure 6B is a table representing resulting rule set after rotating the TCP and UDP sub-tries. The TCP sub-trie in Figure 6A has an expected packet matching probability of 0.095 while the UDP sub-trie has an expected packet matching probability of 0.155. Thus, the UDP sub-trie and the TCP sub-trie should be rotated so that the UDP sub-trie becomes the left sub-trie and the TCP sub-trie becomes the right sub-trie. In the table illustrated in Figure 6B, which lists the ordering of the rules after rotating the sub-tries, the average number of comparisons per packets has been reduced to 3.70.

Rule Splitting

Rule splitting takes a general rule and creates more specific rules that collectively perform the same action over the same set of packets. Here, rule splitting is used to reduce the average number of comparisons. For example in Figure 7A, rule r_5 is split into two separate rules, r_{5a} for UDP and r_{5b} for TCP. Figure 7B is a table corresponding to the policy trie of Figure 7A. Once the rules are positioned based on their probabilities and their relation to other rules, the average number of rule comparisons is reduced to 2.98 (after sort, trie rotation, and splitting) which is a 30% less.

It many not be advantageous to split a general rule since it adds another rule to the policy. For example, assume a policy contains 20 rules where the first 19 rules have the same probability. Assume the last rule can be split and the new specific rule has a probability that is $\frac{3}{4}$ of the last rule. The impact of the probability of the last rule and the location of the new split rule, m, is depicted in Figures 8A and 8B. More particularly, Figure 8A is a graph illustrating the average number of comparisons for firewalls with and without rule splitting. Figure 8B is a graph illustrating average number of comparisons versus different locations of the new rule and probabilities. The average number of comparisons is reduced as the split rule is located closer to the first rule (surface decreases as m approaches one). Furthermore, splitting yields

IPR2023-01353 Page 00948

better results when the general rule has a high probability. However as illustrated in Figure 8B, the closer the specific rule is to the location of the original rule the average number of comparisons increases, which is the penalty of adding one more rule to the policy.

The effect of splitting a single rule can be described mathematically as follows. Consider n rules where rule r_n can be split into rules $r_{n,1}$ and $r_{n,r}$ (whose union is the original rule). In addition, assume the split rule $r_{n,1}$ will be located at the mth position ($1 \leq m < n$), while rule $r_{n,r}$ will remain at the nth location. The goal is to determine the best location m, which yields a lower average number of comparisons as compared to the original rule set. This can be defined mathematically in the following formula.

$$\sum_{i=1}^{n} i \cdot p_i > \sum_{i=1}^{m-1} i \cdot p_i + m \cdot p_{n,l} + \sum_{i=m}^{n-1} (i+1) \cdot p_i + (n+1) \cdot p_{n,r}$$

The left side of the inequality is the average number of comparisons for the original rule set. The right hand side of the inequality is the average number of comparisons with the specific rule at location m. If it is assumed that the rules located between m and n have an equal probability (denoted as p) the previous equation can be solved for m.

$$m < \frac{n \cdot p - n \cdot p_n + (n+1) \cdot p_{n,r}}{p - p_{n,l}}$$

The new rule must be located between the first and m^{th}; however, its final location will depend on the relationship with the other rules (cannot be placed before any rule for which it is a superset). This result can be applied iteratively to multiple rules and repeatedly to the same rule.

Rule Optimization for QoS

In the preceding sections, optimizing the entire security policy has been discussed. However, it may be desirable to optimize the rules for a certain type of traffic, in order to reduce the average number of comparisons this traffic encounters. This can be done by organizing the policy trie via traffic

IPR2023-01353 Page 00949

classification, then optimizing the sub-tries. The result is an ordering where the rules for the most important traffic are tested first.

Network Firewall Optimization Using Rule Compression

Another method for network firewall optimization is referred to as rule compression. Given a security policy R, one objective in optimizing firewall performance is to reduce number of rules in the policy while maintaining the original integrity. A method for reducing the number of rules called *rule compression*, which combines and replaces two rules in the policy will now be described. Although compression does reduce the number of rules, it can only be done if the integrity of the original policy is maintained.

Consider two rules r_i and r_j in policy R, where r_i appears before r_j. The two rules can be compressed if and only if the following conditions are true

1. The two rules intersect, $r_i \cap r_j \neq \emptyset$ (an edge between r_i and r_j exists in the policy DAG) (Tuples that are proper subsets and supersets are considered. Compression may still occur if the rules do not intersect or are adjacent in the set space, the resulting rule may contain ranges.)
2. The action of r_i and r_j is the same.
3. Rule r_i is not dependent on any rule r_k that is directly or indirectly dependent on r_j (only one path from r_i to r_j exists in the policy DAG).

The result of compression is a new rule, $r_{i,j}$ whose tuples are the union of the corresponding tuples of rules r_i and r_j. The new rule $r_{i,j}$ replaces both r_i and r_j in R; however, the location of $r_{i,j}$ in R may require the relocation of other rules. Let D_j be the ordered set of rules that appear after r_i that directly or indirectly depend upon r_j. The new rule is placed at the original location of r_i in R and the rules D_j are placed before $r_{i,j}$.

For example, consider the policy given in Table 2. It is possible to compress rules r_1 and r_5, creating the new rule $r_{1,5}$ =[TCP, *, *, 2.*, *, accept]. The ordered set D_j in this example consists of the rules r_3 and r_4. As a result, the new rule $r_{1,5}$ is placed at the original location of r_1 while the rules in D_j are placed before the new rule. Figures 9A and 9B respectively illustrate the

IPR2023-01353 Page 00950

original policy DAG for Table 2 and the policy DAG after rules r_1 and r_5 are compressed.

No.	Proto.	Source IP	Source Port	Destination IP	Destination Port	Action
1	TCP	1.1.*	*	2.2.*	*	accept
2	TCP	1.*	*	*	80	deny
3	TCP	3.3.*	*	*	80	accept
4	TCP	3.*	*	*	*	deny
5	TCP	*	*	2.*	90	accept

Table 2: Example Policy Consisting of Multiple Ordered Rules

Theorem *Compressing rules r_i and r_j of policy R will maintain integrity if the three conditions are met and method described is used.*

Proof Before r_i and r_j are compressed, relocate the rules D_j immediately before r_i and D_i. This relocation will maintain integrity due to the third requirement for compression (no rule in D_i can be dependent on any rule in D_j). Now place r_j directly after r_i. This relocation will not affect integrity since no rule in D_i can have an edge to r_j (again, due to the third condition). Compressing the neighboring rules r_i and r_j creates $r_{i,j}$. This does not affect integrity since the resulting rule only matches the packets that match r_i or r_j in the original policy. Therefore, the result is the compression of rule r_i and r_j, and the integrity of the policy is maintained.

Determining if Compression Should Occur

Although compression may be possible, it may not improve the performance of the policy due to the relocation of rules. Compression should only occur if the average number of comparisons required for the new policy R_c is less than the original policy R; therefore, $E[R_c] < E[R]$.

IPR2023-01353 Page 00951

Firewall Rule Sub-Trie and Sub-List Ordering

Policy Sub-Trie Ordering

Additional methods for firewall optimization are referred to as firewall sub-trie and sub-list ordering. Given a policy trie T, it may be desirable to order all sub-tries such that the average number of tuple comparisons is reduced and policy integrity is maintained. Consider a policy trie T that contains sub-tries T_i and T_j having the same parent node, as seen in Figures 10A-10C. Let P_i denote the sum of the probabilities of the rules contained in sub-trie i, while C_i denotes the number of comparisons required to completely traverse sub-trie i (which is equal to the number of branches). In order to reduce the average number of tuple comparisons, sub-tries that share the same parent node should be ordered such that the following two conditions are observed.

1. Sub-tries that share a parent node are ordered such that the P_i values are non-ascending (higher match probabilities occur first, from left to right).
2. If sub-tries that share a parent node have the same probability (P_i equals P_j), then order the sub-tries such that the C_i values are non-descending (sub-tries consisting of fewer comparisons occur first, from left to right).

These conditions are recursively applied throughout the policy trie as seen in Figures 11A-11D. However, these conditions do not necessarily maintain policy integrity.

Consider a group of n sub-tries that share the same parent node that are numbered sequentially from left to right. Consider any two non-intersecting sub-tries T_i and T_{i+k} in this group that are out of order. If two sub-tries do not intersect, denoted as $T_i \cap T_j$, then no rule in one sub-trie will intersect with any rule in the other sub-trie, $r_k \cap r_l$, $\forall r_k \in T_i$, $\forall r_l \in T_j$. The following *integrity condition* must be observed when reordering the sub-tries. The two sub-tries can be exchanged (rotated about the parent) if and only the sub-tries T_i through T_{i+k-1} do not intersect with T_{i+k} and the sub-tries T_{i+1} through T_{i+k} do not intersect with T_i. Similar to finding the linear sequence of a policy DAGs, these conditions maintain the policy trie integrity. This is observed in Figure 10A,

IPR2023-01353 Page 00952

where no edge exists between any rule in T_1 and T_2. More particularly, Figure 10A is a policy DAG of a six-rule firewall list. Figure 10B is a policy trie representation of the firewall rule list represented by Figure 10A. In Figure 10B, the average number of tuple compares required per packet is 5.5. Figure 10C illustrates the policy trie after rotating sub-tries T_1 and T_2. In Figure 10C, the average number of tuple compares required is 4.5. In Figures 10A-10C, the following probabilities are assumed:

$P_1 = 0.5$

$C_1 = 4$

$P_2 = 0.5$; and

$C_2 = 2$.

The average number of tuple comparisons is an estimate based on the longest trie path.

Using the conditions for comparing and ordering, the following sorting algorithm sorts neighboring sub-tries that share the same parent node.

```
m = a parent node in T
done = false
while(!done)
    done = true
    for each sub-trie Ti having parent m AND a right neighboring sub-trie
        if((Pi < Pi+1 AND (Pi == Pi+1 AND Ci > Ci+1)) AND Ti ⇏ Ti+1) then
        interchange Ti and Ti+1
        done = false
        endif
    endfor
endwhile
```

Figures 11A-11D illustrate policy trie rotation for an eight-rule firewall policy. More particularly, Figure 11A illustrates the policy DAG representation of an eight-rule firewall policy. Figure 11B is a policy trie representation of the original firewall policy. In this example, ordering occurs in three stages, starting at the bottom level and moving towards the top level of the trie. In Figure 11C, the leaves are ordered at the lowest level. More particularly, nodes r_1 and r_2 are exchanged with nodes r_7 and r_8. The average number of tuple compares for the ordering represented by Figure 11C is 10.45. In Figure 11D, the sub-

IPR2023-01353 Page 00953

tries on the second lowest level (T_3, T_4, T_5, and T_6) are ordered. More particularly, Figure 11D illustrates the policy trie after exchanging sub-tries T_3 and T_4. The average number of tuple compares for the representation in Figure 11D is 10.275. In Figure 11E, the sub-tries on the second highest level, (T_1 and T_2) are ordered. Figure 11E illustrates the final ordering after exchanging T_1 and T_2. In the policy trie of Figure 11E, the average number of tuple compares required is 6.775. The average number of tuple comparisons is an estimate based on the longest trie path.

Policy Sub-List Ordering

The preceding section describes conditions for sorting policy sub-tries, which allow the exchange of groups of rules (sub-tries). The same conditions can be applied to list based policies, where sub-lists are ordered to improve the average number of rule comparisons.

In the sections above relating to rule sorting, a method is described to exchange neighboring rules in a list-based security policy, as seen in Figure 12B. However, it was observed that when ordering rules in this fashion it is possible that one rule can block the exchange of another. If groups of rules were allowed to be exchanged the average number of rule comparisons could be further reduced.

Given a policy list L, it may be desirable to order all sub-lists such that the average number of rule comparisons is reduced and policy integrity is maintained. Consider a policy list L that contains sub-tries L_i and L_j. Let P_i denote the sum of the probabilities of the rules contained in sub-list i, while C_i denotes the number of rules in sub-list i. In order to reduce the average number of rule comparisons, sub-lists should be ordered such that the following two conditions are observed.

1. Sub-lists are ordered such that the P_i values are non-ascending (higher match probabilities occur first, from left to right).
2. If sub-lists have the same probability (P_i equals P_j), then order the sub-lists such that the C_i values are non-descending (sub-lists consisting of fewer comparisons occur first, from left to right).

IPR2023-01353 Page 00954

These conditions are applied throughout the policy list; however as with policy tries, these conditions do not necessarily maintain policy integrity.

Consider a group of n sub-lists that are numbered sequentially from left to right. Consider any two non-intersecting sub-lists L_i and L_{i+k} in this group that are out of order. If two sub-lists do not intersect, denoted as $L_i \pitchfork L_j$, then no rule in one sub-list will intersect with any rule in the other sub-list, $r_k \pitchfork r_l$, $\forall r_k \in L_i$, $\forall r_l \in L_j$. The following *integrity condition* must be observed when reordering the sub-lists. The two sub-lists can be exchanged if and only if the sub-lists L_i through L_{i+k-1} do not intersect with any rule in L_{i+k} and sub-lists L_{i+1} through L_{i+k} do not intersect with L_i. Similar to finding the linear sequence of a policy DAGs, these conditions maintain the policy trie integrity. For example, consider the sub-lists $L_1 = \{r_1, r_3\}$ and $L_2 = \{r_2, r_4\}$ given in Figure 12B. No edge exists between any rule in L_1 and L_2. The two sub-lists are out of order with respect to P and should be exchanged. The resulting list is given in Figure 12C. Figure 12A illustrates the original policy order and results in an average number of rule comparisons of 3.4. In the sorted version in Figure 12B, the average number of rule comparisons is 3.3. In Figure 12C, the average number of rule comparisons has been reduced to 2.7. Using the conditions for comparing and ordering, the following sorting algorithm sorts neighboring sub-lists.

```
done = false
while(!done)
    done = true
    for each sub-list Li having a right neighboring sub-list
        if ((Pi < Pi+1 OR (Pi == Pi+1 AND Ci > Ci+1)) AND Li ⋔ Li+1)then
            interchange rules and probabilities
            done = false
        endif
    endfor
endwhile
```


IPR2023-01353 Page 00955

Summary of Firewall Policy Optimization Techniques and Methods

As stated above, the methods and systems described herein for a network firewall policy optimization can be implemented using any combination of hardware, software, and/or firmware. Alternatively, the steps for firewall policy optimization can be performed manually with the number of rules being optimized is small enough. Figure 13 is a flow chart illustrating an exemplary process for firewall policy optimization according to an embodiment of the subject matter described herein. Referring to Figure 13, in block **1300**, a firewall policy including an ordered list of firewall rules is defined. In block **1302**, for each rule, a probability indicating a likelihood of receiving a packet that matches each rule is specified. In block **1304**, neighboring rules are sorted in order of non-increasing probability in a manner that preserves firewall policy. In block **1306**, groups of rules are sorted in order of non-increasing probability in a manner that preserves firewall policy. For example, any of the policy-trie-based methods described above may be used. In block **1308**, one or more rules are split to reduce the number of average packet comparisons. In block **1310**, intersecting rules having common actions are identified and collapsed into single rules.

As described above, the subject matter described herein may be implemented in hardware, software, and/or firmware. In one implementation, the subject matter described herein for firewall policy optimization may be implemented on a general purpose computing platform. Figure 14 is a block diagram of a general purpose computing platform including software for firewall policy optimization according to an embodiment of the subject matter described herein. Referring to Figure 14, computing platform **1400** may be a general purpose computer, such as a personal computer, that includes a microprocessor **1402**, memory **1404**, and I/O interfaces **1406**. Microprocessor **1400** may be any suitable microprocessor, such as any of the Intel or AMD families of microprocessors. Memory **1404** may include volatile memory for running programs and persistent memory, such as one or more disk storage devices. I/O interface **1406** may include interfaces with I/O devices, such as user input devices and output devices.

IPR2023-01353 Page 00956

In the illustrated example, software that may be resident in memory **1404** includes a firewall policy editor **1408** and a firewall policy optimizer **1410**. Firewall policy editor **1408** may allow a user to define a firewall policy. For example, firewall policy editor **1408** may allow a user to define a firewall policy by specifying different rules. The rules may be specified in a graphical manner, for example using policy DAGs as described above. Alternatively, firewall policy editor **1408** may allow a user to input rules in a tabular manner, as illustrated in any of the tables described herein. Firewall policy optimizer **1410** may implement any or all of the firewall policy optimization techniques to order rules entered via firewall policy editor **1408** in a manner that preserves policy integrity and that enhances firewall performance.

A firewall rule set that is optimized using the subject matter described herein may be implemented on any firewall system that includes one or more firewalls. For example, an optimized firewall rule set according to embodiments of the subject matter described herein may be implemented using any of the hierarchical, multi-node firewall systems described in commonly assigned, co-pending U.S. patent application no. 11/316,331, filed December 22, 2005, the disclosure of which is incorporated herein by reference in its entirety. Figure 15 illustrates an example of a multi-node firewall system suitable for implementing the firewall rules according to an embodiment of the subject matter described herein. Referring to Figure 15, a plurality of firewall nodes **1500**, **1502**, **1504**,and **1506** may collectively implement any of the optimized firewall policies described herein. For example, the firewall nodes may collectively implement different portions of a firewall policy data structure including an ordered list of firewall rules. A firewall policy engine **1508** resident on each firewall node may filter packets using its local portion **1510** of the firewall policy data structure. A control node **1512** may include a firewall policy optimizer **1514** that measures the average number of comparisons per packets for the current rule configuration and may dynamically reorder the rules to reduce the average number of comparisons per packet. For example, firewall policy optimizer **1514** may utilize any of the methods described herein to rearrange rules and improve firewall performance. Rules may be tested for rearrangement at user-specified

intervals or when average number of packet comparisons increases by a user-specified amount.

References

The disclosure of each of the following references is hereby incorporated herein by reference in its entirety.

[1] E. Al-Shaer and H. Hamed. Firewall Policy Management Advisor for Anomaly Detection and Rule Editing. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*, 2003.

[2] E. Al-Shaer and H. Hamed. Modeling and Management of Firewall Policies. *IEEE Transactions on Network and Service Management*, 1(1), 2004.

[3] S. M. Bellovin and W. Cheswick. Network Firewalls. *IEEE Communications Magazine*, pages 50–57, Sept. 1994.

[4] C. Benecke. A Parallel Packet Screen for High Speed Networks. In *Proceedings of the 15th Annual Computer Security Applications Conference*, 1999.

[5] G. Brightwell and P. Winkler. Counting Linear Extensions is #P-Complete. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, 1991.

[6] M. Christiansen and E. Fleury. Using Interval Decision Diagrams for Packet Filtering. Technical report, BRICS, 2002.

[7] D. Decasper, Z. Dittia, G. Parulkar, and B. Plattner. Router Plugins: A Software Architecture for Next-Generation Routers. *IEEE/ACM Transactions on Networking*, 8(1), February 2000.

[8] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink. Small Forwarding Tables for Fast Routing Lookups. In *Proceedings of ACM SIGCOMM*, pages 4 – 13, 1997.

[9] U. Ellermann and C. Benecke. Firewalls for ATM Networks. In Proceedings of *INFOSEC'COM*, 1998.

[10] A. Feldmann and S. Muthukrishnan. Tradeoffs for Packet Classification. In Proceedings of the *IEEE INFOCOM*, pages 397 – 413, 2000.

IPR2023-01353 Page 00958

[11] E. W. Fulp. Firewall Architectures for High Speed Networks. Technical Report 20026, Wake Forest University Computer Science Department, 2002.

[12] E. W. Fulp. Firewall Policy Models Using Ordered-Sets. Technical report, Wake Forest University Computer Science Department, 2004.

[13] R. Funke, A. Grote, and H.-U. Heiss. Performance Evaluation of Firewalls in Gigabit-Networks. In *Proceedings of the Symposium on Performance Evaluation of Computer and Telecommunication Systems*, 1999.

[14] S. Goddard, R. Kieckhafer, and Y. Zhang. An Unavailability Analysis of Firewall Sandwich Configurations. In *Proceedings of the 6th IEEE Symposium on High Assurance Systems Engineering*, 2001.

[15] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan. Optimizing and Approximation in Determinstic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics*, 5:287 – 326, 1979.

[16] A. Hari, S. Suri, and G. Parulkar. Detecting and Resolving Packet Filter Conflicts. In *Proceedings of IEEE INFOCOM*, pages 1203–1212, 2000.

[17] HiPAC. High Performance Packet Classification. http://www.hipac.org.

[18] E. Horowitz, S. Sahni, and D. Mehta. *Fundamentals of Data Structures in C++*. Compuer Science Press, 1995.

[19] E. L. Lawler. Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints. *Annals of Discrete Mathematics*, 2:75 – 90, 1978.

[20] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the Self-Similar Nature of Ethernet Traffic. *IEEE Transactions on Networking*, 2:1 – 15, 1994.

[21] J. K. Lenstra and A. H. G. R. Kan. Complexity of Scheduling under Precedence Constraints. *Operations Research*, 26(1):22 – 35, 1978.

[22] O. Paul and M. Laurent. A Full Bandwidth ATM Firewall. In Proceedings of the 6th European Symposium on *Research in Computer Security ESORICS'2000*, 2000.

IPR2023-01353 Page 00959

[23] B. R. Preiss. *Data Structures and Algorithms with Object-Oriented Design Patterns in C++*. John Wiley & Sons, 1999.

[24] L. Qui, G. Varghese, and S. Suri. Fast Firewall Implementations for Software and Hardware-Based Routers. In *Proceedings of ACM SIGMETRICS*, June 2001.

[25] V. P. Ranganath and D. Andresen. A Set-Based Approach to Packet Classification. In *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, pages 889–894, 2003.

[26] R. Rivest. On Self-Organizing Sequential Search Heuristics. *Communications of the ACM*, 19(2), 1976.

[27] W. E. Smith. Various Optimizers for Single-Stage Production. *Naval Research Logistics Quarterly*, 3:59 – 66, 1956.

[28] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel. Fast and Scalable Layer Four Switching. In *Proceedings of ACM SIGCOMM*, pages 191–202, 1998.

[29] S. Suri and G. Varghese. Packet Filtering in High Speed Networks. In *Proceedings of the Symposium on Discrete Algorithms*, pages 969 – 970, 1999.

[30] R. L. Ziegler. *Linux Firewalls*. New Riders, second edition, 2002.

[31] E. D. Zwicky, S. Cooper, and D. B. Chapman. *Building Internet Firewalls*. O'Reilly, 2000.

It is to be understood that, if any prior art publication is referred to herein, such reference does not constitute an admission that the publication forms a part of the common general knowledge in the art, in Australia or any other country.

In the claims which follow and in the preceding description of the invention, except where the context requires otherwise due to express language or necessary implication, the word "comprise" or variations such as "comprises" or "comprising" is used in an inclusive sense, i.e. to specify the presence of the stated features but not to preclude the presence or addition of further features in various embodiments of the invention.

It will be understood that various details of the invention may be changed without departing from the scope of the invention. Furthermore, the foregoing description is for the purpose of illustration only, and not for the purpose of limitation.

5

2006230171 06 Jun 2012

CLAIMS

The claims defining the invention are as follows:

1. A method for producing a performance optimized firewall policy, the method comprising:
   (a) defining a firewall policy including an ordered list of firewall rules;
   (b) for each rule, defining a first match probability indicating the likelihood that the rule will be the first match for a given packet;
   (c) sorting neighboring rules in order of non-increasing probability for the first match probabilities and in a manner that preserves the firewall policy; and
   (d) sorting groups of two or more rules in order of non-increasing probability in a manner that preserves the firewall policy, wherein at least one of the groups of two or more rules includes a first rule that prevents a second rule in the group from being moved, within the group, closer to a beginning of the ordered list that implements the firewall policy and wherein sorting the groups of rules includes moving the first and second rules closer to the beginning of the ordered list while maintaining their relative position.

2. The method of claim 1 wherein sorting neighboring rules in order of non-increasing probability in a manner that preserves the firewall policy includes rearranging the rules if a first rule has a lower probability than a second rule and the first rule is not a subset of the second rule with a different action.

3. The method of claim 1 comprising sorting groups of two or more rules in order of non-increasing probability in a manner that preserves the firewall policy.

COMS ID No: ARCS-372276 Received by IP Australia: Time (H:m) 11:49 Date (Y-M-d) 2012-06-06

2006230171 06 Jun 2012

4. The method of claim 3 wherein sorting groups of two or more rules in a manner that preserves firewall policy includes assigning each group of rules to a sub-trie in a policy trie, computing a sum of packet matching probabilities for rules in each sub-trie, and, in response to determining that a first sub-trie has a lower sum than a second sub-trie, rotating the first and second sub-tries about a common parent so that rules in the second sub-trie will be applied before the rules in the first sub-trie in the firewall policy.

5. The method of claim 4 comprising, in response to determining that the first and second sub-tries have equal sums, determining whether the second sub-trie has an lower number of branches than the first sub-trie, and, in response to determining that the second sub-trie has a lower number of branches than the first sub-trie, rotating the first and second sub-tries about a common parent so that the rules in the second sub-trie will be applied before the rules in the first sub-trie in the firewall policy.

6. The method of claim 3 wherein sorting groups of two or more rules in a manner that preserves the firewall policy includes:
   (a) identifying first and second sub-lists of rules in the ordered list of firewall rules, the first sub-list preceding the second sub-list in the ordered list of firewall rules;
   (b) determining a sum of packet matching probabilities for each of the first and second sub-lists;
   (c) determining whether the first and second sub-lists intersect; and
   (d) in response to determining that the sum of the packet matching probabilities for the second sub-list is greater than the sum of the packet matching probabilities for the first sub-list and that the first and second sub-lists do not intersect, switching the order of the first and second sub-lists in the ordered list of firewall rules.

7. The method of claim 6 comprising, in response to determining that the sums of the packet matching probabilities for the first and sub-lists are

COMS ID No: ARCS-372276 Received by IP Australia: Time (H:m) 11:49 Date (Y-M-d) 2012-06-06

2006230171 06 Jun 2012

equal, determining whether the second sub-list has a lower number of rules than the first sub-list, and, in response to determining that the second sub-list has a lower number of rules than the first sub-list, switching the order of the first and second sub-lists in the ordered list of firewall rules.

8. The method of claim 1 comprising splitting at least one of the firewall rules to reduce an average number of comparisons required per received packet.

9. The method of claim 1 comprising identifying intersecting rules having a common action and collapsing the rules into a single rule representing a union of the intersecting rules.

10. The method of claim 1 wherein sorting neighboring rules includes sorting rules such that comparisons for a particular class of packets are reduced.

11. The method of claim 1 comprising assigning the rules to vertices in a directed acyclical graph (DAG), determining precedence relationships between the rules, assigning the precedence relationships to edges that connect the vertices in the DAG, and wherein sorting the neighboring rules in order of non-increasing probability in a manner that preserves the firewall policy includes sorting the neighboring rules in a manner that preserves the precedence relationships represented by the DAG.

12. The method of claim 1 wherein the rules comprise intrusion detection rules.

13. The method of claim 1 wherein the rules comprise intrusion protection rules.

COMS ID No: ARCS-372276 Received by IP Australia: Time (H:m) 11:49 Date (Y-M-d) 2012-06-06

2006230171 06 Jun 2012

14. A system for firewall policy optimization, for implementing the steps of any one of claims 1-13.

15. A network firewall comprising:

(a) a firewall policy data structure for storing an ordered list of firewall rules;

(b) a firewall policy engine for implementing a firewall policy by comparing packets to the rules in the order specified by the firewall policy data structure; and

(c) a firewall policy optimizer for optimizing performance of the network firewall by reordering the rules in a manner that reduces an average number of rule comparisons per packet and that preserves the firewall policy, wherein the firewall policy optimizer is further configured to sort groups of two or more rules in order of non-increasing probability in a manner that preserves the firewall policy, wherein at least one of the groups of two or more rules includes a first rule that prevents a second rule in the group from being moved, within the group, closer to a beginning of the ordered list that implements the firewall policy and wherein sorting the groups of rules includes moving the first and second rules closer to the beginning of the ordered list while maintaining their relative position.

16. The network firewall of claim 15 wherein the firewall policy optimizer is adapted to measure average number of comparisons per packet for a rule ordering specified by the firewall policy data structure and to dynamically re-order to rules to reduce the average number of comparisons per packet.

17. A computer program product comprising computer executable instructions embodied in a computer readable medium which, when executed on one or more computing devices or platforms, performs steps in accordance with any one of claims 1 - 13.

COMS ID No: ARCS-372276 Received by IP Australia: Time (H:m) 11:49 Date (Y-M-d) 2012-06-06

2006230171 06 Jun 2012

18. A method for producing a performance optimized firewall policy, the method substantially as herein described with reference to the accompanying drawings.

19. A network firewall substantially as herein described with reference to the accompanying drawings.

COMS ID No: ARCS-372276 Received by IP Australia: Time (H:m) 11:49 Date (Y-M-d) 2012-06-06

IPR2023-01353 Page 00966

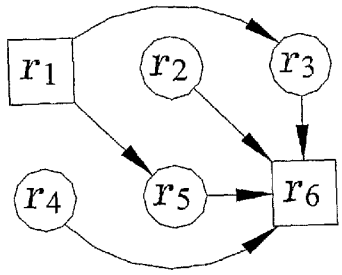


FIG. 1A

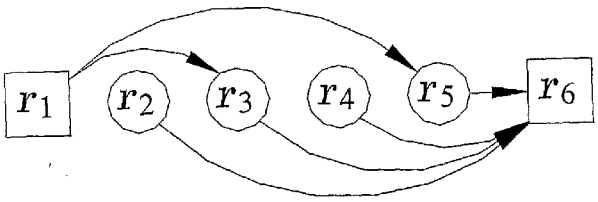


FIG. 1B

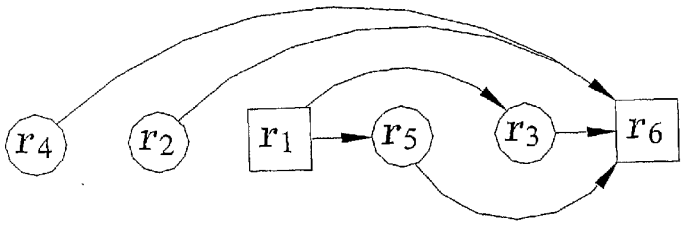


FIG. 2A

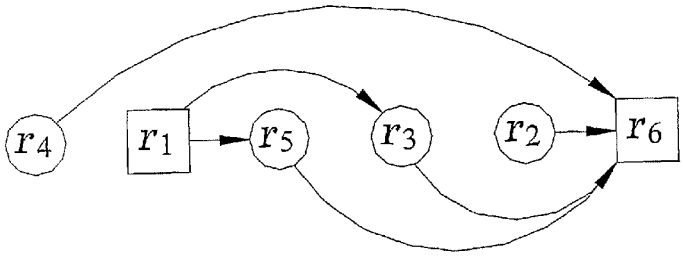


FIG. 2B

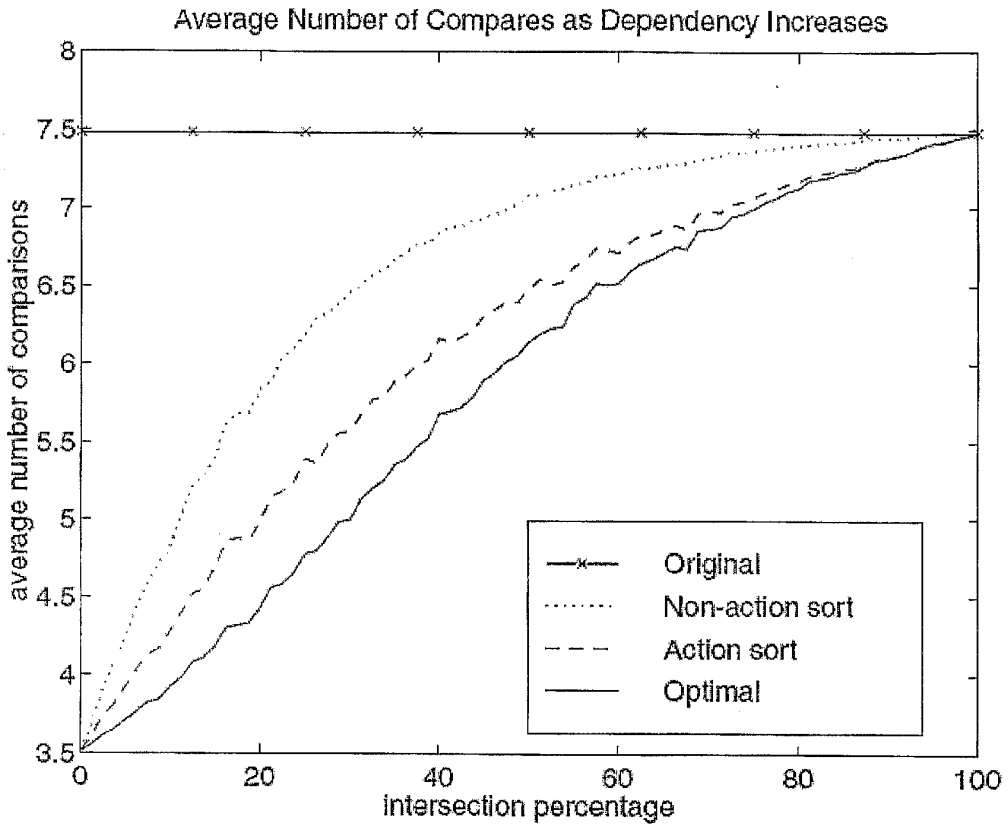

Average Number of Compares as Dependency Increases
average number of comparisons
intersection percentage
8
7.5
7
6.5
6
5.5
5
4.5
4
3.5
0
20
40
60
80
100
Original
Non-action sort
Action sort
Optimal

FIG. 3A

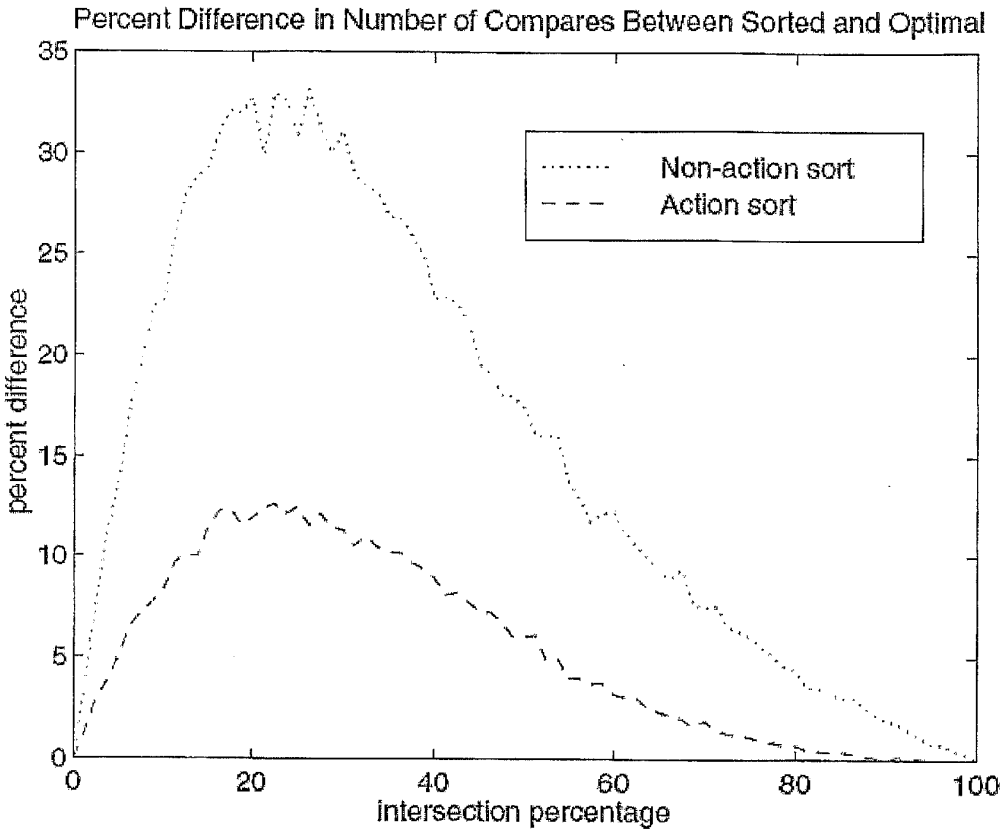

Percent Difference in Number of Compares Between Sorted and Optimal
percent difference
intersection percentage
Non-action sort
Action sort
0
5
10
15
20
25
30
35
0
20
40
60
80
100

FIG. 3B

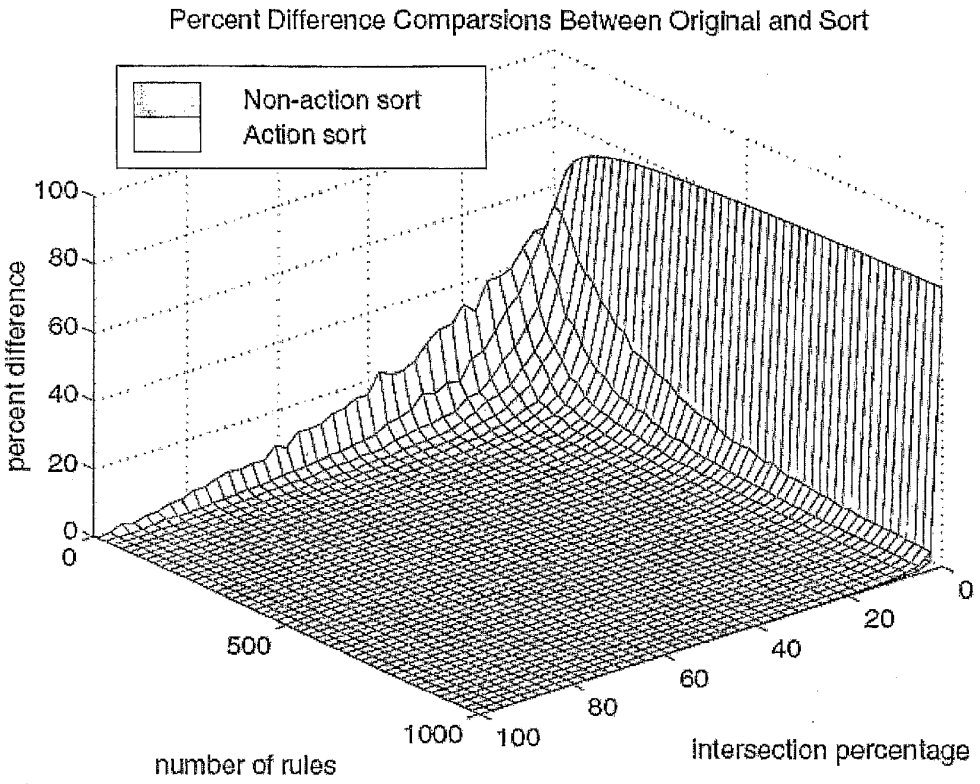

Percent Difference Comparsions Between Original and Sort
Non-action sort
Action sort
percent difference
100
80
60
40
20
0
0
500
1000
number of rules
0
20
40
60
80
100
intersection percentage

FIG. 4A

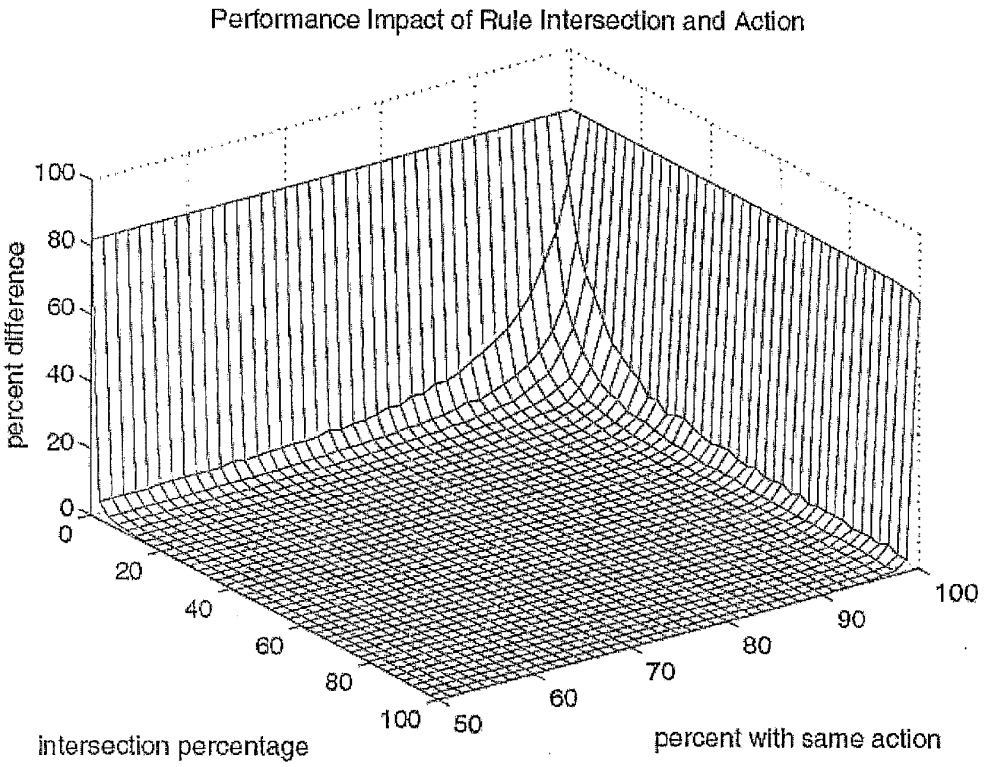

Performance Impact of Rule Intersection and Action
percent difference
100
80
60
40
20
0
0
20
40
60
80
100
intersection percentage
50
60
70
80
90
100
percent with same action

FIG. 4B

r_1	(UDP,	1.1.*,	ACCEPT)	$p_1 = 0.01$
r_2	(TCP,	1.*,	ACCEPT)	$p_2 = 0.02$
r_3	(TCP,	2.*,	ACCEPT)	$p_3 = 0.17$
r_4	(UDP,	1.*,	DENY)	$p_4 = 0.30$
r_5	(*,	*,	DENY)	$p_5 = 0.50$

FIG. 5A

r_3	(TCP,	2.*,	ACCEPT)	$p_3 = 0.17$
r_2	(TCP,	1.*,	ACCEPT)	$p_2 = 0.02$
r_1	(UDP,	1.1.*,	ACCEPT)	$p_1 = 0.01$
r_4	(UDP,	1.*,	DENY)	$p_4 = 0.30$
r_5	(*,	*,	DENY)	$p_5 = 0.50$

FIG. 5B

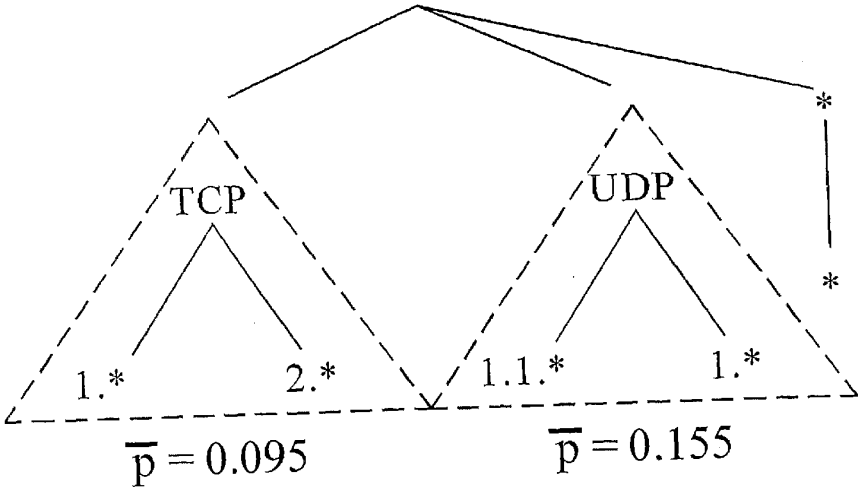


FIG. 6A

r_1	(UDP,	1.1.*,	ACCEPT)	$p_1 = 0.01$
r_4	(UDP,	1.*,	DENY)	$p_4 = 0.30$
r_3	(TCP,	2.*,	ACCEPT)	$p_3 = 0.17$
r_2	(TCP,	1.*,	ACCEPT)	$p_2 = 0.02$
r_5	(*,	*,	DENY)	$p_5 = 0.50$

FIG. 6B

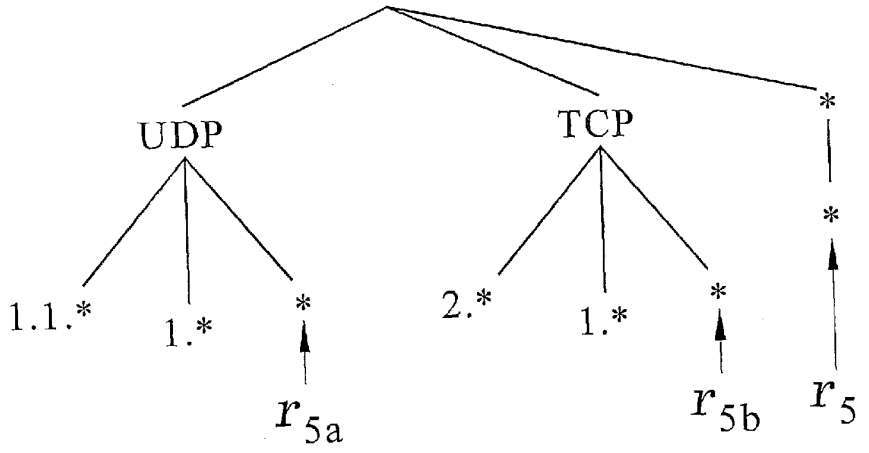


FIG. 7A

r_1	(UDP,	1.1.*,	ACCEPT)	$p_1 = 0.01$
r_4	(UDP,	1.*,	ACCEPT)	$p_4 = 0.30$
r_{5a}	(UDP,	*,	DENY)	$p_{5a} = 0.47$
r_3	(TCP,	2.*,	ACCEPT)	$p_3 = 0.17$
r_2	(TCP,	1.*,	ACCEPT)	$p_2 = 0.02$
r_{5b}	(TCP,	*,	DENY)	$p_{5b} = 0.03$

FIG. 7B

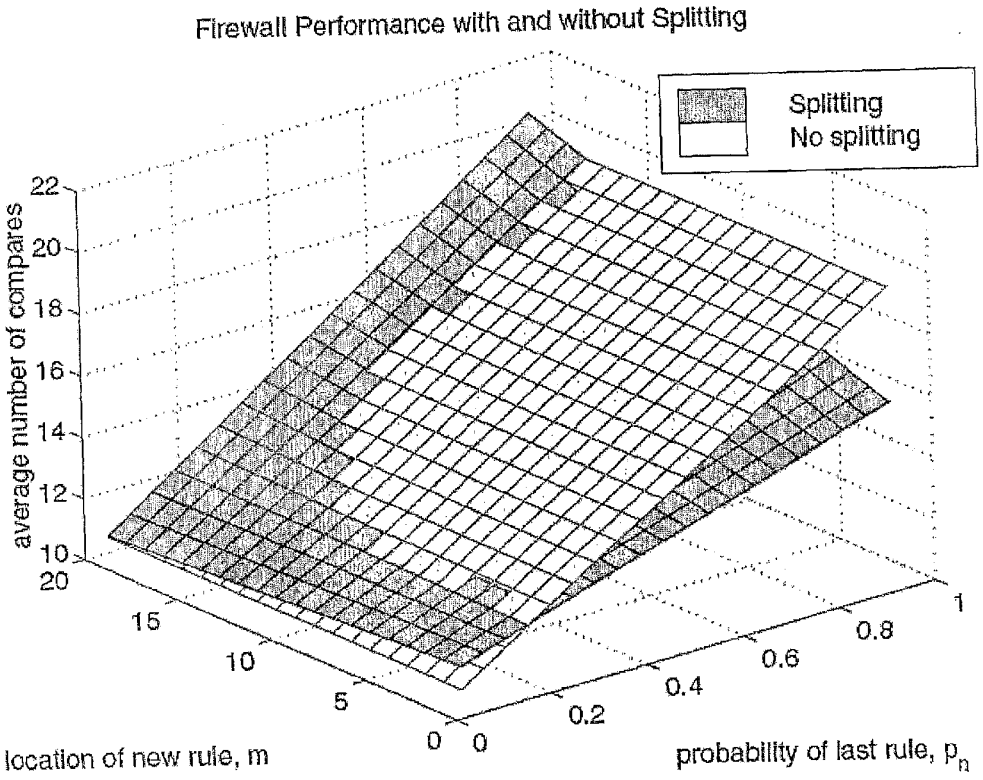

Firewall Performance with and without Splitting
Splitting
No splitting
average number of compares
22
20
18
16
14
12
10
20
15
10
5
0
location of new rule, m
0
0.2
0.4
0.6
0.8
1
probability of last rule, p_n

FIG. 8A

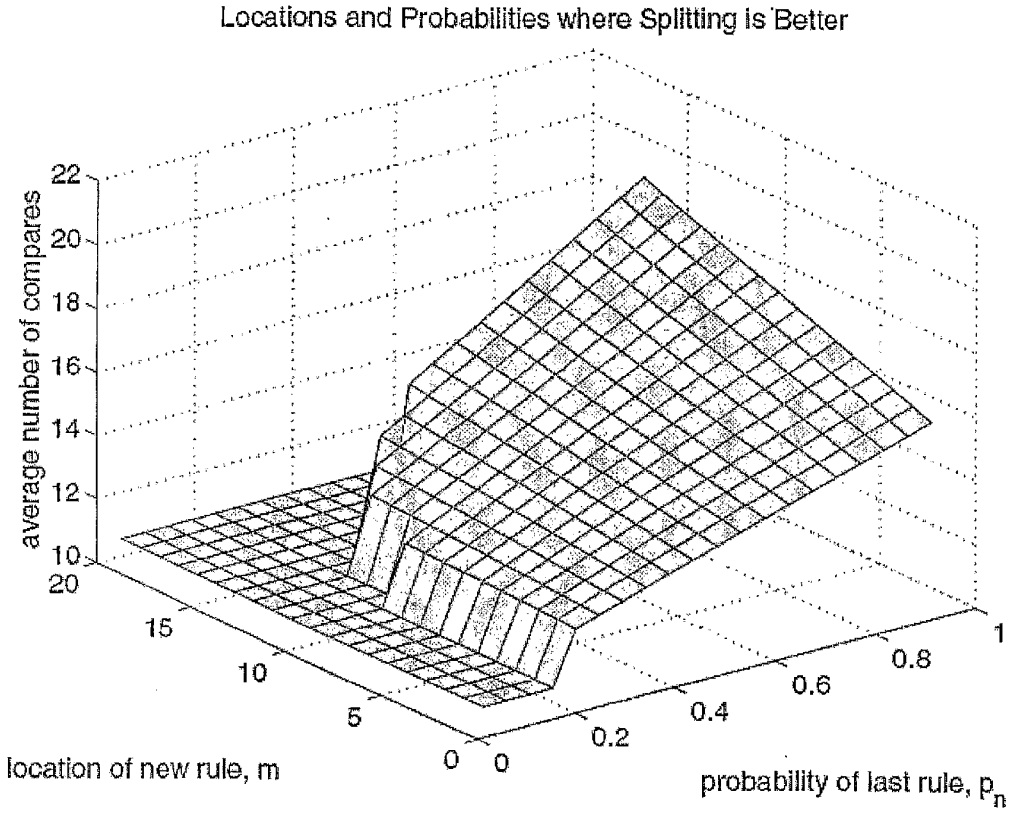

Locations and Probabilities where Splitting is Better
average number of compares
22
20
18
16
14
12
10
20
15
10
5
0
location of new rule, m
0
0.2
0.4
0.6
0.8
1
probability of last rule, p_n

FIG. 8B

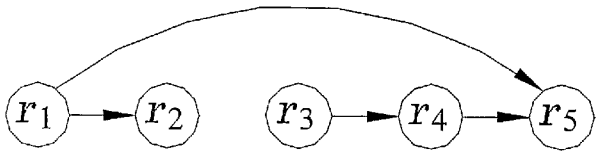


FIG. 9A

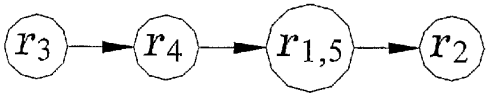


FIG. 9B



FIG. 10A

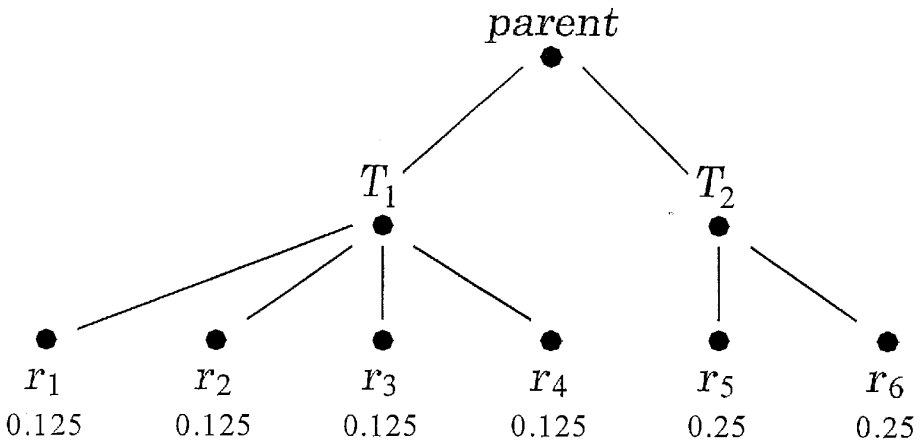


FIG. 10B

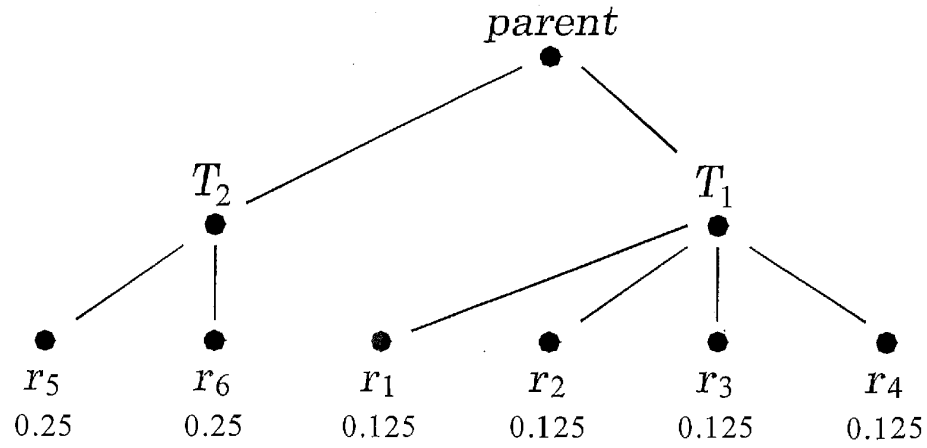


FIG. 10C

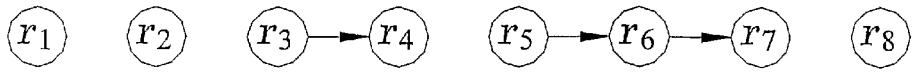


FIG. 11A

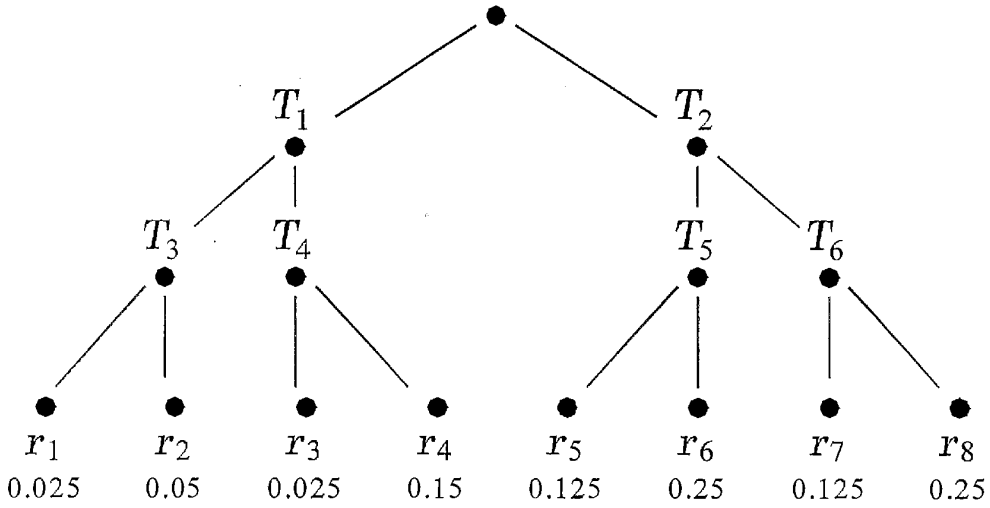


FIG. 11B

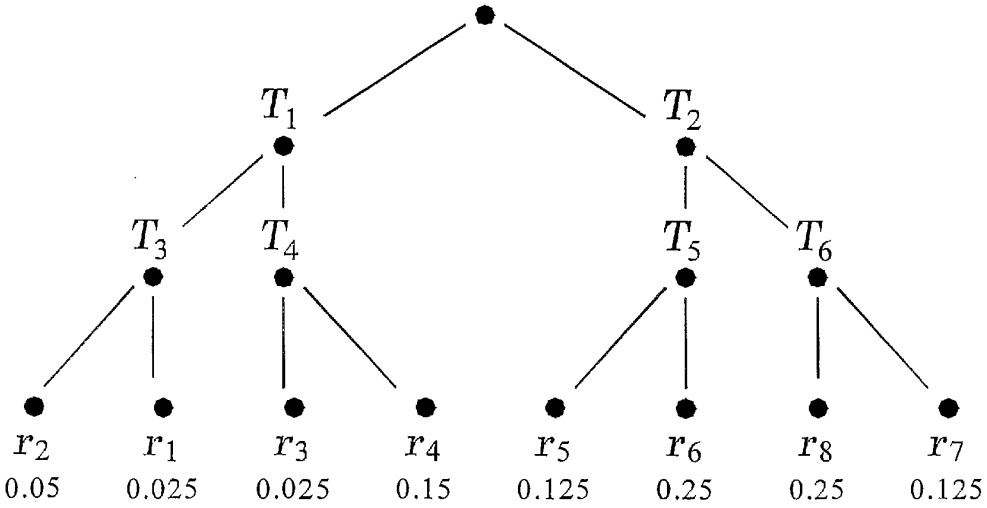


FIG. 11C

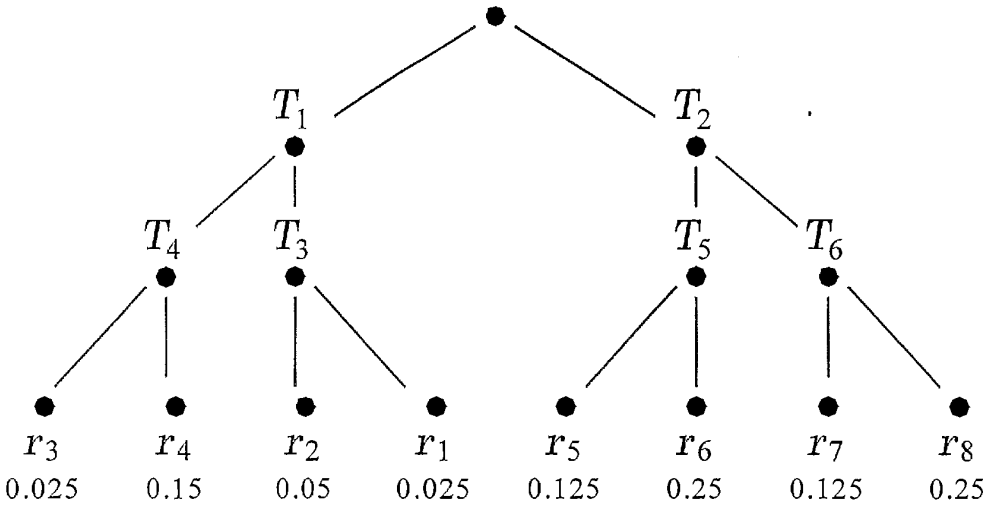


FIG. 11D

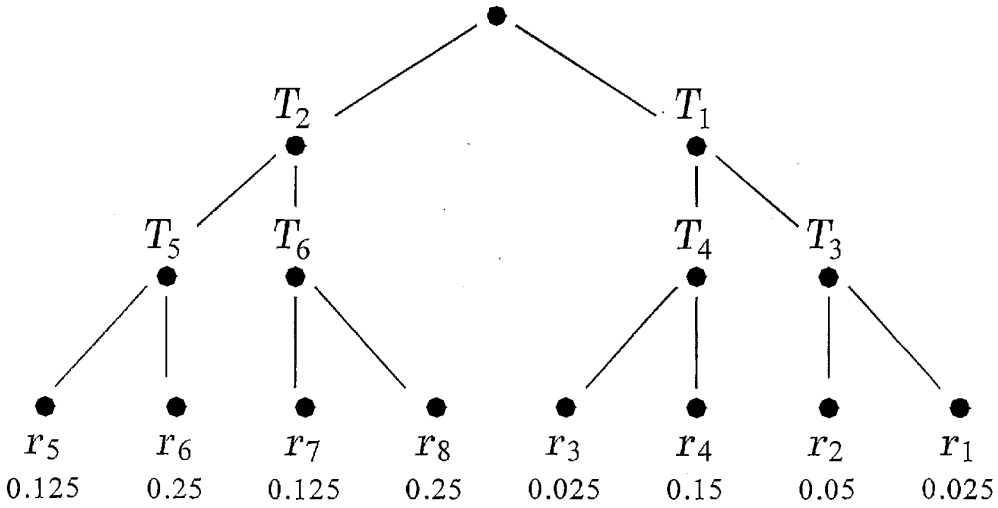


FIG. 11E

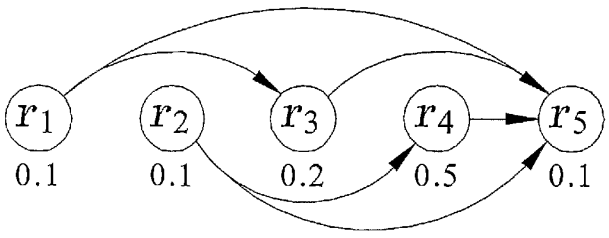


FIG. 12A

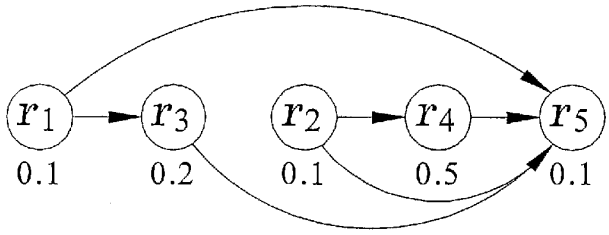


FIG. 12B

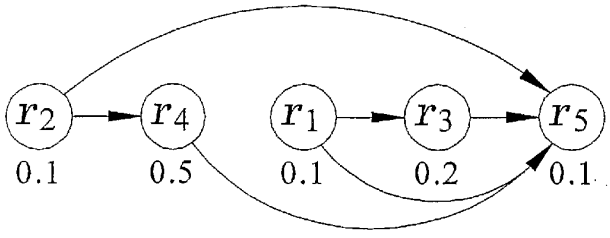


FIG. 12C

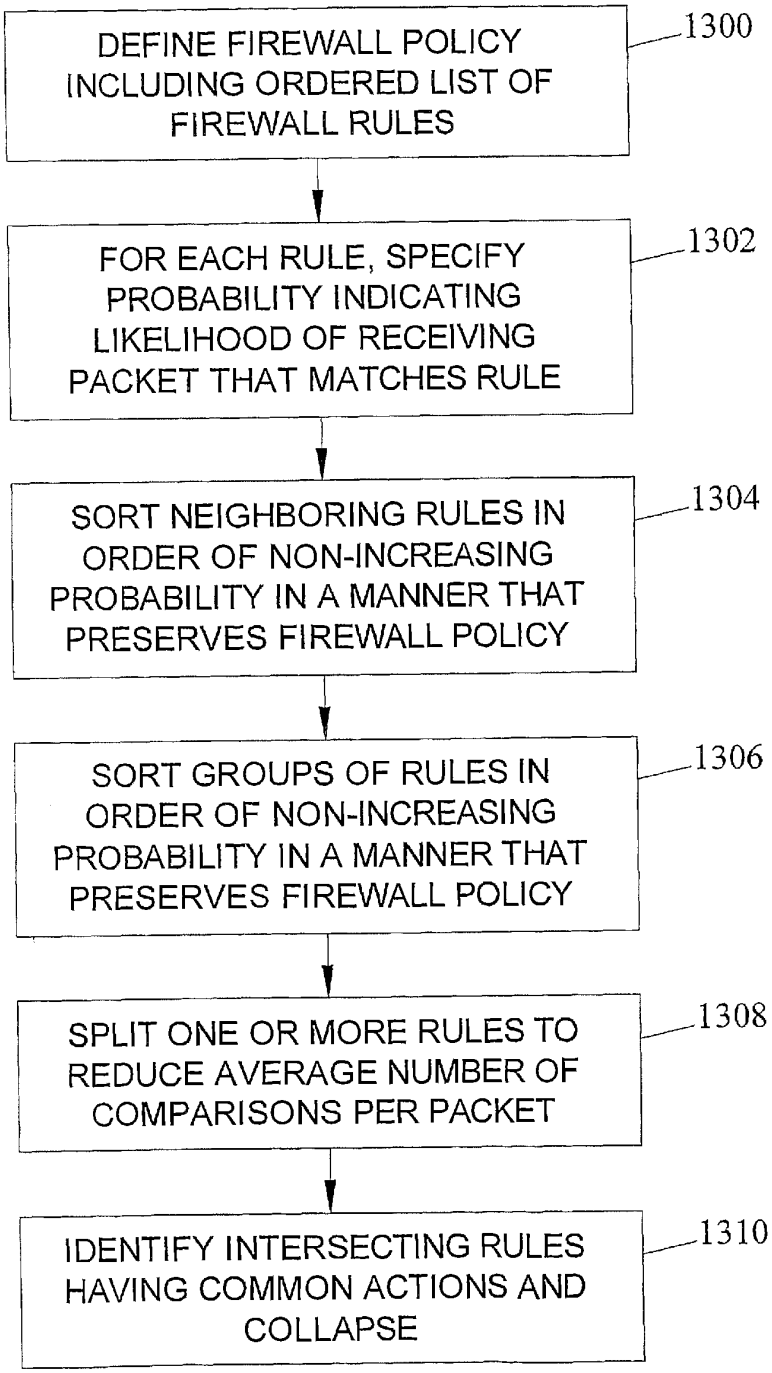

DEFINE FIREWALL POLICY INCLUDING ORDERED LIST OF FIREWALL RULES
1300
FOR EACH RULE, SPECIFY PROBABILITY INDICATING LIKELIHOOD OF RECEIVING PACKET THAT MATCHES RULE
1302
SORT NEIGHBORING RULES IN ORDER OF NON-INCREASING PROBABILITY IN A MANNER THAT PRESERVES FIREWALL POLICY
1304
SORT GROUPS OF RULES IN ORDER OF NON-INCREASING PROBABILITY IN A MANNER THAT PRESERVES FIREWALL POLICY
1306
SPLIT ONE OR MORE RULES TO REDUCE AVERAGE NUMBER OF COMPARISONS PER PACKET
1308
IDENTIFY INTERSECTING RULES HAVING COMMON ACTIONS AND COLLAPSE
1310

FIG. 13

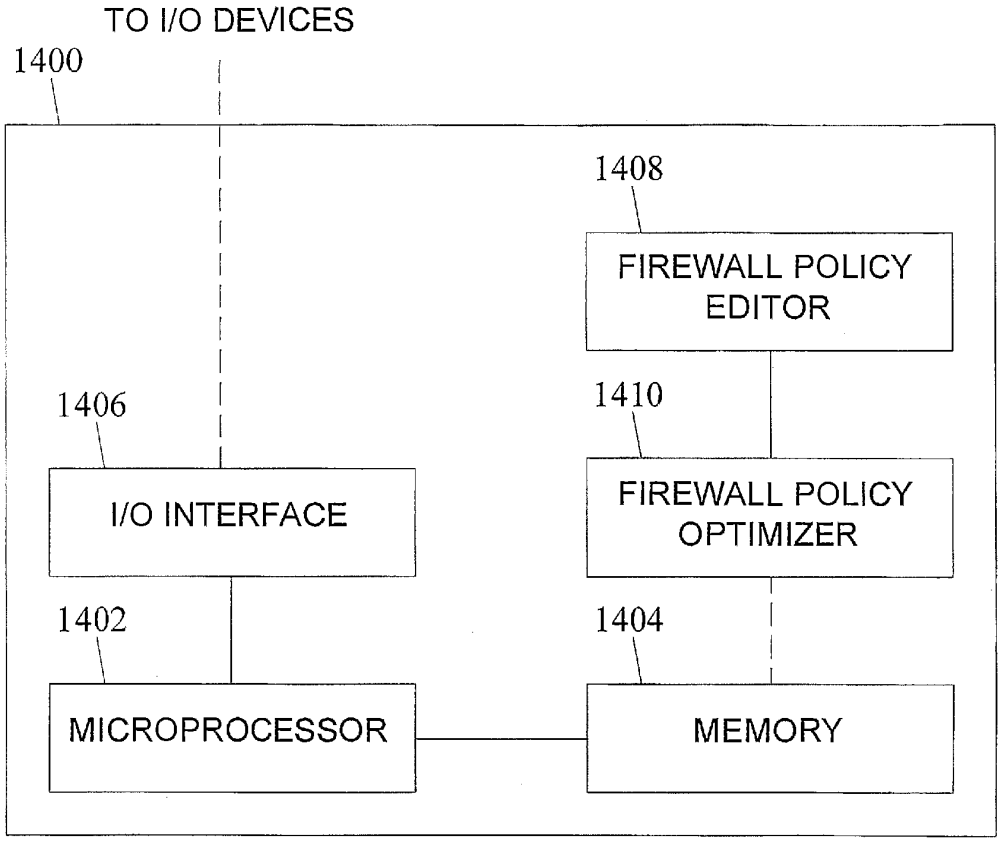

TO I/O DEVICES
1400
1408
FIREWALL POLICY EDITOR
1406
1410
I/O INTERFACE
FIREWALL POLICY OPTIMIZER
1402
1404
MICROPROCESSOR
MEMORY

FIG. 14

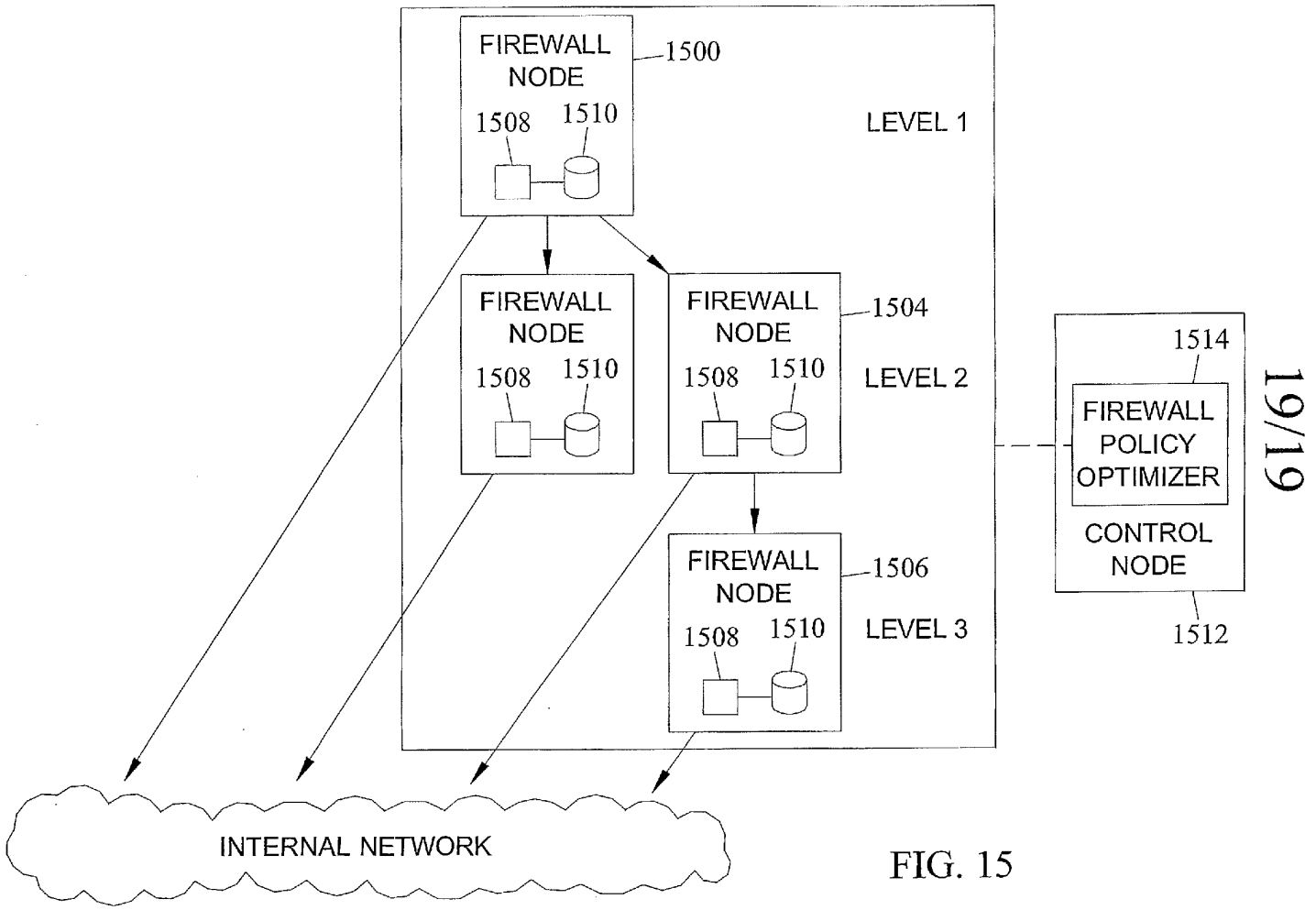

FIREWALL NODE
1500
1508
1510
LEVEL 1
FIREWALL NODE
1508
1510
FIREWALL NODE
1504
1508
1510
LEVEL 2
FIREWALL NODE
1506
1508
1510
LEVEL 3
1514
FIREWALL POLICY OPTIMIZER
CONTROL NODE
1512
INTERNAL NETWORK

FIG. 15

Espacenet

Bibliographic data: EP1864226 (B1) — 2013-05-15

METHODS, SYSTEMS, AND COMPUTER PROGRAM PRODUCTS FOR NETWORK FIREWALL POLICY OPTIMIZATION

Inventor(s): FULP ERRIN W [US]; TARSA STEPHEN J [US] ± (FULP, ERRIN, W, ; TARSA, STEPHEN, J)

Applicant(s): UNIV WAKE FOREST [US] ± (WAKE FOREST UNIVERSITY)

Classification: **- international:** ***H04L29/06***
- cooperative: **H04L63/0263**; **H04L63/20**

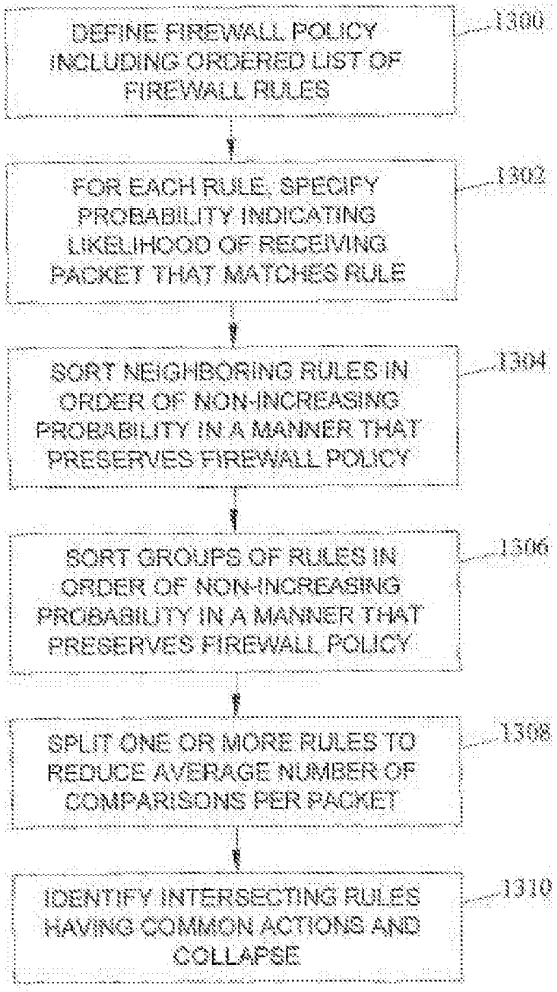
Application number: EP20060758213 20060328

Priority number (s): WO2006US11291 20060328 ; US20050665664P 20050328

Also published as: EP1864226 (A2) EP1864226 (A4) WO2006105093 (A2) WO2006105093 (A3) US2006248580 (A1) more

Abstract not available for EP1864226 (B1)
Abstract of corresponding document: WO2006105093 (A2)

Methods, systems, and computer program products for firewall policy optimization are disclosed. According to one method, a firewall policy including an ordered list of firewall rules is defined. For each rule, a probability indicating a likelihood of receiving a packet matching the rule is determined. The rules are sorted in order of non-increasing probability in a manner that preserves the firewall policy.


DEFINE FIREWALL POLICY INCLUDING ORDERED LIST OF FIREWALL RULES
1300
FOR EACH RULE, SPECIFY PROBABILITY INDICATING LIKELIHOOD OF RECEIVING PACKET THAT MATCHES RULE
1302
SORT NEIGHBORING RULES IN ORDER OF NON-INCREASING PROBABILITY IN A MANNER THAT PRESERVES FIREWALL POLICY
1304
SORT GROUPS OF RULES IN ORDER OF NON-INCREASING PROBABILITY IN A MANNER THAT PRESERVES FIREWALL POLICY
1306
SPLIT ONE OR MORE RULES TO REDUCE AVERAGE NUMBER OF COMPARISONS PER PACKET
1308
IDENTIFY INTERSECTING RULES HAVING COMMON ACTIONS AND COLLAPSE
1310

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 864 226 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
15.05.2013 Bulletin 2013/20

(21) Application number: **06758213.0**

(22) Date of filing: **28.03.2006**

(51) Int Cl.:
H04L 29/06 *(2006.01)*

(86) International application number:
PCT/US2006/011291

(87) International publication number:
WO 2006/105093 (05.10.2006 Gazette 2006/40)

(54) **METHODS, SYSTEMS, AND COMPUTER PROGRAM PRODUCTS FOR NETWORK FIREWALL POLICY OPTIMIZATION**

VERFAHREN, SYSTEME UND COMPUTERPROGRAMM ZUR OPTIMIERUNG DER FIREWALL-REGELUNG EINES NETZWERKES

PROCEDES, SYSTEMES ET PRODUITS DE PROGRAMMES INFORMATIQUES POUR L'OPTIMISATION DE LA POLITIQUE DE PARE-FEU D'UN RESEAU

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IS IT LI LT LU LV MC NL PL PT RO SE SI SK TR

(30) Priority: **28.03.2005 US 665664 P**

(43) Date of publication of application:
12.12.2007 Bulletin 2007/50

(73) Proprietor: **Wake Forest University**
Winston Salem, NC 27101 (US)

(72) Inventors:
- **FULP, Errin, W.**
  Winston-Salem, North Carolina 27104 (US)
- **TARSA, Stephen, J.**
  Duxbury, Massachusetts 02332-3503 (US)

(74) Representative: **Isarpatent**
Patent- und Rechtsanwälte
Postfach 44 01 51
80750 München (DE)

(56) References cited:
EP-A2- 1 006 701
WO-A2-2006/093557
US-A1- 2002 198 981
US-B1- 6 484 261
US-B1- 6 662 235
US-B1- 7 089 581
US-B2- 7 237 267

- **Errin W. Fulp: "Optimization of Network Firewall Policies Using Ordered Sets and Directed Acyclical Graphs", Wake Forest University , 2004, pages 1-12, XP002618343, Retrieved from the Internet: URL:www.cs.wfu.edu/~fulp/Papers/ewflist.ps [retrieved on 2011-01-24]**
- **Errin W. Fulp and Stephen J. Tarsa: "Network Firewall Policy Tries", Wake Forest University , 2004, pages 1-17, XP002618344, Retrieved from the Internet: URL:http://citeseerx.ist.psu.edu/viewdoc/s ummary?doi=10.1.1.143.5727 [retrieved on 2011-01-24]**
- **Lili Qiu, George Varghese, Subhash Suri: "Fast Firewall Implementations for Software and Hardware-based Routers", , 2001, pages 241-250, XP002618345, Retrieved from the Internet: URL: http://www.ieee-icnp.org/2001/papers/2001-26.pdf [retrieved on 2011-01-25]**
- **Errin W. Fulp: "Errin Fulp", , 2010, XP002618346, Retrieved from the Internet: URL: www.cs.wfu.edu/~fulp/ewfPub.html [retrieved on 2011-01-24]**

Note: Within nine months of the publication of the mention of the grant of the European patent in the European Patent Bulletin, any person may give notice to the European Patent Office of opposition to that patent, in accordance with the Implementing Regulations. Notice of opposition shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

Printed by Jouve, 75001 PARIS (FR)

EP 1 864 226 B1

Description

TECHNICAL FIELD

[0001] The subject matter described herein relates to optimizing polices for network firewalls. More particularly, the subject matter described herein relates to methods, systems, and computer program products for optimizing performance of network firewalls.

BACKGROUND ART

[0002] Network firewalls remain the forefront defense for most computer systems. Guided by a security policy, these devices provide access control, auditing, and traffic control [3, 30, 31]. A security policy is a list of ordered rules, as seen in Table 1, that defines the action to perform on matching packets. For example, an accept action passes the packet into or from the secure network, while deny causes the packet to be discarded. In many implementations, the rule set is stored internally as a linked list. A packet is sequentially compared to the rules, starting with the first, until a match is found; otherwise, a default action is performed [30, 31]. This is referred to as a first- match policy and is used in many firewall systems including the Linux firewall implementation iptables [25].

Table 1: Example Security Policy Consisting of Multiple Ordered Rules

		Source		Destination			
No.	Proto.	IP	Port	IP	Port	Action	Prob.
1	UDP	1.1.*	*	*	80	deny	0.01
2	TCP	2.*	*	1.*	90	accept	0.02
3	UDP	*	*	1.*	*	accept	0.10
4	TCP	2.*	*	1.*	20	accept	0.17
5	UDP	1.*	*	*	*	accept	0.20
6	*	*	*	*	*	deny	0.50

[0003] Traditional firewall implementations consist of a single, dedicated machine, similar to a router, that sequentially applies rules to each arriving packet. However, packet filtering represents a significantly higher processing load than routing decisions [24, 29, 31]. For example, a firewall that interconnects two 100 Mbps networks would have to process over 300,000 packets per second [30]. Successfully handling these traffic loads becomes more difficult as rule sets become more complex [4, 22, 31]. Furthermore, firewalls must be capable of processing even more packets as interface speeds increase. In a high-speed environment (e.g. Gigabit Ethernet), a single firewall can easily become a bottleneck and is susceptible to DoS attacks [4, 9, 13, 14]. An attacker could simply inundate the firewall with traffic, delaying or preventing legitimate packets from being processed.

[0004] One approach to increase firewall performance focuses on improving hardware design. Current research is investigating different distributed firewall designs to reduce processing delay [4, 9, 22], and possibly provide service differentiation [11]. Another approach focuses on improving performance via better firewall software [6, 7, 12, 16, 17, 24]. Similar to approaches that address the longest matching prefix problem for packet classification [8,10,25, 28], solutions typically represent the firewall rule set in different fashions (e.g. tree structures) to improve performance. While both approaches, together or separate, show great promise, each requires radical changes to the firewall system, and therefore are not amenable to current or legacy systems.

[0005] EP Patent Publication No. 1006701 discloses a packet data filter which stores ordered rules and sequentially applies the rules to received data packets to determine the disposition of the data packet. The packet filter maintains a match count in memory which indicates the number of times each rule matched an incoming data packet. Periodically, at the initiation of a user, or based on operating parameters of the filter, the rules are automatically re-ordered based on the match count. As a result of the re-ordering, rules with higher match counts are moved earlier in the sequential evaluation order and rules with lower match counts are moved later in the sequential evaluation order. As such, rules which are more likely to match incoming data packets are evaluated earlier, thus avoiding the evaluation of later rules.; In order to prevent a re-ordering which would change the overall security policy of the packet filter, pairs of rules are compared to determine if they conflict (i.e., the swapping of the two rules would result in a change in the overall security policy). During re-ordering, the swapping of conflicting rules is prevented.

[0006] US Patent Publication No. 2002/198981 discloses a method and system for determining whether to enforce a

IPR2023-01353 Page 00989

plurality of filter rules for a packet including a key in a computer network is disclosed. Each of the plurality of filter rules has a priority. The method and system include accumulating statistics for each of the plurality of filter rules. The statistics indicate a frequency of enforcement for each of the plurality of filter rules. The method and system also include placing the plurality of filter rules in an order for testing against the key. The order is based on the frequency of each filter rule of the portion of the plurality of filter rules. Consequently, more frequently enforced filter rules may be tested first.

[0007] WO Publication No. 2006093557 discloses methods, systems, and computer program products for providing function-paralell firewalls are disclosed. According to one aspect, a function-parallel firewall includes a first firewall node for filtering received packets using a first portion of a rule set including a plurality of rules. The first portion includes less than all of the rules in the rule set. At least one second firewall node filters packets using a second portion of the rule set. The second portion includes at least one rule in the rule set that is not present in the first portion. The first and second portions together include all of the rules in the rule set.

[0008] Accordingly, there exists a need for improved methods, systems, and computer program products for network firewall policy optimization.

SUMMARY

[0009] The subject matter described herein includes methods, systems, and computer program products for network firewall policy optimization. According to one method, a firewall policy including an ordered list of firewall rules is defined. For each rule, a probability indicating a likelihood of receiving a packet matching the rule is defined. Neighbouring rules are sorted in order of non- increasing probability in a manner that preserves the firewall policy.

[0010] As used herein, the term "firewall" refers to a logical entity that is adapted to filter packets at the ingress and/or egress portions of a network based on a policy. The term "firewall" is intended to include systems that block packets from entering or leaving a network and systems that perform intrusion detection and intrusion protection functions for packets entering or leaving a network. Thus, the methods and systems described herein for firewall policy optimization can be used to optimize policies for intrusion detection and intrusion protection firewalls.

[0011] The subject matter described herein includes a method to improve firewall performance and lower packet delay that can be applied to both legacy and current systems. In one implementation, a firewall rule set may be reordered to minimize the average number of rule comparisons to determine the action, while maintaining the integrity of the original policy. Integrity is preserved if the reordered and original rules always arrive at the same result. To maintain integrity, a firewall rule set may be modeled as a Directed Acyclical Graph (DAG), where vertices are firewall rules and edges indicate precedence relationships. Given this representation, any linear arrangement of the DAG is proven to maintain the original policy integrity. Unfortunately, determining the optimal rule order from all the possible linear arrangements is proven to be NP- complete, since it is equivalent to sequencing jobs with precedence constraints for a single machine [15]. Although determining the optimal order is NP- complete, a heuristic will be described below to order firewall rules that reduces the average number of comparisons while maintaining integrity. Simulation results show the proposed reordering method yields rule orders that are comparable to optimal; thus, provides a simple means to significantly improve firewall performance and lower packet delay.

[0012] The subject matter described herein for network firewall policy optimization may be implemented using any combination of hardware, software, or firmware. For example, the subject matter described herein may be implemented using a computer program product comprising computer executable instructions embodied in a computer readable medium. Exemplary computer readable media suitable for implementing the subject matter described herein include disk memory devices, chip memory devices, programmable logic devices, application specific integrated circuits, and downloadable electrical signals. In addition, a computer program product that implements the subject matter described herein may be implemented using a single device or computing platform or may be distributed across multiple devices or computing platforms.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] Preferred embodiments of the subject matter described herein will now be explained with reference to the accompanying drawings of which:

Figure 1A is a block diagram of a rule list Directed Acyclical Graph (DAG) for the firewall rules in Table 1;
Figure 1 B is a linear arrangement of firewall rules corresponding to the original rule order;
Figure 2A is a policy DAG representing a reordered version of the firewall rule set in Figure 1 after sorting;
Figure 2B is a policy DAG for the rule set illustrated in Table 1 illustrating an optimal rule order;
Figure 3A is a graph illustrating average numbers of packet comparisons versus intersection percentage for different orderings of a firewall policy;
Figure 3B is a graph illustrating the percent difference in average numbers of packet comparisons between sorted

IPR2023-01353 Page 00990

and optimal sortings of a firewall policy;
Figure 4A is a graph of percent difference comparisons between original and sorted versions of a firewall policy;
Figure 4B is a graph illustrating the performance impact of considering rule intersection and actions in performing firewall rule sorting;
Figure 5A is a table illustrating a firewall policy before sorting;
Figure 5B is a table illustrating a sorted version of the firewall policy of Figure 5A;
Figure 6A is a policy trie representation of the firewall policy of Figure 5B;
Figure 6B is a table representing the firewall policy of Figure 5A after TCP and UDP sub-trie rotation;
Figure 7A is a policy trie illustrating the splitting of rule r_5 into rules r_{5a} and r_{5b};
Figure 7B is a table illustrating the firewall policy of Figure 5A after splitting rule r_5 into rules r_{5a} and r_{5b};
Figure 8A is a graph illustrating firewall performance with and without rule splitting;
Figure 8B is a graph illustrating locations and probabilities where splitting is better;
Figure 9A is a policy DAG of a firewall rule set;
Figure 9B is a policy DAG illustrating compression of rules r_1 and r_5;
Figure 10A is a policy DAG of a firewall rule list;
Figure 10B is an original policy trie before sorting;
Figure 10C is a policy trie illustrating the firewall rules after rotating sub-tries T_1 and T_2;
Figure 11A is a policy DAG illustrating a firewall policy;
Figure 11B is a policy trie illustrating firewall rules of the policy before sorting;
Figure 11C is a policy trie illustrating the firewall policy after exchanging nodes r_1 and r_2 and nodes r_7 and r_8;
Figure 11 D is a policy trie illustrating the firewall policy after exchanging sub-tries T_3 and T_4;
Figure 11 E is a policy trie illustrating the firewall policy after exchanging sub-tries T_1 and T_2;
Figure 12A is a policy DAG illustrating an original order for a firewall policy;
Figure 12B is a policy DAG illustrating a sorted version of the firewall policy of Figure 12A after exchanging neighbors;
Figure 12C is a policy DAG illustrating a sorted version of the firewall policy after exchanging lists L_1 including rules r_1 and r_3 and lists L_2 including rules r_2 and r_4;
Figure 13 is a flow chart illustrating exemplary steps for network firewall policy optimization according to an embodiment of the subject matter described herein;
Figure 14 is a block diagram illustrating exemplary components of a system for firewall policy optimization according to an embodiment of the subject matter described herein; and
Figure 15 is a block diagram illustrating a plurality of firewall nodes for implementing a firewall policy according to an embodiment of the subject matter described herein.

DETAILED DESCRIPTION OF THE INVENTION

Modeling Firewall Security Policies

[0014] As described above, a firewall rule set, also known as a firewall policy, is traditionally an ordered list of firewall rules. Firewall policy models have been the subject of recent research [1, 2, 16]; however, the primary purpose is anomaly detection and policy verification. In contrast, the policy model described in this herein is designed for firewall performance optimization and integrity. Firewall performance refers to reducing the average number of comparisons required to determine an action, while integrity refers to maintaining the original policy intent. Although improving the worst-case performance is important, it is not possible without changing the list-based representation [16, 24].

Firewall Rule and Policy Models

[0015] In the examples described herein, a rule r is modeled as an ordered tuple of sets, $r = (r[1], r[2], ..., r[k])$. Order is necessary among the tuples since comparing rules and packets requires the comparison of corresponding tuples. Each tuple $r[i]$ is a set that can be fully specified, specify a range, or contain wildcards '*' in standard prefix format. For the Internet, security rules are commonly represented as a 5-tuple consisting of: protocol type, IP source address, source port number, IP destination address, and destination port number [30, 31]. Given this model, the ordered tuples can be supersets and subsets of each other, which forms the basis of precedence relationships. In addition to the prefixes, each filter rule has an action, which is to accept or deny. However, the action will not be considered when comparing packets and rules. Similar to a rule, a packet (IP datagram) d can be viewed as an ordered k-tuple $d = (d[1], d[2], ..., d[k])$; however, ranges and wildcards are not possible for any packet tuple.

[0016] Using the previous rule definition, a standard security policy can be modeled as an ordered set (list) of n rules, denoted as $R = \{r_1, r_2, ..., r_n\}$. A packet d is sequentially compared against each rule r_i starting with the first, until a match is found ($d \Rightarrow r_i$) then the associated action is performed. A match is found between a packet and rule when every tuple

IPR2023-01353 Page 00991

of the packet is a subset of the corresponding tuple in the rule.
Definition Packet d matches r_i if

$$d \Rightarrow r_i \qquad \text{iff} \qquad d[l] \subseteq r_i[l], \qquad l = 1, \ldots, k$$

[0017] The rule list R is comprehensive if for every possible legal packet d a match is found using R. Furthermore, two rule lists R and R' are equivalent if for every possible legal packet d the same action is performed by the two rule lists. If R and R' are different (e.g. a reorder) and the lists are equivalent, then the policy integrity is maintained.

[0018] As previously mentioned, a rule list has an implied precedence relationship where certain rules must appear before others if the integrity of the policy is to be maintained. For example consider the rule list in Table 1. Rule r_1 must appear before rules r_3 and r_5, likewise rule r_6 must be the last rule in the policy. If for example, rule r_3 was moved to the beginning of the policy, then it will shadow [2] the original rule r_1. However, there is no precedence relationship between rules r_2 and r_4 given in Table 1. Therefore, the relative ordering of these two rules will not impact the policy integrity and can be changed to improve performance. The present example assumes that the original policy is free from any anomalies. Likewise, when a policy is reordered to improve performance it should not introduce any anomalies, which will occur if precedence relationships are not maintained. As a result, a model is needed to effectively represent precedence relationships.

Modeling Rule List Precedence Relationships

[0019] The precedence relationship between rules in a policy will be modeled as a Directed Acyclical Graph (DAG) [23, 18]. Such graphs have been successfully used to represent the relative order of individual tasks that must take place to complete a job (referred to as a task graph model). Since certain rules must appear before others to maintain policy integrity, this structure is well suited for modeling the precedence of firewall rules. Let $G = (R, E)$ be a rule list DAG for a rule list R, where vertices are rules and edges E are the precedence relationships (constraint). A precedence relationship, or edge, exists between rules r_i and r_j, if $i < j$, the actions for each rule are different, and the rules intersect.
Definition The intersection of rule r_i and r_j, denoted as $r_i \cap r_j$ is

$$r_i \cap r_j = (r_i[l] \cap r_j[l]), \qquad l = 1, \ldots, k$$

Therefore, the intersection of two rules results in an ordered set of tuples that collectively describes the packets that match both rules. The rules r_i and r_j intersect if every tuple of the resulting operation is non-empty. In contrast, the rules r_i and r_j do not intersect, denoted as $r_i \not\cap r_j$, if at least one tuple is the empty set. Note the intersection operation is symmetric; therefore, if r_i intersects r_j then r_j will intersect r_i. The same is true for rules that do not intersect.

[0020] For example consider the rules given in Table 1, the intersection of r_1 and r_3 yields (UDP, 1.1.*, *, 1.*, 80). Again, the rule actions are not considered in the intersection or match operation. Since these two rules intersect, a packet can match both rules for example d = (UDP, 1.1.1.1, 80, 1.1.1.1, 80). Furthermore, the actions of the two rules are different. Therefore, the relative order must be maintained between these two rules and an edge drawn from r_1 to r_3 must be present in the rule list DAG, as seen in Figures 1A and 1B. More particularly, Figure 1 A illustrates a rule list DAG for the rules in Table 1. Figure 1B illustrates a linear arrangement of the rules in Figure 1A. In Figures 1A and 1 B, the vertices represent rules, the circles represents an accept rules, and the squares represents deny rules. Edges that connect the vertices represent precedence requirements. As can be seen from Figure 1 B, because of the edge between rules r_1 and r_3, precedence between rules r_1 and r_3 must be maintained. In contrast consider the intersection of rules r_3 and r_5. These two rules intersect, indicating packets belonging to the set (UDP, 1.*, *, 1.*, *) would match both rules. However, it does not matter which of the two rules a packet matches first, since the action is the same for both rules. Therefore, an edge does not exist between rules r_3 and r_5 in the diagram. Similarly, rules r_2 and r_4 do not intersect due to the fifth tuple (destination port). A packet cannot match both rules indicating the relative order can change; therefore, an edge will not exist between them.

[0021] The match operation can be used to identify precedence relationships, but it cannot do so in every case. Consider a partial-match example [1], where r_a = (UDP, *, 80, 10.*, 90, accept) and r_b = (UDP, 10.*, 80, *, 90, deny). The intersection of r_a and r_b is (UDP, 10.*, 80, 10.*, 90); therefore a packet, such as d = (UDP, 10.10.10.10, 80, 10.10.10.10, 90), can match both rules. If r_a appears before r_b then the packet d is accepted, but if r_b occurs before r_a then d is rejected. As a result, the order of r_a and r_b in the original policy must be maintained. However, the match operation is unable to identify the precedence in this example. A partial match exists in between rules r_3 and r_5 in Table

IPR2023-01353 Page 00992

1, but as previously discussed an edge does not exist between the rules since the actions are the same.

[0022] Using the rule list DAG representation a linear arrangement is sought that improves the firewall performance. As depicted in Figure 1 B, a linear arrangement (permutation or topological sort) is a list of DAG vertices where all the successors of a vertex appear in sequence after that vertex [23]. Therefore it follows that a linear arrangement of a rule list DAG represents a rule order, if the vertices are read from left to right. Furthermore, it is proven in the following theorem that any linear arrangement of a rule list DAG maintains integrity: **Theorem** Any linear arrangement of a rule list DAG maintains integrity.

[0023] **Proof** Assume a rule list DAG G is constructed from the security policy R that is free of anomalies. Consider any two rules r_i and r_j in the policy, where $i < j$. If an edge between r_i and r_j in G does not exist, then a linear arrangement of G can interchange the order of the two rules. An edge will not exist if the rules do not intersect; however, a reorder will not affect integrity since a packet cannot match both rules. Shadowing is not introduced due to the reorder since the intersection operation is symmetric. An edge will not exist if the two rules intersect but have the same action; however, a reorder will not affect integrity since the same action will occur regardless of which rule is matched first. If an edge does exist between the rules, then their relative order will be maintained in every linear arrangement of G; thus maintaining precedence and integrity.

Rule List Optimization

[0024] As described above, it is important to inspect packets as quickly as possible given increasing network speeds and QoS requirements. Using the rule list DAG to maintain policy integrity, a linear arrangement is sought that minimizes the average number of comparisons required. However, this will require information not present in the firewall rule list. Certain firewall rules have a higher probability of matching a packet than others. As a result, it is possible to develop a policy profile over time that indicates frequency of rule matches (similar to cache hit ratio). Let $P = \{p_1, p_2, ..., p_n\}$ be the policy profile, where p_i is the probability that a packet will match rule i (first match in the policy). Furthermore, assume a packet will always find a match, $\sum_{i=1}^{n} p_i = 1$; therefore R is comprehensive. Using this information, the average number of rule comparisons required is

$$E[n] = \sum_{i=1}^{n} i \cdot p_i \qquad (1)$$

For example, the average number of comparisons required for the rule set in Table 1 is 5.03.

[0025] Given a rule list DAG $G = (R, E)$ and policy profile $P = \{p_1, p_2, ..., p_n\}$ a linear arrangement π of G is sought that minimizes Equation 1. In the absence of precedence relationships, the average number of comparisons is minimized if the rules are sorted in non-increasing order according to the probabilities [26], which is also referred to as Smith's algorithm [27]. Precedence constraints cause the problem to be more realistic; however, such constraints also make determining the optimal permutation more problematic.

[0026] Determining the optimal rule list permutation can be viewed as job scheduling for a single machine with precedence constraints [15, 21]. The notation for such scheduling problems is $\alpha|\beta|\gamma|\delta$, where α is the number of machines, β is the precedence (or absence of) which can be represented as a DAG, γ is a restriction on processing time, and δ is the optimality criterion [15]. Determining the optimal rule order is similar to the $1|\beta|1|\sum \omega C_j$ scheduling problem, or optimality criterion, where ω is a weight associated with a job (for example, importance) and C_j is the completion time. As previously noted, the $1|||\sum \omega C_j$ problem can be solved in linear time the using Smith's algorithm [27], which orders jobs according to non-decreasing $\frac{t_i}{\omega}$ ratio, where t_j is the processing time of job i. In this case set $t_j = 1$ and $\omega = p_j \forall_j$. However, Lawler [19] and Lenstra et al. [21] proved $1|\beta|1|\sum \omega C_j$ to be *NP*-complete via the linear arrangement problem, which implies determining the optimal firewall rule order is also *NP*-complete. Note, determining the number of possible permutations has been proven to be *NP*-hard [5].

Theorem $1|\beta|1|\sum \omega C_j$ α *Determining the optimal order of a firewall rule list*

[0027] **Proof** Consider the $1|\beta|1|\sum \omega C_j$ problem. Each of n jobs J_i, $i \in I$, has to be processed without preemption on a single machine that can handle at most one job at a time. For each $i \in I$, let ω be the associated weight. Furthermore, let $G = (V, E)$ be a DAG that represents the precedence order of the jobs J_i. Assume the processing time of each job equals 1 time unit, the weights to be $0 \leq \omega \leq 1$ such that $\sum \omega = 1$, and β, which is G, to be a rule list DAG. In this case, the

IPR2023-01353 Page 00993

optimization criterion $\sum \omega \cdot C_i$ is the same as $\sum p_i \cdot i$, which is given in equation 1. Clearly, the optimal firewall rule ordering problem has a solution if and only if $1|\beta|1|\sum \omega C_i$ has a solution. Therefore, determining the optimal permutation of firewall rules is *NP*-complete.

Exemplary Rule Sorting Algorithm

[0028] Although determining the optimal rule permutation was proven to be *NP*-complete, reducing the average number of comparisons required to process a packet remains an important objective. As previously discussed, a sorting algorithm must maintain the precedence relationships among rules. Of course an exhaustive search is possible if the number of rules is small (generate and test all possible topological orders); however as proven in the previous section, this is not feasible with a realistic rule list.

[0029] One exemplary algorithm starts with the original the rule set, then sorts neighboring rules based on nonincreasing probabilities. However, an exchange of neighbors should never occur if the rules intersect and have different actions. This test preserves any precedence relationships in the policy. For example, the following sorting algorithm uses such a comparison to determine if neighboring rules should be exchanged. Note a_i denotes the action associated with the ith rule.

```
done = false
while(!done)
    done = true
    for(i = 1; i <n; i++)
        if(p_i < p_i+1 AND (r_i ∩ r_i+1 OR a_i == a_i+1))then
            exchange rules, actions, and probabilities
            done = false
        endif
    endfor
endwhile
```

[0030] Other sorting algorithms are possible if the completion time of the sort is an issue; however, the method presented is easy to implement and only requires a simple neighbor comparison. Assume the match probabilities for the rule list given in Table 1. Applying the sorting algorithm to this rule list results in the ordering depicted in Figure 2A, which has 10% fewer comparisons on average. However when using the algorithm, it is possible that one rule can prevent another rule from being reordered. For example, rule r_1 prevents rule r_5 from being placed closer to the beginning of the rule set. However, if both rules r_1 and r_5 are placed closer to the beginning of the policy while maintaining their relative order, the average number of comparison will be further reduced, 15% fewer. This is the optimal order for the 6 rule set, which is depicted in Figure 2B. Although this simple sorting algorithm is unable to move groups of rules, it can still improve the performance of the firewall system. Its effectiveness is measured experimentally in the next section.

Experimental Results

[0031] In this section, the average number of rule comparisons required after sorting the rules is compared with average number of comparisons required using the original order and the optimal order. Note the optimal rule ordering was determined via an exhaustive search. As a result, it was not feasible to determine the optimal ordering once the number of rules equaled 15 given the number of permutations to consider. Two different variations of the rule sorting method described in the preceding section were implemented. The difference between the methods was the comparison used to determine if neighboring rules should be exchanged. The first sorting algorithm exchanged neighboring rules if they were out of order and did not intersect. The if-condition was as follows.

if ($p_i < p_{i+1}$ **AND** $r_i \cap r_{i+1}$) **then**

Therefore, rule actions were not considered and the method will be referred to as *non-action sort.* The second sorting method did consider the rule action (as described in the preceding section) and will be referred to as *action sort.* The comparison between the two sorting algorithms will indicate the importance of considering rule actions when ordering rules.

IPR2023-01353 Page 00994

[0032] In the first experiment, lists of 10 firewall rules were generated with random precedence relationships. The match probability of each rule was given by a Zipf distribution [20], which assigns probabilities according to the rank of an item. For this simulation, the last rule had the highest probability which is consistent with most policies (last rules are more general). As a result, the original order yields the worst average number of comparisons. The intersection percentage measured the percentage of rules that intersect in the policy. This metric gives a value for the dependency level in the policy; however note, rule action is not considered when calculating this metric. Rules were equally likely to have an accept or deny action and the results (average number of rule comparisons) for a particular intersection percentage were averaged over 1000 simulations.

[0033] Results of the first experiment are given in Figures 3A and 3B. More particularly, Figure 3A is a graph illustrating the average number of packet comparisons versus intersection percentage for the original rule sorting, the non-action sort, the action sort, and the optimal sort. Figure 3B is a graph illustrating the percent difference in the average number of comparisons required between the sorted and optimal firewall policy configurations. The average number of comparisons required was lower for the sorted and the optimal lists when the intersection percentage was low, as seen in Figure 3A. This was expected since there is a large number of possible rule permutations (few edges in the DAG). When the intersection percentage approached 100%, the values converged to the number required for the original list. This is due to the limited number of possible rule orders, only one order in the extreme case. The percent difference between the sorted and optimal order is shown in Figure 3B. At zero intersection percentage the sorted and optimal orders are equal, since any ordering is possible. Similarly at an intersection percentage of 100, the sorted and optimal orders are equal since only one order is possible. Between these two extremes, the action sorting algorithm remains close to the optimal value. In contrast, the non-action sort had a maximum difference of 33%.

[0034] The benefit of rule sorting on larger policies is also of interest; however as previously described, it was not possible to determine the optimal ordering once the number of rules approaches 15. Therefore, the second experiment used larger rule sets, but only compared the original order and the sorted orders. The number of rules ranged from 10 to 1000, while the matching probabilities and intersection percentages were the same as the previous experiment. The results of this experiment are depicted in Figure 4A. The sorted rule sets always performed equal to or better than the original order, while the action sort consistently performed better than the non-action sort. As noted in the previous experiment, the percent difference is very large given few intersections (e.g. 80% decrease for 1000 rules with 0% dependency), but approaches zero as all the rules intersect. As the number of rules increases, sorting is increasingly beneficial, although only at low intersection percentages. The benefit of sorting drops more significantly as the intersection percentage increases with larger rule sets. This is primarily due to the low matching probabilities of each rule, which requires a complete reordering to have a significant impact on the average number of comparisons. As a result, large rule sets can benefit from sorting if the intersection percentage is low.

[0035] The impact of considering rule actions when sorting is illustrated in Figure 4B. In this experiment lists of 1000 rules were generated, where the intersection percentages ranged from 0% to 100% and the percentage of rules with the same action varied from 50% to 100%. The performance of the action sort to the original list was then compared. As the percentage of rules with the same action increased, the percent difference (reduction) in the average number of comparisons also increased. This is due to the increased number of permutations possible when the rule actions are increasingly the same (fewer edges in the rule list DAG to consider). This is depicted in Figure 4B where a policy with a 100% intersection percentage can significantly reduce the number of comparisons (80% reduction) if all the rules have the same action. This performance increase is not possible with the non-action sort. Therefore, considering the rule action increases the number of possible rule orders, thereby providing more possibilities to improve firewall performance.

Conclusions

[0036] Network firewalls enforce a security policy by comparing arriving packets to a list of rules. An action, such as accept or deny, is then performed on the packet based on the matching rule. Unfortunately packet filtering can impose significant delays on traffic due to the complexity and size of rule sets. Therefore, improving firewall performance is important, given network Quality of Service (QoS) requirements and increasing network traffic loads.

[0037] The sections above describe rule ordering methods to improve the performance of network firewalls. Assuming each rule has a probability of a packet matching, firewall rules should be sorted such that the matching probabilities are non-increasing. This reduces the average number of comparisons and the delay across the firewall. However, a simple sort is not possible given precedence relations across rules. It is common in a security policy that two rules may match the same packet yet have different actions. It is this precedence relationship between rules that must be maintained to preserve integrity. The method described above uses Directed Acyclical Graphs (DAG) to represent the precedence order rules must maintain. Given this representation, a topological sort can be used to determine the optimal order (minimum average number of comparisons); however, the examples above prove this problem to be *NP*-complete (similar to job scheduling for a single nonpreemptive machine with precedence constraints). As an alternative, a simple sorting method was introduced that maintained the precedence order of the rules. Simulation results indicate this method can

IPR2023-01353 Page 00995

significantly reduce the average number of comparisons required and is comparable to optimal ordering.

[0038] Several areas exist for future research in optimizing firewall rule lists. The sorting method proposed above is based on a simple algorithm. Although it can offer an improvement over the original rule order, algorithms that can move groups of rules could provide a larger reduction in the average number of comparisons. The effect of stateful firewalls should also be addressed in future research. Security can be enhanced with connection state and packet audit information. For example, a table can be used to record the state of each connection, which is useful for preventing certain types of attacks (e.g., TCP SYN flood) [30, 31]. The impact of such rules on the firewall needs to be investigated and whether sorting can be done on-line to reflect temporal changes. In addition, more research is needed to determine more accurate probability distributions for packet matching and dependency percentages. Given this information, better algorithms can be designed and more realistic simulations can be performed.

Trie-Based Network Firewall Optimization Methods and Rule Splitting Overview

[0039] As described above, while assuring firewall policy integrity is critical, performance is equally important given increasing network speeds and traffic volumes. Firewall processing delay can be reduced via hardware as well as policy optimization, where rules are manipulated to reduce the number of comparisons. For a given firewall installation, it can be determined that certain security rules have a higher probability of matching a packet than others. As a result, it is possible to develop a *policy profile* over time that indicates frequency of rule matches (similar to cache hit ratio). Given this information, trie-based methods are described herein to minimize the average number of comparisons required per packet. Although most firewall systems still utilize an ordered list representation, the proposed enhancements still are applicable to current firewall systems since a policy trie can be converted into an ordered list using an inorder traversal.

Rule Sorting

[0040] As described above, rule sorting of neighboring rules based on ascending probabilities in a manner that considers rules intersection and rule actions can be used to improve firewall performance by reducing the number of packet comparisons. The methods described above use Directed Acyclic Goal Graphs (DAGs) to represent firewalls. The methods described in this section will use the rule sorting algorithms in combination with policy trie representations to perform sorting of groups of rules in a manner that improves firewall performance.

[0041] In this example, it is assumed that a security policy contains a list of n ordered rules, $\{r_1, r_2, ..., r_n\}$. In addition, let p_i represent the probability that a packet will match rule i (first match in the policy). Therefore, the policy is comprehensive, which means every packet will have a match, if $\sum_{i=1}^{n} p_i = 1$. Using this information, the average number of rule comparisons required can be calculated as

$$E[n] = \sum_{i=1}^{n} i \cdot p_i$$

The average number of comparisons is minimized if the rules are sorted in non-ascending order according to the probabilities. However, this can only be achieved if the sorting algorithm considers the probabilities and whether one rule is a superset of another. For example, sorting algorithms must use the following comparison to determine if neighboring rules should be interchanged Note, a_i denotes the action associated with the ith rule.

if ($p_i < p_{i+1}$ **AND** $r_i \not\subset r_{i+1}$ **OR** $a_i == a_{i+1}$)) **then**

[0042] Sorting rules in this fashion can have positive impact on the average number of comparisons required. Figure 5A is a table illustrating a firewall policy prior to rule sorting. The expected number of comparisons for the policy of Figure 5A is 4.26 comparisons per packet. Figure 5B is a table illustrating the firewall policy of Figure 5A after sorting using the above-described algorithm. In Figure 5B, the expected number of comparisons per packet has been reduced to 3.94. Thus, using the method described above, the number of comparisons per packet can be reduced by 8%.

[0043] However, it can be seen from Figures 5A and 5B that one rule can prevent another rule from being reordered. For example in Figure 5B, rule r_1 prevents rule r_4 from being placed closer to the beginning of the rule set. However, if both rules r_1 and r_4 are placed closer to the beginning of the policy while maintaining their relative order, the average number of comparison will be reduced.

[0044] To solve this problem, rule sets can be sorted using policy tries. First, the rule list is converted into an equivalent

IPR2023-01353 Page 00996

policy trie. Each sub-trie will have an associated probability $\overline{p}$ that is average probability (hit-ratio) of the rules comprising the sub-trie. Sub-tries can be rotated around their parent node to increase performance, using the method described earlier. Since the policy trie (or equivalent rule set) is tested from left to right, rotation should occur if the probability of the right sub-trie is greater than the left sub-trie. As a result, the average number of comparisons required will be reduced and the policy integrity is maintained. Figure 6A is a trie representation of the rules in Figure 5B. Figure 6B is a table representing resulting rule set after rotating the TCP and UDP sub-tries. The TCP sub-trie in Figure 6A has an expected packet matching probability of 0.095 while the UDP sub-trie has an expected packet matching probability of 0.155. Thus, the UDP sub-trie and the TCP sub-trie should be rotated so that the UDP sub-trie becomes the left sub-trie and the TCP sub-trie becomes the right sub-trie. In the table illustrated in Figure 6B, which lists the ordering of the rules after rotating the sub-tries, the average number of comparisons per packets has been reduced to 3.70.

Rule Splitting

[0045] Rule splitting takes a general rule and creates more specific rules that collectively perform the same action over the same set of packets. Here, rule splitting is used to reduce the average number of comparisons. For example in Figure 7A, rule r_5 is split into two separate rules, r_{5a} for UDP and r_{5b} for TCP. Figure 7B is a table corresponding to the policy trie of Figure 7A. Once the rules are positioned based on their probabilities and their relation to other rules, the average number of rule comparisons is reduced to 2.98 (after sort, trie rotation, and splitting) which is a 30% less.

[0046] It many not be advantageous to split a general rule since it adds another rule to the policy. For example, assume a policy contains 20 rules where the first 19 rules have the same probability. Assume the last rule can be split and the new specific rule has a probability that is $\frac{3}{4}$ of the last rule. The impact of the probability of the last rule and the location of the new split rule, *m,* is depicted in Figures 8A and 8B. More particularly, Figure 8A is a graph illustrating the average number of comparisons for firewalls with and without rule splitting. Figure 8B is a graph illustrating average number of comparisons versus different locations of the new rule and probabilities. The average number of comparisons is reduced as the split rule is located closer to the first rule (surface decreases as *m* approaches one). Furthermore, splitting yields better results when the general rule has a high probability. However as illustrated in Figure 8B, the closer the specific rule is to the location of the original rule the average number of comparisons increases, which is the penalty of adding one more rule to the policy.

[0047] The effect of splitting a single rule can be described mathematically as follows. Consider *n* rules where rule r_n can be split into rules $r_{n,1}$ and $r_{n,r}$ (whose union is the original rule). In addition, assume the split rule $r_{n,1}$ will be located at the mth position ($1 \leq m < n$), while rule $r_{n,r}$ will remain at the nth location. The goal is to determine the best location *m*, which yields a lower average number of comparisons as compared to the original rule set. This can be defined mathematically in the following formula.

$$\sum_{i=1}^{n} i \cdot p_i > \sum_{i=1}^{m-1} i \cdot p_i + m \cdot p_{n,l} + \sum_{i=m}^{n-1} (i+1) \cdot p_i + (n+1) \cdot p_{n,r}$$

The left side of the inequality is the average number of comparisons for the original rule set. The right hand side of the inequality is the average number of comparisons with the specific rule at location *m.* If it is assumed that the rules located between *m* and *n* have an equal probability (denoted as *p*) the previous equation can be solved for *m*.

$$m < \frac{n \cdot p - n \cdot p_n + (n+1) \cdot p_{n,r}}{p - p_{n,l}}$$

The new rule must be located between the first and m^{th}; however, its final location will depend on the relationship with the other rules (cannot be placed before any rule for which it is a superset). This result can be applied iteratively to multiple rules and repeatedly to the same rule.

Rule Optimization for QoS

[0048] In the preceding sections, optimizing the entire security policy has been discussed. However, it may be desirable to optimize the rules for a certain type of traffic, in order to reduce the average number of comparisons this traffic

IPR2023-01353 Page 00997

encounters. This can be done by organizing the policy trie via traffic classification, then optimizing the sub-tries. The result is an ordering where the rules for the most important traffic are tested first.

Network Firewall Optimization Using Rule Compression

[0049] Another method for network firewall optimization is referred to as rule compression. Given a security policy R, one objective in optimizing firewall performance is to reduce number of rules in the policy while maintaining the original integrity. A method for reducing the number of rules called *rule compression*, which combines and replaces two rules in the policy will now be described. Although compression does reduce the number of rules, it can only be done if the integrity of the original policy is maintained.

[0050] Consider two rules r_i and r_j in policy R, where r_i appears before r_j. The two rules can be compressed if and only if the following conditions are true

1. The two rules intersect, $r_i \cap r_j \neq 0$ (an edge between r_i and r_j exists in the policy DAG) (Tuples that are proper subsets and supersets are considered. Compression may still occur if the rules do not intersect or are adjacent in the set space, the resulting rule may contain ranges.)
2. The action of r_i and r_j is the same.
3. Rule r_i is not dependent on any rule r_k that is directly or indirectly dependent on r_j (only one path from r_i to r_j exists in the policy DAG).

[0051] The result of compression is a new rule, $r_{i,j}$ whose tuples are the union of the corresponding tuples of rules r_i and r_j. The new rule $r_{i,j}$ replaces both r_i and r_j in R; however, the location of $r_{i,j}$ in R may require the relocation of other rules. Let D_j be the ordered set of rules that appear after r_i that directly or indirectly depend upon r_j. The new rule is placed at the original location of r_i in R and the rules D_j are placed before $r_{i,j}$.

[0052] For example, consider the policy given in Table 2. It is possible to compress rules r_1 and r_5, creating the new rule $r_{1,5}$ =[TCP, *, *, 2.*, *, accept]. The ordered set D_j in this example consists of the rules r_3 and r_4. As a result, the new rule $r_{1,5}$ is placed at the original location of r_1 while the rules in D_j are placed before the new rule. Figures 9A and 9B respectively illustrate the original policy DAG for Table 2 and the policy DAG after rules r_1 and r_5 are compressed.

Table 2: Example Policy Consisting of Multiple Ordered Rules

		Source		Destination		
No.	Proto.	IP	Port	IP	Port	Action
1	TCP	1.1.*	*	2.2.*	*	accept
2	TCP	1.*	*	*	80	deny
3	TCP	3.3.*	*	*	80	accept
4	TCP	3.*	*	*	*	deny
5	TCP	*	*	2.*	90	accept

[0053] **Theorem** *Compressing rules r_i and r_j of policy R will maintain integrity if the three conditions are met and method described is used.*

[0054] **Proof** Before r_i and r_j are compressed, relocate the rules D_j immediately before r_i and D_i. This relocation will maintain integrity due to the third requirement for compression (no rule in D_i can be dependent on any rule in D_j). Now place r_j directly after r_i. This relocation will not affect integrity since no rule in D_i can have an edge to r_j (again, due to the third condition). Compressing the neighboring rules r_i and r_j creates $r_{i,j}$. This does not affect integrity since the resulting rule only matches the packets that match r_i or r_j in the original policy. Therefore, the result is the compression of rule r_i and r_j, and the integrity of the policy is maintained.

Determining if Compression Should Occur

[0055] Although compression may be possible, it may not improve the performance of the policy due to the relocation of rules. Compression should only occur if the average number of comparisons required for the new policy R_c is less than the original policy R; therefore, $E[R_c] < E[R]$.

IPR2023-01353 Page 00998

Firewall Rule Sub-Trie and Sub-List Ordering

Policy Sub-Trie Ordering

[0056] Additional methods for firewall optimization are referred to as firewall sub-trie and sub-list ordering. Given a policy trie T, it may be desirable to order all sub-tries such that the average number of tuple comparisons is reduced and policy integrity is maintained. Consider a policy trie T that contains sub-tries T_i and T_j having the same parent node, as seen in Figures 10A-10C. Let P_i denote the sum of the probabilities of the rules contained in sub-trie i, while C_i denotes the number of comparisons required to completely traverse sub-trie i (which is equal to the number of branches). In order to reduce the average number of tuple comparisons, sub-tries that share the same parent node should be ordered such that the following two conditions are observed.

1. Sub-tries that share a parent node are ordered such that the P_i values are non-ascending (higher match probabilities occur first, from left to right).
2. If sub-tries that share a parent node have the same probability (P_i equals P_j), then order the sub-tries such that the C_i values are non-descending (sub-tries consisting of fewer comparisons occur first, from left to right).

These conditions are recursively applied throughout the policy trie as seen in Figures 11A-11D. However, these conditions do not necessarily maintain policy integrity.

[0057] Consider a group of n sub-tries that share the same parent node that are numbered sequentially from left to right. Consider any two non-intersecting sub-tries T_i and T_{i+k} in this group that are out of order. If two sub-tries do not intersect, denoted as $T_i \pitchfork T_j$, then no rule in one sub-trie will intersect with any rule in the other sub-trie, $r_k \pitchfork r_l$, $\forall r_k \in T_i$, $\forall r_l \in T_j$. The following *integrity condition* must be observed when reordering the sub-tries. The two sub-tries can be exchanged (rotated about the parent) if and only the sub-tries T_i through T_{i+k-1} do not intersect with T_{i+k} and the sub-tries T_{i+1} through T_{i+k} do not intersect with T_i. Similar to finding the linear sequence of a policy DAGs, these conditions maintain the policy trie integrity. This is observed in Figure 10A, where no edge exists between any rule in T_1 and T_2. More particularly, Figure 10A is a policy DAG of a six-rule firewall list. Figure 10B is a policy trie representation of the firewall rule list represented by Figure 10A. In Figure 10B, the average number of tuple compares required per packet is 5.5. Figure 10C illustrates the policy trie after rotating sub-tries T_1 and T_2. In Figure 10C, the average number of tuple compares required is 4.5. In Figures 10A-10C, the following probabilities are assumed:

$P_1 = 0.5$
$C_1 = 4$
$P_2 = 0.5$; and
$C_2 = 2$.

The average number of tuple comparisons is an estimate based on the longest trie path.

[0058] Using the conditions for comparing and ordering, the following sorting algorithm sorts neighboring sub-tries that share the same parent node.

```
m = a parent node in T
done = false
while(!done)
    done = true
    for each sub-trie Ti having parent m AND a right neighboring sub-trie
        if((Pi < Pi+1 AND (Pi == Pi+1 AND Ci > Ci+1)) AND Ti ⋔ Ti+1) then
        interchange Ti and Ti+1
        done = false
        endif
    endfor
endwhile
```


IPR2023-01353 Page 00999

[0059] Figures 11A-11D illustrate policy trie rotation for an eight-rule firewall policy. More particularly, Figure 11A illustrates the policy DAG representation of an eight-rule firewall policy. Figure 11B is a policy trie representation of the original firewall policy. In this example, ordering occurs in three stages, starting at the bottom level and moving towards the top level of the trie. In Figure 11C, the leaves are ordered at the lowest level. More particularly, nodes r_1 and r_2 are exchanged with nodes r_7 and r_8. The average number of tuple compares for the ordering represented by Figure 11C is 10.45. In Figure 11D, the sub-tries on the second lowest level (T_3, T_4, T_5, and T_6) are ordered. More particularly, Figure 11 D illustrates the policy trie after exchanging sub-tries T_3 and T_4. The average number of tuple compares for the representation in Figure 11 D is 10.275. In Figure 11 E, the sub-tries on the second highest level, (T_1 and T_2) are ordered. Figure 11E illustrates the final ordering after exchanging T_1 and T_2. In the policy trie of Figure 11E, the average number of tuple compares required is 6.775. The average number of tuple comparisons is an estimate based on the longest trie path.

Policy Sub-List Ordering

[0060] The preceding section describes conditions for sorting policy sub-tries, which allow the exchange of groups of rules (sub-tries). The same conditions can be applied to list based policies, where sub-lists are ordered to improve the average number of rule comparisons.

[0061] In the sections above relating to rule sorting, a method is described to exchange neighboring rules in a list-based security policy, as seen in Figure 12B. However, it was observed that when ordering rules in this fashion it is possible that one rule can block the exchange of another. If groups of rules were allowed to be exchanged the average number of rule comparisons could be further reduced.

[0062] Given a policy list L, it may be desirable to order all sub-lists such that the average number of rule comparisons is reduced and policy integrity is maintained. Consider a policy list L that contains sub-tries L_i and L_j. Let P_i denote the sum of the probabilities of the rules contained in sub-list i, while C_i denotes the number of rules in sub-list i. In order to reduce the average number of rule comparisons, sub-lists should be ordered such that the following two conditions are observed.

1. Sub-lists are ordered such that the P_i values are non-ascending (higher match probabilities occur first, from left to right).
2. If sub-lists have the same probability (P_i equals P_j), then order the sub-lists such that the C_i values are non-descending (sub-lists consisting of fewer comparisons occur first, from left to right).

These conditions are applied throughout the policy list; however as with policy tries, these conditions do not necessarily maintain policy integrity.

[0063] Consider a group of n sub-lists that are numbered sequentially from left to right. Consider any two non-intersecting sub-lists L_i and L_{i+k} in this group that are out of order. If two sub-lists do not intersect, denoted as $L_i \pitchfork L_j$ then no rule in one sub-list will intersect with any rule in the other sub-list, $r_k \pitchfork r_l, \forall r_k \in L_i, \forall r_l \in L_j$. The following *integrity condition* must be observed when reordering the sub-lists. The two sub-lists can be exchanged if and only if the sub-lists L_i through L_{i+k-1} do not intersect with any rule in L_{i+k} and sub-lists L_{i+1} through L_{i+k} do not intersect with L_i. Similar to finding the linear sequence of a policy DAGs, these conditions maintain the policy trie integrity. For example, consider the sub-lists $L_1 = \{r_1, r_3\}$ and $L_2 = \{r_2, r_4\}$ given in Figure 12B. No edge exists between any rule in L_1 and L_2. The two sub-lists are out of order with respect to P and should be exchanged. The resulting list is given in Figure 12C. Figure 12A illustrates the original policy order and results in an average number of rule comparisons of 3.4. In the sorted version in Figure 12B, the average number of rule comparisons is 3.3. In Figure 12C, the average number of rule comparisons has been reduced to 2.7. Using the conditions for comparing and ordering, the following sorting algorithm sorts neighboring sub-lists.

IPR2023-01353 Page 01000

```
done = false
while(!done)
    done = true
    for each sub-list Li having a right neighboring sub-list
        if ((Pi < Pi+1 OR (Pi == Pi+1 AND Ci > Ci+1)) AND Li ⌒ Li+1)then
            interchange rules and probabilities
            done = false
        endif
    endfor
endwhile
```

Summary of Firewall Policy Optimization Techniques and Methods

[0064] As stated above, the methods and systems described herein for a network firewall policy optimization can be implemented using any combination of hardware, software, and/or firmware. Alternatively, the steps for firewall policy optimization can be performed manually with the number of rules being optimized is small enough. Figure 13 is a flow chart illustrating an exemplary process for firewall policy optimization according to an embodiment of the subject matter described herein. Referring to Figure 13, in block **1300**, a firewall policy including an ordered list of firewall rules is defined. In block **1302**, for each rule, a probability indicating a likelihood of receiving a packet that matches each rule is specified. In block **1304**, neighboring rules are sorted in order of non-increasing probability in a manner that preserves firewall policy. In block **1306**, groups of rules are sorted in order of non-increasing probability in a manner that preserves firewall policy. For example, any of the policy-trie-based methods described above may be used. In block **1308**, one or more rules are split to reduce the number of average packet comparisons. In block **1310**, intersecting rules having common actions are identified and collapsed into single rules.

[0065] As described above, the subject matter described herein may be implemented in hardware, software, and/or firmware. In one implementation, the subject matter described herein for firewall policy optimization may be implemented on a general purpose computing platform. Figure 14 is a block diagram of a general purpose computing platform including software for firewall policy optimization according to an embodiment of the subject matter described herein. Referring to Figure 14, computing platform **1400** may be a general purpose computer, such as a personal computer, that includes a microprocessor **1402**, memory **1404**, and I/O interfaces **1406**. Microprocessor **1400** may be any suitable microprocessor, such as any of the Intel or AMD families of microprocessors. Memory **1404** may include volatile memory for running programs and persistent memory, such as one or more disk storage devices. I/O interface **1406** may include interfaces with I/O devices, such as user input devices and output devices.

[0066] In the illustrated example, software that may be resident in memory 1404 includes a firewall policy editor 1408 and a firewall policy optimizer 1410. Firewall policy editor 1408 may allow a user to define a firewall policy. For example, firewall policy editor 1408 may allow a user to define a firewall policy by specifying different rules. The rules may be specified in a graphical manner, for example using policy DAGs as described above. Alternatively, firewall policy editor 1408 may allow a user to input rules in a tabular manner, as illustrated in any of the tables described herein. Firewall policy optimizer 1410 may implement any or all of the firewall policy optimization techniques to order rules entered via firewall policy editor 1408 in a manner that preserves policy integrity and that enhances firewall performance.

[0067] A firewall rule set that is optimized using the subject matter described herein may be implemented on any firewall system that includes one or more firewalls. For example, an optimized firewall rule set according to embodiments of the subject matter described herein may be implemented using any of the hierarchical, multi-node firewall systems described in commonly assigned, co- pending U.S. patent application no. 11/316,331, filed December 22, 2005. Figure 15 illustrates an example of a multi-node firewall system suitable for implementing the firewall rules according to an embodiment of the subject matter described herein. Referring to Figure 15, a plurality of firewall nodes 1500, 1502, 1504, and 1506 may collectively implement any of the optimized firewall policies described herein. For example, the firewall nodes may collectively implement different portions of a firewall policy data structure including an ordered list of firewall rules. A firewall policy engine 1508 resident on each firewall node may filter packets using its local portion 1510

IPR2023-01353 Page 01001

of the firewall policy data structure. A control node 1512 may include a firewall policy optimizer 1514 that measures the average number of comparisons per packets for the current rule configuration and may dynamically reorder the rules to reduce the average number of comparisons per packet. For example, firewall policy optimizer 1514 may utilize any of the methods described herein to rearrange rules and improve firewall performance. Rules may be tested for rearrangement at user-specified intervals or when average number of packet comparisons increases by a user-specified amount.

References

[0068] The disclosure of each of the following references is hereby incorporated herein by reference in its entirety.

[1] E. Al-Shaer and H. Hamed. Firewall Policy Management Advisor for Anomaly Detection and Rule Editing. In Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, 2003.
[2] E. Al-Shaer and H. Hamed. Modeling and Management of Firewall Policies. IEEE Transactions on Network and Service Management, (1), 2004.
[3] S. M. Bellovin and W. Cheswick. Network Firewalls. IEEE Communications Magazine, pages 50-57, Sept. 1994.
[4] C. Benecke. A Parallel Packet Screen for High Speed Networks. In Proceedings of the 15th Annual Computer Security Applications Conference, 1999.
[5] G. Brightwell and P. Winkler. Counting Linear Extensions is #P-Complete. In Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing, 1991.
[6] M. Christiansen and E. Fleury. Using Interval Decision Diagrams for Packet Filtering. Technical report, BRICS, 2002.
[7] D. Decasper, Z. Dittia, G. Parulkar, and B. Plattner. Router Plugins: A Software Architecture for Next-Generation. Routers. IEEE/ACM Transactions on Networking, 8(1), February 2000.
[8] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink. Small Forwarding Tables for Fast Routing Lookups. In Proceedings of ACM SIGCOMM, pages 4-13, 1997.
[9] U. Ellermann and C. Benecke. Firewalls for ATM Networks. In Proceedings of INFOSEC'COM, 1998.
[10] A. Feldmann and S. Muthukrishnan. Tradeoffs for Packet Classification. In Proceedings of the IEEE INFOCOM, pages 397 - 413, 2000.
[11] E. W. Fulp. Firewall Architectures for High Speed Networks. Technical Report 20026, Wake Forest University Computer Science Department, 2002.
[12] E. W. Fulp. Firewall Policy Models Using Ordered-Sets. Technical report, Wake Forest University Computer Science Department, 2004.
[13] R. Funke, A. Grote, and H.-U. Heiss. Performance Evaluation of Firewalls in Gigabit-Networks. In Proceedings of the Symposium on Performance Evaluation of Computer and Telecommunication Systems, 1999.
[14] S. Goddard, R. Kieckhafer, and Y. Zhang. An Unavailability Analysis of Firewall Sandwich Configurations. In Proceedings of the 6th IEEE Symposium on High Assurance Systems Engineering, 2001.
[15] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan. Optimizing and Approximation in Determinstic Sequencing and Scheduling: A Survey. Annals of Discrete Mathematics, 5:287 - 326, 1979.
[16] A. Hari, S. Suri, and G. Parulkar. Detecting and Resolving Packet Filter Conflicts. In Proceedings of IEEE INFOCOM, pages 1203-1212, 2000.
[17] HiPAC. High Performance Packet Classification. http://www.hipac.org.
[18] E. Horowitz, S. Sahni, and D. Mehta. Fundamentals of Data Structures in C++. Compuer Science Press, 1995.
[19] E. L. Lawler. Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints. Annals of Discrete Mathematics, 2:75 - 90, 1978.
[20] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the Self-Similar Nature of Ethernet Traffic. IEEE Transactions on Networking, 2:1-15, 1994.
[21] J. K. Lenstra and A. H. G. R. Kan. Complexity of Scheduling under Precedence Constraints. Operations Research, 26(1):22 - 35, 1978.
[22] O. Paul and M. Laurent. A Full Bandwidth ATM Firewall. In Proceedings of the 6th European Symposium on Research in Computer Security ESORICS'2000, 2000.
[23] B. R. Preiss. Data Structures and Algorithms with Object-Oriented Design Patterns in C++. John Wiley & Sons, 1999.
[24] L. Qui, G. Varghese, and S. Suri. Fast Firewall Implementations for Software and Hardware-Based Routers. In Proceedings of ACM SIGMETRICS, June 2001.
[25] V. P. Ranganath and D. Andresen. A Set-Based Approach to Packet Classification. In Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems, pages 889-894, 2003.
[26] R. Rivest. On Self-Organizing Sequential Search Heuristics. Communications of the ACM, 19(2), 1976.
[27] W. E. Smith. Various Optimizers for Single-Stage Production. Naval Research Logistics Quarterly, 3:59 - 66,

IPR2023-01353 Page 01002

1956.
[28] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel. Fast and Scalable Layer Four Switching. In Proceedings of ACM SIGCOMM, pages 191-202, 1998.
[29] S. Suri and G. Varghese. Packet Filtering in High Speed Networks. In Proceedings of the Symposium on Discrete Algorithms, pages 969 - 970, 1999.
[30] R. L. Ziegler. Linux Firewalls. New Riders, second edition, 2002.
[31] E. D. Zwicky, S. Cooper, and D. B. Chapman. Building Internet Firewalls. O'Reilly, 2000.

[0069] It will be understood that various details of the invention may be changed without departing from the scope of the invention. Furthermore, the foregoing description is for the purpose of illustration only, and not for the purpose of limitation.

Claims

1. A method for producing a performance optimized firewall policy, the method comprising:

   (a) defining a firewall policy including an ordered list of firewall rules;
   (b) for each rule, defining a probability indicating the likelihood of receiving a packet matching each rule; and
   (c) sorting neighboring rules in order of non-increasing probability in a manner that preserves the firewall policy;

   the method **characterized in that**

   (d) sorting groups of two or more rules in order of non-increasing probability in a manner that preserves the firewall policy, wherein at least one of the groups of two or more rules includes a first rule that prevents a second rule in the group from being moved, within the group, closer to a beginning of the ordered list that implements the firewall policy and wherein sorting the groups of rules includes moving the first and second rules closer to the beginning of the ordered list while maintaining their relative order.

2. The method of claim 1 wherein sorting neighboring rules in order of non-increasing probability in a manner that preserves the firewall policy includes, for each pair of neighboring rules, rearranging the rules if a first rule of the pair has a lower probability than a second rule of the pair and the first rule of the pair is not a subset of the second rule of the pair with a different action.

3. The method of claim 1, wherein sorting groups of two or more rules in a manner that preserves firewall policy includes assigning each group of rules to a sub-trie in a policy trie, computing a sum of packet matching probabilities for rules in each sub-trie, and, in response to determining that a first sub-trie has a lower sum than a second sub-trie, rotating the first and second sub-tries about a common parent so that rules in the second sub-trie will be applied before the rules in the first sub-trie in the firewall policy.

4. The method of claim 3, comprising, in response to determining that the first and second sub-tries have equal sums, determining whether the second sub-trie has an lower number of branches than the first sub-trie, and, in response to determining that the second sub-trie has a lower number of branches than the first sub-trie, rotating the first and second sub-tries about a common parent so that the rules in the second sub-trie will be applied before the rules in the first sub-trie in the firewall policy.

5. The method of claim 1 wherein sorting groups of two or more rules in a manner that preserves the firewall policy includes:

   (e) identifying first and second sub-lists of rules in the ordered list of firewall rules, the first sub-list preceding the second sub-list in the ordered list of firewall rules;
   (f) determining a sum of packet matching probabilities for each of the first and second sub-lists;
   (g) determining whether the first and second sub-lists intersect; and
   (h) in response to determining that the sum of the packet matching probabilities for the second sub-list is greater than the sum of the packet matching probabilities for the first sub-list and that the first and second sub-lists do not intersect, switching the order of the first and second sub-lists in the ordered list of firewall rules.

6. The method of claim 5, comprising, in response to determining that the sums of the packet matching probabilities

IPR2023-01353 Page 01003

for the first and sub-lists are equal, determining whether the second sub-list has a lower number of rules than the first sub-list, and, in response to determining that the second sub-list has a lower number of rules than the first sub-list, switching the order of the first and second sub-lists in the ordered list of firewall rules.

7. The method of claim 1, comprising splitting at least one of the firewall rules to reduce an average number of comparisons required per received packet.

8. The method of claim 1, comprising identifying intersecting rules having a common action and collapsing the rules into a single rule representing a union of the intersecting rules.

9. The method of claim 1, wherein sorting neighboring rules includes sorting rules such that comparisons for a particular class of packets are reduced.

10. The method of claim 1, comprising assigning the rules to vertices in a directed acyclical graph, DAG, determining precedence relationships between the rules, assigning the precedence relationships to edges that connect the vertices in the DAG, and wherein sorting the neighboring rules in order of non-increasing probability in a manner that preserves the firewall policy includes sorting the neighboring rules in a manner that preserves the precedence relationships represented by the DAG.

11. The method of claim 1, wherein the rules comprise intrusion detection rules.

12. The method of claim 1, wherein the rules comprise intrusion protection rules.

13. A network firewall, comprising:

(a) a firewall policy data structure (1510) for storing an ordered list of firewall rules;
(b) a firewall policy engine (1508) for implementing a firewall policy by comparing packets to the rules in the order specified by the firewall policy data structure (1510); and
(c) a firewall policy optimizer (1514) for optimizing performance of the network firewall by reordering the rules in a manner that reduces an average number of rule comparisons per packet and hat preserves the firewall policy;

the network firewall **characterized in that** the firewall policy optimizer (1514) is adapted to (d) sort groups of two or more rules in order of non-increasing probability in a manner that preserves the firewall policy, wherein at least one of the groups of two or more rules includes a first rule that prevents a second rule in the group from being moved, within the group, closer to a beginning of the ordered list that implements the firewall policy and wherein sorting the groups of rules includes moving the first and second rules closer to the beginning of the ordered list while maintaining their relative order.

14. The network firewall of claim 13, wherein the firewall policy optimizer (1514) is adapted to measure average number of comparisons per packet for a rule ordering specified by the firewall policy data structure (1510) and to dynamically re-order to rules to reduce the average number of comparisons per packet.

Patentansprüche

1. Verfahren zum Generieren einer leistungsoptimierten Firewall-Vorschrift, wobei das Verfahren Folgendes umfasst:

(a) Definieren einer Firewall-Vorschrift, die eine geordnete Liste mit Firewall-Regeln umfasst;
(b) für jede Regel, Definieren einer Wahrscheinlichkeit, welche die Wahrscheinlichkeit des Empfangens eines mit jeder Regel übereinstimmenden Paketes anzeigt; und
(c) Sortieren benachbarter Regeln in der Reihenfolge einer nicht-zunehmenden Wahrscheinlichkeit in einer Weise, welche die Firewall-Vorschrift wahrt;

wobei das Verfahren **gekennzeichnet ist durch**:

(d) Sortieren von Gruppen von zwei oder mehr Regeln in der Reihenfolge einer nicht-zunehmenden Wahrscheinlichkeit in einer Weise, welche die Firewall-Vorschrift wahrt, wobei mindestens eine der Gruppen von zwei oder mehr Regeln eine erste Regel umfasst, die verhindert, dass eine zweite Regel in der Gruppe innerhalb

IPR2023-01353 Page 01004

der Gruppe näher zu einem Anfang der geordneten Liste, welche die Firewall-Vorschrift implementiert, verschoben wird, und wobei das Sortieren der Gruppen von Regeln umfasst, die erste und die zweite Regel näher an den Anfang der geordneten Liste zu bewegen, während ihre relative Reihenfolge beibehalten wird.

2. Verfahren nach Anspruch 1, wobei das Sortieren benachbarter Regeln in der Reihenfolge einer nicht-zunehmenden Wahrscheinlichkeit in einer Weise, welche die Firewall-Vorschrift wahrt, für jedes Paar benachbarter Regeln umfasst, die Regeln umzuordnen, wenn eine erste Regel das Paares einer geringere Wahrscheinlichkeit als eine zweite Regel das Paares hat und die erste Regel das Paares keine Teilmenge der zweiten Regel das Paares mit einer anderen Aktion ist.

3. Verfahren nach Anspruch 1, wobei das Sortieren von Gruppen von zwei oder mehr Regeln in einer Weise, welche die Firewall-Vorschrift wahrt, Folgendes umfasst: Zuordnen jeder Gruppe von Regeln zu einem Sub-Trie in einem Vorschrifts-Trie; Berechnen einer Summe von Wahrscheinlichkeiten der Paketübereinstimmung für Regeln in jedem Sub-Trie; und in Reaktion auf die Feststellung, dass ein erster Sub-Trie eine kleinere Summe als ein zweiter Sub-Trie hat, Drehen des ersten und des zweiten Sub-Tries um einen gemeinsamen Ursprung, so dass Regeln in dem zweiten Sub-Trie vor den Regeln in dem ersten Sub-Trie in der Firewall-Vorschrift angewendet werden.

4. Verfahren nach Anspruch 3, das in Reaktion auf die Feststellung, dass der erste und der zweite Sub-Trie gleiche Summen haben, das Feststellen umfasst, ob der zweite Sub-Trie eine kleinere Anzahl von Abzweigungen als der erste Sub-Trie hat, und in Reaktion auf die Feststellung, dass der zweite Sub-Trie eine kleinere Anzahl von Abzweigungen als der erste Sub-Trie hat, das Drehen des ersten und des zweiten Sub-Tries um einen gemeinsamen Ursprung umfasst, so dass die Regeln in dem zweiten Sub-Trie vor den Regeln in dem ersten Sub-Trie in der Firewall-Vorschrift angewendet werden.

5. Verfahren nach Anspruch 1, wobei das Sortieren von Gruppen von zwei oder mehr Regeln in einer Weise, welche die Firewall-Vorschrift wahrt, Folgendes umfasst:

(e) Identifizieren einer ersten und einer zweiten Teilliste von Regeln in der geordneten Liste mit Firewall-Regeln, wobei die erste Teilliste der zweiten Teilliste in der geordneten Liste mit Firewall-Regeln vorausgeht;
(f) Bestimmen einer Summe von Wahrscheinlichkeiten der Paketübereinstimmung für die erste und für die zweite Teilliste;
(g) Feststellen, ob sich die erste und die zweite Teilliste überschneiden; und
(h) in Reaktion auf die Feststellung, dass die Summe der Wahrscheinlichkeiten der Paketübereinstimmung für die zweite Teilliste größer als die Summe der Wahrscheinlichkeiten der Paketübereinstimmung für die erste Teilliste ist und dass sich die erste und die zweite Teilliste nicht überschneiden, Wechseln der Reihenfolge der ersten und der zweiten Teilliste in der geordneten Liste mit Firewall-Regeln.

6. Verfahren nach Anspruch 5, das, in Reaktion auf die Feststellung, dass die Summen der Wahrscheinlichkeiten der Paketübereinstimmung für die erste und Teilliste gleich sind, das Feststellen umfasst, ob die zweite Teilliste eine kleinere Anzahl von Regeln als die erste Teilliste hat, und in Reaktion auf die Feststellung, dass die zweite Teilliste eine kleinere Anzahl von Regeln als die erste Teilliste hat, das Wechseln der Reihenfolge der ersten und der zweiten Teilliste in der geordneten Liste mit Firewall-Regeln umfasst.

7. Verfahren nach Anspruch 1, das umfasst, mindestens eine der Firewall-Regeln zu teilen, um eine durchschnittliche Anzahl von Vergleichen, die je empfangenem Paket erforderlich sind, zu verringern.

8. Verfahren nach Anspruch 1, das umfasst, sich überschneidende Regeln zu identifizieren, die eine gemeinsame Aktion haben, und die Regeln zu einer einzigen Regel zusammenzulegen, die eine Vereinigung der sich überschneidenden Regeln darstellt.

9. Verfahren nach Anspruch 1, wobei das Sortieren benachbarter Regeln umfasst, die Regeln so zu sortieren, dass Vergleiche für eine bestimmte Klasse von Paketen verringert werden.

10. Verfahren nach Anspruch 1, das Folgendes umfasst: Zuweisen der Regeln zu Scheitelpunkten in einem gerichteten azyklischen Graph (DAG); Bestimmen von Präzedenzbeziehungen zwischen den Regeln; Zuweisen der Präzedenzbeziehungen zu Flanken, welche die Scheitelpunkte in dem DAG verbinden; und wobei das Sortieren der benachbarten Regeln in der Reihenfolge einer nicht-zunehmenden Wahrscheinlichkeit in einer Weise, welche die Firewall-Vorschrift wahrt, umfasst, die benachbarten Regeln in einer Weise zu sortieren, welche die durch den DAG

IPR2023-01353 Page 01005

dargestellten Präzedenzbeziehungen wahrt.

11. Verfahren nach Anspruch 1, wobei die Regeln Eindringdetektionsregeln umfassen.

12. Verfahren nach Anspruch 1, wobei die Regeln Eindringschutzregeln umfassen.

13. Netzwerk-Firewall, die Folgendes umfasst:

(a) eine Firewall-Vorschrifts-Datenstruktur (1510) zum Speichern einer geordneten Liste mit Firewall-Regeln;
(b) eine Firewall-Vorschrifts-Engine (1508) zum Implementieren einer Firewall-Vorschrift durch Vergleichen von Paketen mit den Regeln in der durch die Firewall-Vorschrifts-Datenstruktur (1510) spezifizierten Reihenfolge; und
(c) einen Firewall-Vorschrifts-Optimierer (1514) zum Optimieren der Leistung der Netzwerk-Firewall durch Umordnen der Regeln in einer Weise, die eine durchschnittliche Anzahl von Regelvergleichen je Paket verringert und die Firewall-Vorschrift wahrt;

wobei die Netzwerk-Firewall **dadurch gekennzeichnet ist, dass** der Firewall-Vorschrifts-Optimierer (1514) dafür geeignet ist:

(d) Gruppen von zwei oder mehr Regeln in der Reihenfolge einer nicht-zunehmenden Wahrscheinlichkeit in einer Weise zu sortieren, welche die Firewall-Vorschrift wahrt, wobei mindestens eine der Gruppen von zwei oder mehr Regeln eine erste Regel umfasst, die verhindert, dass eine zweite Regel in der Gruppe innerhalb der Gruppe näher zu einem Anfang der geordneten Liste, welche die Firewall-Vorschrift implementiert, verschoben wird, und wobei das Sortieren der Gruppen von Regeln umfasst, die erste und die zweite Regel näher an den Anfang der geordneten Liste zu bewegen, während ihre relative Reihenfolge beibehalten wird.

14. Netzwerk-Firewall nach Anspruch 13, wobei der Firewall-Vorschrifts-Optimierer (1514) dafür geeignet ist, die durchschnittliche Anzahl von Vergleichen je Paket für eine durch die Firewall-Vorschrifts-Datenstruktur (1510) spezifizierte Regelordnung zu messen und die Regeln dynamisch umzuordnen, um die durchschnittliche Anzahl von Vergleichen je Paket zu verringern.

Revendications

1. Procédé de génération d'une politique de pare-feu à performances optimisées, le procédé consistant à :

(a) définir une politique de pare-feu comprenant une liste ordonnée de règles de pare-feu ;
(b) pour chaque règle, définir une probabilité indiquant la probabilité de recevoir un paquet correspondant à chaque règle ; et
(c) trier des règles voisines dans l'ordre de probabilité non croissante d'une manière qui préserve la politique de pare-feu ;

le procédé étant **caractérisé par** le fait de

(d) trier des groupes de deux règles ou plus dans l'ordre de probabilité non croissante d'une manière qui préserve la politique de pare-feu, dans lequel au moins l'un des groupes de deux règles ou plus comprend une première règle qui empêche une seconde règle du groupe d'être déplacée, au sein du groupe, plus près d'un début de la liste ordonnée qui met en oeuvre la politique de pare-feu, et dans lequel le tri des groupes de règles consiste à déplacer les première et seconde règles plus près du début de la liste ordonnée, tout en maintenant leur ordre relatif.

2. Procédé selon la revendication 1, dans lequel le tri de règles voisines dans l'ordre de probabilité non croissante d'une manière qui préserve la politique de pare-feu consiste, pour chaque paire de règles voisines, à réarranger les règles si une première règle de la paire a une probabilité plus faible qu'une seconde règle de la paire et si la première règle de la paire n'est pas un sous-ensemble de la seconde règle de la paire avec une action différente.

3. Procédé selon la revendication 1, dans lequel le tri de groupes de deux règles ou plus d'une manière qui préserve la politique de pare-feu consiste à affecter chaque groupe de règles à un sous-arbre dans un arbre de politique,

IPR2023-01353 Page 01006

calculer une somme de probabilités de correspondance de paquet pour les règles de chaque sous-arbre, et, en réponse à la détermination que le premier sous-arbre a une somme inférieure à un second sous-arbre, faire tourner les premier et second sous-arbres autour d'un parent commun, de sorte que les règles du second sous-arbre soient appliquées avant les règles du premier sous-arbre dans la politique de pare-feu.

4. Procédé selon la revendication 3, consistant, en réponse à la détermination que les premier et second sous-arbres ont des sommes égales, à déterminer si le second sous-arbre a un plus petit nombre de branches que le premier sous-arbre, et, en réponse à la détermination que le second sous-arbre a un plus petit nombre de branches que le premier sous-arbre, faire tourner les premier et second sous-arbres autour d'un parent commun, de sorte que les règles du second sous-arbre soient appliquées avant les règles du premier sous-arbre dans la politique de pare-feu.

5. Procédé selon la revendication 1, dans lequel le tri de groupes de deux règles ou plus d'une manière qui préserve la politique de pare-feu consiste à :

(e) identifier de première et seconde sous-listes de règles dans la liste ordonnée de règles de pare-feu, la première sous-liste précédant la seconde sous-liste dans la liste ordonnée de règles de pare-feu ;
(f) déterminer une somme de probabilités de correspondance de paquet pour chacune des première et seconde sous-listes ;
(g) déterminer si les première et seconde sous-listes se croisent ; et
(h) en réponse à la détermination que la somme des probabilités de correspondance de paquet pour la seconde sous-liste est supérieure à la somme des probabilités de correspondance de paquet pour la première sous-liste et que les première et seconde sous-listes ne se croisent pas, inverser l'ordre des première et seconde sous-listes dans la liste ordonnée de règles de pare-feu.

6. Procédé selon la revendication 5, consistant, en réponse à la détermination que les sommes des probabilités de correspondance de paquet pour les première et seconde sous-listes sont égales, déterminer si la seconde sous-liste a un plus petit nombre de règles que la première sous-liste, et, en réponse à la détermination que la seconde sous-liste a un plus petit nombre de règles que la première sous-liste, inverser l'ordre des première et seconde sous-listes dans la liste ordonnée de règles de pare-feu.

7. Procédé selon la revendication 1, consistant à diviser au moins l'une des règles de pare-feu afin de réduire un nombre moyen de comparaisons nécessaire par paquet reçu.

8. Procédé selon la revendication 1, consistant à identifier des règles qui se croisent ayant une action commune et à regrouper les règles en une règle unique qui représente une union des règles qui se croisent.

9. Procédé selon la revendication 1, dans lequel le tri de règles voisines consiste à trier des règles de telle sorte que les comparaisons relatives à une catégorie particulière de paquets sont réduites.

10. Procédé selon la revendication 1, consistant à affecter les règles à des sommets dans un graphe acyclique dirigé, DAG, déterminer des relations de priorité entre les règles, affecter les relations de priorité à des bords qui relient les sommets dans le DAG, et dans lequel le tri des règles voisines dans l'ordre de probabilité non croissante d'une manière qui préserve la politique de pare-feu consiste à trier les règles voisines d'une manière qui préserve les relations de priorité représentées par le DAG.

11. Procédé selon la revendication 1, dans lequel les règles comprennent des règles de détection des intrusions.

12. Procédé selon la revendication 1, dans lequel les règles comprennent des règles de protection contre les intrusions.

13. Pare-feu de réseau, comprenant :

(a) une structure de données de politique de pare-feu (1510) pour stocker une liste ordonnée de règles de pare-feu ;
(b) un moteur de politique de pare-feu (1508) pour mettre en oeuvre une politique de pare-feu en comparant des paquets aux règles dans l'ordre spécifié par la structure de données de politique de pare-feu (1510) ; et
(c) un optimiseur de politique de pare-feu (1514) pour optimiser la performance du pare-feu de réseau en réordonnant les règles d'une manière qui réduit un nombre moyen de comparaisons de règles par paquet et qui préserve la politique de pare-feu ;

IPR2023-01353 Page 01007

le pare-feu de réseau étant **caractérisé en ce que** l'optimiseur de politique de pare-feu (1514) est conçu pour :

(d) trier des groupes de deux règles ou plus dans l'ordre de probabilité non croissante d'une manière qui préserve la politique de pare-feu, dans lequel au moins l'un des groupes de deux règles ou plus comprend une première règle qui empêche une seconde règle du groupe d'être déplacée, au sein du groupe, plus près d'un début de la liste ordonnée qui met en oeuvre la politique de pare-feu, et dans lequel le tri des groupes de règles consiste à déplacer les première et seconde règles plus près du début de la liste ordonnée, tout en maintenant leur ordre relatif.

14. Pare-feu de réseau selon la revendication 13, dans lequel l'optimiseur de politique de pare-feu (1514) est conçu pour mesurer un nombre moyen de comparaisons par paquet pour un ordonnancement de paquets spécifié par la structure de données de politique de pare-feu (1510) et pour réordonner de manière dynamique les règles afin de réduire le nombre moyen de comparaisons par paquet.

IPR2023-01353 Page 01008

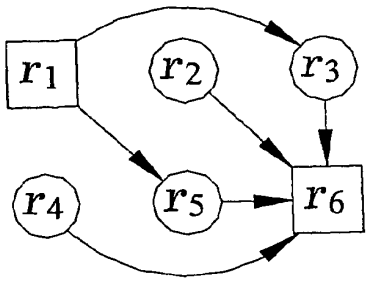


FIG. 1A

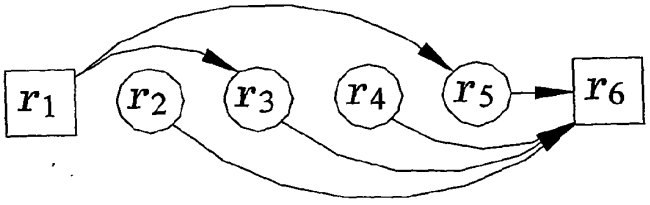


FIG. 1B

IPR2023-01353 Page 01009

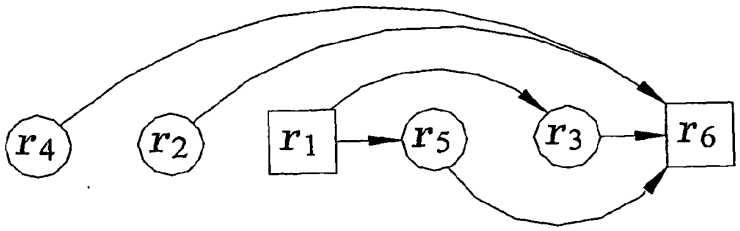


FIG. 2A

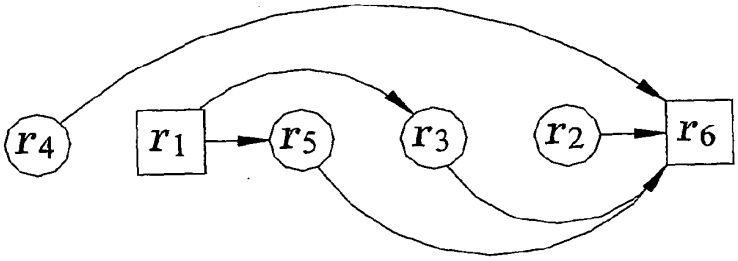


FIG. 2B

IPR2023-01353 Page 01010

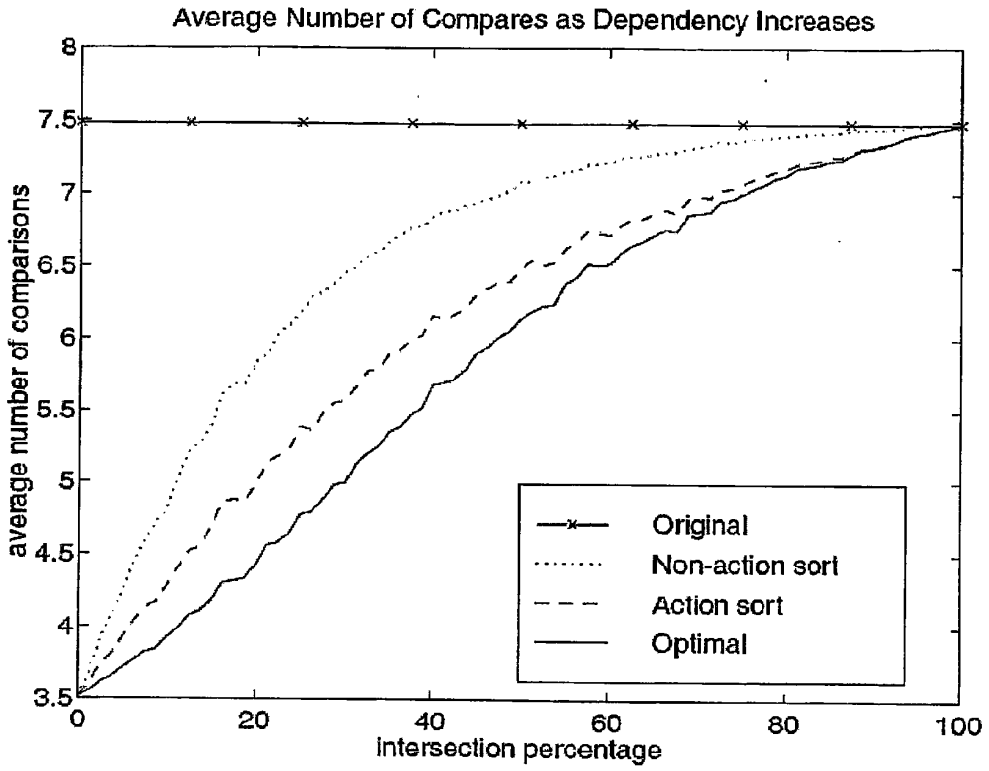

Average Number of Compares as Dependency Increases
average number of comparisons
intersection percentage
8
7.5
7
6.5
6
5.5
5
4.5
4
3.5
0
20
40
60
80
100
Original
Non-action sort
Action sort
Optimal

FIG. 3A

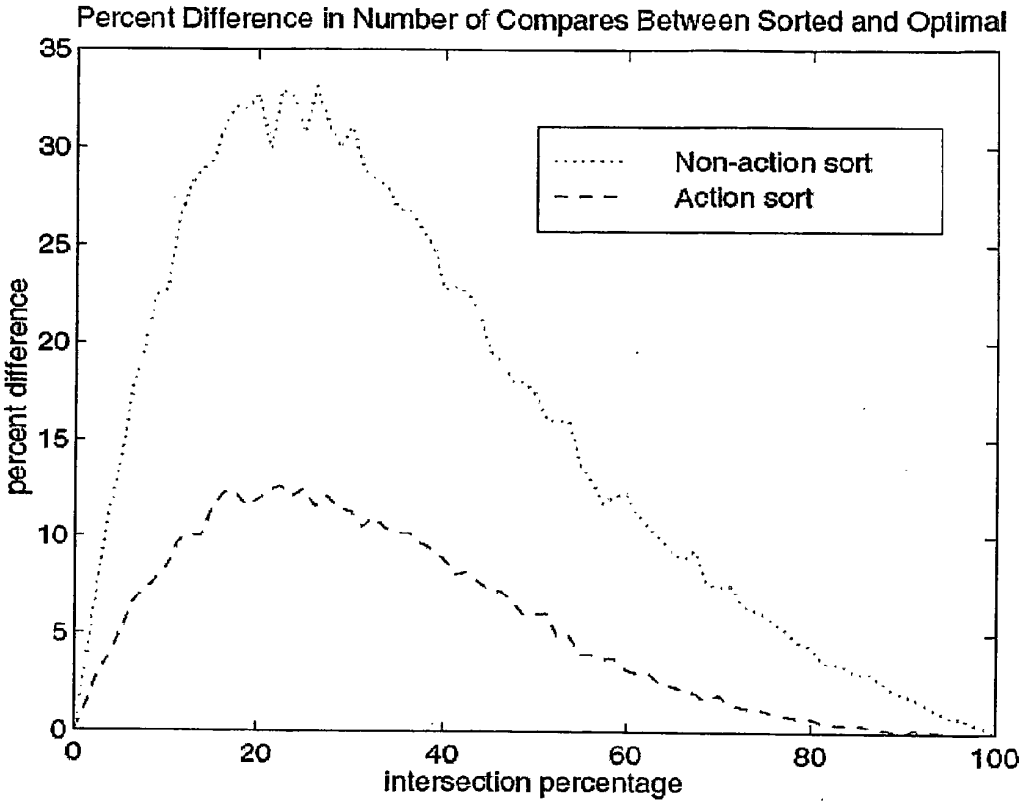

Percent Difference in Number of Compares Between Sorted and Optimal
35
30
25
20
15
10
5
0
percent difference
0
20
40
60
80
100
intersection percentage
Non-action sort
Action sort

FIG. 3B

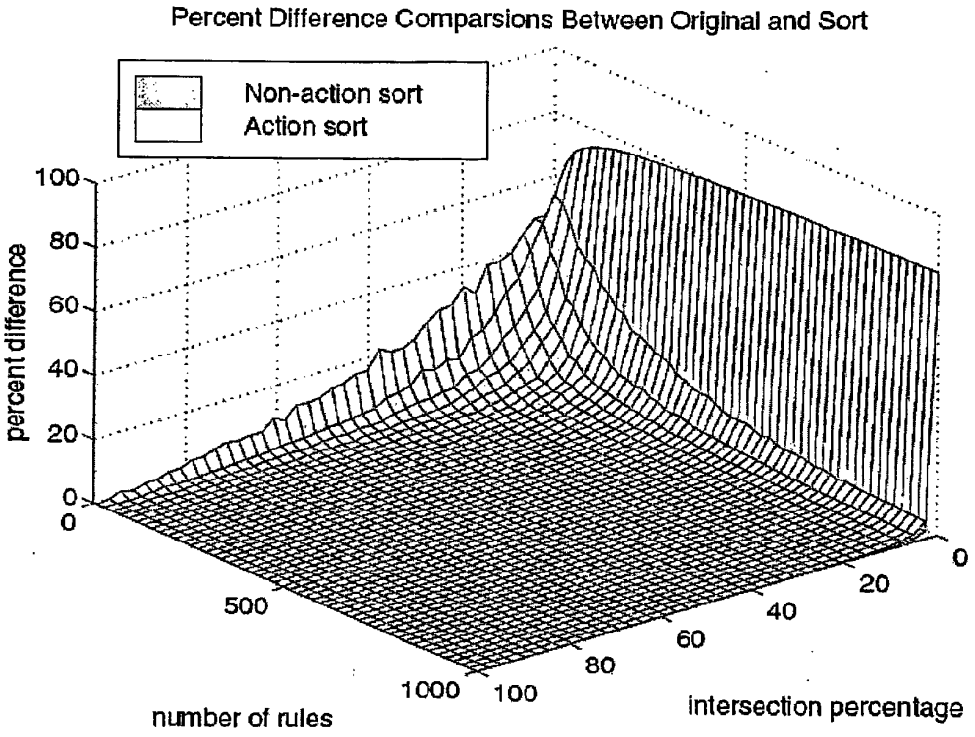

Percent Difference Comparsions Between Original and Sort
Non-action sort
Action sort
percent difference
100
80
60
40
20
0
0
500
1000
number of rules
0
20
40
60
80
100
intersection percentage

FIG. 4A

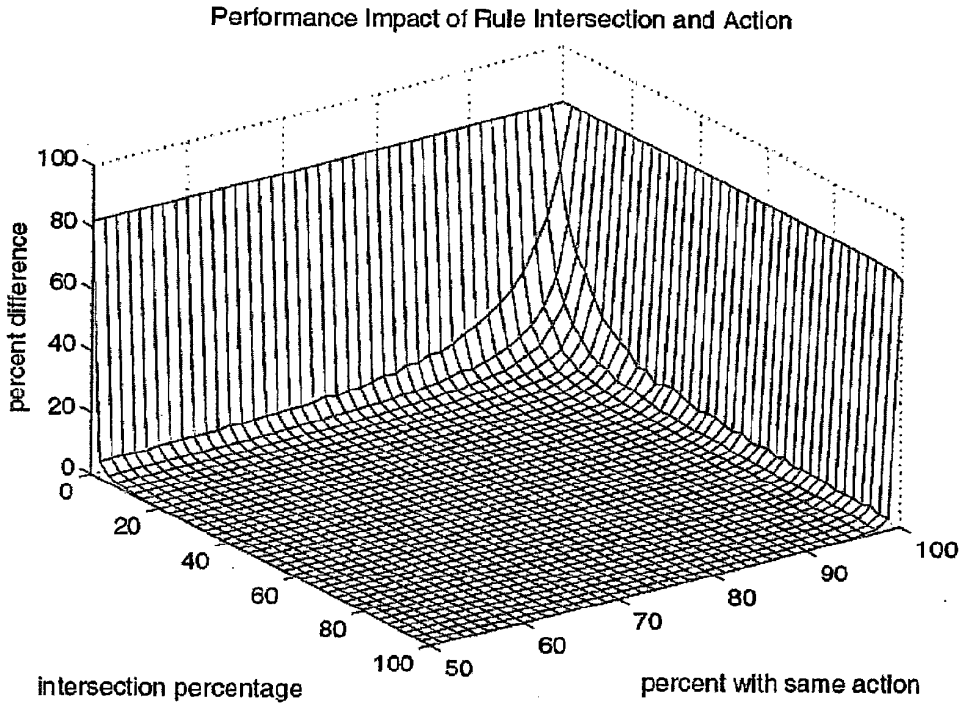

Performance Impact of Rule Intersection and Action
percent difference
100
80
60
40
20
0
0
20
40
60
80
100
intersection percentage
50
60
70
80
90
100
percent with same action

FIG. 4B

r_1	(UDP,	1.1.*,	ACCEPT)	$p_1 = 0.01$
r_2	(TCP,	1.*,	ACCEPT)	$p_2 = 0.02$
r_3	(TCP,	2.*,	ACCEPT)	$p_3 = 0.17$
r_4	(UDP,	1.*,	DENY)	$p_4 = 0.30$
r_5	(*,	*,	DENY)	$p_5 = 0.50$

FIG. 5A

r_3	(TCP,	2.*,	ACCEPT)	$p_3 = 0.17$
r_2	(TCP,	1.*,	ACCEPT)	$p_2 = 0.02$
r_1	(UDP,	1.1.*,	ACCEPT)	$p_1 = 0.01$
r_4	(UDP,	1.*,	DENY)	$p_4 = 0.30$
r_5	(*,	*,	DENY)	$p_5 = 0.50$

FIG. 5B

IPR2023-01353 Page 01015

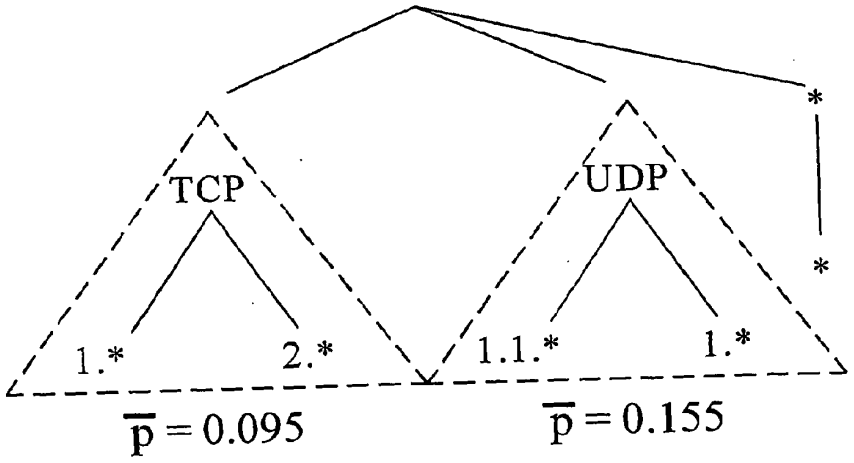


FIG. 6A

r_1	(UDP,	1.1.*,	ACCEPT)	$p_1 = 0.01$
r_4	(UDP,	1.*,	DENY)	$p_4 = 0.30$
r_3	(TCP,	2.*,	ACCEPT)	$p_3 = 0.17$
r_2	(TCP,	1.*,	ACCEPT)	$p_2 = 0.02$
r_5	(*,	*,	DENY)	$p_5 = 0.50$

FIG. 6B

IPR2023-01353 Page 01016

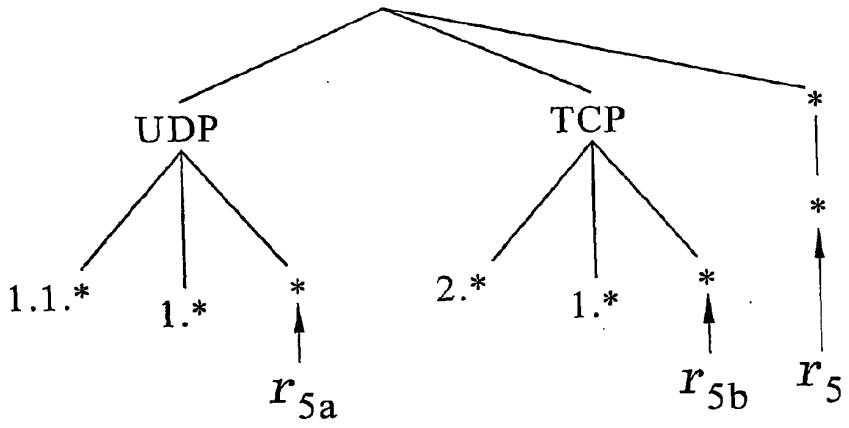


FIG. 7A

r_1	(UDP,	1.1.*,	ACCEPT)	$p_1 = 0.01$
r_4	(UDP,	1.*,	ACCEPT)	$p_4 = 0.30$
r_{5a}	(UDP,	*,	DENY)	$p_{5a} = 0.47$
r_3	(TCP,	2.*,	ACCEPT)	$p_3 = 0.17$
r_2	(TCP,	1.*,	ACCEPT)	$p_2 = 0.02$
r_{5b}	(TCP,	*,	DENY)	$p_{5b} = 0.03$

FIG. 7B

IPR2023-01353 Page 01017

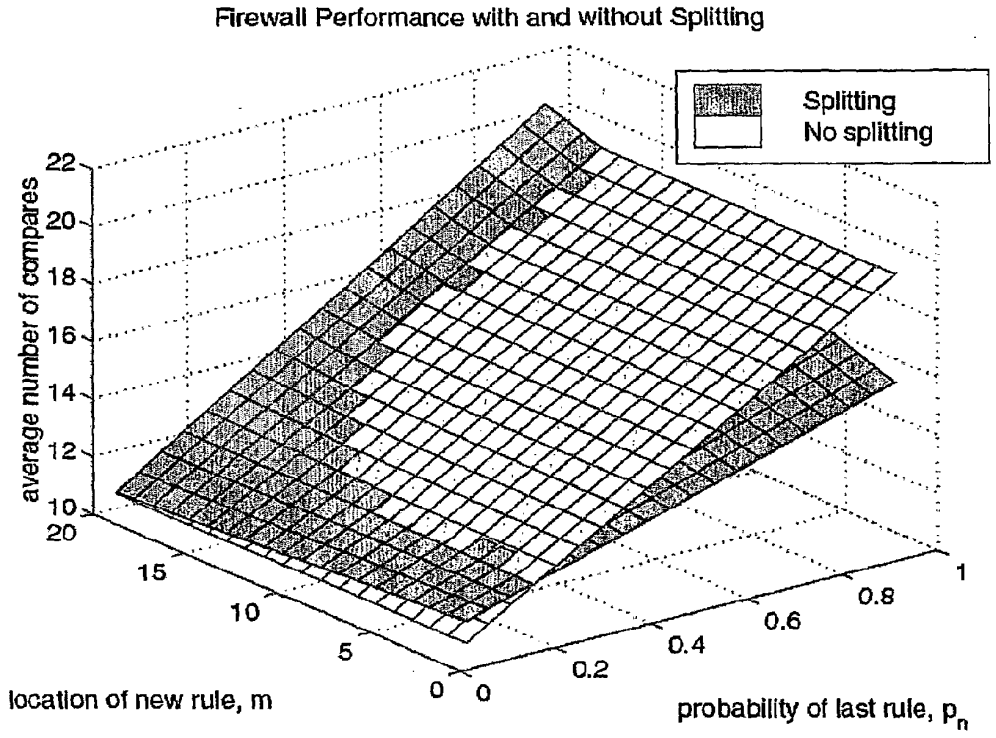

Firewall Performance with and without Splitting
Splitting
No splitting
average number of compares
22
20
18
16
14
12
10
20
15
10
5
0
0
0.2
0.4
0.6
0.8
1
location of new rule, m
probability of last rule, p_n

FIG. 8A

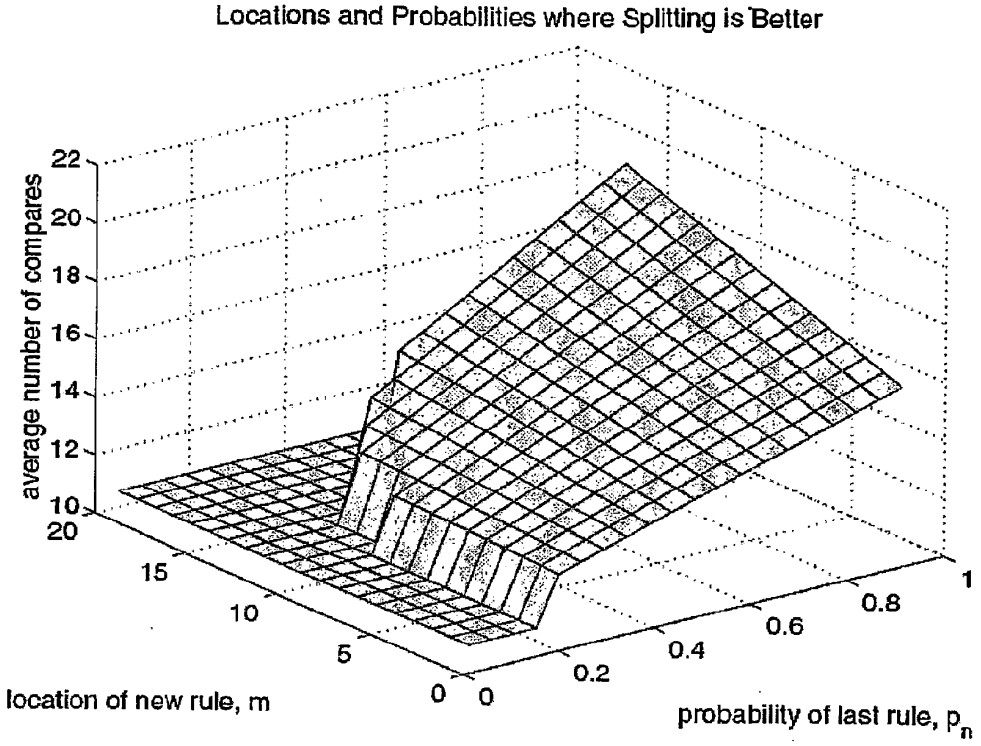


FIG. 8B

IPR2023-01353 Page 01019

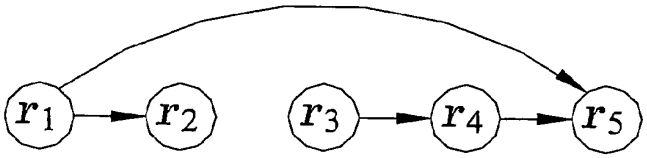


FIG. 9A

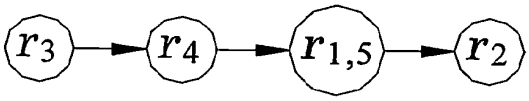


FIG. 9B

IPR2023-01353 Page 01020

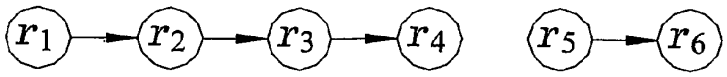


FIG. 10A

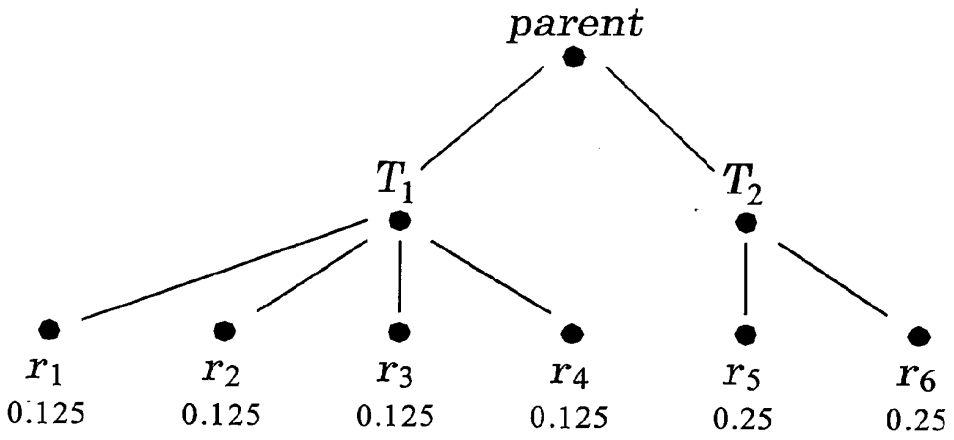


FIG. 10B

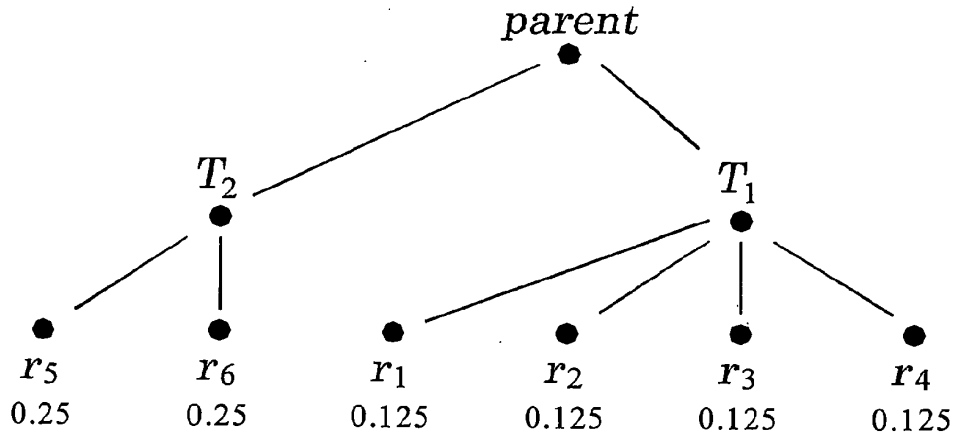


FIG. 10C

IPR2023-01353 Page 01021

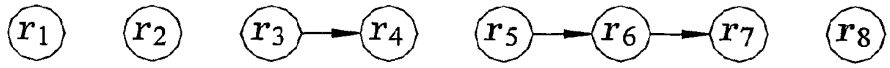


FIG. 11A

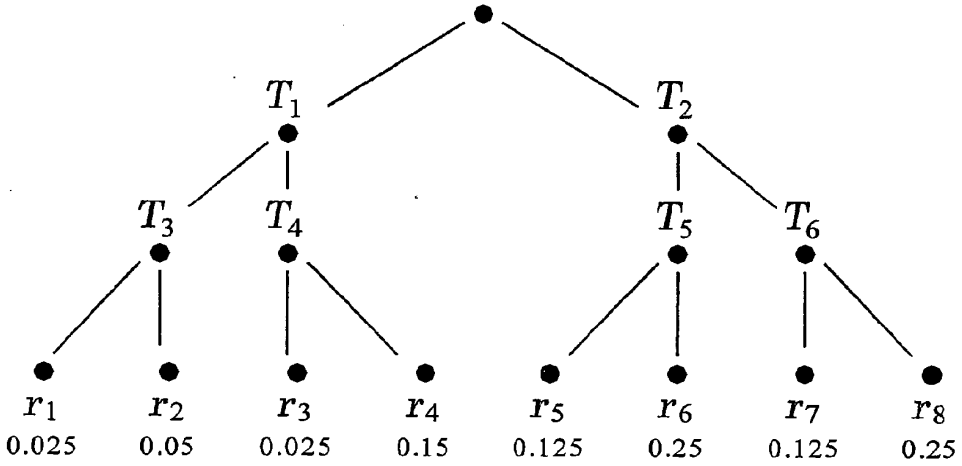


FIG. 11B

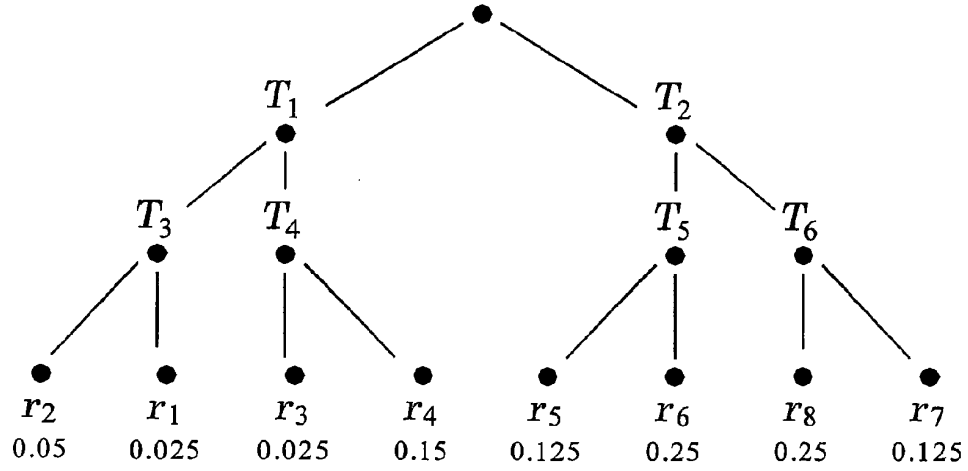


FIG. 11C

IPR2023-01353 Page 01022

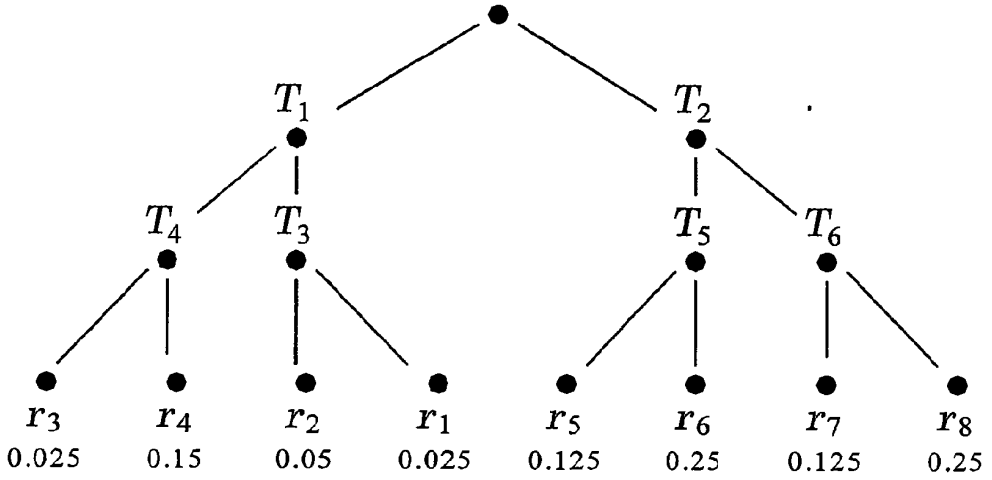


FIG. 11D

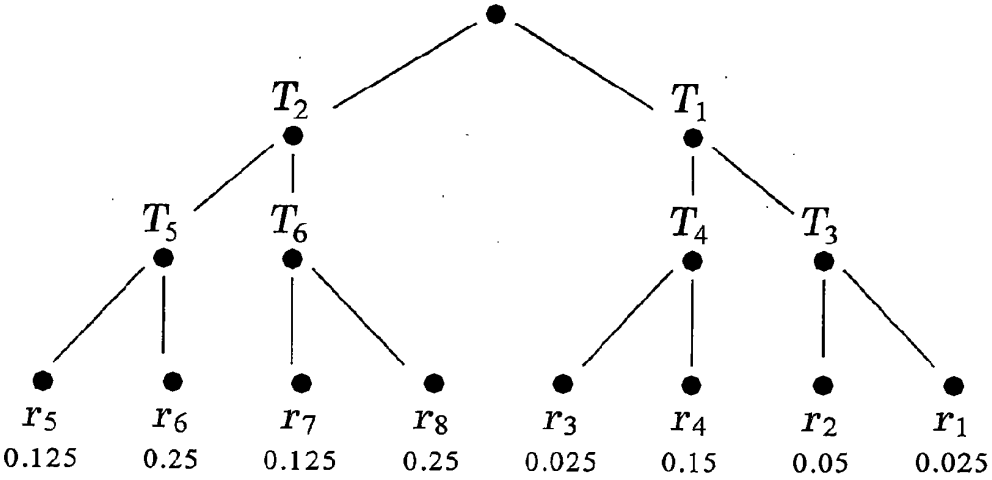


FIG. 11E

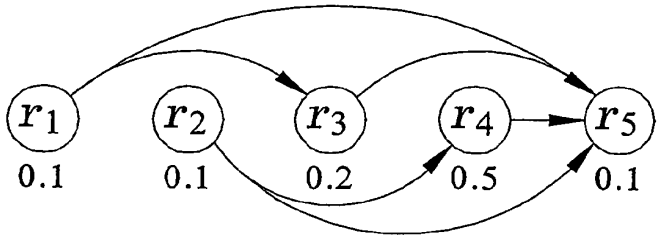


FIG. 12A

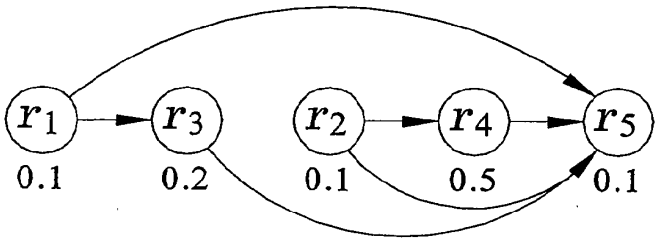


FIG. 12B

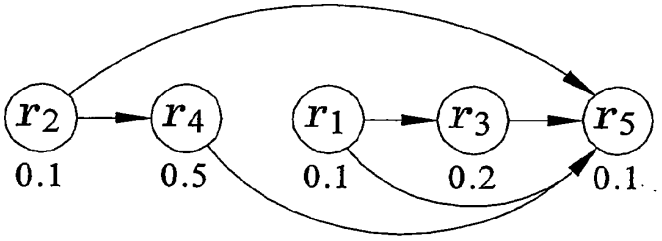


FIG. 12C

IPR2023-01353 Page 01024

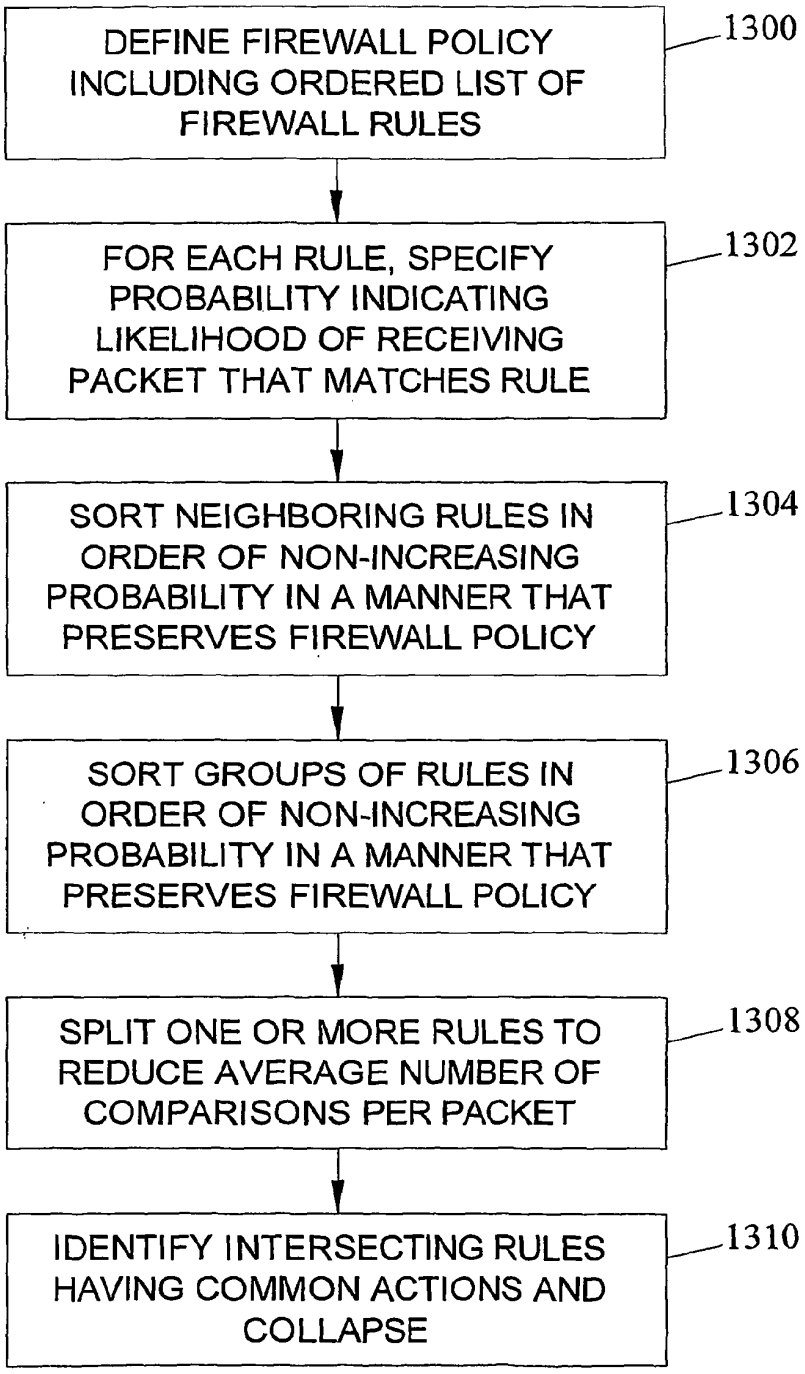


FIG. 13

IPR2023-01353 Page 01025

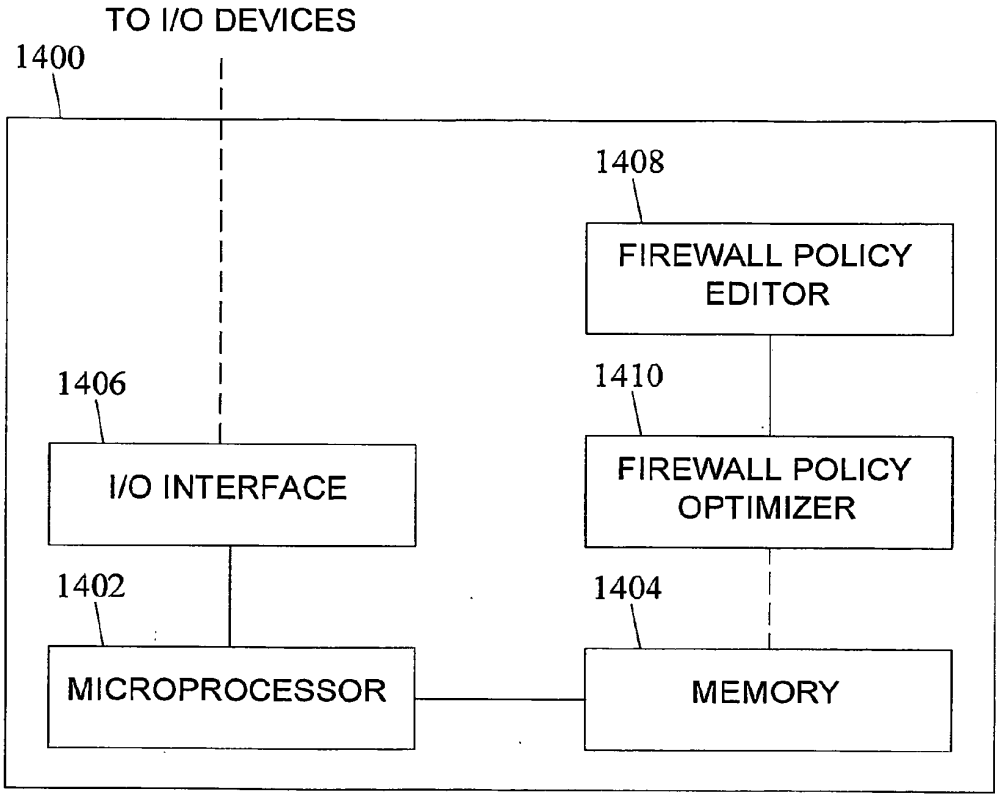


FIG. 14

IPR2023-01353 Page 01026

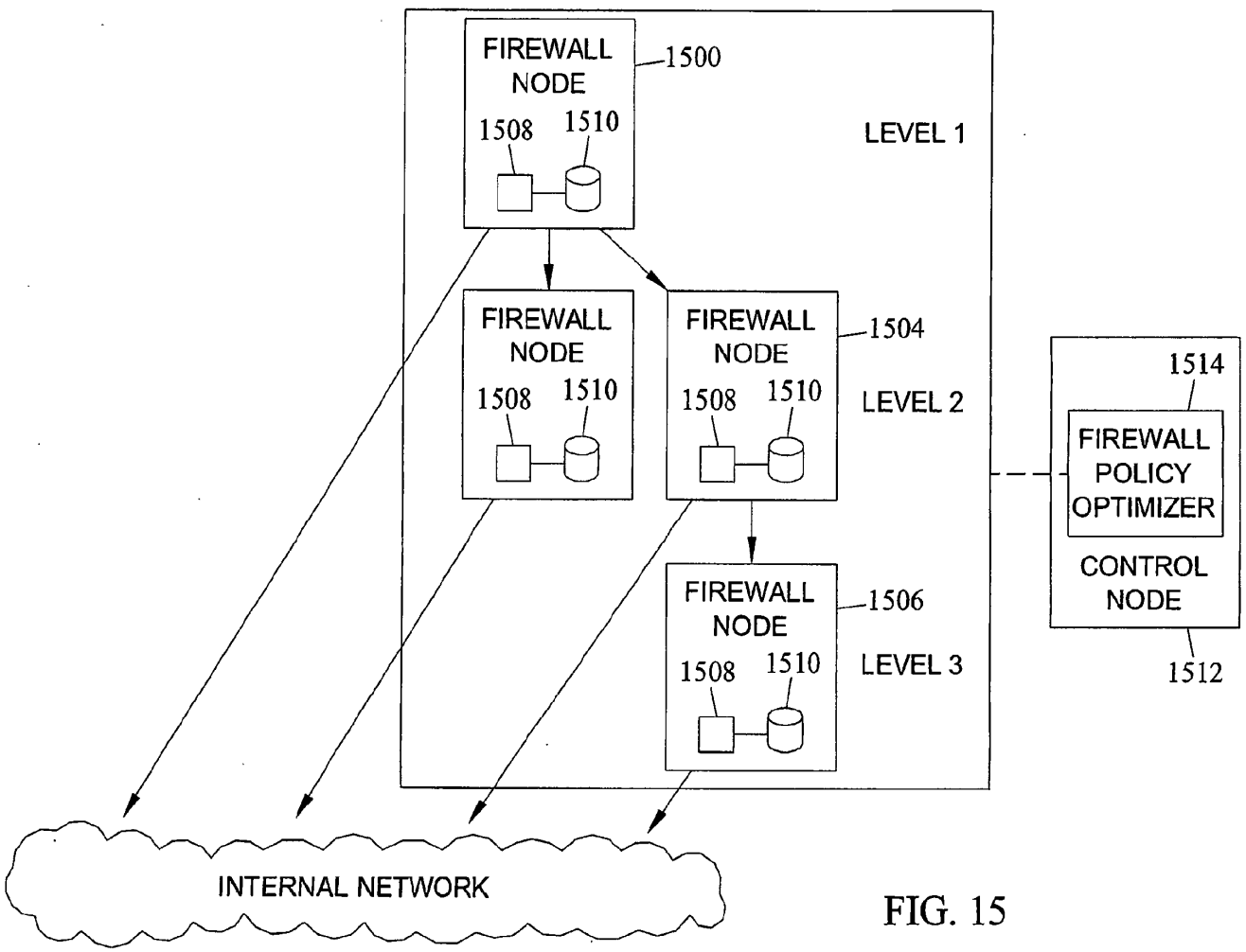

FIREWALL NODE
1500
1508
1510
LEVEL 1
FIREWALL NODE
1508
1510
FIREWALL NODE
1504
1508
1510
LEVEL 2
1514
FIREWALL POLICY OPTIMIZER
CONTROL NODE
1512
FIREWALL NODE
1506
1508
1510
LEVEL 3
INTERNAL NETWORK

FIG. 15

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- EP 1006701 A **[0005]**
- US 2002198981 A **[0006]**
- WO 2006093557 A **[0007]**
- US 31633105 A **[0067]**

Non-patent literature cited in the description

- **E. AL-SHAER ; H. HAMED.** Firewall Policy Management Advisor for Anomaly Detection and Rule Editing. *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management,* 2003 **[0068]**
- **E. AL-SHAER ; H. HAMED.** Modeling and Management of Firewall Policies. *IEEE Transactions on Network and Service Management,* 2004 **[0068]**
- **S. M. BELLOVIN ; W. CHESWICK.** Network Firewalls. *IEEE Communications Magazine,* September 1994, 50-57 **[0068]**
- **C. BENECKE.** A Parallel Packet Screen for High Speed Networks. *Proceedings of the 15th Annual Computer Security Applications Conference,* 1999 **[0068]**
- **G. BRIGHTWELL ; P. WINKLER.** Counting Linear Extensions is #P-Complete. *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing* **[0068]**
- **M. CHRISTIANSEN ; E. FLEURY.** Using Interval Decision Diagrams for Packet Filtering. *Technical report, BRICS,* 2002 **[0068]**
- **D. DECASPER ; Z. DITTIA ; G. PARULKAR ; B. PLATTNER.** Router Plugins: A Software Architecture for Next-Generation. Routers. *IEEE/ACM Transactions on Networking,* February 2000, vol. 8 (1 **[0068]**
- **M. DEGERMARK ; A. BRODNIK ; S. CARLSSON ; S. PINK.** Small Forwarding Tables for Fast Routing Lookups. *Proceedings of ACM SIGCOMM,* 1997, 4-13 **[0068]**
- **U. ELLERMANN ; C. BENECKE.** Firewalls for ATM Networks. *Proceedings of INFOSEC'COM,* 1998 **[0068]**
- **A. FELDMANN ; S. MUTHUKRISHNAN.** Tradeoffs for Packet Classification. *Proceedings of the IEEE INFOCOM,* 2000, 397-413 **[0068]**
- **E. W. FULP.** Firewall Architectures for High Speed Networks. *Technical Report,* 2002 **[0068]**
- **E. W. FULP.** Firewall Policy Models Using Ordered-Sets. *Technical report,* 2004 **[0068]**
- **R. FUNKE ; A. GROTE ; H.-U. HEISS.** Performance Evaluation of Firewalls in Gigabit-Networks. *Proceedings of the Symposium on Performance Evaluation of Computer and Telecommunication Systems,* 1999 **[0068]**
- **S. GODDARD ; R. KIECKHAFER ; Y. ZHANG.** An Unavailability Analysis of Firewall Sandwich Configurations. *Proceedings of the 6th IEEE Symposium on High Assurance Systems Engineering,* 2001 **[0068]**
- Optimizing and Approximation in Determinstic Sequencing and Scheduling: A Survey. **R. L. GRAHAM ; E. L. LAWLER ; J. K. LENSTRA ; A. H. G. R. KAN.** Annals of Discrete Mathematics. 1979, vol. 5, 287-326 **[0068]**
- **A. HARI ; S. SURI ; G. PARULKAR.** Detecting and Resolving Packet Filter Conflicts. *Proceedings of IEEE INFOCOM,* 2000, 1203-1212 **[0068]**
- *High Performance Packet Classification, http://www.hipac.org* **[0068]**
- **E. HOROWITZ ; S. SAHNI ; D. MEHTA.** Fundamentals of Data Structures in C++. Compuer Science Press, 1995 **[0068]**
- Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints. **E. L. LAWLER.** Annals of Discrete Mathematics. 1978, vol. 2, 75-90 **[0068]**
- **W. E. LELAND ; M. S. TAQQU ; W. WILLINGER ; D. V. WILSON.** On the Self-Similar Nature of Ethernet Traffic. *IEEE Transactions on Networking,* 1994, vol. 2, 1-15 **[0068]**
- **J. K. LENSTRA ; A. H. G. R. KAN.** Complexity of Scheduling under Precedence Constraints. *Operations Research,* 1978, vol. 26 (1), 22-35 **[0068]**
- **O. PAUL ; M. LAURENT.** A Full Bandwidth ATM Firewall. *Proceedings of the 6th European Symposium on Research in Computer Security ESORICS'2000,* 2000 **[0068]**
- **B. R. PREISS.** Data Structures and Algorithms with Object-Oriented Design Patterns in C++. John Wiley & Sons, 1999 **[0068]**

IPR2023-01353 Page 01028

- **L. QUI ; G. VARGHESE ; S. SURI.** Fast Firewall Implementations for Software and Hardware-Based Routers. *Proceedings of ACM SIGMETRICS,* June 2001 **[0068]**
- **V. P. RANGANATH ; D. ANDRESEN.** A Set-Based Approach to Packet Classification. *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems,* 2003, 889-894 **[0068]**
- **R. RIVEST.** On Self-Organizing Sequential Search Heuristics. *Communications of the ACM,* 1976, vol. 19 (2 **[0068]**
- **W. E. SMITH.** Various Optimizers for Single-Stage Production. *Naval Research Logistics Quarterly,* 1956, vol. 3, 59-66 **[0068]**
- **V. SRINIVASAN ; G. VARGHESE ; S. SURI ; M. WALDVOGEL.** Fast and Scalable Layer Four Switching. *Proceedings of ACM SIGCOMM,* 1998, 191-202 **[0068]**
- **S. SURI ; G. VARGHESE.** Packet Filtering in High Speed Networks. *Proceedings of the Symposium on Discrete Algorithms,* 1999, 969-970 **[0068]**
- **R. L. ZIEGLER.** Linux Firewalls. New Riders. 2002 **[0068]**
- **E. D. ZWICKY ; S. COOPER ; D. B. CHAPMAN.** *Building Internet Firewalls,* 2000 **[0068]**

IPR2023-01353 Page 01029

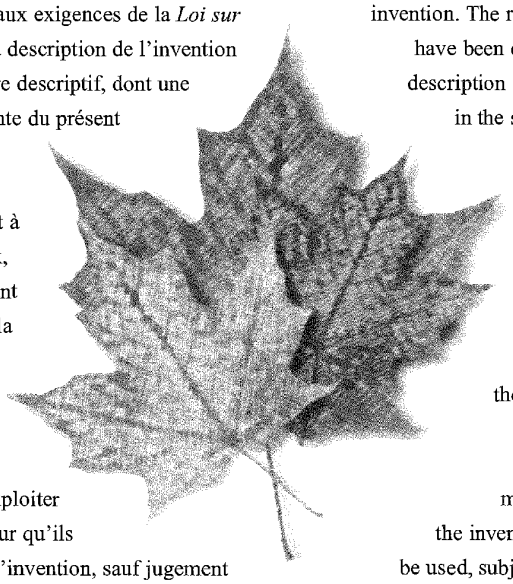
Office de la propriété intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An Agency of Industry Canada

Brevet canadien / Canadian Patent

Le commissaire aux brevets a reçu une demande de délivrance de brevet visant une invention. Ladite requête satisfait aux exigences de la *Loi sur les brevets*. Le titre et la description de l'invention figurent dans le mémoire descriptif, dont une copie fait partie intégrante du présent document.

Le présent brevet confère à son titulaire et à ses représentants légaux, pour une période expirant vingt ans à compter de la date du dépôt de la demande au Canada, le droit, la faculté et le privilège exclusif de fabriquer, construire, exploiter et vendre à d'autres, pour qu'ils l'exploitent, l'objet de l'invention, sauf jugement en l'espèce rendu par un tribunal compétent, et sous réserve du paiement des taxes périodiques.



The Commissioner of Patents has received a petition for the grant of a patent for an invention. The requirements of the *Patent Act* have been complied with. The title and a description of the invention are contained in the specification, a copy of which forms an integral part of this document.

The present patent grants to its owner and to the legal representatives of its owner, for a term which expires twenty years from the filing date of the application in Canada, the exclusive right, privilege and liberty of making, constructing and using the invention and selling it to others to be used, subject to adjudication before any court of competent jurisdiction, and subject to the payment of maintenance fees.

BREVET CANADIEN **2,600,236** CANADIAN PATENT

Date à laquelle le brevet a été accordé et délivré	**2014/08/12**	Date on which the patent was granted and issued
Date du dépôt de la demande	**2006/03/28**	Filing date of the application
Date à laquelle la demande est devenue accessible au public pour consultation	**2006/10/05**	Date on which the application was made available for public inspection

Commissaire aux brevets / Commissioner of Patents

Canada

3256 (CIPO 91) 04/14

OPIC CIPO

Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2600236 C 2014/08/12

(11)(21) 2 600 236

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2006/03/28
(87) Date publication PCT/PCT Publication Date: 2006/10/05
(45) Date de délivrance/Issue Date: 2014/08/12
(85) Entrée phase nationale/National Entry: 2007/09/04
(86) N° demande PCT/PCT Application No.: US 2006/011291
(87) N° publication PCT/PCT Publication No.: 2006/105093
(30) Priorité/Priority: 2005/03/28 (US60/665,664)

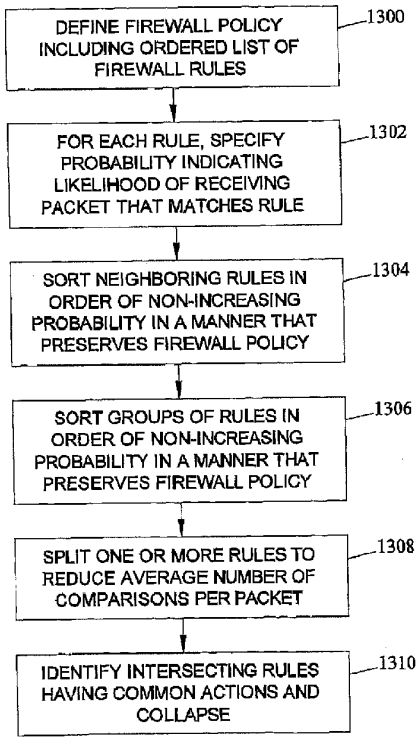
(51) Cl.Int./Int.Cl. *G06F 15/16* (2006.01)
(72) Inventeurs/Inventors:
FULP, ERRIN W., US;
TARSA, STEPHEN J., US

(73) Propriétaire/Owner:
WAKE FOREST UNIVERSITY, US

(74) Agent: NORTON ROSE FULBRIGHT CANADA LLP/S.E.N.C.R.L., S.R.L.

(54) Titre : PROCEDES, SYSTEMES ET PRODUITS DE PROGRAMMES INFORMATIQUES POUR L'OPTIMISATION DE LA POLITIQUE DE PARE-FEU D'UN RESEAU

(54) Title: METHODS, SYSTEMS, AND COMPUTER PROGRAM PRODUCTS FOR NETWORK FIREWALL POLICY OPTIMIZATION



(57) Abrégé/Abstract:
Methods, systems, and computer program products for firewall policy optimization are disclosed. According to one method, a firewall policy including an ordered list of firewall rules is defined. For each rule, a probability indicating a likelihood of receiving a packet matching the rule is determined. The rules are sorted in order of non-increasing probability in a manner that preserves the firewall policy.

METHODS, SYSTEMS, AND COMPUTER PROGRAM PRODUCTS FOR NETWORK FIREWALL POLICY OPTIMIZATION

GOVERNMENT INTEREST

This invention was made with Government support under Grant No. DE-FG02-03ER25581 awarded by U.S. Department of Energy, Mathematical and Computational Information Sciences Division. The Government has certain rights in the invention.

TECHNICAL FIELD

The subject matter described herein relates to optimizing polices for network firewalls. More particularly, the subject matter described herein relates to methods, systems, and computer program products for optimizing performance of network firewalls.

BACKGROUND ART

Network firewalls remain the forefront defense for most computer systems. Guided by a security policy, these devices provide access control, auditing, and traffic control [3, 30, 31]. A security policy is a list of ordered rules, as seen in Table 1, that defines the action to perform on matching, packets. For example, an accept action passes the packet into or from the secure network, while deny causes the packet to be discarded. In many implementations, the rule set is stored internally as a linked list. A packet is sequentially compared to the rules, starting with the first, until a match is found; otherwise, a default action is performed [30, 31]. This is referred to as a first- match policy and is used in many firewall systems including the Linux firewall implementation iptables [25].

No.	Proto.	Source IP	Source Port	Destination IP	Destination Port	Action	Prob.
1	UDP	1.1.*	*	*	80	deny	0.01
2	TCP	2.*	*	1.*	90	accept	0.02
3	UDP	*	*	1.*	*	accept	0.10
4	TCP	2.*	*	1.*	20	accept	0.17
5	UDP	1.*	*	*	*	accept	0.20
6	*	*	*	*	*	deny	0.50

Table 1: Example Security Policy Consisting of Multiple Ordered Rules

Traditional firewall implementations consist of a single, dedicated machine, similar to a router, that sequentially applies rules to each arriving packet. However, packet filtering represents a significantly higher processing load than routing decisions [24, 29, 31]. For example, a firewall that interconnects two 100 Mbps networks would have to process over 300,000 packets per second [30]. Successfully handling these traffic loads becomes more difficult as rule sets become more complex [4, 22, 31]. Furthermore, firewalls must be capable of processing even more packets as interface speeds increase. In a high-speed environment (e.g. Gigabit Ethernet), a single firewall can easily become a bottleneck and is susceptible to DoS attacks [4, 9, 13, 14]. An attacker could simply inundate the firewall with traffic, delaying or preventing legitimate packets from being processed.

One approach to increase firewall performance focuses on improving hardware design. Current research is investigating different distributed firewall designs to reduce processing delay [4, 9, 22], and possibly provide service differentiation [11]. Another approach focuses on improving performance via better firewall software [6, 7, 12, 16, 17, 24]. Similar to approaches that address the longest matching prefix problem for packet classification [8, 10, 25, 28], solutions typically represent the firewall rule set in different fashions (e.g. tree structures) to improve performance. While both approaches, together or separate, show great promise, each requires radical changes to the firewall system, and therefore are not amenable to current or legacy systems.

Accordingly, there exists a need for improved methods, systems, and computer program products for network firewall policy optimization.

SUMMARY

The subject matter described herein includes methods, systems, and computer program products for network firewall policy optimization. According to one method, a firewall policy including an ordered list of firewall rules is defined. For each rule, a probability indicating a likelihood of receiving a packet matching the rule is defined. Neighboring rules are sorted in order of non-increasing probability in a manner that preserves the firewall policy.

As used herein, the term "firewall" refers to a logical entity that is adapted to filter packets at the ingress and/or egress portions of a network based on a policy. The term "firewall" is intended to include systems that block packets from entering or leaving a network and systems that perform intrusion detection and intrusion protection functions for packets entering or leaving a network. Thus, the methods and systems described herein for firewall policy optimization can be used to optimize policies for intrusion detection and intrusion protection firewalls.

The subject matter described herein includes a method to improve firewall performance and lower packet delay that can be applied to both legacy and current systems. In one implementation, a firewall rule set may be re-ordered to minimize the average number of rule comparisons to determine the action, while maintaining the integrity of the original policy. Integrity is preserved if the reordered and original rules always arrive at the same result. To maintain integrity, a firewall rule set may be modeled as a Directed Acyclical Graph (DAG), where vertices are firewall rules and edges indicate precedence relationships. Given this representation, any linear arrangement of the DAG is proven to maintain the original policy integrity. Unfortunately, determining the optimal rule order from all the possible linear arrangements is proven to be *NP*-complete, since it is equivalent to sequencing jobs with precedence constraints for a single machine [15]. Although determining the optimal order is *NP*-complete, a heuristic will be described below to order firewall rules that reduces the average number of comparisons while maintaining integrity. Simulation

results show the proposed reordering method yields rule orders that are comparable to optimal; thus, provides a simple means to significantly improve firewall performance and lower packet delay.

The subject matter described herein for network firewall policy optimization may be implemented using any combination of hardware, software, or firmware. For example, the subject matter described herein may be implemented using a computer program product comprising computer executable instructions embodied in a computer readable medium. Exemplary computer readable media suitable for implementing the subject matter described herein include disk memory devices, chip memory devices, programmable logic devices, application specific integrated circuits, and downloadable electrical signals. In addition, a computer program product that implements the subject matter described herein may be implemented using a single device or computing platform or may be distributed across multiple devices or computing platforms.

BRIEF DESCRIPTION OF THE DRAWINGS

Preferred embodiments of the subject matter described herein will now be explained with reference to the accompanying drawings of which:

Figure 1A is a block diagram of a rule list Directed Acyclical Graph (DAG) for the firewall rules in Table 1;

Figure 1B is a linear arrangement of firewall rules corresponding to the original rule order;

Figure 2A is a policy DAG representing a reordered version of the firewall rule set in Figure 1 after sorting;

Figure 2B is a policy DAG for the rule set illustrated in Table 1 illustrating an optimal rule order;

Figure 3A is a graph illustrating average numbers of packet comparisons versus intersection percentage for different orderings of a firewall policy;

Figure 3B is a graph illustrating the percent difference in average numbers of packet comparisons between sorted and optimal sortings of a firewall policy;

IPR2023-01353 Page 01035

Figure 4A is a graph of percent difference comparisons between original and sorted versions of a firewall policy;

Figure 4B is a graph illustrating the performance impact of considering rule intersection and actions in performing firewall rule sorting;

Figure 5A is a table illustrating a firewall policy before sorting;

Figure 5B is a table illustrating a sorted version of the firewall policy of Figure 5A;

Figure 6A is a policy trie representation of the firewall policy of Figure 5B;

Figure 6B is a table representing the firewall policy of Figure 5A after TCP and UDP sub-trie rotation;

Figure 7A is a policy trie illustrating the splitting of rule r_5 into rules r_{5a} and r_{5b};

Figure 7B is a table illustrating the firewall policy of Figure 5A after splitting rule r_5 into rules r_{5a} and r_{5b};

Figure 8A is a graph illustrating firewall performance with and without rule splitting;

Figure 8B is a graph illustrating locations and probabilities where splitting is better;

Figure 9A is a policy DAG of a firewall rule set;

Figure 9B is a policy DAG illustrating compression of rules r_1 and r_5;

Figure 10A is a policy DAG of a firewall rule list;

Figure 10B is an original policy trie before sorting;

Figure 10C is a policy trie illustrating the firewall rules after rotating sub-tries T_1 and T_2;

Figure 11A is a policy DAG illustrating a firewall policy;

Figure 11B is a policy trie illustrating firewall rules of the policy before sorting;

Figure 11C is a policy trie illustrating the firewall policy after exchanging nodes r_1 and r_2 and nodes r_7 and r_8;

Figure 11D is a policy trie illustrating the firewall policy after exchanging sub-tries T_3 and T_4;

Figure 11E is a policy trie illustrating the firewall policy after exchanging sub-tries T_1 and T_2;

IPR2023-01353 Page 01036

Figure 12A is a policy DAG illustrating an original order for a firewall policy;

Figure 12B is a policy DAG illustrating a sorted version of the firewall policy of Figure 12A after exchanging neighbors;

Figure 12C is a policy DAG illustrating a sorted version of the firewall policy after exchanging lists L_1 including rules r_1 and r_3 and lists L_2 including rules r_2 and r_4;

Figure 13 is a flow chart illustrating exemplary steps for network firewall policy optimization according to an embodiment of the subject matter described herein;

Figure 14 is a block diagram illustrating exemplary components of a system for firewall policy optimization according to an embodiment of the subject matter described herein; and

Figure 15 is a block diagram illustrating a plurality of firewall nodes for implementing a firewall policy according to an embodiment of the subject matter described herein.

DETAILED DESCRIPTION OF THE INVENTION

Modeling Firewall Security Policies

As described above, a firewall rule set, also known as a firewall policy, is traditionally an ordered list of firewall rules. Firewall policy models have been the subject of recent research [1, 2, 16]; however, the primary purpose is anomaly detection and policy verification. In contrast, the policy model described in this herein is designed for firewall performance optimization and integrity. Firewall performance refers to reducing the average number of comparisons required to determine an action, while integrity refers to maintaining the original policy intent. Although improving the worst-case performance is important, it is not possible without changing the list-based representation [16, 24].

Firewall Rule and Policy Models

In the examples described herein, a rule r is modeled as an ordered tuple of sets, $r = (r[1], r[2], ..., r[k])$. Order is necessary among the tuples since

comparing rules and packets requires the comparison of corresponding tuples. Each tuple $r[i]$ is a set that can be fully specified, specify a range, or contain wildcards '*' in standard prefix format. For the Internet, security rules are commonly represented as a 5-tuple consisting of: protocol type, IP source address, source port number, IP destination address, and destination port number [30, 31]. Given this model, the ordered tuples can be supersets and subsets of each other, which forms the basis of precedence relationships. In addition to the prefixes, each filter rule has an action, which is to accept or deny. However, the action will not be considered when comparing packets and rules. Similar to a rule, a packet (IP datagram) d can be viewed as an ordered k-tuple $d = (d[1], d[2], ..., d[k])$; however, ranges and wildcards are not possible for any packet tuple.

Using the previous rule definition, a standard security policy can be modeled as an ordered set (list) of n rules, denoted as $R = \{r_1, r_2, ..., r_n\}$. A packet d is sequentially compared against each rule r_i starting with the first, until a match is found ($d \Rightarrow r_i$) then the associated action is performed. A match is found between a packet and rule when every tuple of the packet is a subset of the corresponding tuple in the rule.

Definition Packet d matches r_i if

$$d \Rightarrow r_i \quad \text{iff} \quad d[i] \subseteq r_i[i], \quad i = 1, ..., k$$

The rule list R is comprehensive if for every possible legal packet d a match is found using R. Furthermore, two rule lists R and R' are equivalent if for every possible legal packet d the same action is performed by the two rule lists. If R and R' are different (e.g. a reorder) and the lists are equivalent, then the policy integrity is maintained.

As previously mentioned, a rule list has an implied precedence relationship where certain rules must appear before others if the integrity of the policy is to be maintained. For example consider the rule list in Table 1. Rule r_1 must appear before rules r_3 and r_5, likewise rule r_6 must be the last rule in the policy. If for example, rule r_3 was moved to the beginning of the policy, then it will shadow [2] the original rule r_1. However, there is no precedence relationship between rules r_2 and r_4 given in Table 1. Therefore, the relative ordering of these two rules will not impact the policy integrity and can be

changed to improve performance. The present example assumes that the original policy is free from any anomalies. Likewise, when a policy is reordered to improve performance it should not introduce any anomalies, which will occur if precedence relationships are not maintained. As a result, a model is needed to effectively represent precedence relationships.

Modeling Rule List Precedence Relationships

The precedence relationship between rules in a policy will be modeled as a Directed Acyclical Graph (DAG) [23, 18]. Such graphs have been successfully used to represent the relative order of individual tasks that must take place to complete a job (referred to as a task graph model). Since certain rules must appear before others to maintain policy integrity, this structure is well suited for modeling the precedence of firewall rules. Let $G = (R, E)$ be a rule list DAG for a rule list R, where vertices are rules and edges E are the precedence relationships (constraint). A precedence relationship, or edge, exists between rules r_i and r_j, if $i < j$, the actions for each rule are different, and the rules intersect.

Definition The intersection of rule r_i and r_j, denoted as $r_i \cap r_j$ is

$$r_i \cap r_j = (r_i[l] \cap r_j[l]), \qquad l = 1, \ldots, k$$

Therefore, the intersection of two rules results in an ordered set of tuples that collectively describes the packets that match both rules. The rules r_i and r_j intersect if every tuple of the resulting operation is non-empty. In contrast, the rules r_i and r_j do not intersect, denoted as $r_i \not\cap r_j$, if at least one tuple is the empty set. Note the intersection operation is symmetric; therefore, if r_i intersects r_j, then r_j will intersect r_i. The same is true for rules that do not intersect.

For example consider the rules given in Table 1, the intersection of r_1 and r_3 yields (UDP, 1.1.*, *, 1.*, 80). Again, the rule actions are not considered in the intersection or match operation. Since these two rules intersect, a packet can match both rules for example d = (UDP, 1.1.1.1, 80, 1.1.1.1, 80). Furthermore, the actions of the two rules are different. Therefore, the relative order must be maintained between these two rules and an edge drawn from r_1 to r_3 must be present in the rule list DAG, as seen in Figures 1A and 1B. More

IPR2023-01353 Page 01039

particularly, Figure 1A illustrates a rule list DAG for the rules in Table 1. Figure 1B illustrates a linear arrangement of the rules in Figure 1A. In Figures 1A and 1B, the vertices represent rules, the circles represents an accept rules, and the squares represents deny rules. Edges that connect the vertices represent precedence requirements. As can be seen from Figure 1B, because of the edge between rules r_1 and r_3, precedence between rules r_1 and r_3 must be maintained. In contrast consider the intersection of rules r_3 and r_5. These two rules intersect, indicating packets belonging to the set (UDP, 1.*, *, 1.*, *) would match both rules. However, it does not matter which of the two rules a packet matches first, since the action is the same for both rules. Therefore, an edge does not exist between rules r_3 and r_5 in the diagram. Similarly, rules r_2 and r_4 do not intersect due to the fifth tuple (destination port). A packet cannot match both rules indicating the relative order can change; therefore, an edge will not exist between them.

The match operation can be used to identify precedence relationships, but it cannot do so in every case. Consider a partial-match example [1], where r_a = (UDP, *, 80, 10.*, 90, accept) and r_b = (UDP, 10.*, 80, *, 90, deny). The intersection of r_a and r_b is (UDP, 10.*, 80, 10.*, 90); therefore a packet, such as d = (UDP, 10.10.10.10, 80, 10.10.10.10, 90), can match both rules. If r_a appears before r_b then the packet d is accepted, but if r_b occurs before r_a then d is rejected. As a result, the order of r_a and r_b in the original policy must be maintained. However, the match operation is unable to identify the precedence in this example. A partial match exists in between rules r_3 and r_5 in Table 1, but as previously discussed an edge does not exist between the rules since the actions are the same.

Using the rule list DAG representation a linear arrangement is sought that improves the firewall performance. As depicted in Figure 1B, a linear arrangement (permutation or topological sort) is a list of DAG vertices where all the successors of a vertex appear in sequence after that vertex [23]. Therefore it follows that a linear arrangement of a rule list DAG represents a rule order, if the vertices are read from left to right. Furthermore, it is proven in the following theorem that any linear arrangement of a rule list DAG maintains integrity.

IPR2023-01353 Page 01040

Theorem Any linear arrangement of a rule list DAG maintains integrity.

Proof Assume a rule list DAG *G* is constructed from the security policy *R* that is free of anomalies. Consider any two rules r_i and r_j in the policy, where $i < j$. If an edge between r_i and r_j in *G* does not exist, then a linear arrangement of *G* can interchange the order of the two rules. An edge will not exist if the rules do not intersect; however, a reorder will not affect integrity since a packet cannot match both rules. Shadowing is not introduced due to the reorder since the intersection operation is symmetric. An edge will not exist if the two rules intersect but have the same action; however, a reorder will not affect integrity since the same action will occur regardless of which rule is matched first. If an edge does exist between the rules, then their relative order will be maintained in every linear arrangement of *G*; thus maintaining precedence and integrity.

Rule List Optimization

As described above, it is important to inspect packets as quickly as possible given increasing network speeds and QoS requirements. Using the rule list DAG to maintain policy integrity, a linear arrangement is sought that minimizes the average number of comparisons required. However, this will require information not present in the firewall rule list. Certain firewall rules have a higher probability of matching a packet than others. As a result, it is possible to develop a policy profile over time that indicates frequency of rule matches (similar to cache hit ratio). Let $P = \{p_1, p_2, ..., p_n\}$ be the policy profile, where p_i is the probability that a packet will match rule *i* (first match in the policy). Furthermore, assume a packet will always find a match, $\sum_{i=1}^{n} p_i = 1$; therefore *R* is comprehensive. Using this information, the average number of rule comparisons required is

$$E[n] = \sum_{i=1}^{n} i \cdot p_i \qquad (1)$$

For example, the average number of comparisons required for the rule set in Table 1 is 5.03.

Given a rule list DAG $G = (R, E)$ and policy profile $P = \{p_1, p_2, ..., p_n\}$ a linear arrangement π of G is sought that minimizes Equation 1. In the absence of precedence relationships, the average number of comparisons is minimized if the rules are sorted in non-increasing order according to the probabilities [26], which is also referred to as Smith's algorithm [27]. Precedence constraints cause the problem to be more realistic; however, such constraints also make determining the optimal permutation more problematic.

Determining the optimal rule list permutation can be viewed as job scheduling for a single machine with precedence constraints [15, 21]. The notation for such scheduling problems is $\alpha|\beta|\gamma|\delta$, where α is the number of machines, β is the precedence (or absence of) which can be represented as a DAG, γ is a restriction on processing time, and δ is the optimality criterion [15]. Determining the optimal rule order is similar to the $1|\beta|1|\sum \omega C_i$ scheduling problem, or optimality criterion, where ω is a weight associated with a job (for example, importance) and C_i is the completion time. As previously noted, the $1|||\sum \omega C_i$ problem can be solved in linear time the using Smith's algorithm [27], which orders jobs according to non-decreasing $\frac{t_i}{\omega}$ ratio, where t_i is the processing time of job i. In this case set $t_i = 1$ and $\omega = p_i \ \forall i$. However, Lawler [19] and Lenstra et al. [21] proved $1|\beta|1|\sum \omega C_i$ to be *NP*-complete via the linear arrangement problem, which implies determining the optimal firewall rule order is also *NP*-complete. Note, determining the number of possible permutations has been proven to be *NP*-hard [5].

Theorem $1|\beta|1|\sum \omega C_i$ α *Determining the optimal order of a firewall rule list*

Proof Consider the $1|\beta|1|\sum \omega C_i$ problem. Each of n jobs J_i, $i \in I$, has to be processed without preemption on a single machine that can handle at most one job at a time. For each $i \in I$, let ω be the associated weight. Furthermore, let $G = (V, E)$ be a DAG that represents the precedence order of the jobs J_i. Assume the processing time of each job equals 1 time unit, the weights to be $0 \leq \omega \leq 1$ such that $\sum \omega = 1$, and β, which is G, to be a rule list DAG. In this case, the

optimization criterion $\sum \omega \cdot C_i$ is the same as $\sum p_i \cdot i$, which is given in equation 1. Clearly, the optimal firewall rule ordering problem has a solution if and only if $1|\beta|1|\sum \omega C_i$ has a solution. Therefore, determining the optimal permutation of firewall rules is *NP*-complete.

Exemplary Rule Sorting Algorithm

Although determining the optimal rule permutation was proven to be *NP*-complete, reducing the average number of comparisons required to process a packet remains an important objective. As previously discussed, a sorting algorithm must maintain the precedence relationships among rules. Of course an exhaustive search is possible if the number of rules is small (generate and test all possible topological orders); however as proven in the previous section, this is not feasible with a realistic rule list.

One exemplary algorithm starts with the original the rule set, then sorts neighboring rules based on nonincreasing probabilities. However, an exchange of neighbors should never occur if the rules intersect and have different actions. This test preserves any precedence relationships in the policy. For example, the following sorting algorithm uses such a comparison to determine if neighboring rules should be exchanged. Note a_i denotes the action associated with the ith rule.

```
done = false
while(!done)
    done = true
    for(i = 1; i <n; i++)
        if(p_i < p_i+1 AND (r_i ∩ r_i+1 OR a_i == a_i+1))then
            exchange rules, actions, and probabilities
            done = false
        endif
    endfor
endwhile
```

Other sorting algorithms are possible if the completion time of the sort is an issue; however, the method presented is easy to implement and only requires a simple neighbor comparison. Assume the match probabilities for the rule list given in Table 1. Applying the sorting algorithm to this rule list results in the ordering depicted in Figure 2A, which has 10% fewer comparisons on

average. However when using the algorithm, it is possible that one rule can prevent another rule from being reordered. For example, rule r_1 prevents rule r_5 from being placed closer to the beginning of the rule set. However, if both rules r_1 and r_5 are placed closer to the beginning of the policy while maintaining their relative order, the average number of comparison will be further reduced, 15% fewer. This is the optimal order for the 6 rule set, which is depicted in Figure 2B. Although this simple sorting algorithm is unable to move groups of rules, it can still improve the performance of the firewall system. Its effectiveness is measured experimentally in the next section.

Experimental Results

In this section, the average number of rule comparisons required after sorting the rules is compared with average number of comparisons required using the original order and the optimal order. Note the optimal rule ordering was determined via an exhaustive search. As a result, it was not feasible to determine the optimal ordering once the number of rules equaled 15 given the number of permutations to consider. Two different variations of the rule sorting method described in the preceding section were implemented. The difference between the methods was the comparison used to determine if neighboring rules should be exchanged. The first sorting algorithm exchanged neighboring rules if they were out of order and did not intersect. The if-condition was as follows.

if ($p_i < p_{i+1}$ **AND** $r_i \not\cap r_{i+1}$) **then**

Therefore, rule actions were not considered and the method will be referred to as *non-action sort*. The second sorting method did consider the rule action (as described in the preceding section) and will be referred to as *action sort*. The comparison between the two sorting algorithms will indicate the importance of considering rule actions when ordering rules.

In the first experiment, lists of 10 firewall rules were generated with random precedence relationships. The match probability of each rule was given by a Zipf distribution [20], which assigns probabilities according to the rank of an item. For this simulation, the last rule had the highest probability

IPR2023-01353 Page 01044

which is consistent with most policies (last rules are more general). As a result, the original order yields the worst average number of comparisons. The intersection percentage measured the percentage of rules that intersect in the policy. This metric gives a value for the dependency level in the policy; however note, rule action is not considered when calculating this metric. Rules were equally likely to have an accept or deny action and the results (average number of rule comparisons) for a particular intersection percentage were averaged over 1000 simulations.

Results of the first experiment are given in Figures 3A and 3B. More particularly, Figure 3A is a graph illustrating the average number of packet comparisons versus intersection percentage for the original rule sorting, the non-action sort, the action sort, and the optimal sort. Figure 3B is a graph illustrating the percent difference in the average number of comparisons required between the sorted and optimal firewall policy configurations. The average number of comparisons required was lower for the sorted and the optimal lists when the intersection percentage was low, as seen in Figure 3A. This was expected since there is a large number of possible rule permutations (few edges in the DAG). When the intersection percentage approached 100%, the values converged to the number required for the original list. This is due to the limited number of possible rule orders, only one order in the extreme case. The percent difference between the sorted and optimal order is shown in Figure 3B. At zero intersection percentage the sorted and optimal orders are equal, since any ordering is possible. Similarly at an intersection percentage of 100, the sorted and optimal orders are equal since only one order is possible. Between these two extremes, the action sorting algorithm remains close to the optimal value. In contrast, the non-action sort had a maximum difference of 33%.

The benefit of rule sorting on larger policies is also of interest; however as previously described, it was not possible to determine the optimal ordering once the number of rules approaches 15. Therefore, the second experiment used larger rule sets, but only compared the original order and the sorted orders. The number of rules ranged from 10 to 1000, while the matching probabilities and intersection percentages were the same as the previous

IPR2023-01353 Page 01045

experiment. The results of this experiment are depicted in Figure 4A. The sorted rule sets always performed equal to or better than the original order, while the action sort consistently performed better than the non-action sort. As noted in the previous experiment, the percent difference is very large given few intersections (e.g. 80% decrease for 1000 rules with 0% dependency), but approaches zero as all the rules intersect. As the number of rules increases, sorting is increasingly beneficial, although only at low intersection percentages. The benefit of sorting drops more significantly as the intersection percentage increases with larger rule sets. This is primarily due to the low matching probabilities of each rule, which requires a complete reordering to have a significant impact on the average number of comparisons. As a result, large rule sets can benefit from sorting if the intersection percentage is low.

The impact of considering rule actions when sorting is illustrated in Figure 4B. In this experiment lists of 1000 rules were generated, where the intersection percentages ranged from 0% to 100% and the percentage of rules with the same action varied from 50% to 100%. The performance of the action sort to the original list was then compared. As the percentage of rules with the same action increased, the percent difference (reduction) in the average number of comparisons also increased. This is due to the increased number of permutations possible when the rule actions are increasingly the same (fewer edges in the rule list DAG to consider). This is depicted in Figure 4B where a policy with a 100% intersection percentage can significantly reduce the number of comparisons (80% reduction) if all the rules have the same action. This performance increase is not possible with the non-action sort. Therefore, considering the rule action increases the number of possible rule orders, thereby providing more possibilities to improve firewall performance.

Conclusions

Network firewalls enforce a security policy by comparing arriving packets to a list of rules. An action, such as accept or deny, is then performed on the packet based on the matching rule. Unfortunately packet filtering can impose significant delays on traffic due to the complexity and size of rule sets.

IPR2023-01353 Page 01046

Therefore, improving firewall performance is important, given network Quality of Service (QoS) requirements and increasing network traffic loads.

The sections above describe rule ordering methods to improve the performance of network firewalls. Assuming each rule has a probability of a packet matching, firewall rules should be sorted such that the matching probabilities are non-increasing. This reduces the average number of comparisons and the delay across the firewall. However, a simple sort is not possible given precedence relations across rules. It is common in a security policy that two rules may match the same packet yet have different actions. It is this precedence relationship between rules that must be maintained to preserve integrity. The method described above uses Directed Acyclical Graphs (DAG) to represent the precedence order rules must maintain. Given this representation, a topological sort can be used to determine the optimal order (minimum average number of comparisons); however, the examples above prove this problem to be *NP*-complete (similar to job scheduling for a single nonpreemptive machine with precedence constraints). As an alternative, a simple sorting method was introduced that maintained the precedence order of the rules. Simulation results indicate this method can significantly reduce the average number of comparisons required and is comparable to optimal ordering.

Several areas exist for future research in optimizing firewall rule lists. The sorting method proposed above is based on a simple algorithm. Although it can offer an improvement over the original rule order, algorithms that can move groups of rules could provide a larger reduction in the average number of comparisons. The effect of stateful firewalls should also be addressed in future research. Security can be enhanced with connection state and packet audit information. For example, a table can be used to record the state of each connection, which is useful for preventing certain types of attacks (e.g., TCP SYN flood) [30, 31]. The impact of such rules on the firewall needs to be investigated and whether sorting can be done on-line to reflect temporal changes. In addition, more research is needed to determine more accurate probability distributions for packet matching and dependency percentages.

IPR2023-01353 Page 01047

Given this information, better algorithms can be designed and more realistic simulations can be performed.

Trie-Based Network Firewall Optimization Methods and Rule Splitting Overview

As described above, while assuring firewall policy integrity is critical, performance is equally important given increasing network speeds and traffic volumes. Firewall processing delay can be reduced via hardware as well as policy optimization, where rules are manipulated to reduce the number of comparisons. For a given firewall installation, it can be determined that certain security rules have a higher probability of matching a packet than others. As a result, it is possible to develop a *policy profile* over time that indicates frequency of rule matches (similar to cache hit ratio). Given this information, trie-based methods are described herein to minimize the average number of comparisons required per packet. Although most firewall systems still utilize an ordered list representation, the proposed enhancements still are applicable to current firewall systems since a policy trie can be converted into an ordered list using an inorder traversal.

Rule Sorting

As described above, rule sorting of neighboring rules based on ascending probabilities in a manner that considers rules intersection and rule actions can be used to improve firewall performance by reducing the number of packet comparisons. The methods described above use Directed Acyclic Goal Graphs (DAGs) to represent firewalls. The methods described in this section will use the rule sorting algorithms in combination with policy trie representations to perform sorting of groups of rules in a manner that improves firewall performance.

In this example, it is assumed that a security policy contains a list of n ordered rules, $\{r_1, r_2, ..., r_n\}$. In addition, let p_i represent the probability that a packet will match rule i (first match in the policy). Therefore, the policy is comprehensive, which means every packet will have a match, if $\sum_{i=1}^{n} p_i = 1$.

IPR2023-01353 Page 01048

Using this information, the average number of rule comparisons required can be calculated as

$$E[n] = \sum_{i=1}^{n} i \cdot p_i$$

The average number of comparisons is minimized if the rules are sorted in non-ascending order according to the probabilities. However, this can only be achieved if the sorting algorithm considers the probabilities and whether one rule is a superset of another. For example, sorting algorithms must use the following comparison to determine if neighboring rules should be interchanged Note, a_i denotes the action associated with the ith rule.

if ($p_i < p_{i+1}$ **AND** $r_i \not\cap r_{i+1}$ **OR** $a_i == a_{i+1}$)) **then**

Sorting rules in this fashion can have positive impact on the average number of comparisons required. Figure 5A is a table illustrating a firewall policy prior to rule sorting. The expected number of comparisons for the policy of Figure 5A is 4.26 comparisons per packet. Figure 5B is a table illustrating the firewall policy of Figure 5A after sorting using the above-described algorithm. In Figure 5B, the expected number of comparisons per packet has been reduced to 3.94. Thus, using the method described above, the number of comparisons per packet can be reduced by 8%.

However, it can be seen from Figures 5A and 5B that one rule can prevent another rule from being reordered. For example in Figure 5B, rule r_1 prevents rule r_4 from being placed closer to the beginning of the rule set. However, if both rules r_1 and r_4 are placed closer to the beginning of the policy while maintaining their relative order, the average number of comparison will be reduced.

To solve this problem, rule sets can be sorted a using policy tries. First, the rule list is converted into an equivalent policy trie. Each sub-trie will have an associated probability $\overline{p}$ that is average probability (hit-ratio) of the rules comprising the sub-trie. Sub-tries can be rotated around their parent node to increase performance, using the method described earlier. Since the policy trie (or equivalent rule set) is tested from left to right, rotation should occur if the

probability of the right sub-trie is greater than the left sub-trie. As a result, the average number of comparisons required will be reduced and the policy integrity is maintained. Figure 6A is a trie representation of the rules in Figure 5B. Figure 6B is a table representing resulting rule set after rotating the TCP and UDP sub-tries. The TCP sub-trie in Figure 6A has an expected packet matching probability of 0.095 while the UDP sub-trie has an expected packet matching probability of 0.155. Thus, the UDP sub-trie and the TCP sub-trie should be rotated so that the UDP sub-trie becomes the left sub-trie and the TCP sub-trie becomes the right sub-trie. In the table illustrated in Figure 6B, which lists the ordering of the rules after rotating the sub-tries, the average number of comparisons per packets has been reduced to 3.70.

Rule Splitting

Rule splitting takes a general rule and creates more specific rules that collectively perform the same action over the same set of packets. Here, rule splitting is used to reduce the average number of comparisons. For example in Figure 7A, rule r_5 is split into two separate rules, r_{5a} for UDP and r_{5b} for TCP. Figure 7B is a table corresponding to the policy trie of Figure 7A. Once the rules are positioned based on their probabilities and their relation to other rules, the average number of rule comparisons is reduced to 2.98 (after sort, trie rotation, and splitting) which is a 30% less.

It many not be advantageous to split a general rule since it adds another rule to the policy. For example, assume a policy contains 20 rules where the first 19 rules have the same probability. Assume the last rule can be split and the new specific rule has a probability that is $\frac{3}{4}$ of the last rule. The impact of the probability of the last rule and the location of the new split rule, m, is depicted in Figures 8A and 8B. More particularly, Figure 8A is a graph illustrating the average number of comparisons for firewalls with and without rule splitting. Figure 8B is a graph illustrating average number of comparisons versus different locations of the new rule and probabilities. The average number of comparisons is reduced as the split rule is located closer to the first rule (surface decreases as m approaches one). Furthermore, splitting yields

IPR2023-01353 Page 01050

better results when the general rule has a high probability. However as illustrated in Figure 8B, the closer the specific rule is to the location of the original rule the average number of comparisons increases, which is the penalty of adding one more rule to the policy.

The effect of splitting a single rule can be described mathematically as follows. Consider n rules where rule r_n can be split into rules $r_{n,l}$ and $r_{n,r}$ (whose union is the original rule). In addition, assume the split rule $r_{n,l}$ will be located at the mth position ($1 \leq m < n$), while rule $r_{n,r}$ will remain at the nth location. The goal is to determine the best location m, which yields a lower average number of comparisons as compared to the original rule set. This can be defined mathematically in the following formula.

$$\sum_{i=1}^{n} i \cdot p_i > \sum_{i=1}^{m-1} i \cdot p_i + m \cdot p_{n,l} + \sum_{i=m}^{n-1} (i+1) \cdot p_i + (n+1) \cdot p_{n,r}$$

The left side of the inequality is the average number of comparisons for the original rule set. The right hand side of the inequality is the average number of comparisons with the specific rule at location m. If it is assumed that the rules located between m and n have an equal probability (denoted as p) the previous equation can be solved for m.

$$m < \frac{n \cdot p - n \cdot p_n + (n+1) \cdot p_{n,r}}{p - p_{n,l}}$$

The new rule must be located between the first and m^{th}; however, its final location will depend on the relationship with the other rules (cannot be placed before any rule for which it is a superset). This result can be applied iteratively to multiple rules and repeatedly to the same rule.

Rule Optimization for QoS

In the preceding sections, optimizing the entire security policy has been discussed. However, it may be desirable to optimize the rules for a certain type of traffic, in order to reduce the average number of comparisons this traffic encounters. This can be done by organizing the policy trie via traffic

IPR2023-01353 Page 01051

classification, then optimizing the sub-tries. The result is an ordering where the rules for the most important traffic are tested first.

Network Firewall Optimization Using Rule Compression

Another method for network firewall optimization is referred to as rule compression. Given a security policy R, one objective in optimizing firewall performance is to reduce number of rules in the policy while maintaining the original integrity. A method for reducing the number of rules called *rule compression*, which combines and replaces two rules in the policy will now be described. Although compression does reduce the number of rules, it can only be done if the integrity of the original policy is maintained.

Consider two rules r_i and r_j in policy R, where r_i appears before r_j. The two rules can be compressed if and only if the following conditions are true

1. The two rules intersect, $r_i \cap r_j \neq \emptyset$ (an edge between r_i and r_j exists in the policy DAG) (Tuples that are proper subsets and supersets are considered. Compression may still occur if the rules do not intersect or are adjacent in the set space, the resulting rule may contain ranges.)
2. The action of r_i and r_j is the same.
3. Rule r_i is not dependent on any rule r_k that is directly or indirectly dependent on r_j (only one path from r_i to r_j exists in the policy DAG).

The result of compression is a new rule, $r_{i,j}$ whose tuples are the union of the corresponding tuples of rules r_i and r_j. The new rule $r_{i,j}$ replaces both r_i and r_j in R; however, the location of $r_{i,j}$ in R may require the relocation of other rules. Let D_j be the ordered set of rules that appear after r_i that directly or indirectly depend upon r_j. The new rule is placed at the original location of r_i in R and the rules D_j are placed before $r_{i,j}$.

For example, consider the policy given in Table 2. It is possible to compress rules r_1 and r_5, creating the new rule $r_{1,5}$ =[TCP, *, *, 2.*, *, accept]. The ordered set D_j in this example consists of the rules r_3 and r_4. As a result, the new rule $r_{1,5}$ is placed at the original location of r_1 while the rules in D_j are placed before the new rule. Figures 9A and 9B respectively illustrate the

IPR2023-01353 Page 01052

original policy DAG for Table 2 and the policy DAG after rules r_1 and r_5 are compressed.

No.	Proto.	Source IP	Source Port	Destination IP	Destination Port	Action
1	TCP	1.1.*	*	2.2.*	*	accept
2	TCP	1.*	*	*	80	deny
3	TCP	3.3.*	*	*	80	accept
4	TCP	3.*	*	*	*	deny
5	TCP	*	*	2.*	90	accept

Table 2: Example Policy Consisting of Multiple Ordered Rules

Theorem *Compressing rules r_i and r_j of policy R will maintain integrity if the three conditions are met and method described is used.*

Proof Before r_i and r_j are compressed, relocate the rules D_j immediately before r_i and D_i. This relocation will maintain integrity due to the third requirement for compression (no rule in D_i can be dependent on any rule in D_j). Now place r_j directly after r_i. This relocation will not affect integrity since no rule in D_i can have an edge to r_j (again, due to the third condition). Compressing the neighboring rules r_i and r_j creates $r_{i,j}$. This does not affect integrity since the resulting rule only matches the packets that match r_i or r_j in the original policy. Therefore, the result is the compression of rule r_i and r_j, and the integrity of the policy is maintained.

Determining if Compression Should Occur

Although compression may be possible, it may not improve the performance of the policy due to the relocation of rules. Compression should only occur if the average number of comparisons required for the new policy R_c is less than the original policy R; therefore, $E[R_c] < E[R]$.

IPR2023-01353 Page 01053

Firewall Rule Sub-Trie and Sub-List Ordering

Policy Sub-Trie Ordering

Additional methods for firewall optimization are referred to as firewall sub-trie and sub-list ordering. Given a policy trie T, it may be desirable to order all sub-tries such that the average number of tuple comparisons is reduced and policy integrity is maintained. Consider a policy trie T that contains sub-tries T_i and T_j having the same parent node, as seen in Figures 10A-10C. Let P_i denote the sum of the probabilities of the rules contained in sub-trie i, while C_i denotes the number of comparisons required to completely traverse sub-trie i (which is equal to the number of branches). In order to reduce the average number of tuple comparisons, sub-tries that share the same parent node should be ordered such that the following two conditions are observed.

1. Sub-tries that share a parent node are ordered such that the P_i values are non-ascending (higher match probabilities occur first, from left to right).
2. If sub-tries that share a parent node have the same probability (P_i equals P_j), then order the sub-tries such that the C_i values are non-descending (sub-tries consisting of fewer comparisons occur first, from left to right).

These conditions are recursively applied throughout the policy trie as seen in Figures 11A-11D. However, these conditions do not necessarily maintain policy integrity.

Consider a group of n sub-tries that share the same parent node that are numbered sequentially from left to right. Consider any two non-intersecting sub-tries T_i and T_{i+k} in this group that are out of order. If two sub-tries do not intersect, denoted as $T_i \pitchfork T_j$, then no rule in one sub-trie will intersect with any rule in the other sub-trie, $r_k \pitchfork r_l$, $\forall r_k \in T_i$, $\forall r_l \in T_j$. The following *integrity condition* must be observed when reordering the sub-tries. The two sub-tries can be exchanged (rotated about the parent) if and only the sub-tries T_i through T_{i+k-1} do not intersect with T_{i+k} and the sub-tries T_{i+1} through T_{i+k} do not intersect with T_i. Similar to finding the linear sequence of a policy DAGs, these conditions maintain the policy trie integrity. This is observed in Figure 10A,

where no edge exists between any rule in T_1 and T_2. More particularly, Figure 10A is a policy DAG of a six-rule firewall list. Figure 10B is a policy trie representation of the firewall rule list represented by Figure 10A. In Figure 10B, the average number of tuple compares required per packet is 5.5. Figure 10C illustrates the policy trie after rotating sub-tries T_1 and T_2. In Figure 10C, the average number of tuple compares required is 4.5. In Figures 10A-10C, the following probabilities are assumed:

$P_1 = 0.5$
$C_1 = 4$
$P_2 = 0.5$; and
$C_2 = 2$.

The average number of tuple comparisons is an estimate based on the longest trie path.

Using the conditions for comparing and ordering, the following sorting algorithm sorts neighboring sub-tries that share the same parent node.

```
m = a parent node in T
done = false
while(!done)
    done = true
    for each sub-trie Ti having parent m AND a right neighboring sub-trie
        if((Pi < Pi+1 AND (Pi == Pi+1 AND Ci > Ci+1)) AND Ti ⋔ Ti+1) then
        interchange Ti and Ti+1
        done = false
        endif
    endfor
endwhile
```

Figures 11A-11D illustrate policy trie rotation for an eight-rule firewall policy. More particularly, Figure 11A illustrates the policy DAG representation of an eight-rule firewall policy. Figure 11B is a policy trie representation of the original firewall policy. In this example, ordering occurs in three stages, starting at the bottom level and moving towards the top level of the trie. In Figure 11C, the leaves are ordered at the lowest level. More particularly, nodes r_1 and r_2 are exchanged with nodes r_7 and r_8. The average number of tuple compares for the ordering represented by Figure 11C is 10.45. In Figure 11D, the sub-

IPR2023-01353 Page 01055

tries on the second lowest level (T_3, T_4, T_5, and T_6) are ordered. More particularly, Figure 11D illustrates the policy trie after exchanging sub-tries T_3 and T_4. The average number of tuple compares for the representation in Figure 11D is 10.275. In Figure 11E, the sub-tries on the second highest level, (T_1 and T_2) are ordered. Figure 11E illustrates the final ordering after exchanging T_1 and T_2. In the policy trie of Figure 11E, the average number of tuple compares required is 6.775. The average number of tuple comparisons is an estimate based on the longest trie path.

Policy Sub-List Ordering

The preceding section describes conditions for sorting policy sub-tries, which allow the exchange of groups of rules (sub-tries). The same conditions can be applied to list based policies, where sub-lists are ordered to improve the average number of rule comparisons.

In the sections above relating to rule sorting, a method is described to exchange neighboring rules in a list-based security policy, as seen in Figure 12B. However, it was observed that when ordering rules in this fashion it is possible that one rule can block the exchange of another. If groups of rules were allowed to be exchanged the average number of rule comparisons could be further reduced.

Given a policy list L, it may be desirable to order all sub-lists such that the average number of rule comparisons is reduced and policy integrity is maintained. Consider a policy list L that contains sub-tries L_i and L_j. Let P_i denote the sum of the probabilities of the rules contained in sub-list i, while C_i denotes the number of rules in sub-list i. In order to reduce the average number of rule comparisons, sub-lists should be ordered such that the following two conditions are observed.

1. Sub-lists are ordered such that the P_i values are non-ascending (higher match probabilities occur first, from left to right).
2. If sub-lists have the same probability (P_i equals P_j), then order the sub-lists such that the C_i values are non-descending (sub-lists consisting of fewer comparisons occur first, from left to right).

IPR2023-01353 Page 01056

These conditions are applied throughout the policy list; however as with policy tries, these conditions do not necessarily maintain policy integrity.

Consider a group of n sub-lists that are numbered sequentially from left to right. Consider any two non-intersecting sub-lists L_i and L_{i+k} in this group that are out of order. If two sub-lists do not intersect, denoted as $L_i \pitchfork L_j$, then no rule in one sub-list will intersect with any rule in the other sub-list, $r_k \pitchfork r_l$, $\forall r_k \in L_i$, $\forall r_l \in L_j$. The following *integrity condition* must be observed when reordering the sub-lists. The two sub-lists can be exchanged if and only if the sub-lists L_i through L_{i+k-1} do not intersect with any rule in L_{i+k} and sub-lists L_{i+1} through L_{i+k} do not intersect with L_i. Similar to finding the linear sequence of a policy DAGs, these conditions maintain the policy trie integrity. For example, consider the sub-lists $L_1 = \{r_1, r_3\}$ and $L_2 = \{r_2, r_4\}$ given in Figure 12B. No edge exists between any rule in L_1 and L_2. The two sub-lists are out of order with respect to P and should be exchanged. The resulting list is given in Figure 12C. Figure 12A illustrates the original policy order and results in an average number of rule comparisons of 3.4. In the sorted version in Figure 12B, the average number of rule comparisons is 3.3. In Figure 12C, the average number of rule comparisons has been reduced to 2.7. Using the conditions for comparing and ordering, the following sorting algorithm sorts neighboring sub-lists.

```
done = false
while(!done)
    done = true
    for each sub-list Li having a right neighboring sub-list
        if ((Pi < Pi+1 OR (Pi == Pi+1 AND Ci > Ci+1)) AND Li ⋔ Li+1)then
            interchange rules and probabilities
            done = false
        endif
    endfor
endwhile
```

Summary of Firewall Policy Optimization Techniques and Methods

As stated above, the methods and systems described herein for a network firewall policy optimization can be implemented using any combination of hardware, software, and/or firmware. Alternatively, the steps for firewall policy optimization can be performed manually with the number of rules being optimized is small enough. Figure 13 is a flow chart illustrating an exemplary process for firewall policy optimization according to an embodiment of the subject matter described herein. Referring to Figure 13, in block **1300**, a firewall policy including an ordered list of firewall rules is defined. In block **1302**, for each rule, a probability indicating a likelihood of receiving a packet that matches each rule is specified. In block **1304**, neighboring rules are sorted in order of non-increasing probability in a manner that preserves firewall policy. In block **1306**, groups of rules are sorted in order of non-increasing probability in a manner that preserves firewall policy. For example, any of the policy-trie-based methods described above may be used. In block **1308**, one or more rules are split to reduce the number of average packet comparisons. In block **1310**, intersecting rules having common actions are identified and collapsed into single rules.

As described above, the subject matter described herein may be implemented in hardware, software, and/or firmware. In one implementation, the subject matter described herein for firewall policy optimization may be implemented on a general purpose computing platform. Figure 14 is a block diagram of a general purpose computing platform including software for firewall policy optimization according to an embodiment of the subject matter described herein. Referring to Figure 14, computing platform **1400** may be a general purpose computer, such as a personal computer, that includes a microprocessor **1402**, memory **1404**, and I/O interfaces **1406**. Microprocessor **1400** may be any suitable microprocessor, such as any of the Intel or AMD families of microprocessors. Memory **1404** may include volatile memory for running programs and persistent memory, such as one or more disk storage devices. I/O interface **1406** may include interfaces with I/O devices, such as user input devices and output devices.

IPR2023-01353 Page 01058

In the illustrated example, software that may be resident in memory **1404** includes a firewall policy editor **1408** and a firewall policy optimizer **1410**. Firewall policy editor **1408** may allow a user to define a firewall policy. For example, firewall policy editor **1408** may allow a user to define a firewall policy by specifying different rules. The rules may be specified in a graphical manner, for example using policy DAGs as described above. Alternatively, firewall policy editor **1408** may allow a user to input rules in a tabular manner, as illustrated in any of the tables described herein. Firewall policy optimizer **1410** may implement any or all of the firewall policy optimization techniques to order rules entered via firewall policy editor **1408** in a manner that preserves policy integrity and that enhances firewall performance.

A firewall rule set that is optimized using the subject matter described herein may be implemented on any firewall system that includes one or more firewalls. For example, an optimized firewall rule set according to embodiments of the subject matter described herein may be implemented using any of the hierarchical, multi-node firewall systems described in commonly assigned, co-pending U.S. patent application no. 11/316,331, filed December 22, 2005, the disclosure of which is incorporated herein by reference in its entirety. Figure 15 illustrates an example of a multi-node firewall system suitable for implementing the firewall rules according to an embodiment of the subject matter described herein. Referring to Figure 15, a plurality of firewall nodes **1500**, **1502**, **1504**,and **1506** may collectively implement any of the optimized firewall policies described herein. For example, the firewall nodes may collectively implement different portions of a firewall policy data structure including an ordered list of firewall rules. A firewall policy engine **1508** resident on each firewall node may filter packets using its local portion **1510** of the firewall policy data structure. A control node **1512** may include a firewall policy optimizer **1514** that measures the average number of comparisons per packets for the current rule configuration and may dynamically reorder the rules to reduce the average number of comparisons per packet. For example, firewall policy optimizer **1514** may utilize any of the methods described herein to rearrange rules and improve firewall performance. Rules may be tested for rearrangement at user-specified

intervals or when average number of packet comparisons increases by a user-specified amount.

References

The disclosure of each of the following references is hereby incorporated herein by reference in its entirety.

[1] E. Al-Shaer and H. Hamed. Firewall Policy Management Advisor for Anomaly Detection and Rule Editing. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*, 2003.

[2] E. Al-Shaer and H. Hamed. Modeling and Management of Firewall Policies. *IEEE Transactions on Network and Service Management*, 1(1), 2004.

[3] S. M. Bellovin and W. Cheswick. Network Firewalls. *IEEE Communications Magazine*, pages 50–57, Sept. 1994.

[4] C. Benecke. A Parallel Packet Screen for High Speed Networks. In *Proceedings of the 15th Annual Computer Security Applications Conference*, 1999.

[5] G. Brightwell and P. Winkler. Counting Linear Extensions is #P-Complete. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, 1991.

[6] M. Christiansen and E. Fleury. Using Interval Decision Diagrams for Packet Filtering. Technical report, BRICS, 2002.

[7] D. Decasper, Z. Dittia, G. Parulkar, and B. Plattner. Router Plugins: A Software Architecture for Next-Generation Routers. *IEEE/ACM Transactions on Networking*, 8(1), February 2000.

[8] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink. Small Forwarding Tables for Fast Routing Lookups. In *Proceedings of ACM SIGCOMM*, pages 4 – 13, 1997.

[9] U. Ellermann and C. Benecke. Firewalls for ATM Networks. In Proceedings of *INFOSEC'COM*, 1998.

[10] A. Feldmann and S. Muthukrishnan. Tradeoffs for Packet Classification. In Proceedings of the *IEEE INFOCOM*, pages 397 – 413, 2000.

IPR2023-01353 Page 01060

[11] E. W. Fulp. Firewall Architectures for High Speed Networks. Technical Report 20026, Wake Forest University Computer Science Department, 2002.

[12] E. W. Fulp. Firewall Policy Models Using Ordered-Sets. Technical report, Wake Forest University Computer Science Department, 2004.

[13] R. Funke, A. Grote, and H.-U. Heiss. Performance Evaluation of Firewalls in Gigabit-Networks. In *Proceedings of the Symposium on Performance Evaluation of Computer and Telecommunication Systems*, 1999.

[14] S. Goddard, R. Kieckhafer, and Y. Zhang. An Unavailability Analysis of Firewall Sandwich Configurations. In *Proceedings of the 6th IEEE Symposium on High Assurance Systems Engineering*, 2001.

[15] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan. Optimizing and Approximation in Determinstic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics*, 5:287 – 326, 1979.

[16] A. Hari, S. Suri, and G. Parulkar. Detecting and Resolving Packet Filter Conflicts. In *Proceedings of IEEE INFOCOM*, pages 1203–1212, 2000.

[17] HiPAC. High Performance Packet Classification. http://www.hipac.org.

[18] E. Horowitz, S. Sahni, and D. Mehta. *Fundamentals of Data Structures in C++*. Compuer Science Press, 1995.

[19] E. L. Lawler. Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints. *Annals of Discrete Mathematics*, 2:75 – 90, 1978.

[20] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the Self-Similar Nature of Ethernet Traffic. *IEEE Transactions on Networking*, 2:1 – 15, 1994.

[21] J. K. Lenstra and A. H. G. R. Kan. Complexity of Scheduling under Precedence Constraints. *Operations Research*, 26(1):22 – 35, 1978.

[22] O. Paul and M. Laurent. A Full Bandwidth ATM Firewall. In Proceedings of the 6th European Symposium on *Research in Computer Security ESORICS'2000*, 2000.

[23] B. R. Preiss. *Data Structures and Algorithms with Object-Oriented Design Patterns in C++*. John Wiley & Sons, 1999.

[24] L. Qui, G. Varghese, and S. Suri. Fast Firewall Implementations for Software and Hardware-Based Routers. In *Proceedings of ACM SIGMETRICS*, June 2001.

[25] V. P. Ranganath and D. Andresen. A Set-Based Approach to Packet Classification. In *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, pages 889–894, 2003.

[26] R. Rivest. On Self-Organizing Sequential Search Heuristics. *Communications of the ACM*, 19(2), 1976.

[27] W. E. Smith. Various Optimizers for Single-Stage Production. *Naval Research Logistics Quarterly*, 3:59 – 66, 1956.

[28] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel. Fast and Scalable Layer Four Switching. In *Proceedings of ACM SIGCOMM*, pages 191–202, 1998.

[29] S. Suri and G. Varghese. Packet Filtering in High Speed Networks. In *Proceedings of the Symposium on Discrete Algorithms*, pages 969 – 970, 1999.

[30] R. L. Ziegler. *Linux Firewalls*. New Riders, second edition, 2002.

[31] E. D. Zwicky, S. Cooper, and D. B. Chapman. *Building Internet Firewalls*. O'Reilly, 2000.

It will be understood that various details of the invention may be changed without departing from the scope of the invention. Furthermore, the foregoing description is for the purpose of illustration only, and not for the purpose of limitation.

CLAIMS

What is claimed is:

1. A computer-implemented method for producing a performance optimized firewall policy, the method comprising:

(a) defining a firewall policy including an ordered list of firewall rules;

(b) for each rule, defining a probability indicating the likelihood of receiving a packet matching each rule;

(c) sorting neighboring rules in order of non-increasing probability in a manner that preserves the firewall policy; and

(d) sorting groups of two or more rules in order of non-increasing probability in a manner that preserves the firewall policy.

2. The method of claim 1, wherein sorting neighboring rules in order of non-increasing probability in a manner that preserves the firewall policy includes, for each pair of neighboring rules, rearranging the rules if a first rule of the pair has a lower probability than a second rule of the pair and the first rule of the pair is not a subset of the second rule of the pair with a different action.

3. The method of claim 1, wherein sorting groups of two or more rules in a manner that preserves firewall policy includes assigning each group of rules to a sub-trie in a policy trie, computing a sum of packet matching probabilities for rules in each sub-trie, and, in response to determining that a first sub-trie has a lower sum than a second sub-trie, rotating the first and second sub-tries about a common parent so that rules in the second sub-trie will be applied before the rules in the first sub-trie in the firewall policy.

4. The method of claim 3, comprising, in response to determining that the first and second sub-tries have equal sums, determining whether the second sub-trie has a lower number of branches than the first sub-trie, and, in response to determining that the second sub-trie has a lower number of branches than the first sub-trie, rotating

IPR2023-01353 Page 01063

the first and second sub-tries about a common parent so that the rules in the second sub-trie will be applied before the rules in the first sub-trie in the firewall policy.

5. The method of claim 1, wherein sorting groups of two or more rules in a manner that preserves the firewall policy includes:

(a) identifying first and second sub-lists of rules in the ordered list of firewall rules, the first sub-list preceding the second sub-list in the ordered list of firewall rules;

(b) determining a sum of packet matching probabilities for each of the first and second sub-lists;

(c) determining whether the first and second sub-lists intersect; and

(d) in response to determining that the sum of the packet matching probabilities for the second sub-list is greater than the sum of the packet matching probabilities for the first sub-list and that the first and second sub-lists do not intersect, switching the order of the first and second sub-lists in the ordered list of firewall rules.

6. The method of claim 5, comprising, in response to determining that the sums of the packet matching probabilities for the first and sub-lists are equal, determining whether the second sub-list has a lower number of rules than the first sub-list, and, in response to determining that the second sub-list has a lower number of rules than the first sub-list, switching the order of the first and second sub-lists in the ordered list of firewall rules.

7. The method of claim 1, comprising splitting at least one of the firewall rules to reduce an average number of comparisons required per received packet.

8. The method of claim 1, comprising identifying intersecting rules having a common action and collapsing the rules into a single rule representing a union of the intersecting rules.

9. The method of claim 1, wherein sorting neighboring rules includes sorting rules such that comparisons for a particular class of packets are reduced.

10. The method of claim 1, comprising assigning the rules to vertices in a directed acyclical graph (DAG). determining precedence relationships between the rules,

assigning the precedence relationships to edges that connect the vertices in the DAG, and wherein sorting the neighboring rules in order of non-increasing probability in a manner that preserves the firewall policy includes sorting the neighboring rules in a manner that preserves the precedence relationships represented by the DAG.

11. The method of claim 1, wherein the rules comprise intrusion detection rules.

12. The method of claim 1, wherein the rules comprise intrusion protection rules.

13. A system for firewall policy optimization, for implementing the steps of any one of claims 1-12.

14. A network firewall comprising:

(a) a firewall policy data structure for storing an ordered list of firewall rules;

(b) a firewall policy engine for implementing a firewall policy by comparing packets to the rules in the order specified by the firewall policy data structure; and

(c) a firewall policy optimizer for optimizing performance of the network firewall by reordering the rules in a manner that reduces an average number of rule comparisons per packet and that preserves the firewall policy, the firewall policy optimizer sorting groups of two or more rules in order of non-increasing probability in a manner that preserves the firewall policy.

15. The network firewall of claim 14, wherein the firewall policy optimizer is adapted to measure average number of comparisons per packet for a rule ordering specified by the firewall policy data structure and to dynamically re-order to rules to reduce the average number of comparisons per packet

16. A computer program product comprising computer executable instructions embodied in a computer readable medium for performing steps comprising:

(a) defining a firewall policy including an ordered list of firewall rules;

(b) for each rule, defining a probability indicating the likelihood of receiving a packet matching each rule;

(c) sorting neighboring rules in order of non-increasing probability in a manner that preserves the firewall policy; and

(d) sorting groups of two or more rules in order of non-increasing probability in a manner that preserves the firewall policy.

1/19

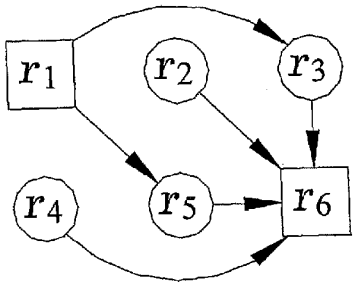


FIG. 1A

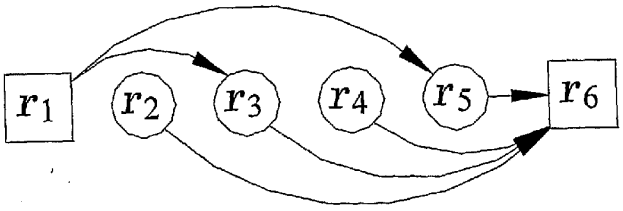


FIG. 1B

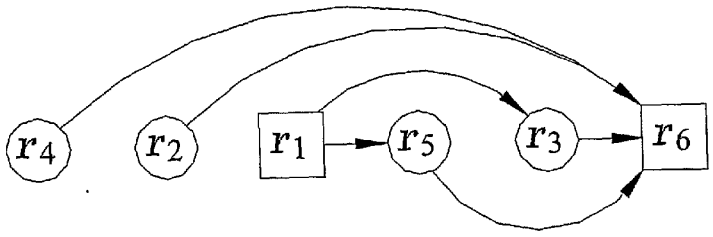


FIG. 2A

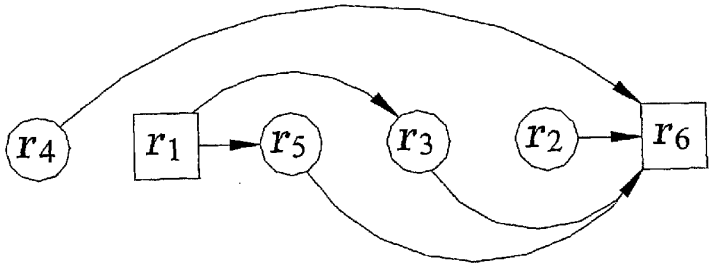


FIG. 2B

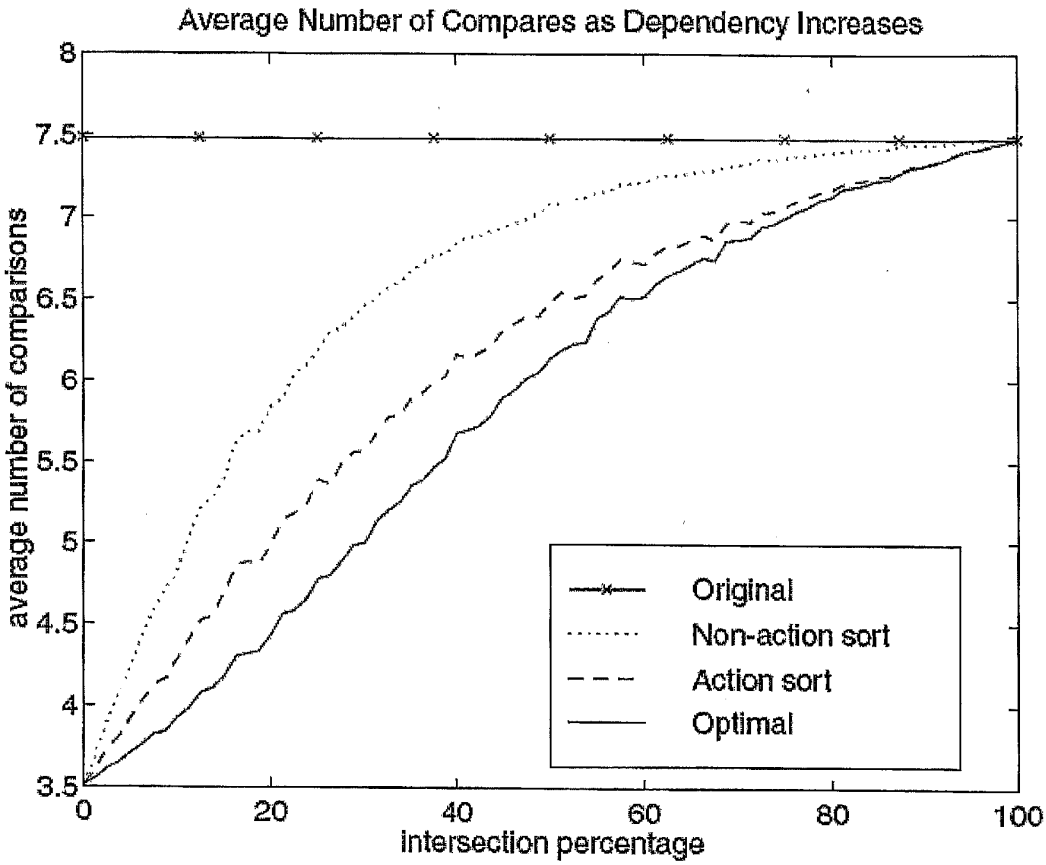

Average Number of Compares as Dependency Increases
average number of comparisons
intersection percentage
8
7.5
7
6.5
6
5.5
5
4.5
4
3.5
0
20
40
60
80
100
Original
Non-action sort
Action sort
Optimal

FIG. 3A

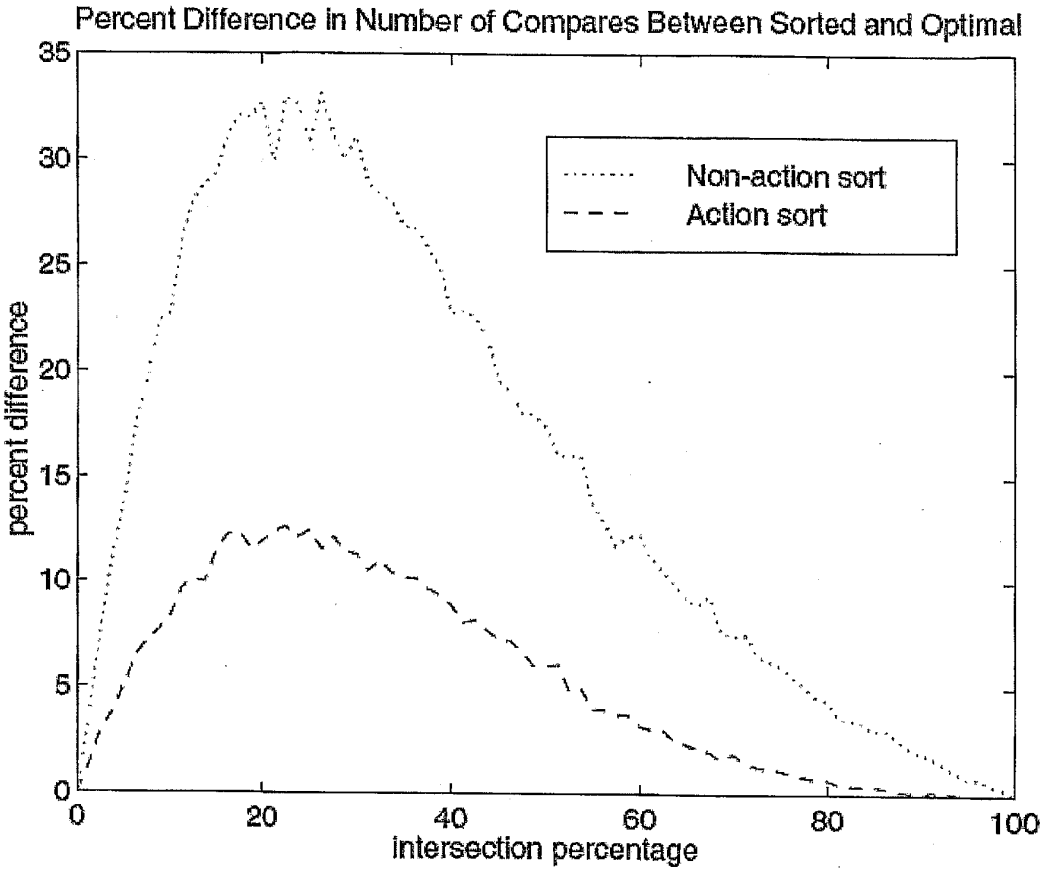

Percent Difference in Number of Compares Between Sorted and Optimal
percent difference
intersection percentage
Non-action sort
Action sort
0
5
10
15
20
25
30
35
0
20
40
60
80
100

FIG. 3B

5/19

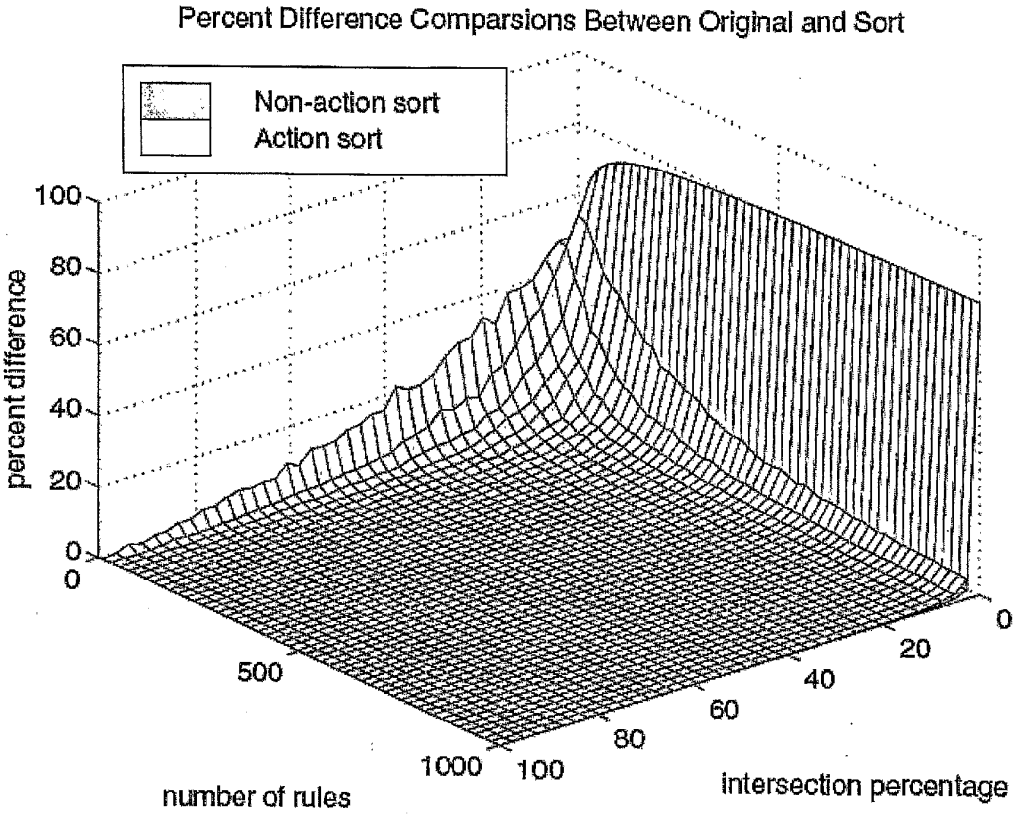

Percent Difference Comparsions Between Original and Sort
Non-action sort
Action sort
percent difference
100
80
60
40
20
0
0
500
1000
number of rules
0
20
40
60
80
100
intersection percentage

FIG. 4A

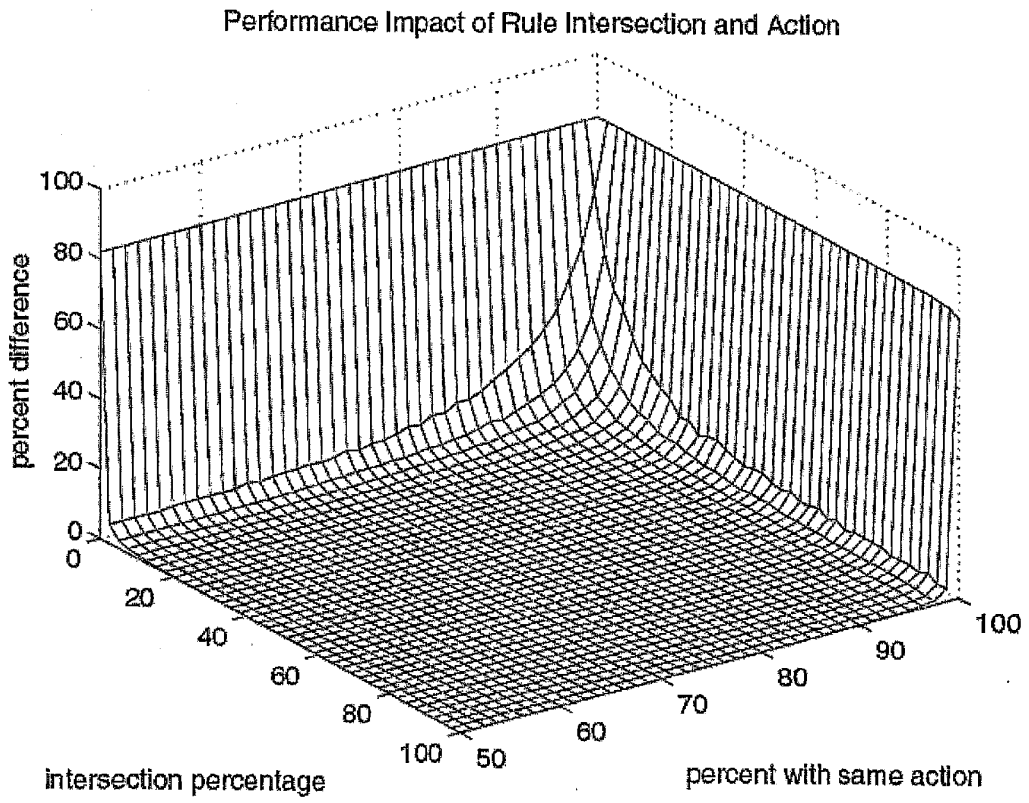

Performance Impact of Rule Intersection and Action
percent difference
0
20
40
60
80
100
intersection percentage
0
20
40
60
80
100
percent with same action
50
60
70
80
90
100

FIG. 4B

7/19

r_1	(UDP,	1.1.*,	ACCEPT)	$p_1 = 0.01$
r_2	(TCP,	1.*,	ACCEPT)	$p_2 = 0.02$
r_3	(TCP,	2.*,	ACCEPT)	$p_3 = 0.17$
r_4	(UDP,	1.*,	DENY)	$p_4 = 0.30$
r_5	(*,	*,	DENY)	$p_5 = 0.50$

FIG. 5A

r_3	(TCP,	2.*,	ACCEPT)	$p_3 = 0.17$
r_2	(TCP,	1.*,	ACCEPT)	$p_2 = 0.02$
r_1	(UDP,	1.1.*,	ACCEPT)	$p_1 = 0.01$
r_4	(UDP,	1.*,	DENY)	$p_4 = 0.30$
r_5	(*,	*,	DENY)	$p_5 = 0.50$

FIG. 5B

8/19

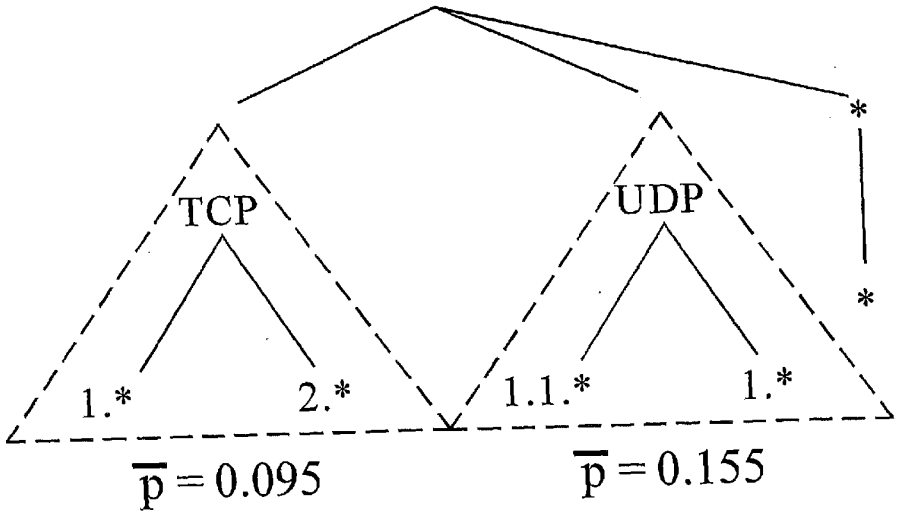


FIG. 6A

r_1	(UDP,	1.1.*,	ACCEPT)	$p_1 = 0.01$
r_4	(UDP,	1.*,	DENY)	$p_4 = 0.30$
r_3	(TCP,	2.*,	ACCEPT)	$p_3 = 0.17$
r_2	(TCP,	1.*,	ACCEPT)	$p_2 = 0.02$
r_5	(*,	*,	DENY)	$p_5 = 0.50$

FIG. 6B

9/19

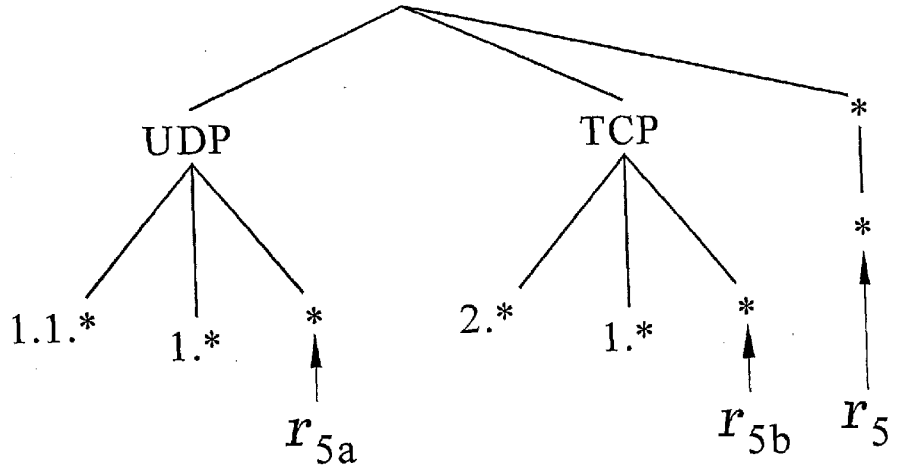


FIG. 7A

r_1	(UDP,	1.1.*,	ACCEPT)	$p_1 = 0.01$
r_4	(UDP,	1.*,	ACCEPT)	$p_4 = 0.30$
r_{5a}	(UDP,	*,	DENY)	$p_{5a} = 0.47$
r_3	(TCP,	2.*,	ACCEPT)	$p_3 = 0.17$
r_2	(TCP,	1.*,	ACCEPT)	$p_2 = 0.02$
r_{5b}	(TCP,	*,	DENY)	$p_{5b} = 0.03$

FIG. 7B

10/19

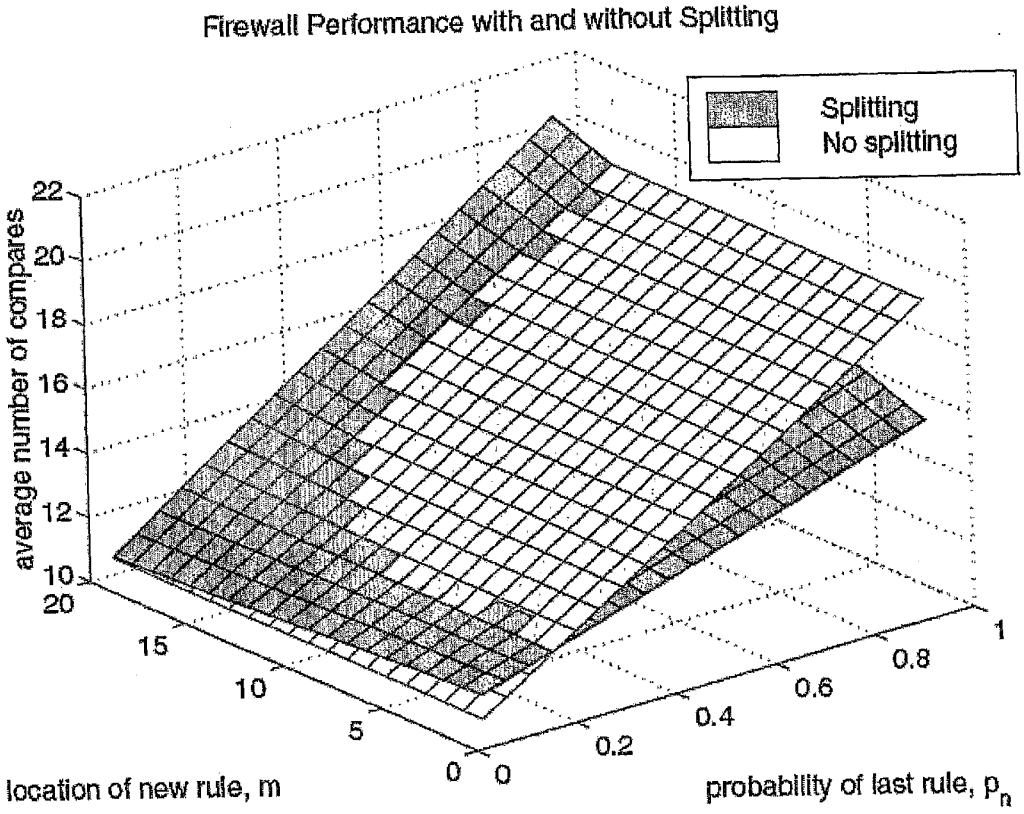

Firewall Performance with and without Splitting
Splitting
No splitting
average number of compares
22
20
18
16
14
12
10
20
15
10
5
0
location of new rule, m
0
0.2
0.4
0.6
0.8
1
probability of last rule, p_n

FIG. 8A

11/19

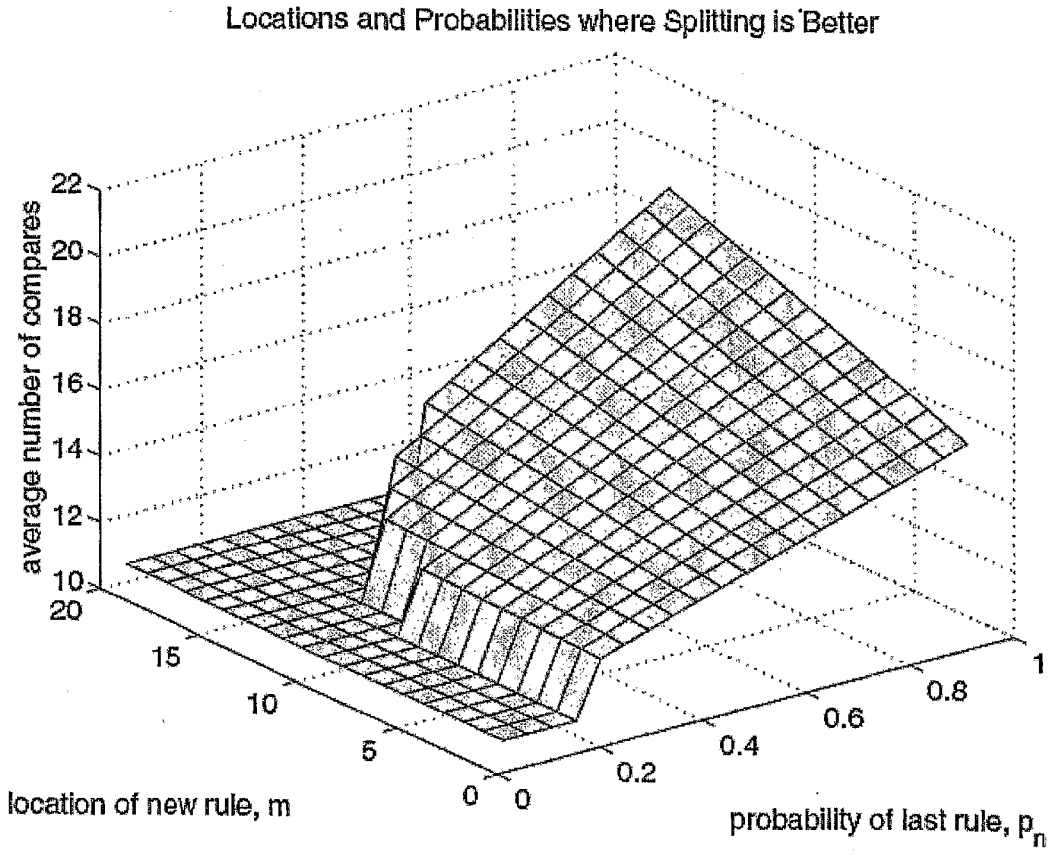

Locations and Probabilities where Splitting is Better
average number of compares
22
20
18
16
14
12
10
20
15
10
5
0
location of new rule, m
0
0.2
0.4
0.6
0.8
1
probability of last rule, p_n

FIG. 8B

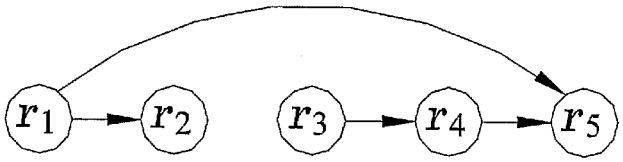


FIG. 9A

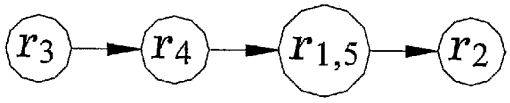


FIG. 9B

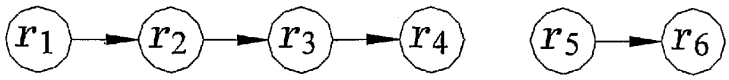


FIG. 10A

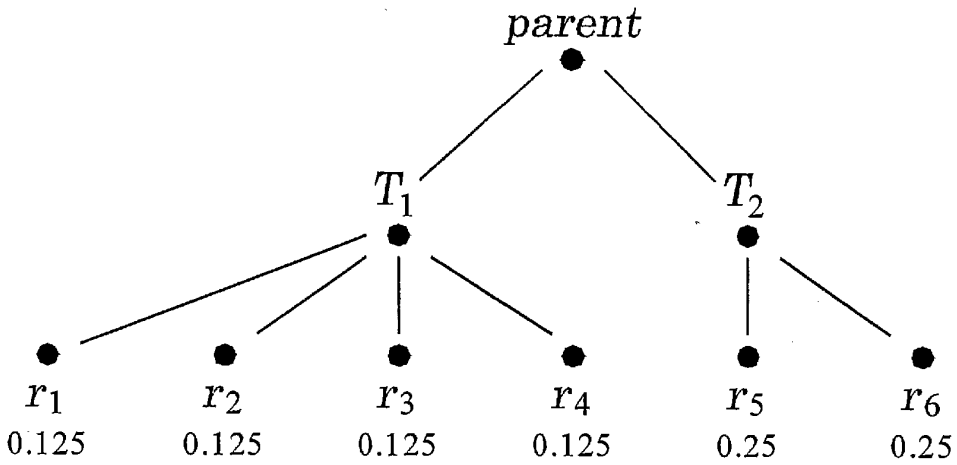


FIG. 10B

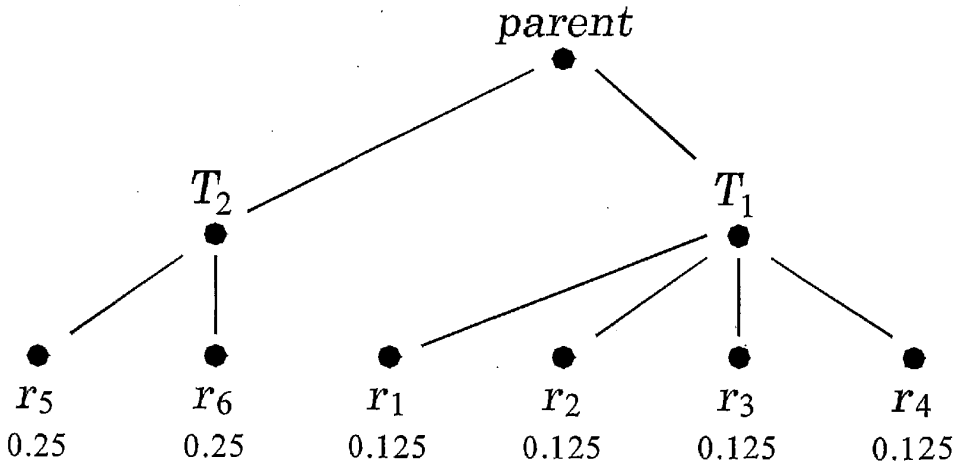


FIG. 10C



FIG. 11A

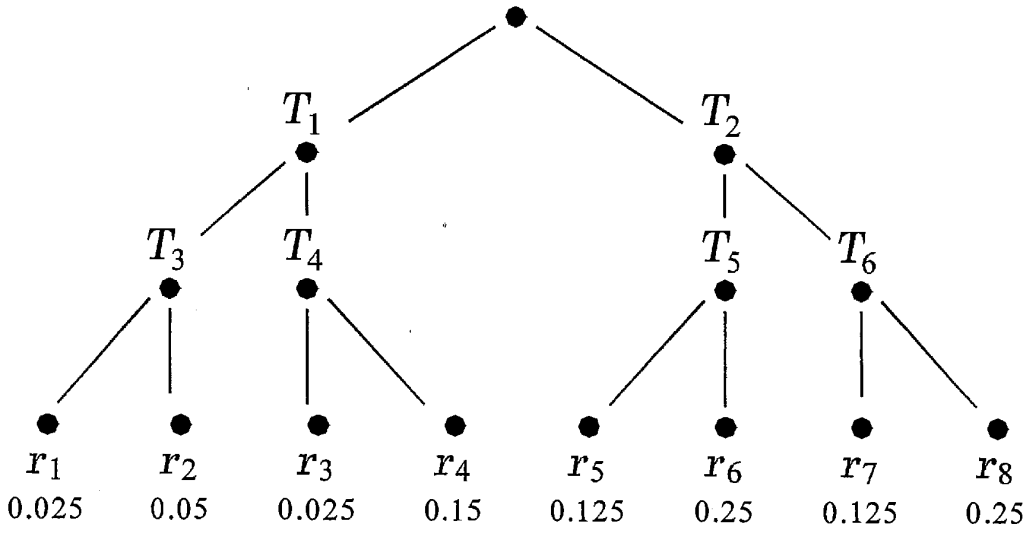


FIG. 11B

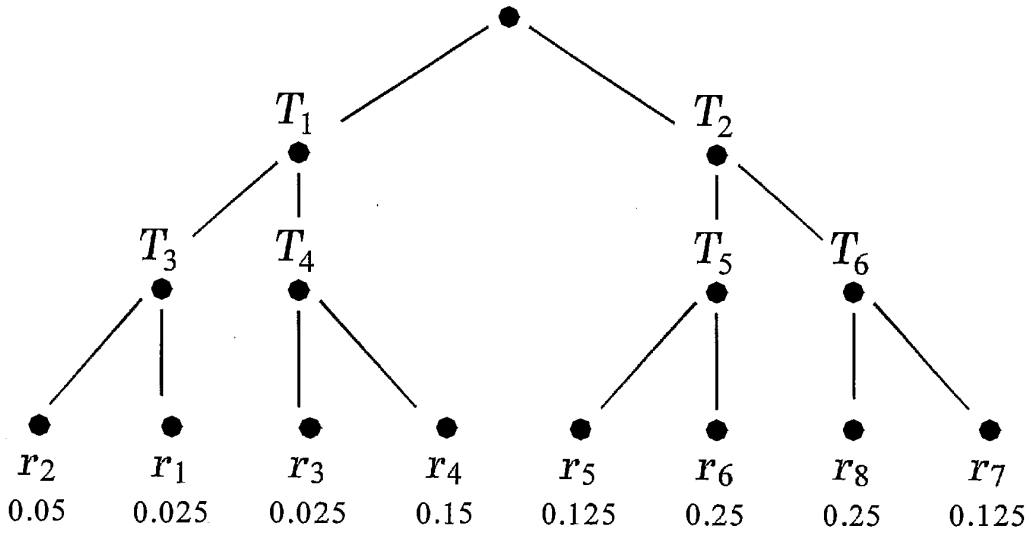


FIG. 11C

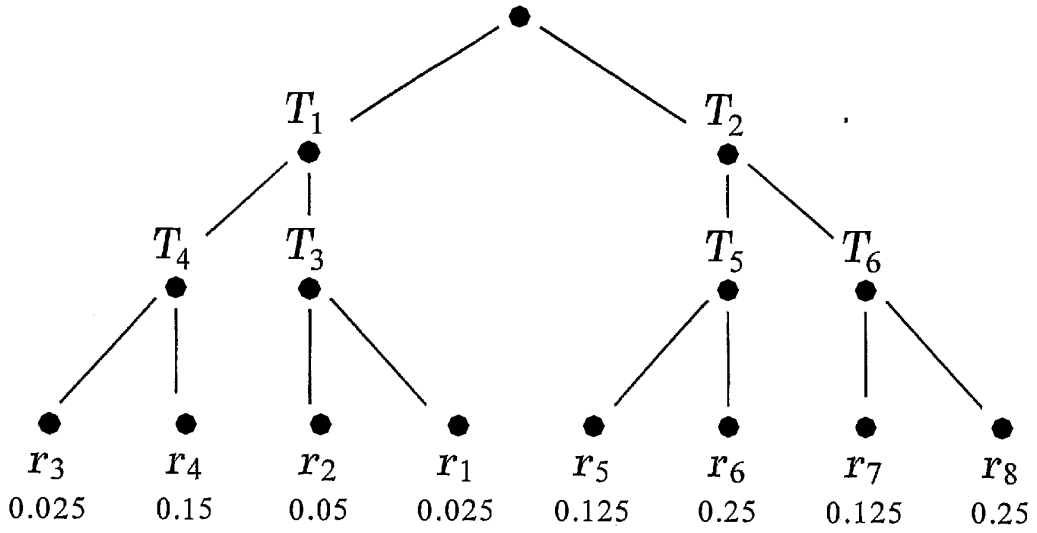


FIG. 11D

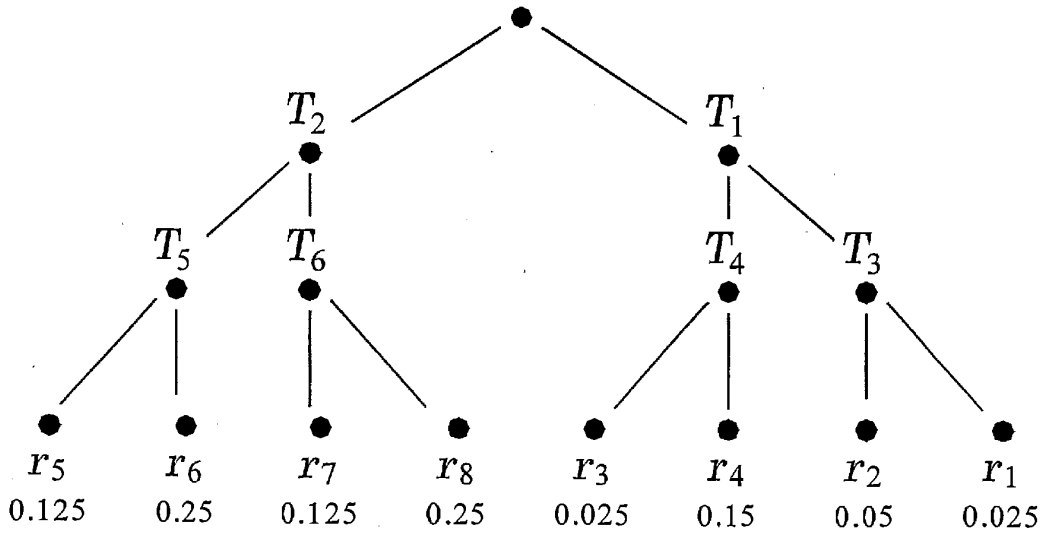


FIG. 11E

16/19

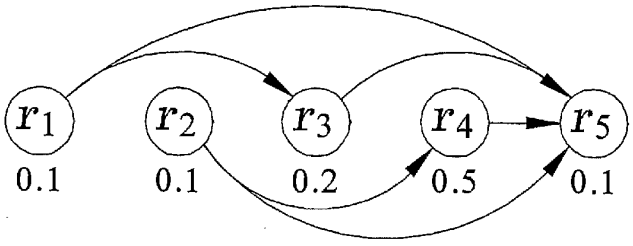


FIG. 12A

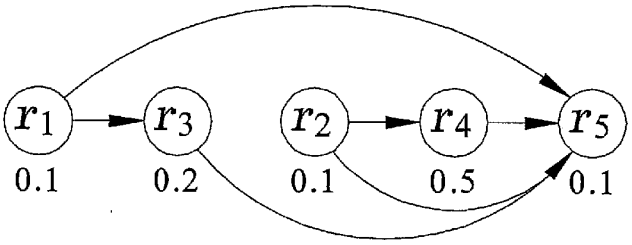


FIG. 12B

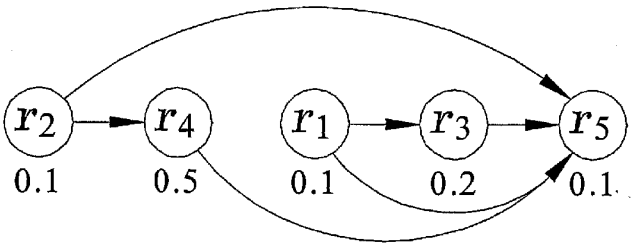


FIG. 12C

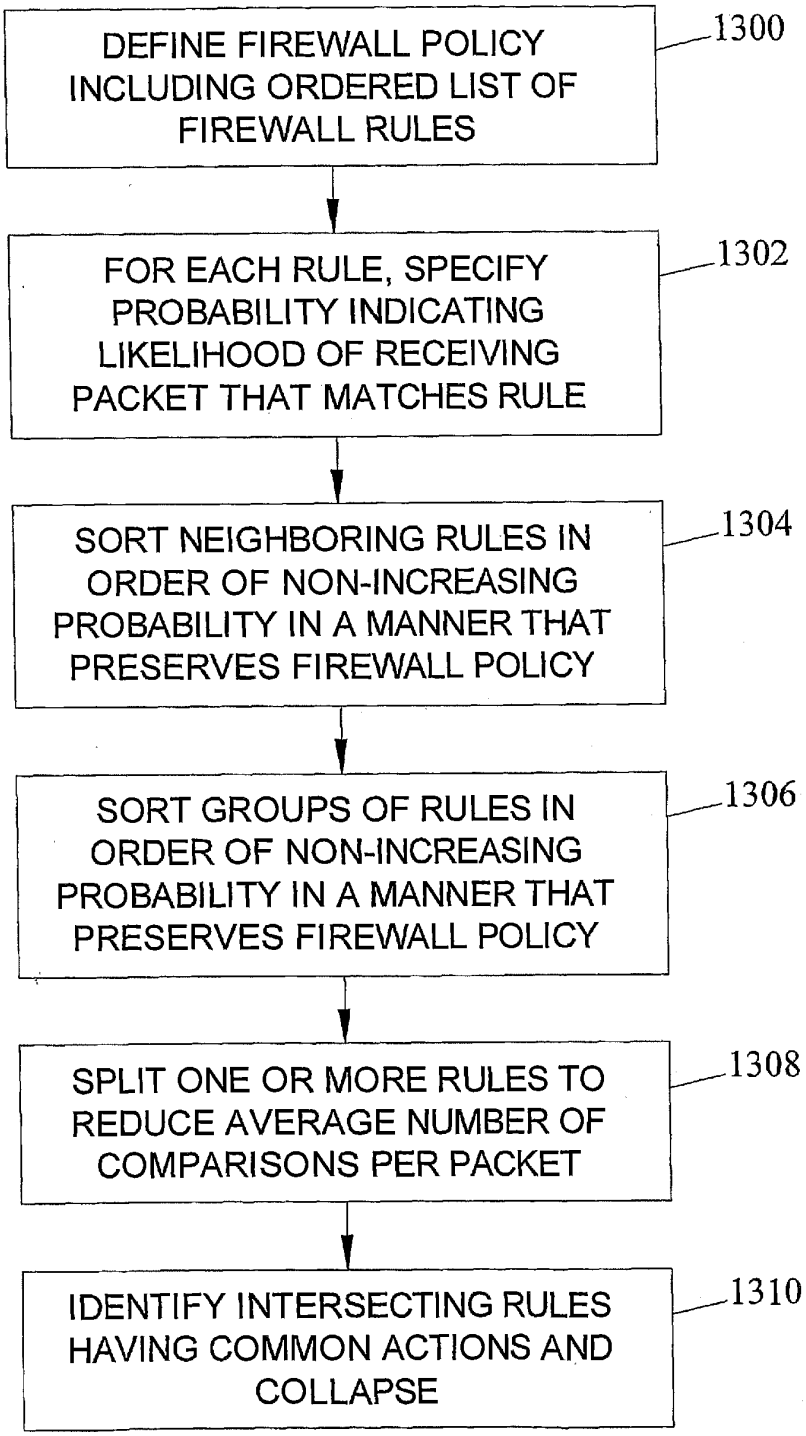

DEFINE FIREWALL POLICY INCLUDING ORDERED LIST OF FIREWALL RULES
1300
FOR EACH RULE, SPECIFY PROBABILITY INDICATING LIKELIHOOD OF RECEIVING PACKET THAT MATCHES RULE
1302
SORT NEIGHBORING RULES IN ORDER OF NON-INCREASING PROBABILITY IN A MANNER THAT PRESERVES FIREWALL POLICY
1304
SORT GROUPS OF RULES IN ORDER OF NON-INCREASING PROBABILITY IN A MANNER THAT PRESERVES FIREWALL POLICY
1306
SPLIT ONE OR MORE RULES TO REDUCE AVERAGE NUMBER OF COMPARISONS PER PACKET
1308
IDENTIFY INTERSECTING RULES HAVING COMMON ACTIONS AND COLLAPSE
1310

FIG. 13

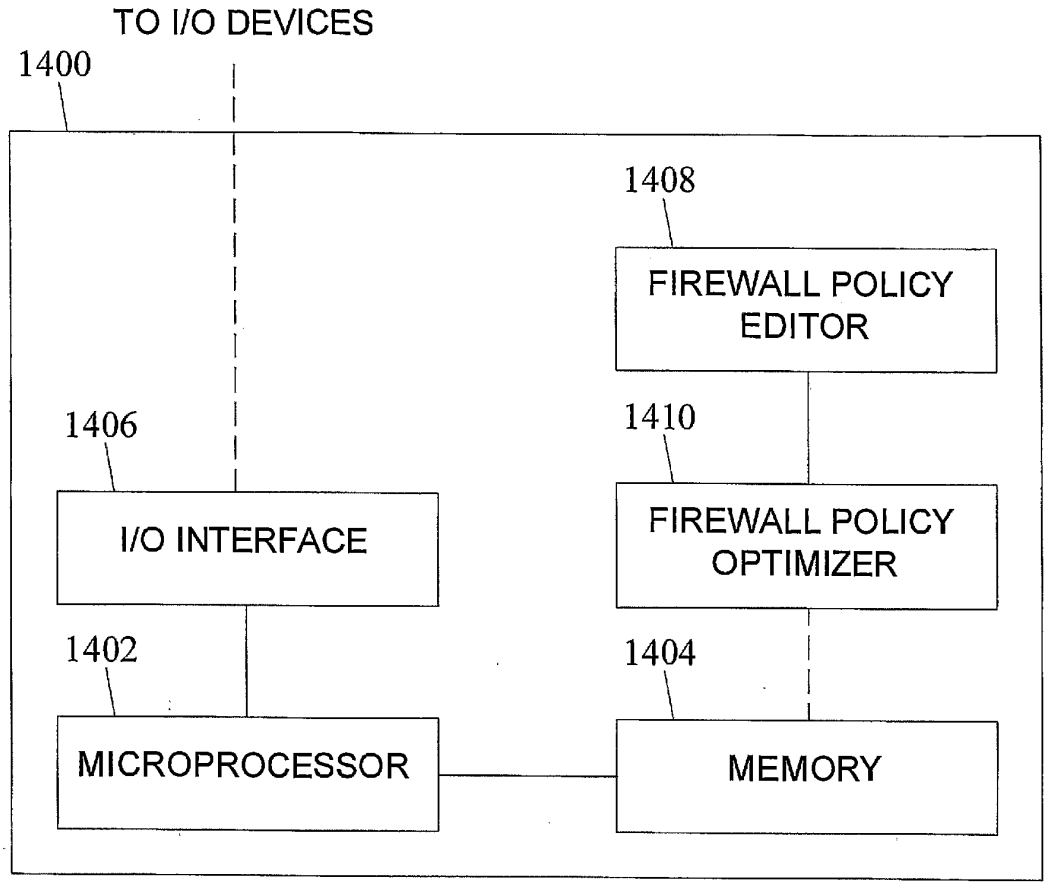

TO I/O DEVICES
1400
1408
FIREWALL POLICY EDITOR
1406
1410
I/O INTERFACE
FIREWALL POLICY OPTIMIZER
1402
1404
MICROPROCESSOR
MEMORY

FIG. 14

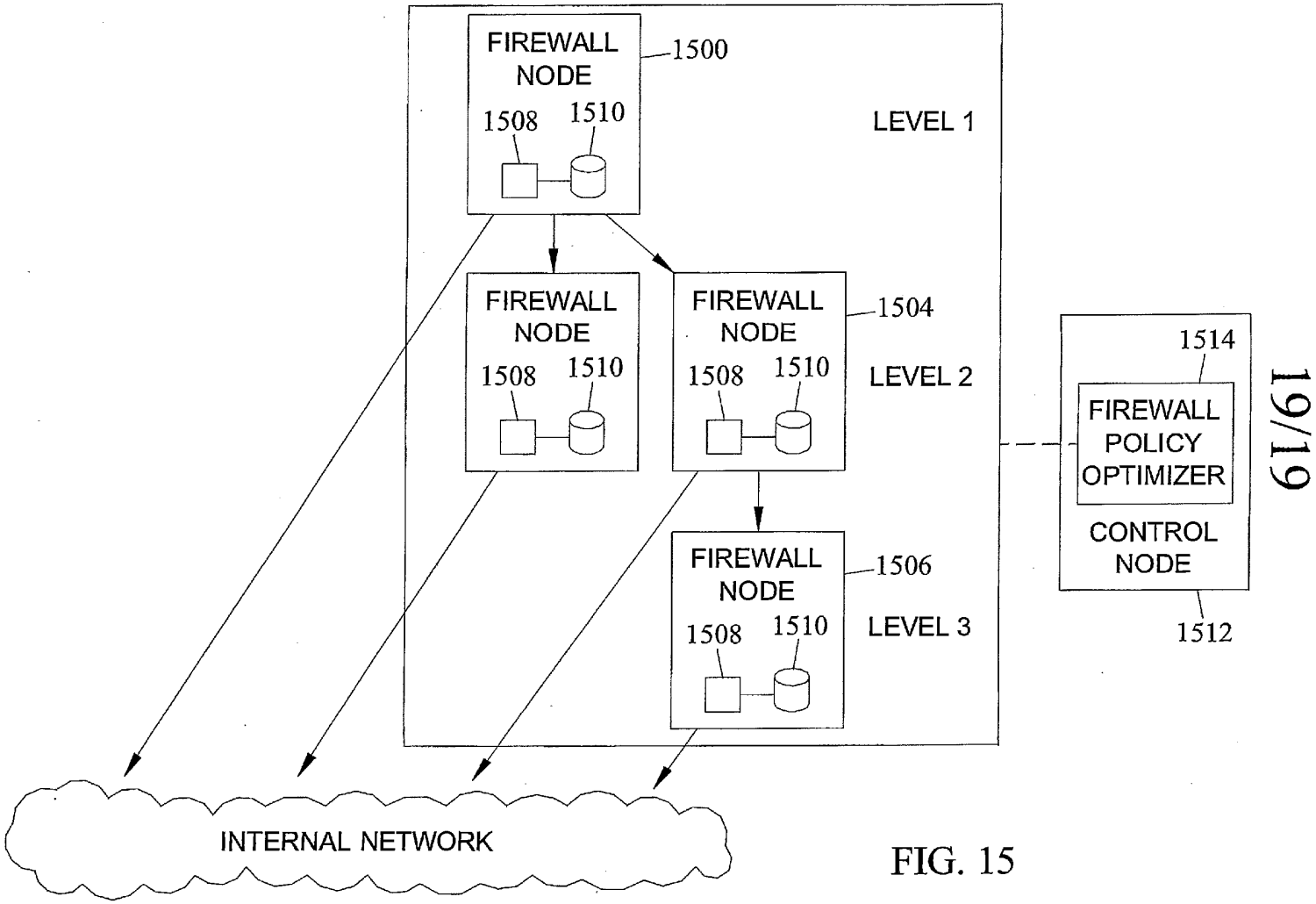

FIREWALL NODE
1500
1508
1510
LEVEL 1
FIREWALL NODE
FIREWALL NODE
1504
1508
1510
1508
1510
LEVEL 2
1514
FIREWALL POLICY OPTIMIZER
CONTROL NODE
1512
FIREWALL NODE
1506
1508
1510
LEVEL 3
INTERNAL NETWORK

FIG. 15

Espacenet

Bibliographic data: WO2012146265 (A1) — 2012-11-01

CORRELATION OF MEDIA PLANE AND SIGNALING PLANE OF MEDIA SERVICES IN A PACKET-SWITCHED NETWORK

Inventor(s): SCHOLZ HENDRIK [DE]; WALLBAUM MICHAEL [DE]; KRUEGER MICHAEL [DE]; BASTIAN JAN [DE] ± (SCHOLZ, HENDRIK, ; WALLBAUM, MICHAEL, ; KRUEGER, MICHAEL, ; BASTIAN, JAN)

Applicant(s): VOIPFUTURE GMBH [DE]; SCHOLZ HENDRIK [DE]; WALLBAUM MICHAEL [DE]; KRUEGER MICHAEL [DE]; BASTIAN JAN [DE] ± (VOIPFUTURE GMBH, ; SCHOLZ, HENDRIK, ; WALLBAUM, MICHAEL, ; KRUEGER, MICHAEL, ; BASTIAN, JAN)

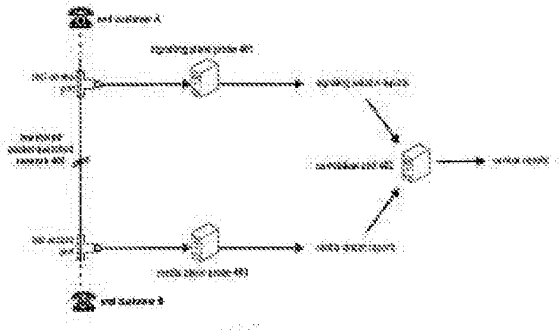
Classification: - **international:** ***H04L12/24; H04L29/06;*** *H04L12/26*
- **cooperative:** **H04L41/5032**; **H04L43/062**; **H04L43/12**; **H04L65/1006**; **H04L65/1009**; **H04L65/1069**; **H04L65/607**; **H04L65/80**

Application number: WO2011EP02145 20110428

Priority number(s): WO2011EP02145 20110428

Also published as: US2014089504 (A1) EP2702740 (A1) CA2834168 (A1)

Abstract of WO2012146265 (A1)



This invention relates to methods for correlating media streams and signaling sessions of services, for example, in a passive monitoring system of a packet-switched network. Furthermore, the invention also relates to an implementation of these methods in hardware and software, and provides a signaling plane probe, a media plane probe and a correlation unit. Moreover, a passive monitoring system comprising one or more of these hardware devices is provided. To correlate media streams and

signaling sessions of services, the invention proposes to independently generate correlation keys in a media plane probe for monitored media streams and correlation keys for signaling sessions that are monitored by a signaling plane probe in a fashion that matching correlation keys are generated for a respective service. By identifying reports on the media streams and reports on the signaling session that contain matching correlation keys respectively, the media streams and the signaling streams are correlated in a correlation unit.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 November 2012 (01.11.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/146265 A1

(51) International Patent Classification:
H04L 29/06 (2006.01) *H04L 12/26* (2006.01)
H04L 12/24 (2006.01)

(21) International Application Number:
PCT/EP2011/002145

(22) International Filing Date:
28 April 2011 (28.04.2011)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **VOIPFUTURE GMBH** [DE/DE]; Wendenstrasse 4, 20097 Hamburg (DE).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **SCHOLZ, Hendrik** [DE/DE]; C/o Voipfuture Gmbh, Wendenstrasse 4, 20097 Hamburg (DE). **WALLBAUM, Michael** [DE/DE]; C/o Voipfuture Gmbh, Wendenstrasse 4, 20097 Hamburg (DE). **KRUEGER, Michael** [DE/DE]; C/o Voipfuture Gmbh, Wendenstrasse 4, 20097 Hamburg (DE). **BASTIAN, Jan** [DE/DE]; C/o Voipfuture Gmbh, Wendenstrasse 4, 20097 Hamburg (DE).

(74) Agent: **BOEHMERT & BOEHMERT**; Felix Hermann, Hollerallee 32, 28209 Bremen (DE).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

(54) Title: CORRELATION OF MEDIA PLANE AND SIGNALING PLANE OF MEDIA SERVICES IN A PACKET-SWITCHED NETWORK

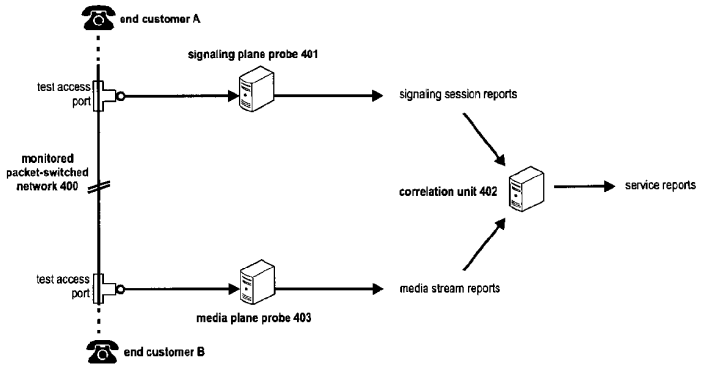


Fig. 4

(57) Abstract: This invention relates to methods for correlating media streams and signaling sessions of services, for example, in a passive monitoring system of a packet-switched network. Furthermore, the invention also relates to an implementation of these methods in hardware and software, and provides a signaling plane probe, a media plane probe and a correlation unit. Moreover, a passive monitoring system comprising one or more of these hardware devices is provided. To correlate media streams and signaling sessions of services, the invention proposes to independently generate correlation keys in a media plane probe for monitored media streams and correlation keys for signaling sessions that are monitored by a signaling plane probe in a fashion that matching correlation keys are generated for a respective service. By identifying reports on the media streams and reports on the signalmg session that contain matching correlation keys respectively, the media streams and the signaling streams are correlated in a correlation unit.

WO 2012/146265 A1

Correlation of Media Plane and Signaling Plane of Media Services in a Packet-Switched Network

FIELD OF THE INVENTION

This invention relates to methods for correlating media streams and signaling sessions of services, for example, in a passive monitoring system of a packet-switched network. Furthermore, the invention also relates to an implementation of these methods in hardware and software, and provides a signaling plane probe, a media plane probe and a correlation unit. Moreover, a passive monitoring system comprising one or more of these hardware devices is provided.

TECHNICAL BACKGROUND

VoIP Services

VoIP ("Voice over IP" – IP denoting the Internet Protocol) services may be considered to consist of a signaling plane and a media plane. On the signaling plane various protocols describe the session (call) flow in terms of involved parties, intermediary VoIP entities (i.e. VoIP proxies, routers) and the characteristics of the VoIP service (call).The media plane typically carries the media information (i.e. audio and/or video data) between the involved parties. Neither the media plane nor the signaling plane alone is sufficient to carry a VoIP service. On the signaling plane protocols like SIP (see IETF RFC 3261, "SIP: Session Initiation Protocol", available at http://www.ietf.org) or ITU-T recommendation H.323 (see H.323, "Packet-based multimedia communications systems", Edition 7, 2009, available at http://www.itu.int) are commonly used, whereas protocols like RTP (Real-time Transport Protocol, see IETF RFC 3550, "RTP: A Transport Protocol for Real-Time Applications", available at http://www.ietf.org), MSRP (see IETF RFC 4975, "The Message Session Relay Protocol (MSRP)", available at http://www.ietf.org) or ITU recommendation T.38 (see T.38, "Procedures for real-time Group 3 facsimile communication over IP networks", Edition 5 (2007) or Edition 6 (2010), available at http://www.itu.int) may be present on the media plane.

In contrast to the traditional PSTN (Public Switched Telephone Network) network both planes may be on different infrastructure using different protocols and even take different routes through a network.

Correlation Information

The property of disjunct signaling stream(s) and media stream(s) of a VoIP service within packet-switched networks may necessitate a correlation mechanism. This correlation may only be based on the common parameters carried on both planes of the session. Monitoring equipment used in VoIP networks may be limited to either the signaling or media plane due to the network structure. A correlation of signaling stream(s) and media stream(s) is thus needed outside the VoIP network in between the monitoring components.

In RTP-based VoIP networks the data basis is rather limited as defined by the RTP protocol and the underlying transport and Internet layer protocols. An RTP stream which typically carries voice or video information from one party to another of a VoIP service is made up of plural RTP packets. The header of each RTP packet does not contain an identifier which can be used to distinctively associate single RTP packets or the entire RTP stream to a VoIP service (defined by the VoIP signaling). The same way the VoIP signaling protocol does not carry information to relate a signaling session (signaling plane of the VoIP service) to RTP protocol header information.

The parties originating or receiving a VoIP service actively dictate the transport and Internet layer address (IP address and port) on which they want to receive media streams. This configuration is done on the signaling plane. Using SIP as a signaling protocol, the address tuple(s) to be used on the media plane is/are defined in a so-called SDP body (see IETF RFC 3264, "An Offer/Answer Model with the Session Description Protocol (SDP)", available at http://www.ietf.org). The address tuple(s) - usually consisting of an IP address and port – is/are the sole information to correlate media plane of a VoIP service with the signaling plane of the VoIP service, and vice versa.

Challenges in Correlation of Signaling Plane and Media Plane

Beside the limited information available for correlation, the correlation of the signaling plane and media plane imposes also numerous further challenges, especially in passive mid-point monitoring scenarios. While an address tuple (IP address and port) may be enough information to identify a single RTP stream at a given point in time, multiple streams

identified by different address tuples may exist in a regular VoIP call. In fact at least two media streams are needed to allow two-way communication between the caller and callee.

It will be outlined in further detail in the following how multiple media streams may occur in a single VoIP service and how they are communicated between the parties involved (which may not always be the case).

Mid-point monitoring equipment (i.e. the signaling and media probes) is introduced into the network path which active components (e.g. VoIP phones and servers) use to communicate with each other. This passive equipment is not involved in the communication itself and has to derive all information from the network packets it observes on said network path. The correlation challenge increases as the passive equipment lacks information the active components internally have. A VoIP phone may internally assign a media port to a signaling session but never send out this information combined so that a mid-point monitoring probe could benefit from this information.

Time Constraints due to Port Reuse

The IP address and port combination transmitted inside the signaling plane of a VoIP service as well as the IP address and port observed on the media plane of the VoIP service may be reused. For exemplary purposes it can be assumed that each VoIP device has a single IP address. The port range from which the client can draw is limited to roughly 65000 ports by the underlying transport layer protocol UDP (see RFC 768, "User Datagram Protocol", available at http://www.ietf.org). In practice additional specifications and configuration parameters significantly further limit this port range. RFC 3550 which defines the Real-Time Transport Protocol mandates that only even port numbers may be used to transport media streams. The odd ports are reserved for Real-Time Transport Control Protocol (RTCP) packets. This standardization alone effectively halves the number of ports available for VoIP communication.

As a result a VoIP device has to re-use a certain IP address and port combination, thus eliminating a unique identification criterion. Essentially the IP address and port combination can only be assumed to be unique for a limited amount of time. The time-frame depends on the utilization of the VoIP device as well as the number of IP addresses required in an Internet Service Provider's network. If a correlation of the signaling and media plane takes place without a time criteria, it would therefore produce so-called false positives: Multiple media streams would match the requested IP address and port combination, since the signaling plane may have caused re-use of the very same IP address and/or port for another VoIP service.

Media Streams

Multiple streams may exist per direction of each VoIP service. The caller sends data to the callee and vice versa resulting in at least one media stream per direction. Desirable features, such as for example codec changes, may trigger additional streams. On the network multiple streams may use the same IP address and port tuple depending on the media protocol. For example, each RTP stream is additionally identified by a unique identifier - the Sender Synchronization Source (SSRC) – which is used to distinguish between multiple streams.

Fig. 1 shows exemplary situations where multiple media streams may be triggered. The focus is on the media plane traffic and its timing. The x-axis denotes the time basis, starting on the left side. The vertical lines in Fig. 1 denote signaling events and the bold horizontal lines are RTP streams present in the respective exemplified VoIP service example (please note that the different VoIP service examples are separated by broken horizontal lines). Only the RTP stream direction from the callee to the caller is shown per VoIP service example. The RTP stream from the caller to the callee can be assumed to be less complex (e.g. as shown in the "normal" VoIP service example).

The three vertical lines denote important events in the phase of a call. The leftmost line denotes the start timestamp. This is the time at which the caller submitted the call setup request to the network. The connected point is the timestamp at which a callee or his endpoint/handset picked up and moved the service into a fully established state. The service ends when one of the parties sends a termination request.

The most common VoIP services will be the trivial session examples where there is only one RTP stream per direction (see "normal" or "early media"). In a "normal" VoIP service flow both parties only start sending once the session is fully established on the signaling plane. More often the callee would start sending early media, i.e. when sending ring-tones or (pre-call) announcements. In "session forking" VoIP service example there may be multiple concurrent or sequential RTP streams at the beginning of a service e.g. due ring-tones being sent to multiple VoIP devices of the callee, while one is answered at the connect event

During a service media codecs may be changed (see "code change" VOIP service example) which is commonly resulting in a new media stream. Another service scenario may be the VoIP call being put on hold and picked up at a later time essentially generating multiple media streams (see "call on hold" VoIP service example). Silence suppression and other RTP features may cause multiple streams sporadically during a service (see "silence suppression" VoIP service example). Also retransmissions on the signaling plane may

desynchronize the signaling plane and media plane (see "SIP retransmissions" VoIP service examples).

Geographic Distribution or Media Plane and Signaling Plane

As indicated above, signaling plane data and media plane data of VoIP services may take different routes. In examples, where SIP is used for session setup, this is indeed the case as those sessions usually contain signaling-only components (i.e. SIP proxies, REGISTRARs). As a result passive mid-point monitoring solutions may be split geographically along the routes of the VoIP traffic.

An exemplary passive mid-point monitoring scenario for this case is shown in Fig. 2. Fig. 2 shows three geographically distributed POPs (Point of Presence) which are linked over the carrier's internal network. The signaling data flows on the signaling plane between POP A and B via the signaling-only POP C. POP C would typically host a centralized routing entity which reduces the complexity in the POPs A and B. To prevent back-hauling traffic from POP A to POP C before sending it to POP B the VoIP system is configured in a way that would allow the media to flow directly on the media plane between POPs A and B.

Typical mid-point monitoring locations would be inside each POP. Due to the layout of the network no media traffic would be visible in POP C. To allow for successful correlation of signaling and media streams by the correlation mechanism it is therefore desirable to support distributed networks. If the same session traverses multiple POPs it would be desirable for the correlation mechanism to identify this and support correlation of the session legs across multiple sites.

Limitations/Challenges in Correlation of Signaling Plane and Media Plane

The correlation may be solely based on the IP address and port used on the transport and Internet layer. On the media plane each RTP stream is destined to exactly one IP address and port on the transport layer. On the signaling plane one IP address and port is typically exchanged inside the Session Description Body of the SDP protocol during the VoIP service setup. However this IP address and port combination may not be the combination used throughout the VoIP service. In this case the above mentioned correlation mechanism fails: It fails upon searching a media stream for a specific signaling session of the VoIP service. The same way it fails to locate a signaling session of the VoIP service for a specific media stream on the media plane. The reasons for using another IP address port combination than advertised upon session setup are manifold.

Network Address Translation (NAT)

Network Address Translation (NAT, RFC 1631) was developed to solve Internet address space depletion and routing problems. In today's Internet one may distinguish between internal addresses which are often only unique locally (i.e. typically used inside company or home networks), and external addresses of global validity and uniqueness as used on the public Internet.

Fig. 3 shows an exemplary network using internal and external addresses. The internal network on the left side uses IP addresses from the so-called private space, as for example defined in IETF RFC 1918, "Address Allocation for Private Internets", available at http://www.ietf.org. The external network on the right side is the public Internet with globally reachable addresses. A router with embedded Network Address Translation (NAT) functionality is located in between the two networks. The router does not directly route packets in between the two connected networks, but "translates" (i.e. replaces) IP source and destination addresses within the routed IP packets on demand.

Essentially, the NAT functionality of the router hides the entire internal network (10.0.0.0/8 in this example) behind one public IP address (192.0.2.1). Every IP connection from within the internal network to the Internet will seem to originate from the sole public IP address (192.0.2.1) of the entire network.

For each node inside the internal network this imposes some limitations. Although Internet access works in general, a node in the internal network may not be reachable from the outside. This is due to the fact that IP traffic from an outside node (i.e. a node in the external network) would be destined to the public IP address (192.0.2.1), but the router with this IP address does not know at this point in time where to forward the traffic to on the internal network, given that there was no prior communication from an node inside of the internal network and the external node.

Additionally the internal node may not be aware that it is behind a router which performs the NAT. As required by the transport protocols, the internal node will use the internal IP address for communication purposes. On the VoIP signaling and media layer (i.e. session layer or application layer), the application will also use the internal IP address (i.e. 10.0.0.123) instead of the public IP address. VoIP signaling protocols (e.g. SIP or H.323) contain various IP addresses on the application layer. An intermediary NAT device (i.e. the router in Fig. 3) will not perform translation of the addresses on the session layer or application layer, but only translates IP addresses in the network layer. The internal IP addresses inside the VoIP

signaling (i.e. SIP or H.323 messages) impose a problem to the application as internal IP addresses are most likely not reachable from the public Internet.

VoIP devices involved in the processing of a certain call may detect this problem by comparing the IP addresses seen on the Internet and transport layers with the IP addresses set on the application layer. The following message is an exemplary SIP message as seen on the network and transport layer, which means that both the Internet and transport layer information (IP addresses and ports) are visible:

```
U 2010/09/09 13:57:31.925226 192.0.2.99:37682 -> 10.0.0.1:5060
```

The Internet and transport layers show that the UDP packet (U) was received from the external address 192.0.2.99 (port 37682).

The following message represents the same exemplary SIP message as seen on the application layer, which means that both the Internet and transport layer information (IP addresses and ports) as well as the application layer (SIP message contents) are visible:

```
INVITE sip:echo@example.com SIP/2.0
Via: SIP/2.0/UDP 192.168.1.216;branch=z9hG4bK3BB64874;rport=37682;received=192.0.2.99
CSeq: 7726 INVITE
To: <sip:echo@example.com>
Content-Type: application/sdp
From: "John Doe" <sip:jdoe@example.com>;tag=693C7725
Call-ID: 1341723561@192.168.1.216
Subject: sip:jdoe@example.com
Content-Length: 230
User-Agent: kphone/4.2
Contact: "John Doe" <sip:jdoe@192.168.1.216;transport=udp>
v=0
o=username 0 0 IN IP4 192.168.1.216
s=The Funky Flow
c=IN IP4 192.168.1.216
t=0 0
m=audio 60942 RTP/AVP 0 97 8 3
a=rtpmap:0 PCMU/8000
a=rtpmap:3 GSM/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
a=fmtp:97 mode=30
```

In contrast to the information available at the network and transport layers claiming that the message was received from the external address 192.0.2.99 (port 37682), the application claims that the source was the internal address 192.168.1.216 (port 5060 - the port 5060 is not viewed for the source address as it is the default port).

In general the translation of the network (IP) address also applies to the port numbers which are used on the transport layer. This feature is named Port Address Translation (PAT). NAT and PAT commonly happen at the same time in the same device.

PAT becomes necessary if multiple devices attempt to use the same port. Since different devices have different IP addresses their IP address and port tuple is unique. If packet streams from both devices traverse the same NAT device, the NAT device has to translate the port of one of the two packet streams. Otherwise there would be two distinct packet streams sharing the same external IP address of the NAT device and the same port. While this may not be a problem if the destination IP address and port tuple of the destination device are different for the two packet streams, it adds overhead which the NAT device wants to prevent. By rewriting the source port of one of the packet streams the address tuples of the external IP address and ports become unique again.

However, in networks where NAT (and PAT) is performed to for example hide private networks from the outside Internet, the correlation of signaling sessions of VoIP services and the associated media plane streams is becoming complex.

Multiple Communication Paths

Numerous solutions have been developed to allow traffic of VoIP services to operate in networks with Network Address Translation (NAT). These mechanisms require the involved caller and callee to exchange information, for instance on additional transport protocols or addresses which can be used to reach the remote party. These include – but are not limited to - IPv6 (Internet Protocol Version 6, see IETF RFC 2460, "Internet Protocol, Version 6 (IPv6), Specification"), TURN (Traversal Using Relays around NAT, see IETF RFC 5766, "Traversal Using Relays around NAT (TURN):Relay Extensions to Session Traversal Utilities for NAT (STUN)") addresses or ICE (Interactive Connection Establishment, see IETF RFC 5245, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols") candidates (all RFCs available at http://www.ietf.org).

These additional tuples of IP addresses and ports communicated in these solutions establish different data paths in media plane and signaling plane of a VoIP service and thereby break

up the previous one-to-one relation of signaling session to media stream to a given VoIP device. Previously one media stream to the given VoIP device could be distinctively identified by exactly one IP address and port. In the new scenario on the signaling plane a list of IP address and port tuples is transported on which the VoIP device may receive media thereby increasing the complexity of correlation between media plane and signaling plane of a VoIP service.

SUMMARY OF THE INVENTION

One object of the invention is to suggest correlation mechanisms that allow overcoming one or more of the limitations and challenges on the correlation of signaling plane and media plane of services.

The object is solved by the subject matter of the independent claims. Advantageous embodiments of the invention are subject to the dependent claims.

One of the various aspects of the invention is to suggest a definition of correlation keys that allow the correlation of media streams of services and the associated signaling session of the respective services based on data records (reports) that provide information on the media plane and signaling plane of the respective services. These correlation keys may be for example provided in form of hash keys. A correlation key may be associated to a given direction of the media data paths of the media streams belonging to a given service (e.g. caller-to-callee or callee-to-caller for a simple bi-directional service, e.g. VoIP call). Hence, assuming that the service comprises at least two media streams of different directions, at least two correlation keys are provided on the signaling plane for later correlation of the media plane and signaling plane of the service. On the media plane one correlation key is provided per media stream.

The correlation keys for use in correlation are generated by a media plane probe and a signaling plane probe independently from each other, but the two probes generate matching correlation keys for a given media service (i.e. the signaling plane probe and the media plane probe independently each generate the same correlation key for a given direction/media data path of the media service, given that there are not any exceptional situations preventing the match of the correlation keys).

In response to the detection of the end of a signaling session, the signaling plane probe generates for this signaling session at least one correlation key for each "direction" of the service to which the signaling session belongs. Each correlation key (e.g. hash key) is generated by the signaling plane probe based on a timestamp indicative of the point in time

at which the end of the respective signaling session has been detected by the signaling plane probe and using information on respective direction of the media plane data flow (e.g. caller-to-callee or callee-to-caller for a simple bi-directional service, e.g. VoIP call). The correlation keys generated for the different directions of the service corresponding to the signaling session the end of which has been detected are then collated in a signaling session report, which may be for example stored in a database or a flat file.

For the media plane, the media plane probe generates a correlation key (e.g. hash key) for each media stream. The correlation keys for the media steams are generated upon the media plane probe detecting the end of the respective media stream. The correlation key for a respective terminated media stream is generated by the media plane probe based on the direction of the respective media stream, e.g. based on the media data path of a media stream, and on a timestamp indicative of the point in time at which the respective media stream has ended. The timestamp at which the respective media stream has ended is for example the time at which the last packet of the media stream has been received by the media plane probe. The media plane probe stores the correlation key (e.g. hash key) for the media stream in a media stream report, which may be for example stored in a database or exported as a flat file.

Another aspect of the invention is the use of the correlation keys comprised in the signaling session reports and the media stream reports to identify the media streams and signaling session belonging to a single media service. As indicated above, the correlation keys generated by the signaling plane probe and the media plane probe should be matching each other (i.e. the correlation keys of a given signaling session report should be found in the media stream reports, respectively, the correlation key of a media stream report should be found in one of the signaling session reports), provided that signaling session and related media streams are terminated nearly at the same point in time, i.e. within a given timeframe. Hence, a correlation unit can identify which media streams belong to which signaling session (or vice versa) and may therefore generate a service record that summarizes a service based on information contained in the media stream reports and information contained in the signaling session report of the respective service.

According to one exemplary embodiment of the invention, a method for correlating media streams and signaling sessions of services in a passive monitoring system of a packet-switched network is provided. It is assumed for exemplary purposes that one service has one signaling session.

In this method a signaling plane probe generates at least one correlation key for each direction of a service based on the information of the signaling session of the respective

service monitored at the signaling plane probe. As noted above the direction may be equivalent to a respective media data path of a media stream belonging to a given service. The correlation keys for a respective service may be generated using a timestamp indicative of the point in time at which the end of the respective signaling session of the respective service has been detected by the signaling plane probe and using information on the direction of the respective service. Furthermore, the signaling plane probe stores a signaling session report for each signaling session in a database. Each of the signaling session reports comprises at least two correlation keys generated for the respective directions of the service.

Moreover, in this method a media plane probe also generates a correlation key for each media stream monitored by the media plane probe. The media plane probe generates the correlation key using a timestamp indicative of the point in time of the end of the respective media stream as detected by the media plane probe and using information on the media data path of the respective media stream. The media plane probe stores a media stream report for each media stream in a database, wherein each media stream report comprises at least the correlation key generated for the respective media.

A correlation unit further correlates the media streams and the signaling sessions based on the correlation keys comprised within media stream reports and signaling session reports.

In a more detailed exemplary embodiment of the invention, the generation of the respective correlation keys for the respective media streams by the media plane probe is based on the media data path information of the respective media stream and the timestamp indicative of the point in time at which the respective media stream ended.

According to another exemplary embodiment of the invention, information on the direction of the respective service – based on which the correlation keys are generated by signaling plane probe for a respective service – is information on one or more media data paths of media streams of the service learned from messages of the signaling session of the service. In other words, the direction of the service may be for example given by the media data path which is yielding a given direction. For example in a simple VoIP call between two peer devices, there are two directions: caller-to-callee and callee-to-caller as noted above.

In a further embodiment of the invention, the signaling plane probe monitors signaling messages of the respective signaling sessions of the services, and detects media streams belonging to each service associated to a respective monitored signaling session based on the signaling messages of the respective signaling sessions. The correlation keys for a respective one of the signaling sessions may be generated by the signaling plane probe

based on information on the media data path of the media streams belonging to the service associated to the respective one of the signaling sessions and the timestamp indicative of the point in time at which the end of the respective signaling session has been detected by the signaling plane probe. For example, in one possible implementation information on a media data path of a media stream is a destination IP address and port number of the media stream.

In another embodiment, the timestamps used for the generation of the correlation keys by the signaling plane probe and the media plane probe could be for example rounded (e.g. in a configurable or fixed interval) before the generation of the correlation keys, in order to ensure that timing differences in the events monitored in the media plane and signaling plane of the services do not lead to generation of non-matching correlation keys for a given service. Alternatively, in another exemplary implementation, the timestamps – by default – have a granularity that is higher than one second, e.g. a granularity in the range of one to ten seconds.

In a further embodiment, instead of storing a single correlation key per direction of a signaling session or per media stream, additional correlation keys may be generated and stored. The input for these additional correlation keys are the same IP address and port information, but a different timestamp. For example, the signaling plane probe could calculate an additional correlation key using a timestamp earlier than the timestamp indicative of the end of the respective signaling session.

In case of using a rounded timestamp this earlier timestamp may be the next earlier rounded timestamp before the rounded timestamp indicative of the end of the respective signaling session. In one exemplary implementation, the signaling plane probe could generate two timestamps from a (non-rounded) timestamp indicative of the end of the respective signaling session which are each used together with the same IP address and port information to generate two respective correlation keys for a given direction: One of these two generated timestamps used for calculation of the correlation keys may be the timestamp indicative of the end of the respective signaling session rounded to the nearest lower value according to the rounding interval, while the other of these two generated timestamps may be the timestamp indicative of the end of the respective signaling session rounded to the nearest higher value according to the rounding interval.

All correlation keys that are generated by the signaling plane probe for a given direction may be stored as part of the signaling session records.

In a similar fashion as described above for the signaling plane probe and the directions indicated in the signaling, also the media plane probe may optionally generate more than one correlation key for a given media data path/media stream. In case of using rounded timestamps also the media plane probe could generate two timestamps from a (non-rounded) timestamp indicative of the end of the respective media stream which are each used together with the same IP address and port information of the media stream to generate two respective correlation keys. One of the two timestamps generated by the media plane probe may be the timestamp indicative of the end of the respective media stream rounded to the nearest lower value according to the rounding interval, while the other of these two generated timestamps may be the timestamp indicative of the end of the respective media stream rounded to the nearest higher value according to the rounding interval. All correlation keys that are generated by the media plane probe for a given media stream may be stored as part of the media stream records.

In case of rounding timestamps in the media plane probe and the signaling plane probe, the rounding interval in both probes should be identical. The gain of having both probes generating additional correlation keys and correlating a multitude of correlation keys may not reflect in the increased processing needs.

In a further exemplary embodiment of the invention, the correlation of the media streams and the signaling session of the respective services is performed by a correlation unit periodically or in response to a query.

Another exemplary embodiment of the invention is related to a method for correlating media streams and signaling sessions of services. According to this method a correlation unit correlates the media streams and the signaling sessions of one or more services based on the correlation keys comprised within media stream reports of the media streams and signaling session reports of the signaling sessions to thereby identify the media stream reports and signaling session report belonging to a respective one of said one or more services. For correlating media streams and the signaling sessions, the correlation unit identifies one or more media streams of a respective service and the corresponding signaling session of the respective service based on the correlation keys comprised in the media stream reports and signaling session reports. Further, the correlation unit generates a service record for a respective one of said at least one service based on information contained in the media stream reports and information contained in the signaling session report of said at least one service.

In one exemplary implementation, these steps are performed by a correlation unit periodically for all signaling session reports and media stream reports generated within a configurable time interval, or in response to a request.

In case the correlation unit is unable to match one or more media streams to said signaling session report, the correlation unit may optionally generate a service record without quality information of the media streams of the service or may optionally generate a corresponding indication of the user/operator of the correlation unit.

A service report generated for a service by the correlation unit may comprise – according to one exemplary implementation – information on the service and the quality information of the media streams of the service.

In one exemplary implementation, the correlation keys provided in a respective signaling session report may have been generated using a timestamp indicative of the point in time at which the end of the respective signaling session of the corresponding service has been detected and using information on the media streams of the service. Similarly, the correlation keys provided in a respective media stream report may have been generated using a timestamp indicative of the point in time at which the respective media stream ended and using information on the media data path of the media stream.

In one further exemplary implementation, the correlation unit first attempts a correlation of media stream reports and signaling session reports using the correlation keys that have been generated based on the timestamps indicative of the end of the signaling session, respectively, the end of the media stream(s). In case the correlation is unsuccessful, the correlation unit may further use optionally available additional correlation keys contained in the signaling session reports and/or media plane reports which are based on other timestamps, e.g. before and after the actual end of the respective signaling session or media stream(s), respectively. This layered approach allows to first conduct the correlation using the correlation keys which are more likely to yield a result and as a fallback conduct the correlation using a wider timeframe.

According to a more detailed exemplary embodiment of the invention, the correlation of the media streams and the signaling sessions of one or more services based on the correlation keys comprised within media stream reports of the media streams and signaling session reports of the signaling sessions by the correlation unit comprises matching the respective correlation keys of the media stream reports of media streams are to the respective correlation keys of the signaling session reports, or vice versa. A match of the correlation keys of a one of the media stream reports and one of the signaling session reports (i.e. the

two correlation keys are identical) yields that the media stream of said one media stream report belongs to the same (media) service to which the signaling session of said one signaling session report belongs.

In another exemplary embodiment of the invention, for correlating the media streams and the signaling sessions of one or more services by the correlation unit, same may obtain from a signaling session report correlation keys identifying the media streams belonging to the same service as the signaling session. Further the correlation unit could then identify those media stream reports which include one of the correlation keys obtained from the signaling session report.

Alternatively, in another exemplary embodiment, the correlation of the media streams and the signaling sessions of one or more services could also be implemented by the correlation unit obtaining from a media stream report a correlation key identifying the media stream, and further the correlation unit identifying a signaling session report among the signaling session reports which includes the same correlation key obtained from the signaling session report.

According to one exemplary embodiment of the invention, a method for correlating media streams and signaling sessions of services in a passive monitoring system of a packet-switched network is provided. It is assumed for exemplary purposes that one service has one signaling session.

Another embodiment of the invention is related to a passive monitoring system for correlating media streams and signaling sessions of services in a packet-switched network. The passive monitoring system comprises a signaling plane probe that generates at least one correlation key for each direction of a service based on the information of the signaling session of the respective service monitored at the signaling plane probe. As noted above, there may be one or more correlation keys for each direction of the service. The correlation keys for a respective service are generated by the signaling plane probe using a timestamp indicative of the point in time at which the end of the respective signaling session of the respective service has been detected by the signaling plane probe and further using information on the direction of the respective service (e.g. information on the media data path of the given direction, such as a destination IP address and UDP port number). The signaling plane probe stores a signaling session report for each signaling session in a database, wherein each signaling session report comprises the at least two correlation keys generated for the respective directions of the service.

Furthermore, the passive monitoring system also comprises a media plane probe for generating a correlation key for each media stream monitored by the media plane probe. The

correlation key is generated using a timestamp indicative of the point in time at which the respective media stream has ended and using information on the media data path of the respective media stream. The media plane probe stores a media stream report for each media stream in a database. Each media stream report comprises at least the correlation key generated for the respective media.

The passive monitoring system further comprises a correlation unit for correlating the media streams and the signaling sessions based on the correlation keys comprised within media stream reports and signaling session reports. The correlation unit may for example use one of the different correlation methods described herein for this purpose.

Another embodiment of the invention is related to the design of the signaling plane probe. According to this embodiment of the invention, the signaling plane probe comprises a network interface card for receiving packets of signaling sessions of multiple media services and a monitoring unit for monitoring the signaling sessions based on the packets of the signaling sessions received by the signaling plane probe to thereby obtain information on a respective one of the signaling sessions. The monitoring unit of the signaling plane probe is further capable of detecting the point in time at which a respective signaling session of the respective service terminates.

The signaling plane probe also comprises a processing unit for generating at least one correlation key for each direction of a service based on a timestamp indicative of the point in time at which the end of the respective signaling session of the respective service has been detected by the monitoring unit and based on information on the direction of the respective service. Moreover, the monitoring unit stores a signaling session report for each signaling session in a database, wherein each signaling session report comprises at least two correlation keys generated for the respective directions of the service.

Another embodiment of the invention is related to a media plane probe which comprises a network interface card for receiving packets of media streams of multiple media services, and a monitoring unit for monitoring the media streams based on the packets of the media streams received by the media plane probe. Thereby the media plane probe obtains information on a media data path a respective one of the media stream. Further, the monitoring unit detects the point in time at which a respective media stream of the respective service terminates, and a processing unit of the media plane probe generates a correlation key for each media stream monitored by the monitoring unit. This correlation key may be for example generated using a timestamp indicative of the point in time at which the respective media stream has ended and using information on the media data path of the respective media stream.

The media plane probe's monitoring unit further stores a media stream report for each media stream in a database, wherein each media stream report comprises at least the correlation key generated for the respective media.

The invention according to another embodiment is related to a correlation unit capable of correlating media streams and signaling sessions of services in a passive monitoring system of a packet-switched network. The correlation unit comprises a database access unit for accessing a database comprising signaling session reports for signaling sessions of services and a database comprising media stream reports of media streams of the services. The signaling session reports each comprise at least one correlation key for each direction of a respective service, and the media stream reports each comprise one correlation key indicating the direction of the media stream. A processing unit of the correlation unit is used to correlate the media streams and the signaling sessions of the services based on the correlation keys comprised within media stream reports and signaling session reports to thereby identify the media stream reports and signaling session report belonging to a respective one of said one or more services. The processing unit identifies one or more media streams of a respective service and the corresponding signaling session of the respective service based on the correlation keys comprised in the media stream reports and signaling session reports. The correlation unit may further comprise a service record generation unit (e.g. implemented by a processing unit) for generating a service record for a respective service based on information comprised in the media stream reports and information comprised the signaling session report of said at least one service.

In a further embodiment of the invention, the correlation unit is further adapted, e.g. by comprised respective configured means, to perform the steps of the correlation methods according to one of the different embodiments described herein. Furthermore, also the signaling plane probe and the media plane probe may be adapted, e.g. by comprised respective configured means, to implement the different functions thereof as described herein.

Further embodiments of the invention relate to the implementation of the different functions of media plane probe, signaling plane probe and correlation unit in software. In this context, another embodiment of the invention provides a computer-readable medium storing instructions that, when executed by a processor of a signaling plane probe, cause the signaling plane probe to receive packets of signaling sessions of multiple media services, and to monitor the signaling sessions based on the packets of the signaling sessions received by the signaling plane probe to thereby obtain information on a respective one of the signaling sessions. The execution of the instructions further cause the signaling plane probe to detect the point in time at which a respective signaling session of the respective

service terminates, and to generate at least one correlation key for each direction of a service based a timestamp indicative of the point in time at which the end of the respective signaling session of the respective service has been detected and based on information on the direction of the respective service. Moreover, the executed instructions may further cause the signaling plane probe to store a signaling session report for each signaling session in a database, wherein each signaling session report comprises the at least two correlation keys generated for the respective directions of the service.

Another exemplary embodiment of the invention is providing a computer-readable medium storing instructions that, when executed by a processor of a media plane probe, cause the media plane probe to receive packets of media streams of multiple media services, and to monitor the media streams based on the packets of the media streams received by the media plane probe to thereby obtain information on a media data path a respective one of the media streams. Moreover, the execution of the instructions cause the media plane probe to detect the point in time at which a respective media stream of the respective service terminates, and to generate a correlation key for each media stream monitored by the monitoring unit, wherein the correlation key is generated using a timestamp indicative of the point in time at which the respective media stream has ended and using information on the media data path of the respective media stream. Furthermore, the execution of the instructions may further cause the media plane probe to store a media stream report for each media stream in a database, wherein each media stream report comprises at least the correlation key generated for the respective media.

Another embodiment of the invention is providing a computer-readable medium storing instructions that, when executed by a correlation unit, cause the correlation unit to correlate media streams and signaling sessions of services in a passive monitoring system of a packet-switched network. The execution of the instructions cause the correlation unit to access a database comprising signaling session reports for signaling sessions of services and a database comprising media stream reports of media streams of the services. The signaling session reports each comprise at least one correlation key for each direction of a respective service, and the media stream reports each comprise one correlation key indication the direction of the media stream. The execution of the instructions further cause the correlation unit to correlate the media streams and the signaling sessions of the services based on the correlation keys comprised within media stream reports and signaling session reports to thereby identify the media stream reports and signaling session report belonging to a respective one of said one or more services and to identify one or more media streams of a respective service and the corresponding signaling session of the respective service based on the correlation keys comprised in the media stream reports and signaling session reports.

The execution of the instructions also cause the correlation unit to generate a service record for a respective one of said at least one service based on information comprised in the media stream reports and information comprised the signaling session report of said at least one service.

Another embodiment is providing a computer readable medium storing instruction that, when executed by the processor of the correlation unit, cause the correlation unit to perform the steps of the method for correlating media streams and signaling sessions of services according to one of the different embodiments described herein.

BRIEF DESCRIPTION OF THE FIGURES

In the following the invention is described in more detail in reference to the attached figures and drawings. Similar or corresponding details in the figures are marked with the same reference numerals.

Fig. 1 shows exemplary situations where multiple media streams may be triggered in a media service,

Fig. 2 shows three geographically distributed POPs (Point of Presence) which are linked over the carrier's internal network,

Fig. 3 shows an exemplary network using internal and external addresses,

Fig. 4 shows an exemplary overview on a system according to an embodiment of the invention comprising signaling plane probe, a correlation unit and a media plane probe in a packet-switched network,

Fig. 5 shows a flow chart of the operation of the correlation unit according to an exemplary embodiment of the invention,

Fig. 6 shows the structure of a correlation unit according to an exemplary embodiment of the invention,

Figs. 7 & 8 shows two exemplary flow charts of the operation of a signaling plane probe according to different exemplary embodiments of the invention,

Fig. 9 shows the structure of a signaling plane probe according to an embodiment of the invention,

Fig 10 shows the structure of a media plane probe according to an embodiment of the invention,

Fig. 11 shows a flow chart of the operation of a media plane probe according to an embodiment of the invention,

Fig. 12 shows a network in which locations are indicated, where signaling and media plane probes of the invention may be advantageously located according to an embodiment of the invention,

Fig. 13 illustrates the use of rounded timestamps within a signaling plane probe and a media plane probe to highlight potential sources of error in correlation, and

Fig. 14 illustrates the use of two rounded timestamps for generating respective correlation keys according to an exemplary embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

The following paragraphs will describe various embodiments of the invention. For exemplary purposes only, most of the embodiments are outlined in relation mainly to VoIP services as an example for a media service, but the invention is not limited to VoIP services. For example, the invention may also be used for other types of media services provided in packet-switched networks. The invention may be advantageously used in media services which have signaling plane and media plane separated and/or taking different routes through the packet-switched network.

The explanations given in the Technical Background section above are intended to better understand the often VoIP call-related exemplary embodiments described herein and should not be understood as limiting the invention to the described specific implementations of processes and functions in a packet-switched communication network.

As indicated above, the invention generally relates to methods for correlating at least one media data path and a signaling session of a service in a passive monitoring system that is monitoring a packet-switched network. Furthermore, the invention also relates to an implementation of these methods in hardware and software, and provides a signaling plane probe, a media plane probe and a correlation unit, as well as passive monitoring system comprising same. Based on the assumption of disjunctive signaling and media components of a service, the correlation mechanism suggested herein supports correlation across

multiple hosts connected via a network (e.g. a Local Area Network (LAN), Wide Area Network (WAN) or the Internet).

One of the various aspects of the invention is to suggest a definition of correlation keys that allow the correlation of media streams of services and the associated signaling session of the respective services based on data records (reports) that provide information on the media plane and signaling plane of the respective services. In the following these correlation keys are implemented as in form of hash keys for exemplary purposes. The correlation keys may be used to query database(s) to correlate media stream reports for media streams monitored by a media plane probe and a signaling session report on the signaling session provided by a signaling plane probe that belongs to the same service – it should be noted that for the proper functioning of the invention, it is not decisive whether media stream reports and signaling session reports are stored in the same or different databases, or whether there are even multiple database partitions provided. The media stream reports and signaling session reports may even be stored in files.

The correlation keys are generated by one or more media plane probes and one or more signaling plane probes that monitor a packet-switched network independently from each other. This independent calculation of the correlation keys facilitates distributed correlation spanning across multiple geographically distributed components or even across multiple vendors of software components. Furthermore, as will become more apparent from the following, for a given media data path – respectively a corresponding given direction –, the parameters that are used for generating correlation keys at the signaling plane probe(s) and the media plane probe(s) ensure that the identical correlation keys are generated for the given media data path by the signaling plane probe(s) and the media plane probe(s) to facilitate correlation. A respective correlation key may be associated to a given direction of the media data paths of the media streams belonging to a given service. For example, assuming for exemplary purposes a simple bi-directional service, e.g. VoIP call, there are two directions: caller-to-callee and callee-to-caller.

The signaling plane probe may generate a correlation key for each direction of the service. For example, the signaling plane probe generates one correlation key for the caller-to-callee direction and another correlation key for the callee-to-caller direction. However, the signaling plane probe may also generate multiple correlation keys per direction, as will be outlined below in further detail. Consequently, in case it is assumed that a service comprises at least two media streams of different directions, then at least two correlation keys are calculated by the signaling plane probe for later correlation of the media plane and signaling plane of the service. The correlation key for a given direction or media data path may be generated based on information on the direction or media data path, such as for example a tuple comprising

the destination IP address and UDP port for a media stream in the service's media plane corresponding to the given direction or media data path. In addition, the generation of the correlation key is further based on a timestamp. With respect to the generation of the correlation key by the signaling plane probe, the timestamp may for example indicate the point in time at which the signaling plane probe has detected the end of the signaling session (and thus the service). In another exemplary implementation, the timestamp used in the generation of a correlation key for a media data path of a media stream in the media plane of the service could also be the point in time of the termination (end) of the media stream in the media plane detected by the signaling plane probe.

The signaling plane probe may generate the correlation keys for the different endpoints (the endpoint indicate a direction of the related media data flow) of a service upon detection of the end of a signaling session and additionally or alternatively, upon detection of the end of a media stream in the service's media plane. Typically but not limited thereto, there is one endpoint per direction, e.g. the caller and callee. The callee may be represented by multiple endpoints in case of more complex media service scenarios, e.g. when session forking is used (see Fig. 1).

The end of a signaling session may be for example detected in the signaling plane probe by detecting a signaling protocol specific hang-up message, a terminal failure to connect the involved endpoints or a session timeout. Each correlation key (e.g. hash key) is then generated by the signaling plane probe based on a timestamp indicative of the point in time at which the end of the respective signaling session has been detected by the signaling plane probe and using information on respective direction of the media plane data flow (e.g. caller-to-callee or callee-to-caller for a simple bi-directional service, e.g. VoIP call).

The direction may be for example given by a media data path of a media stream of the respective service. A media data path could be for example indicated by a tuple comprising the destination IP address and UDP port number for transmitting the media packets of a media stream of the service. The signaling plane probe may thereby generate only one single correlation key per direction or multiple correlation keys per direction. In one exemplary embodiment of the invention, the signaling plane probe generates one correlation key per direction, where the correlation key for a respective direction is based on a media data path of a media stream. This media stream of which the media data path information is used to generate the correlation key should be existent at the end of the signaling session (according to the session signaling monitored by the signaling plane probe), in order to be able to match the correlation key to a corresponding correlation key in a media stream report generated by a media plane probe. The media data path(s) of a service and optionally their

existence (e.g. start time and termination time) could be for example learned by the signaling plane probe from the messages of the signaling session of the service.

The correlation keys generated for the different directions of the service corresponding to the signaling session the end of which has been detected are then comprised in a signaling session report, which may be for example stored in a database or (flat) file.

Alternatively, the signaling plane probe may also generate a correlation key for each media stream in the media plane of the service. In this case it may be assumed that the signaling plane traffic, i.e. the packets of the signaling protocol of the signaling plane, allow the signaling plane probe to detect the termination (i.e. end) of respective media streams in the media plane. In this case, the detection of the end of a media stream could trigger the generation of a correlation key for the respective media stream based on information on its media data path and a timestamp indicating the point in time at which the signaling plane probe detected the termination of the media stream.

Further, in another alternative implementation, the detection of the end of a signaling session, respectively, service as well as the end of a media stream as detected by a signaling plane probe could be used to trigger the generation of correlation keys in the signaling plane probe. For example, during the ongoing service the detection of the termination of a media stream in the media plane as learned from the signaling plane traffic could trigger the signaling plane probe to generate and include respective correlation keys for the media streams terminated during the ongoing session into the signaling session report of the service, while the detection of the end of the signaling session, and thus the service, could be used as a trigger for generating correlation keys for the respective media streams being "alive" in the media plane of the service at the signaling plane probe and their inclusion into the signaling session report of the service.

In the media plane, the media plane probe generates a correlation key (e.g. hash key) for each media stream. The correlation keys for the media steams are generated upon the media plane probe detecting the end of the respective media stream. The correlation key for a respective terminated media stream is calculated by the media plane probe based on the direction of the respective media stream, e.g. based on the media data path of a media stream, and on a timestamp indicative of the point in time at which the respective media stream ended. This point in time is for example the point in time at which the last packet of the media stream has been received at the media plane probe. In this latter case, the media plane probe may thus maintain a timestamp indicating the time of reception of the last packet of the media stream at the media plane probe.

The timestamp used for the generation of the correlation key may indicate another point in time than the point in time at which the media plane probe detects the end of the media stream. For example, in a packet switched network, there may be interruption in the packet stream of the media stream so that there may be a timer that measures the time since the last reception of a packet of the media stream. In case the time since the last reception of a packet of the media stream is exceeding a threshold (e.g. the timer "expires") – for example after 15 or 30 seconds, or the like – then the media plane probe detects the end of the media stream. The timestamp used in the correlation key generation may nevertheless be the point in time at which the last packet of the media stream has been received. The media plane probe stores the correlation key (e.g. hash key) for the media stream in a media stream report, which may be for example stored in a database or exported as a flat file.

Furthermore the timestamp used for the generation of a correlation key may be rounded. This counters differences in the time basis (local time) of multiple signaling and media probes as slightly different timestamps still yield the same rounded timestamp. Additional correlation keys may be generated with different timestamps, e.g. indicative of the – optionally rounded – end of a respective signaling session and/or media stream or a timeframe before or thereafter. These additional correlation keys may be stored in the respective signaling session reports and/or media session reports.

A media stream may also be interrupted due to codec features such as Voice Activity Detection (VAD). If a sender of a media stream is mute the codec may not encode any data resulting in an interruption of the media stream. This is typically indicated in the packet stream transporting the media and should not be considered by the media plane probe as an indication of the end of the media stream itself. In case a timer mechanism is used to detect terminated media streams, the timer should be configured such that codec features that lead to a temporary interruption of the media stream are not triggering the termination of the media stream (according to the detection mechanism for detecting terminated media streams in the media plane probe).

Furthermore, in another exemplary embodiment, the correlation of media streams of services and the signaling sessions of the services may also be based on a compound keys used as a correlation keys. This exemplary alternative implementation may however require a more complex database structure of the media stream reports and signaling session reports, as will be outlined below.

Another further aspect of the invention is the use of the correlation keys comprised in the signaling session reports and the media stream reports to identify the media streams and signaling session belonging to the respective services. This correlation of media streams and

signaling session could be done for example periodically or on request. The correlation keys generated by the signaling plane probe and the media plane probe should be matching each other (i.e. the correlation keys of a given signaling session report should be found in the media stream reports, respectively, the correlation key of a media stream report should be found in one of the signaling session reports). Hence, a correlation unit can identify which media streams belong to which signaling session (or vice versa) based on the correlation keys contained in the media stream reports and the signaling session reports and may therefore generate a service record for a respective service based on information contained in the media stream reports and information contained the signaling session report of the service.

The correlation methods according to the different embodiments of the invention proposed herein are based on a probe monitoring the signaling plane traffic of services (i.e. the signaling plane probe) identifying new services and associated media data paths of the media plane by detecting their corresponding signaling sessions and determining for each service correlation keys allowing to identify at least the media streams of the media plane existent upon termination of the service. It should be noted that "at least the media streams" means that it is optionally possible to generate a correlation key for each media data path of the media plane learned from the signaling plane traffic of the service, in case it is possible to detect the termination of the media data paths of the media plane in the signaling plane traffic. In this case the correlation key for a respective media stream may be based on the information on the media data path of the respective media stream visible in the signaling plane of the service and the point in time at which the signaling plane probe detects the termination of the media stream in the signaling plane.

Regarding the use of the point in time of the termination of the signaling session (e.g. a signaling message in the signaling plane indicating "hang up") for the generation of correlation keys it should be noted that the termination of the signaling session also means that all existent media data paths, respectively, the corresponding media streams of the media plane are terminated. Hence, in case it is not possible to detect the termination of the media data paths of the media plane in the signaling plane traffic, it is assumed to be possible to detect at least the end of the signaling session and to thereby conclude on the termination of the respective media data paths of the media plane and to thereby facilitate correlation of the media streams that have been "alive" at the end of the service and the corresponding signaling session of the service.

As noted above, another possibility may be to use compound keys as correlation keys. A compound key is defining a set of information comprised e.g. in a given set of fields of a database entry of a signaling session report and media stream report that – together with

additional criteria on the point in time of the termination of a media stream or the service – allows to correlate media streams and signaling sessions of the services. In one exemplary embodiment, the one or more databases that are used to store the signaling session reports and the media stream reports are partitioned into partitions covering only a given amount of time (i.e. time interval) of signaling session reports and/or media stream reports, respectively. The "size", i.e. the time interval covered by the partitions may be for example determined based on the average time interval in which port reuse occurs in the target monitored packet-switched network (e.g. the partition could cover ½ of the average time interval in which port reuse occurs). In the given partitions it may thus be assumed that the media data path information, such as destination IP address and UDP port number are unique. In this case, the signaling session reports and media stream reports may not include separate correlation keys, but the correlation key can be generated on-demand as a compound key from the information on the media data paths contained in the signaling session reports and media stream reports.

Hence, it would be sufficient that the signaling session reports and media stream reports store information on the media data paths of the service (e.g. destination IP address and UDP port number) and a timestamp that indicates the point in time of termination of the respective media data path in the media plane, as detected by the signaling plane probe from the monitored signaling sessions and the media plane probe from the monitored media streams. As noted above, the signaling plane probe can also detect the point in time of termination of the media data path(s) in the media plane by detecting the point in time of termination of the signaling session, which is – in most practical cases – coinciding with the termination of all media streams of the service in the media plane as well.

In this case, the correlation could be realized by first selecting the database partition of the media plane reports according to a timestamp of the termination of a media data path/signaling session within a given signaling session report, and by then determining the media stream reports that comprise media stream information matching those indicated in the given signaling session report.

Fig. 4 shows a packet switched network 400 according to an exemplary embodiment of the invention that comprises a signaling plane probe 401, a correlation unit 402 and a media plane probe 403. Please note that more than one signaling plane probe 401 and media plane probe 403 may be provided. Optionally, there may be multiple correlation units 402 present, working independent of each other and not impacting the performance or correlation accuracy of any other component.

The signaling monitoring probe 401 and the media plane probe 402 are monitoring the traffic on a packet-switched network 400, which may be for example an IP based packet switched network. It is assumed for exemplary purposes that the signaling plane probe 401 and the media plane probe 403 monitor the traffic of the packet-switched network 400 at different locations of the packet-switched network 400, but observe the same network traffic.

In one exemplary implementation, the signaling plane probe 401 and the media plane probe 403 are probes of a minimal non-intrusive, passive monitoring system. The respective monitoring probes 401, 403 are connected to the packet-switched network 400 via respective test access ports (TAPs). A TAP is a passive network device that mirrors network traffic of the packet-switched network without interference of the original network traffic. It provides a copy of every packet sent or received on the network, by duplicating the data packets observed on the physical layer of the network. Instead of a TAP a Switch Port Analyzer (SPAN) port may be used to monitor the traffic of the packet-switched network.

In packet-switched environments, the signaling data of the signaling plane and media data of the media plane is packaged into packets on the transmitter side. Packets are transported over the packet-switched network independent of each other. The receiver of the packets – identified by the destination IP address and port – may have to regroup and sometimes reorder packets in order to compile streams from single packets.

On the signaling plane multiple packets belong to one signaling session. The first packet of a signaling session – usually containing a "call setup" message – starts a new signaling session (respectively a new service). At this point, the signaling plane probe 401 generates a session entry along with auxiliary data, i.e. a unique correlation key identifying the session internally (signaling session identifier). Packets following the initial signaling packet are matched to one of the currently ongoing signaling sessions (respectively services) and signaling sessions managed by the signaling plane probe. Packets which do not start a new signaling session and do not belong to existing sessions may be discarded.

Matching single signaling packets to existing signaling sessions is specified in the standards which define the signaling protocol itself. Transport layer information like IP addresses and protocol-specific information (e.g. SIP Call-IDs and tags) may be used.

A signaling session may be torn down upon reception of a special packet – a "call hang-up" message. This message again is specific to the signaling protocol used. Upon reception of this message the signaling plane probe 401 can conclude on the termination (end) of the signaling session and generates the correlation keys for use in correlation of media plane and signaling plane of the services. Furthermore, the signaling session may be removed

from its internal storage. Additionally the signaling monitoring component may remove sessions from its internal storage, if it identifies stale sessions.

The media plane 403 of the service may for example identify the media plane data in form of RTP streams. RTP streams are – analog to the signaling sessions – made up from individual RTP packets. The transmitter of an RTP stream generates RTP packets by splitting the media data into smaller pieces. In general these packets are relatively small to reduce the delay between sender and receiver. As a result many packets are created and form a single RTP stream – typically 50 packets per second. Single RTP packets can be grouped into an RTP stream by media plane probe 403. When using RTP, the media plane probe 403 has transport layer information as well as the RTP protocol header which contains a so called Synchronization source (SSRC). The SSRC is a 32-bit numeric identifier which is contained in every single packet and deemed mandatory by the RTP. When media plane probe 403 first receives a packet which does not match an existing RTP stream in its internal storage it creates an RTP stream. Consecutive RTP packets can then be matched to the existing streams by comparing the transport layer information and SSRC value. If no packets are received for a single RTP stream for a certain time the stream is considered ended and can be removed from the internal storage. A short interruption in the packet stream should not trigger the media plane probe 403 to mark an RTP stream as ended, thus the time before a media plane probe 403 can consider an RTP stream as ended is multiple seconds and has to take features like voice activity detection into account.

The signaling plane probe 401 and the media plane probe 403 generate information about the signaling session and media streams, respectively. This generated information is comprised in signaling session reports and media stream reports, respectively. In order to track running sessions and media streams the entities 401, 403 store information internally while the signaling sessions and media streams are in progress. When a signaling session or media stream is terminated the respective monitoring component 401, 403 may generate a signaling session report with signaling information, respectively media stream report with information on the media stream. As will be outlined below in further detail, the signaling session report provides a summary of the signaling session (service) and includes the correlation keys (e.g. hash keys) for correlation. Similarly, the media stream reports summarize the media stream and also include a respective correlation key (e.g. hash key) to facilitate correlation.

Each signaling session report for a signaling session may be for example written by the signaling plane probe 401 upon completion of a signaling session and contains data which describes the signaling session it monitored. For example, in one embodiment of the invention, the data stored in the signaling session report and describing the signaling session

include – but are not limited to – a set of caller and callee specific properties which will be available per endpoint (caller or callee). These endpoint-specific properties may for example include at least one of the following:

- endpoint address/name (e.g. SIP URI),
- phone number,
- User-Agent (endpoint sóftware and version),
- capabilities/features,
- supported media types,
- IP addresses and ports for signaling and media streams (from SDP body), and
- hash key of the media path (for example described by the destination IP address and UDP port of the expected media stream)
- signaling routing information.

Additionally, one or more of the following signaling session properties which are not endpoint-specific may also be included in the signaling plane report:

- timestamps: e.g. session start, Post-Dial-Delay (PDD), connect time, session end;
- information on the session flow (e.g. signaling message(s) with timestamps),
- session state (e.g. successful, unsuccessful, timed out, reason for failure), and
- events during session (media on hold, codec renegotiation, session forking, authentication)

In one exemplary embodiment of the invention, the signaling session report comprises all of the above noted endpoint-specific properties, and – optionally – all signaling session properties listed above, as far as they are available from the signaling protocol messages.

Depending on the specific implementation and capabilities of the signaling plane probe, the signaling session reports may contain information about multiple endpoints and/or multiple media streams per endpoint. In this case the session end timestamp may be available per involved endpoint, and not per session.

For example, the most trivial VoIP service is a phone call between two parties (including two media streams, one in each direction). In this case, the signaling session report could store each party as one endpoint with the above mentioned information and the media stream's destination IP address and port information. If the VoIP service would contain a video call (i.e. one audio and video stream per direction) the endpoint information may be extended to contain the destination IP address(es) and port numbers of the two media streams per direction (one IP address & port tuple for audio, one IP address & port tuple pair for video). Considering a VoIP service, the respective tuples of destination IP address and UDP port number of the media streams of the session may be for example learned by the signaling plane probe from the SDP body of SIP signaling.

Likewise, the signaling plane probe may extend the signaling session report to include additional endpoints, if multiple endpoints were part of the VoIP service. Multiple endpoints could be for example involved in a VoIP service, in case a caller tries to reach a callee which has call forwarding configured. The caller may then hear a ring-tone from the callee's landline, followed by a ring-tone from the callee's mobile phone. Finally a voicemail box picks up the session generating a third endpoint on the callee's side.

The media plane probe 403 monitors the media traffic, i.e. the individual media streams that convey the media information of the different services. The media plane probe 403 identifies a new media stream of a media service based on the network traffic it observes on the monitored packet switched network 400. The identification of a new media stream for a given service may happen shortly after this service is set up on the signaling plane. Throughout the duration of the media stream information on the media stream may be stored in an internal data record on the media steam. The media plane probe 403 may note for each stream the last RTP packet received for the respective media stream in an internal data record on the media stream in order to allow detecting the media stream's termination.

The media plane probe 403 may optionally perform a real-time analysis of the media streams monitored on the packet switched network and may generate a fine-grained media stream quality report with media stream related information. Fine-grained media stream quality reports could be for example written in predetermined time intervals, e.g. every five seconds, and could provide a summary of the media stream in the respective interval as well as quality information of the media stream in the respective interval. The information of the fine-grained media stream quality reports of a media stream could be used by the media plane probe 403 to generate a single media stream report on the media stream upon the end of the media stream is detected by the media plane probe 403. The media stream report and optionally also the fine-grained media stream quality reports of a media stream may be stored in individual databases.

The media stream report generated by the media plane probe 403 upon detecting the end of the media stream may contain data which describes the unidirectional media stream it monitored as well as correlation key that can be for example generated based on the destination IP address and UDP port number of the media stream and a timestamp that indicates the point in time at which the media stream ended/terminated. The media plane probe 403 could for example learn the destination IP address and UDP port number from the IP header encapsulating the RTP packets on the network layer. The characteristics of each media stream can be grouped into two categories. The transport and RTP header information characterize the media stream, e.g. for correlation purposes, whereas the quality information enable quality related processing of the media stream. In general the transport information is the same for all media packets processed as part of a single media stream. The stream quality information includes information covering all media packets and may be aggregated or stored on a per-packet basis.

In one exemplary embodiment of the invention, the media stream report generated by the media plane probe 403 for a given media stream may for example include one or more of the following transport media information:

- source and destination IP address,
- source and destination UDP port,
- RTP header Synchronization source (SSRC) value,
- RTP payload type, and
- correlation key(s) (e.g. hash value(s)) calculated based on destination IP address, destination UDP port and a timestamp indicating the point in time at which the of the media stream terminated (e.g. the timestamp of the last packet received for the media stream) or timestamp indicative of the timeframe before or after the termination timestamp.

The stream quality information that may be added to the media stream report by the media plane probe 403 may for example include/indicate one or more of the following:

- timestamp information (e.g. start and/or end time),
- RTP format violations,
- protocol conformance (e.g. sequence errors and/or DSCP errors),

- transport information (e.g. packet interval and jitter statistics, information on packet loss, information on duplicate packets, etc.), and

- MOS values (e.g. per time-slice (e.g. every 5 seconds), minimum/average/maximum value for entire stream duration, etc.),

- events (e.g. comfort noise, silence suppression, DTMF tones, codec changes, marker bits).

In one exemplary embodiment of the invention, the media stream report comprises all of the above noted transport media information and optionally further all stream quality information listed above, as far as they are available to the media plane probe from the packets of the media streams.

In addition to "measured values" (e.g. jitter between multiple media packets) the media stream report may also contain calculated information like MOS (Mean Opinion Score) values and information about impairments experienced by the receiver of the stream.

As further shown in Fig. 4, a correlation unit 402 correlates the media stream reports and signaling session reports generated by the signaling plane probe 401 and the media plane probe 403 in order to identify the media streams and the signaling session that belong to a respective service. It should be noted that each service is assumed to have one signaling session, but may have one or more media streams.

Generation of Correlation Key

The correlation key that is used for correlation may be built from:

- the timestamp that indicates the end of the signaling session, respectively the end of a given media stream, and

- information on a media stream, such as for example the destination IP address and UDP port of the media stream.

The correlation of signaling plane and media plane by the correlation unit 402 is based on this compound correlation key.

As noted earlier, the correlation key may be for example a hash key generated based on the before-mentioned parameters. The hash function to calculate the hash key (sometimes also called hash value) from the given data should be known to all parties involved (i.e. signaling plane probe 401, and media plane probe 403 and optionally also correlation unit 402) so that

the result for a given input is the same for all parties. Therefore, the hash keys are determined by the signaling plane probe 401 and media plane probe 403 using media data path information in addition to a timestamp that indicates point in time when the signaling session, respectively, the media stream ended/terminated. The correlation unit 402 may generate additional correlation keys based on the timestamps indicative of the timeframe before or after the timestamp that indicates the point in time when the respective session ended.

The hash function to determine the hash key may be a simple concatenation of one or more of the input parameters. For example, a concatenation of a destination IP address, destination UDP port number and a timestamp may (optionally including a delimiter character in between the parameters) be used for defining the hash keys. Of course more complex mechanisms may also be used to calculate a hash key representing the signaling session identifiers and/or the media path identifiers. For example the MD5 message digest function (see IETF RFC 1321, "The MD5 Message-Digest Algorithm", 1992, available at http://www.ietf.org) could be used to create a valid hash key. The MD5 message digest function may be fed with a concatenation of a destination IP address, destination UDP port number and the timestamp indicating the time of termination of the signaling session, respectively, the media stream and the resulting string can be used as a correlation key. Another hash function that could be used may be selected from the SHA cryptographic hash function family (see FIPS PUB-180, "Secure Hash Standard (SHS)", 2008 available at http://www.itl.nist.gov/fipspubs/). Furthermore, in another embodiment, a linear hash function may be used.

As outlined with respect to Fig. 1 earlier herein, ephemeral media streams may exist at any point during a service, such as a VoIP service. The ephemeral media streams may be generated by early media streams, service forking and codec changes. As a result the start and end timestamps of media streams may not necessarily concur with the start and end time of the VoIP service itself. Yet the timestamps of the start or end of a signaling session or media stream are an important piece of information for a successful correlation, as it significantly reduces the search window and amount of data to search. In typical scenarios, it can be expected that the start time of signaling and media streams deviate significantly and are thus not ideal candidates for a correlation mechanism.

At the end of a VoIP service both caller and callee have an established signaling session and send each other media on the media plane. Once a party hangs up the session a hang-up signal is send on the signaling plane and at the same time stops sending media packets. Once the remote party receives the hang-up signal it also stops transmitting media. The timestamps of the end of the signaling session of a service and the end of the corresponding

media streams in the media plane of the service are commonly very close to each other compared to the start of a media service in signaling plane and media plane. All these events will be subject to the same network- and application-imposed delays. Still the time-frame in which these happen is typically in the few milliseconds range or in the sub-second-range on long-distance or low bandwidth networks. This makes the timestamps of the end of the signaling session and the end of the media streams a good choice for calculating a correlation key.

Nevertheless, since signaling plane probe 401 and media plane probe 403 independently determine the correlation keys based on timestamps (and even minor deviations of the timestamps in the millisecond-range could lead non-matching correlation keys), the granularity of the timestamps used for the generation of the correlation keys may be chosen to account for those deviations of events in signaling plane and media plane events (i.e. deviations in the end of the signaling session in the signaling plane and the end of the corresponding media streams in the media plane of a service). Therefore the timestamps could have a granularity for example in the range of one or more seconds or at least 250 milliseconds or 500 milliseconds. In one exemplary embodiment, UNIX timestamps (which is indicating the seconds passed since January 1, 1970, 00:00:00) are used for the calculation of the correlation keys. Alternatively, also timestamps with a high resolution (e.g. resolution of one or ten or hundred milliseconds) could be used, but the timestamps may be rounded for the calculation of the correlation keys. It should be noted that the choice of the granularity or rounding interval may also impact the performance of a database or databases storing the media stream reports and signaling session reports for correlation. The larger the granularity, respectively the rounding interval and thus the search window the more data have to be processed to match signaling sessions and media streams.

Rounding timestamps could be for example implemented by rounding the timestamp to the nearest n second interval. Assuming for example that the interval is 5 seconds the following UNIX timestamps would be rounded as exemplified in Table 1.

Unix timestamp value	rounded value (5 sec)	description
1283936131	1283936130	rounded down
1283936133	1283936135	rounded up
1283936135	1283936135	unchanged
1283936136	1283936135	rounded down
1283936139	1283936140	rounded up

Table 1

The rounding interval can be any time interval in seconds or even milliseconds whereas the specific value is a trade-off between speed and accuracy. As noted above, the larger the interval the more data may have to be processed to match signaling sessions and media streams. The smaller the search window is the more likely it is to miss a match due to misaligned timestamps or derivations due to the delays outlined above.

The best rounding interval depends on the network monitored. Reasons for increased time differences are possible derivations from a shared time source (NTP) or differences seen in signaling and media events, i.e. the delay between a call being terminated and the endpoints stopping to send media.

For a smaller search window of three seconds the rounded values are different, as for example shown in Table 2.

Unix timestamp value	rounded value (3 sec)	description
1283936131	1283936130	rounded down
1283936133	1283936133	unchanged
1283936135	1283936136	rounded up
1283936136	1283936136	unchanged
1283936139	1283936139	unchanged

Table 2

Using normalized "session end" timestamp (i.e. timestamps of low granularity or timestamps that have been rounded) may be advantageous to counter the effects (e.g. false positive hash key matches) of the re-used port numbers. Since the end of a service (and/or end of media streams) becomes part of the correlation criteria only the port numbers that are used and valid at this point in time are considered in the correlation.

Another issue that could complicate the correlation processing is that deviations in detecting the end of a media stream, respectively a signaling session at the media plane probe 403 and the signaling plane probe 401 leads to the two entities rounding the timestamps to different values. For example, the signaling plane probe 401 could round the timestamp up, while the media plane probe 403 rounds the timestamp down to the next rounding interval value due to (minor) deviations in the timestamps at the signaling plane probe 401 and media plane probe 403. In another exemplary embodiment, additional correlation keys may be therefore generated by either one or both, the signaling plane probe 401 and media plane probe 403.

The signaling plane probe 401 and/or media plane probe 403 may use timestamps indicative of the rounding intervals before or after the timestamp indicative of the point in time where a

particular signaling session or media steam ended. For example a session ended at a point in time identified by the Unix timestamp value 1283936135. The correlation unit applies the rounding mechanism based on 3 second intervals. The rounded value thus is 1283936136. The correlation unit may generate additional keys with the timeframes before or after the mentioned rounded value. In this case the timestamps used for generation of the additional keys are 1283936133 (1283936136 – 3 seconds) and 1283936139 (1283936136 + 3 seconds). Of course, especially in cases where the rounding interval is short, it may also be desirable to more rounded timestamps rounded to the next and subsequent rounding interval(s) based on which correlation keys are calculated.

As noted above, the inclusion of multiple correlation keys calculated based on different timestamps for a given direction of a signaling session and/or media stream may for example be useful to counter correlation failures due to applying different rounding of timestamps in the signaling plane probe 401 and media plane probe 403 as will be outlined in further detail based on Fig. 13 and Fig. 14 in the following. The end of the signaling session $t_{signaling}$ of a service as observed by the signaling plane probe 401 and the end of the one or more media streams t_{stream} of the service as observed in the media plane probe 403 may not perfectly match each other as shown in Fig. 13. Even in case signaling plane probe 401 and media plane probe 403 calculate correlation keys based on the same IP address and UDP port tuple, the use of different timestamps $t_{signaling}$ and t_{stream} for the calculation of the correlation key will result in the correlation keys generated by signaling plane probe 401 and media plane probe 403 not matching each other. Additionally the use of rounded timestamps may not necessarily resolve this problem, as it cannot always be assured that the timestamps $t_{signaling}$ and t_{stream} are rounded to the same value. As also highlighted in Fig. 13 it could well happen that signaling plane probe 401 rounds the timestamp $t_{signaling}$ to the next lower rounded value T_1 in accordance with the rounding interval R, while the media plane probe 403 rounds the timestamp t_{stream} to the next higher rounded value T_2 in accordance with the rounding interval R. Again, correlation keys calculated on the different rounded timestamps T_1 and T_2 would not match each other.

As noted above, one possible solution to this problem is to include additional correlation keys at least to one of the signaling session reports and media stream reports. For example, in one exemplary embodiment, the media plane probe 403 could include two correlation keys per direction to the signaling session report. For a given media data path of a media stream (e.g. IP address and UDP port tuple) observed by the signaling plane probe, the signaling

plane probe could calculate one correlation key $C_{stream,1}$ on the media data path information and a rounded timestamp T_1, i.e. using the timestamp indicative of the end of the signaling session $t_{signaling}$ rounded to the next lower value according to the rounding interval, and the signaling plane probe could further calculate one correlation key $C_{stream,2}$ on the media data path information and a rounded timestamp T_2, i.e. using the timestamp indicative of the end of the signaling session $t_{signaling}$ rounded to the next higher value according to the rounding interval. Both correlation keys could then be included for the direction/media data path to the signaling session report.

A further implementation may include even further correlation keys, e.g. a third correlation key that is based on the rounded timestamp T_0, indicative of the next lower timestamp to T_1 according to the rounding interval.

The media plane probe 403 may calculate only one correlation key C_{stream} based on a rounded timestamp T_1 or T_2, i.e. based on the timestamp indicative of the end of the media stream (media data path) t_{stream} rounded to the next lower or higher value according to the rounding interval (and rounding rules). This correlation key for the media stream would then be included to the media stream report.

The correlation of the signaling session to the media stream(s) of the service would be successful in this embodiment, since either the correlation key $C_{stream,1}$ or the correlation key $C_{stream,2}$ within the signaling session report for the given direction/media stream would correspond to the correlation key C_{stream} included in the media stream's media stream report.

Optionally, in another embodiment of the invention, also the media plane probe 403 may calculate two or more correlation keys for each media stream, one correlation key based on the rounded timestamp T_1, i.e. based on the timestamp indicative of the end of the media stream (media data path) t_{stream} rounded to the next lower value according to the rounding interval, and the other correlation key based on the rounded timestamp T_2, i.e. based on the timestamp indicative of the end of the media stream (media data path) t_{stream} rounded to the next higher value according to the rounding interval. Both timestamps would then be included in the media stream report for the media stream by the media plane probe. In this exemplary implementation, the correlation may check whether the two correlation keys of the service's signaling session report for the direction/media stream match the two correlation keys within

the media stream report of the service's media stream to confirm the signaling session report and media stream report belonging to the same service.

Please note that instead of or in addition to the media plane probe 403 including two or more correlation keys to the media stream reports, also the signaling plane probe 401 could calculate two or more correlation keys per direction/media data path that is to be indicated in the signaling session report. Including multiple correlation keys per direction/media data path in the signaling session reports may be advantageous as it reduces the overhead information, since there are commonly less signaling session reports than media stream reports generated in the passive monitoring system.

Furthermore, in another exemplary implementation, the database(s) for signaling session reports and media stream reports may be further partitioned to reduce the search space for correlation. For example, the (respective) database could be partitioned in chunks of multiple hours, e.g. 4 or 6 hours or the like. The signaling session reports and media stream reports may further include as a separate element the timestamps of the end of the signaling session, respectively, the end of the media streams, based on which the database(s) can be partitioned and based on which the correct partition could be chosen for correlation later. Hence, in a more advanced implementation, besides the correlation keys a service end timestamp can be used as a search criteria to limit the amount of data that are to be searched in a database significantly.

In another exemplary implementation, the signaling plane probe may also generate a respective correlation key for all media streams of a service, in case the end of media streams is detectable within the session signaling, as noted previously. In this case the signaling session reports could further include a timestamp that indicates the end of each media stream, in order to allow choosing the correct partition of media stream reports in the database for correlation.

Correlation Mechanisms

For correlation of the media streams and the signaling sessions, different possibilities exist how the correlation by the correlation unit can be implemented. For example, according to exemplary embodiments of the invention, the correlation unit may correlate the media streams and the signaling session of the respective services periodically (or proactive) or in response to a query (i.e. on demand).

For example, a proactive correlation process could be utilized by a third party entity which has access to both signaling session reports and media stream reports. Their data may be

present in one or more databases to ease access to the raw data. As noted above, the correlation aims at correlating signaling and media information of the services and at merging them into combined service reports that may be then accessed by a third party, e.g. for evaluation of the services.

The proactive correlation process periodically takes all new signaling session reports which outline – but not fully describe – the media services, e.g. the monitored VoIP sessions. For each signaling session report the correlation process draws the calculated correlation keys from the signaling session report. There are at least two correlation keys in each signaling session report, at least one per direction. Additional correlation keys may optionally exist in the signaling session report, for instance if it describes a video call or a session with multiple endpoints. For each correlation key contained in a respective signaling session report the correlation process searches the media stream report database for the media streams with the same correlation key. If the correlation keys in the signaling session report are generated based on a timestamp indicating the end of the signaling session, every media stream of the service that existed at the end of the service in the media plane should have the same correlation key and will be in the result set of the search. If the correlation keys in the signaling session report also contain correlation keys for media streams that have been terminated during the ongoing service, the search may also further detect those media streams in the media stream report database based on the correlation key.

If a search returned two or more media stream reports, the correlation process encountered one of two scenarios. Either both media streams belong to the signaling session (i.e. service) or one of them was returned due to a correlation key collision. Both cases may be recovered by comparing the contents of the media stream report against the signaling session report. If a media stream in the result set starts before or ends after a signaling session, it was added to the result set due to a correlation key collision. Due to time information it cannot match the signaling session and is deleted from the result set by the correlation unit. If the media stream result set still contains multiple entries after the previous correlation key collision check, all remaining media streams (i.e. their media stream reports) belong to the media service (e.g. the VoIP session).

If a search returned no usable entry, i.e. no signaling session reports and media stream reports comprising matching correlation keys have been found, the correlation unit may optionally use additional correlation keys contained in either the signaling session reports and/or media stream reports to run another search, as explained with respect to Fig. 13 and Fig. 14 above.

The correlation process may optionally select only the most significant media stream for each direction for inclusion to the service report on the media service. The parameters to define the significance of the media streams may be for example configurable by the user/administrator. One parameter for deciding the significance of a media stream in the result set could be the duration of a media stream. For example, the service report generated for a given media service by the correlation unit could include the media stream with the longest duration, in case multiple media stream reports match a correlation key of a signaling session report. Another option would be to include in the service report the last media stream that has been alive in the media service, in case multiple media stream reports match a correlation key of a signaling session report. The correlation process may optionally add information from multiple media streams to the service report, e.g. when the service consisted of an audio and video part.

For a given signaling session (i.e. service), the correlation process may run at least twice, one time for each correlation key in a signaling session report. For example, in case the media service is a VoIP service, the correlation process may be for example performed by the correlation unit once for the caller's media stream and once for the callee's media stream, both identified by their respective correlation keys in the signaling session report.

The correlation unit combines the information contained in the media stream reports that are correlated to a respective signaling session report and the information contained in the signaling session report to provide a service report, providing an outline of the service. The content of this combined service report may be for example the sum or a subset of information of the signaling and media stream reports of a given service. Furthermore, the content of the service reports for the respective services may be configurable and/or depending on the application.

In another exemplary embodiment of the invention, the correlation unit may perform the correlation based on the media stream reports as starting points which are matched to their corresponding signaling session report. Ideally, each media stream of a media stream report should match to one signaling session report based on its correlation key. This implementation may optimize the search time in the correlation processing. The above correlation mechanism starting from the signaling session reports searches the rather large media stream report database at least twice for each signaling session. If the correlation process is based on the correlation keys within the media stream reports entries as starting point, the correlation would periodically take all new media stream reports, obtain the correlation keys thereof and search the signaling session reports for matching signaling sessions. At this point the same steps are taken as outlined above.

Another option may be to correlate media stream reports and signaling session reports on demand. An on-demand correlation employs the same mechanism as the proactive correlation mechanism described above, but instead of running periodically, e.g. when introducing new signaling session reports or media stream reports into the database(s), the on-demand correlation process runs when a user requests particular data.

In an exemplary implementation, the database may contain independent sets of signaling session reports and media stream reports. An operator could request service reports of services linked to some given search criteria, e.g. a specific user, a specific source and/destination IP address, a certain service type, services using a particular software application, codec, etc. The user could be for example specified by providing a username, phone number, the user's SIP URI, etc. or another identifier of the user that can be found in a signaling session report. The correlation unit would then search for the signaling session reports for all signaling sessions matching the particular search criteria. The search criteria may be for example provided to the correlation unit through a web application running on a browser.

For each signaling session report matching the search criteria, the correlation unit subsequently searches for the matching media stream reports as described above for the proactive correlation process. The input for this on-demand correlation is a set of signaling session reports matching the search criteria and not all signaling session reports as in the proactive correlation procedure described above. The on-demand correlation process generates for each signaling session report matching the search criteria a service report, for example by enriching the information of the signaling session report with the matching media stream report information and returns the service report for each matching signaling session report. The returned data may be for example viewed by the web application as individual service reports. Optionally the on-demand correlation process may also store the generated service reports in a database. This would save the on-demand correlation processing time in case an operator requests the same information multiple times.

Correlation Unit

Next, the correlation unit 402 as shown in Fig. 4 is discussed in further detail with respect to Fig. 5 and Fig. 6. Fig. 6 shows the structure of a correlation unit 402 according to an exemplary embodiment of the invention.

The correlation unit 402 may be reachable via the packet-switched network, e.g. through a web based application. For this purpose, the correlation unit 402 may comprise one or more network interface cards 601. The correlation unit 402 has a receiving unit 602 and a

transmitter unit 603, for example for receiving correlation related messages, such as for example on-demand correlation requests, and for responding thereto. Furthermore, the receiving unit 602 and a transmitter unit 603 may transmit and receive messages for accessing remote database(s) that contain(s) the signaling session reports and the media stream reports. The receiving unit 602 and transmitter unit 603 may receive and transmit the correlation related messages from its network interface card(s) 601.

The correlation unit 402 may also have a processing unit 604. The processing unit 604 may be used to determine correlation respective signaling sessions to their corresponding media streams. Furthermore, the processing unit 604 may be used to generate services reports. The service reports may for example comprise information on the respective services and the quality information of the media streams of the respective services. This information may be obtained from the signaling session report and media stream reports belonging to a given service. For correlation, it may be necessary to query one or more databases that comprise the signaling session reports and the media stream reports as provided by one or more signaling plane probes 401 and media plane probes 403. The correlation unit 402 may therefore comprise a database access unit 605 which provides functionality to access these one or more databases. In one exemplary implementation, the database(s) may be located remote from the correlation unit 403.

The correlation unit 402 may also have a storage unit 606, e.g. a RAM, for (temporarily) storing information and data related to the correlation processing. For example, the result set of media stream reports for correlation process, the data of the signaling session reports, and the service report generated for a respective service may be stored in the storage unit 606.

Fig. 5 shows a flow chart of the operation of the correlation unit 402 according to an exemplary embodiment of the invention. Exemplarily, the correlation processing for determining the media streams belonging to one given signaling session report is illustrated in Fig. 5. In case a correlation should be performed for more than one signaling session (report), the procedure of Fig. 5 is repeated for each signaling session report. For example, when using a proactive correlation mechanism, the correlation unit 402 could obtain all signaling session reports that have been generated since the previous correlation process from a signaling session database and process each of the signaling session reports according to Fig. 5. In another embodiment related to an on-demand correlation process, the correlation unit 402 could also receive a correlation request that is indicating one or more search criteria, and may first – not shown in Fig. 5 – query a signaling session database for all signaling session reports that match the search criteria. The signaling session reports matching the search one or more criteria may then be each processed according to Fig. 5 and the service report(s) are returned to the querying party.

The correlation unit 402 obtains 501 a signaling session report to be processed and generates 502 a service report for the service corresponding to the given signaling session. This service report may include information contained in the signaling session report and will be further supplemented with additional information as described below. Furthermore, the correlation unit 402 obtains 503 a correlation key contained in the signaling session report – as noted previously there should be at least two correlation keys contained in the signaling session report. Using the obtained correlation key, the correlation unit 402 queries 504 a media report database to receive all media stream reports that match the correlation key i.e. that contain the same correlation key.

As noted above, it may happen that the query results in the database returning more than one media stream report for the given correlation key, for example due to a hash collision of the correlation key. Therefore, the correlation unit 402 checks 505 whether the returned result set of the database query comprises more than one media stream report. If not (no), the correlation unit 402 can immediately enrich 508 the service report with information on the media stream as contained in the one media stream report returned from the media stream database.

In case the result set returned by the database comprises multiple media stream reports, the correlation unit 402 may try to detect 506 and resolve a hash collision. For example, the hash collision could be resolved by checking further information in the media stream reports of the result set for parameters that identify the media stream (report) that belongs to the correlation key that has been queried. For example, in case the media stream reports contain timestamps of the beginning and/or end of the media stream, same could be compared to respective timestamps contained in the signaling session report, which yield the beginning and/or end of the signaling session. The media stream report(s) of media stream(s) that start before or end after the signaling session may be obtained from the database due to a hash collision. Accordingly, the correlation unit 402 may select 507 only the most relevant media stream report, i.e. the media stream report that is most likely corresponding to the service of the signaling session report from the result set and may add 508 at least a part of its information to the service report. Further, the correlation unit checks 509 whether there is a further correlation key contained in the signaling session report and, if so (yes), repeats steps 503 to 508. These steps are repeated for each further correlation key(s) comprised in the signaling session report.

Once all correlation keys contained in the signaling session report have been processed, the service report for the signaling session is complete, and may be optionally stored 510 by the correlation unit 402 in a service report database.

Depending on the correlation type or application, the procedure of Fig. 5 could be supplemented with additional steps. For example, in case of a proactive/periodic correlation, the correlation unit 402 may check whether there is another signaling session report to be matched to its corresponding media stream reports, and if so the procedure of Fig. 5 would be executed for each signaling session report to be processed. In case of performing a correlation on demand, the correlation unit 402 may likewise check whether there is another signaling session report to be matched to its corresponding media stream reports, and if so the procedure of Fig. 5 would be executed for each signaling session report to be processed. Once all signaling session reports have been processed the correlation unit 402 may generate a correlation response message comprising the service reports generated by the correlation unit 402 in response to the search criteria provided in a correlation request, and returns the correlation response message to the querying entity, e.g. another device/computer.

Signaling Plane Probe

Fig. 9 shows an exemplary implementation of a signaling plane probe 401 and Figs. 7 and 8 show two exemplary flow charts of the operation of same according to different embodiments of the invention. In general, the signaling plane probe 401 observes one or more network links to obtain a copy of all packets passing these network links/interfaces. For this purpose, the signaling plane probe 401 comprises a monitoring unit 901 that receives 701 packets of a service on the packet-switched network 400. For example, this monitoring unit 901 may receive the packets from a one or more specialized packet capture cards 906 (network interface card - NIC) which is capable of receiving and processing packets on the physical interface and to provide them to the application layer, nearly without requiring CPU processing time and operation system functionality. An optional filter in each packet capture card of the signaling plane probe 401 may be used to prevent unwanted packets from being copied from the network interfaces to the processing processes. These filters may be for example implemented as stateless or stateful Internet Protocol (IP) or payload filters. For example, this filter may check 702, whether the currently processed packet is a signaling packet or not. If the packet does not belong to a signaling session, i.e. is not packet of a signaling related protocol, the next packet may be processed.

Each signaling packet received at the signaling plane probe 401 will be processed by a detection unit 902 which analyzes the contents of the packets provided by the monitoring unit 901 so as to identify and process packets as signaling packets. Specifically, the detection unit 902 detects signaling packets and matches these to signaling sessions of the media service, and is capable of identifying one or more media data paths of a media plane of the

service based on signaling information of the service comprised in the received packets. Optionally, the detection unit 902 may be provided in one or more specialized packet capture cards.

For example, the detection unit 902 identifies packets containing signaling information (signaling packets) in a parsing process based on their header information. If parsed successfully as a signaling packet, the signaling packet is forwarded to a state engine. The state engine is designed to identify 703 and form signaling sessions from individual signaling packets.

The state engine (which may be considered an internal database with correlation related information maintained in the storage unit 903) of the detection unit 902 has a processing facility and a state table which holds information about signaling sessions (signaling session records). Based on the protocol-specific identifiers in the signaling packets the detection unit 902 can identify the start and end of a signaling session, respectively media service, or "append" a respective processed signaling packet to an existing signaling session. Signaling packets may also contain information on the start and termination of media streams in the media plane of the media service, which are events that may be tracked by the detection unit 902. Depending on whether the identified signaling session is already registered in the signaling plane probe 401 (step 704), the detection unit 902 registers 705 the new signaling session in the state table, including the generation of a new signaling session record for the signaling session.

The signaling session record may be supplemented with information on the signaling session while same is running to account – for example – for endpoint address/name (e.g. SIP URI); the phone number of the user; the user-agent (endpoint software and version); capabilities/features; supported media types; IP addresses and ports for signaling and media streams (from SDP body); signaling routing information; timestamps: e.g. session start, Post-Dial-Delay (PDD), connect time, session end; information on the session flow (e.g. signaling message(s) with timestamps); session state (e.g. successful, unsuccessful, timed out, reason for failure), and/or events during session (media on hold, codec renegotiation, session forking, authentication), etc.

Furthermore, by analyzing 706 the contents of the signaling packets, e.g. a SDP body of a SIP messages comprised in signaling packets, the detection unit 902 may further learn one or more media data paths of the media plane of the service belonging to the media service that is associated to a given signaling session. The signaling plane probe 401 may further comprise a storage unit 903, e.g. a volatile memory, to store the information describing the state of the respective signaling sessions in the state table. In case the signaling packet

indicates the start of a new media stream (step 706: yes), the detection unit 902 may add 707 information on the media data path (e.g. IP address and port number of the destination, and optionally the source of the media stream) of this media stream to the signaling session record. Likewise, in case the signaling packet indicates the end/termination of a new media stream (step 708: yes), the detection unit 902 may remove the information on the respective media stream from the signaling session record maintained in storage unit 903. In case the signaling packet indicates the termination of the signaling session (step 710: yes), the detection unit 902 may either itself or trigger the processing unit 904 to generate 711 a respective correlation key for each media data path indicated in the signaling session record. The correlation key for a respective media data path of a media stream may be generated for example based on information on the directivity of the media stream (e.g. as given by the tuple of destination IP address and UDP port number) and a timestamp indicating the point in time at which the signaling session was terminated. For example, this timestamp may indicate the point in time of reception of the signaling packet indicating the event of the termination of the signaling session at the signaling plane probe 401. For example, the network interface card(s) 906 of the signaling plane probe could generate a timestamp indicating the time of reception of the respective signaling plane packet and could make same available to the other components of the signaling plane probe 401. Furthermore, the detection unit 902 or the processing unit 904 may further generate 912 a signaling session report based on the information on the signaling session gathered in the signaling session record. Further, the signaling session report also included the correlation keys for the media data paths of the service that exist at the end of the service according to the signaling plane information. The signaling session report may then be stored in a signaling session report database, for example using the database access unit 907 of the signaling plane probe 401. Moreover, the signaling session is deregistered 713 from the state table.

Once a given signaling packet has been processed, it may be discarded. It should be also noted that the order of the checks in steps 706, 708 and 710 and the respective actions to be taken is only exemplary. It is only important that the information on the media streams of the service that exist in the media plane are tracked by the signaling plane probe 401, and that the signaling plane probe 401 is able to detect the termination of the signaling session.

Optionally (not shown in Fig. 7), in another exemplary embodiment, the detection unit 902 of the signaling plane probe 401 periodically (e.g. every 5, 10 or 15 seconds) traverses all entries in the state table in order to find and clean up stale signaling sessions. For this purpose, the signaling plane probe 401 may add a timestamp to each signaling session record in the state table that is updated each time a new signaling packet/message of the respective signaling session is received. Based on this timestamp allows to identify stale

signaling session by simply checking based on the signaling sessions' timestamps whether the last signaling packet/message for a given signaling session has been received longer than a threshold time period ago. For example, stale sessions might be those which did not receive new signaling packets/messages for 86400 seconds or signaling sessions may be defined stale and are terminated after a certain time. This clean-up removes unnecessary information from the state.

Fig. 8 shows another flow chart of the operation of a signaling plane probe 401 according to another embodiment of the invention. In the example discussed with respect to Fig. 7, the signaling plane probe 401 keeps track of the media streams currently existing in the media plane of the service, eliminating those media streams from the signaling session record that are terminated during the ongoing signaling session/service. In the flow chart of Fig. 8, which is corresponding to Fig. 7 to a large extent, the termination of one or more media streams in the media plane, as detected in step 708, causes the detection unit 902 or the processing unit 904 (triggered by the detection unit 902) to generate 801 a correlation key for each terminated media stream. Similar to the other correlation keys, the correlation key for a terminated media stream may be generated based on its media data path information (e.g. destination IP address and UDP port number) and a timestamp that indicates the point in time the media stream was terminated. This timestamp may indicate the point in time of reception of the signaling packet indicating the event of the termination of the media stream at the signaling plane probe 401. The correlation key generated is then added 802 to the signaling session record of the signaling session by the detection unit 902 or the processing unit 904 and is later then also stored in the signaling session report in step 712 together with the other correlation keys either generated in step 801 or step 711.

Media Plane Probe

Next, the media plane probe 403 according to an exemplary embodiment of the invention is described with respect to Fig. 10 and Fig. 11. Fig 10 shows an exemplary embodiment of a media plane probe 403, while Fig. 11 shows a flow chart of its exemplary operation according to an embodiment of the invention.

Similar to the signaling plane probe 401, the media plane probe 403 observes one or more network links to obtain a copy of all packets passing these interfaces. The media plane probe 403 may observe one or more network links to obtain a copy of all packets passing these interfaces. The media plane probe 403 comprises a monitoring unit 1001 that receives 1101 packets of a service on the packet-switched network 401. For example, this monitoring unit 1001 received the packets from one or more specialized packet capture cards 1006 (network interface card - NIC) which are capable of receiving and processing packets on the physical

interface and to provide them to the application layer, nearly without requiring CPU processing time and operation system functionality. An optional filter in the packet capture card of the media plane probe 403 may be used to prevent unwanted packets from being copied from the network interfaces to the processing processes. This filter may be for example implemented as stateless or stateful Internet Protocol (IP) or payload filter. For example, this filter may check 1102, whether the currently processed packet is a media plane packet (media packet) or is some other packet. If the packet does not belong to the media plane, i.e. is not packet of a media related protocol, such as for example RTP, the next packet may be processed.

Each media packet received will be processed by the media plane probe 403. The detection unit 1002 of the media plane probe 403 is designed to detect and form media streams from single media packets. The media plane probe 403 may further comprise a processing unit 1002 that may hold a state table which stores information about the media streams monitored by the media plane probe 403 in a local storage unit 1003, e.g. a volatile memory. Based on the protocol-specific identifiers in the media streams the media plane probe 403 (e.g. based on header information in the RTP packets and/or lower layer headers) the detection unit 1002 identifies 1103 the media data path, i.e. the media stream, the media packet belongs to. If the media data path has not been yet registered (step 1104: no), i.e. the processed media packet starts a new media stream (with the given media data path) the detection unit 1002 generates a new media data record in the state table to account for the new media data path observed by the media plane probe 403. This media stream record may contain information on the media stream such as for example the destination (and optionally the source) IP address and port combination of the media stream, the SSRC in the packet header (in case RTP is used for media transport), etc. If the media data path is already known (step 1104: yes), the media stream record already exists. In any case, the detection unit 1002 updates a timestamp 1106 in the media stream record to indicate the point in time at which the last media packet of the media stream has been received. This timestamp may be used for calculating the correlation key for the media stream. Furthermore, the detection unit 1002 may (re)start a timer 1108 for the media stream record, which when timing out is indicating the termination of the media stream as will be outlined below. Alternatively, no timer is used, but the media plane probe 403 periodically checks for media streams the last media packet of which was received more than a threshold time period ago, in order to determine terminated media streams.

The media stream records of the media streams may for example indicate the respective media data paths thereof by an IP address and port tuple of the destination device, i.e. the receiving device of the media stream. The media plane probe's detection unit 1002 stores

information describing the state of the detected media streams in the media stream records maintained in the state table. Optionally, the individual media streams (i.e. their media data paths) may be identified by a media path identifier (e.g. stream hash key) as unique key. Further optionally, upon detecting a new media stream, respectively media path (step 1104: no), the processing unit 1004 could generate 1105 a media path identifier (e.g. a media path identifier (e.g. stream hash key)) for the media stream, respectively media path and stores 1106 the media stream record in the state table as outlined above.

Optionally, the processing unit 1004 could analyze every received packet for quality information 1107 and – further optionally – store them in the storage unit 1003 for a given time period. Periodically the processing unit 1004 may aggregate or compress the quality information to form fine-grained media stream quality reports with media stream related information. Fine-grained media stream quality reports could be for example written in predetermined time intervals, e.g. every five seconds, and could provide a summary of the media stream in the respective interval and the aggregated quality information of the media stream in the respective interval. Upon generation of a respective fine-grained media stream quality report, the packets used for the generation of this fine-grained media stream quality report can be removed from the storage unit 1003.

The processing unit 1004 of the media plane probe may for example periodically check 1109 (e.g. every 5, 10 or 15 seconds); whether a timer set of a media stream (record) in step 1108 has expired. Alternatively, as noted above, the media plane probe 403 may also check timestamps indicating the last update (i.e. last reception of a media packet) of the respective media stream reports in the state table. A media stream may be for example considered terminated, if no new media packets have been received by the monitoring unit 1001 for the existing media stream for a given amount of time (e.g. 60 seconds). Accordingly, if using a timer to monitor the termination, the timer value could be set to this given amount of time. Further alternatively, instead of periodically checking the media stream records or expiry of their timers, the expiry of a timer for a media stream (record) could cause an interrupt invoking the processing unit 1004 to perform steps 1110 to 1112 as outlined below.

Upon detecting the termination of a media stream, the processing unit 1004 processes the respective information in the media stream record of this media stream the state table. The processing unit 1004 generates 1110 a correlation key for the media stream based on its media data path information (e.g. the destination IP address and UDP port number) and the timestamp in the media stream record that is indicating the point in time of receiving the last media packet of the media stream (see step 1106). Furthermore, the processing unit 1004 generates 1111 a media stream report for the media data path, respectively media stream, which includes the generated correlation key and further information on the media stream.

This further information may be for example transport media information and optionally stream quality information. For example, in case RTP is used for media transport, the transport media information could include the source and destination IP address of the RTP stream, the source and destination UDP port of the RTP stream, the RTP header Synchronization source (SSRC) value and the RTP payload type. Furthermore, the processing unit 1004 may also generate information on the media stream quality that is included to the media stream report as stream quality information. This stream quality information could for example include timestamp information (e.g. start and/or end time of the media stream), an indication of RTP format violations (e.g. the number thereof), information on the protocol conformance (e.g. sequence errors and/or DSCP errors), and transport information (e.g. packet interval and jitter statistics, information on packet loss, information on duplicate packets, etc.), MOS values (e.g. per time-slice (e.g. every 5 seconds), minimum/average/maximum value for entire stream duration, etc.), and/or events (e.g. comfort noise, silence suppression, DTMF tones, codec changes, marker bits). The contents of the media stream reports may be configurable by an operator and may be adapted to customers' needs.

The media stream record may be further stored in a media stream report database using the database access unit 1007 of the media plane probe 403. Finally, the processing unit 1004 of the media plane probe 403 deregisters 1112 the media data path from the state table, e.g. by deleting the corresponding media stream record.

Steps 1110 to 1112 are performed for every media stream that is considered terminated by the media plane probe.

Optimizations may be used to speed up processing of entries in the state tables inside the signaling plane probe 401, the correlation unit 402 and the media plane probe 403. A simple list would be sufficient to implement the features as outlined above. Add, modify, delete and search operations on simple lists may however be too slow. These operations may be implemented utilizing more complex but faster data structures, e.g. hash tables or data structures heavily dependent on CPU and Operating System specific behavior and functionality. The Intel Thread Building Block (TBB) library for instance can be used to implement in-memory databases used for the state tables in the signaling plane probe 401, the correlation unit 402 and the media plane probe 403. The Intel Thread Building Block (TBB) library may also be used to aid implementation of the processing queues inside and in between the components outlined above.

Please note that in case the signaling plane probe 401 and the media plane probe 403 should perform passive non-intrusive monitoring of the signaling plane traffic, respectively

media plane traffic, a separate network interface card is employed for the transmission of correlation related messages (i.e. the network interface card(s) for monitoring the traffic are only employed for packet reception but not for packet transmission).

Please also note that in the procedures described above the detection unit 902, 1002 and the processing unit 904, 1004 may use the database access unit 907, 1007 to access the database/state table in the storage unit 903, 1003.

Application

The correlation mechanism for signaling plane and media plane of a service may be used in manifold applications. For example, the correlation mechanism is useful in small networks but is more important in larger scale networks where the various signaling and media components are decomposed and present on different, sometimes geographically split, network entities. This is the case in IP Multimedia Subsystem (IMS) or Next Generation Network (NGN) setups.

Fig. 12 shows a simplified network in which locations are indicated where correlation may be desired and can be implemented using the correlation mechanism according to one of the various embodiments described herein. Possible monitoring locations are for example (and not limited thereto):

- The Access Session Border Controller (SBC) network interfaces with end customers (shown of the left side of Fig. 12),
- a decomposed gateways system, e.g. to the PSTN, is shown as Media Gateway (MGW) and a Media Gateway Controller (MGC) – see In the top right section of Fig. 12, and
- an Interconnection SBC interfaces to one or more interconnection or transit partners (see the lower right side of Fig. 12.

It should be noted that the location of the monitoring probes for signaling plane and media plane may also be subject to requirements on the monitoring solution.

In some cases, signaling plane and media plane may originate and/or terminate at the same endpoint (device). In case of an end customer this endpoint most likely uses the same IP address on the signaling and media plane. These streams may be for example observed, if the signaling plane probes 401 and the media plane probes 403 are placed close to the end customer devices. The Access Session Border Controller may be considered to be close to the carrier internal network's "border" towards end customer devices and to therefore receive

traffic from the end customers on one side and transmits this traffic on the other side towards the carrier internal network. Most commonly these Session Border Controllers (SBC) are used to separate the devices outside the own trust domain from the inside network. Although the traffic is in one geographic location the SBC may use different IP addresses to load balance traffic. In this case a simple IP address based correlation mechanism not using the principles and ideas outlined herein would already fail whereas the correlation mechanism described herein would succeed. The signaling plane probes 401 and media plane probes 403 may be for example placed in a way that they observe the signaling streams and media streams respectively.

In the upper right corner of the diagram shown in Fig. 12, a decomposed gateway is shown which uses different physical entities as media and signaling components (MGC and MGW, respectively). In between those devices a third party protocol may be used to synchronize the devices. For the sake of a minimum size network diagram only one of each component types is shown but in real life scenarios multiple media components are controlled by one signaling component (as it is the case in IMS, NGN). The media components may be co-located in the same location with the signaling component or spread across the internal network of the carrier. In either case the correlation task is rather complex as it involves multiple IP addresses on different physical network links and thus may require multiple signaling plane probes 401 as well as media plane probes 403.

In a third exemplary scenario the correlation has to correlate signaling and media streams from different locations. A media plane probe 403 component on the internal side of the interconnection point (lower right corner of the diagram – Interconnection SBC) observes a media stream from the Access SBC (middle left). The corresponding signaling session is observed next to the aforementioned Access SBC by a separate signaling plane probe 401.

The location of the correlation unit 402 does not impact what signaling sessions or media streams can be correlated. The correlation unit 402 should have access to the signaling session reports and media stream reports of the signaling plane probes 401 and media plane probes 403 and can be implemented as distinct physical machine or co-located on an existing signaling plane probe 401 or media plane probe 403.

The terms probe and correlation unit used herein can be considered to refer to dedicated hardware, a software program or a combination thereof to implement the desired functionality of the respective probe, respectively, correlation unit. Accordingly, another embodiment of the invention relates to the implementation of the above described various embodiments using hardware and software. It is recognized that probes and correlation unit of the invention may be implemented by having computing devices (processors) execute a software

program that causes the desired functionality being executed. A computing device or processor may for example be general purpose processors, digital signal processors (DSP), application specific integrated circuits (ASIC), field programmable gate arrays (FPGA) or other programmable logic devices, etc. Some of the computing devices may store the software program internally.

The various embodiments of the invention may also be performed or embodied by a combination of computing devices and software programs providing the desired functionality stored on any kind of computer readable storage media, for example RAM, EPROM, EEPROM, flash memory, registers, hard disks, CD-ROM, DVD, etc.

It should be further noted that the individual features of the different embodiments of the invention may individually or in arbitrary combination be subject matter to another invention.

It would be appreciated by a person skilled in the art that numerous variations and/or modifications may be made to the present invention as shown in the specific embodiments without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects to be illustrative and not restrictive.

CLAIMS

1. A method for correlating media streams and signaling sessions of services in a passive monitoring system of a packet-switched network, the method comprising:

a) generating at least one correlation key for each direction of a service based on the information of the signaling session of the respective service monitored at the signaling plane probe, wherein the correlation keys for a respective service are generated using a timestamp indicative of the point in time at which the end of the respective signaling session of the respective service has been detected by the signaling plane probe and using information on the direction of the respective service,

b) storing, by the signaling plane probe, a signaling session report for each signaling session in a database, wherein each signaling session report comprises the at least two correlation keys generated for the respective directions of the service,

c) generating a correlation key for each media stream monitored by the media plane probe, wherein the correlation key is generated using a timestamp indicative of the point in time at which the respective media stream has ended and using information on the media data path of the respective media stream,

d) storing, by the media plane probe, a media stream report for each media stream in a database, wherein each media stream report comprises at least the correlation key generated for the respective media, and

e) correlating the media streams and the signaling sessions based on the correlation keys comprised within media stream reports and signaling session reports.

2. The method according to claim 1, wherein the generation of the respective correlation keys for the respective media streams by the media plane probe is based on the media data path information of the respective media stream and the timestamp indicative of the point in time at which the respective media stream ended.

3. The method according to claim 1 or 2, wherein information on the direction of the respective service, based on which the correlation keys are generated by

signaling plane probe for a respective service, is information on one or more media data paths of media streams of the service learned from messages of the signaling session of the service.

4. The method according to one of claims 1 to 3, further comprising the steps of:

f) monitoring, by the signaling plane probe, signaling messages of the respective signaling sessions of the services, and

g) detecting, by the signaling plane probe, media streams belonging to each service associated to a respective monitored signaling session based on the signaling messages of the respective signaling sessions,

wherein the correlation keys for a respective one of the signaling sessions are generated based on information on the media data path of the media streams belonging to the service associated to the respective one of the signaling sessions and the timestamp indicative of the point in time at which the end of the respective signaling session has been detected by the signaling plane probe.

5. The method according to one of claims 1 to 4, wherein information on a media data path of a media stream is a destination IP address and port number of the media stream.

6. The method according to one of claims 1 to 5, wherein the timestamps used for the generation of the correlation keys by the signaling plane probe and the media plane probe are rounded before the generation of the correlation keys or have a granularity in the range of one to ten seconds.

7. The method according to one of claims 1 to 6, wherein the signaling plane probe generates for each direction of the service at least one correlation key using a timestamp indicative of the point in time at which the end of the respective signaling session of the respective service has been detected by the signaling plane probe and being rounded to the next lower timestamp value according to a rounding interval, and at least another correlation key using the timestamp indicative of the point in time at which the end of the respective signaling session of the respective service has been detected by the signaling plane probe and being rounded to the next higher timestamp value according to the rounding interval.

8. The method according to one of claims 1 to 7, wherein the media plane probe generates for each media stream at least one correlation key using a timestamp indicative of the point in time at which the respective media ended and being rounded to the next lower timestamp value according to a rounding interval, and at least another correlation key using the timestamp indicative of the point in time at which the respective media stream ended and being rounded to the next higher timestamp value according to the rounding interval.

9. The method according to one of claims 1 to 8, wherein the correlation of the media streams and the signaling session of the respective services is performed by a correlation unit periodically or in response to a query.

10. A method for correlating media streams and signaling sessions of services, the method comprising the steps of:

a) correlating the media streams and the signaling sessions of one or more services based on the correlation keys comprised within media stream reports of the media streams and signaling session reports of the signaling sessions to thereby identify the media stream reports and signaling session report belonging to a respective one of said one or more services, wherein correlating the media streams and the signaling sessions comprises identifying one or more media streams of a respective service and the corresponding signaling session of the respective service based on the correlation keys contained in the media stream reports and signaling session reports, and

b) generating a service record for a respective one of said at least one service based on information contained in the media stream reports and information contained the signaling session report of said at least one service.

11. The method according to claim 10, wherein correlation keys provided in a respective signaling session report have been generated using a timestamp indicative of the point in time at which the end of the respective signaling session of the corresponding service has been detected and using information on the media streams of the service.

12. The method according to claim 10 or 11, wherein correlation key provided in a respective media stream report has been generated using a timestamp indicative of the point in time at which the respective media stream has ended and using information on the media data path of the media stream.

13. The method according to one of claims 10 to 12, wherein correlating the media streams and the signaling sessions of one or more services based on the correlation keys comprised within media stream reports of the media streams and signaling session reports of the signaling sessions comprises:

matching the respective correlation keys of the media stream reports of media streams are to the respective correlation keys of the signaling session reports, or vice versa,

wherein a match of the correlation keys of a one of the media stream reports and one of the signaling session reports yields the media stream of said one media stream report belonging to the same service to which the signaling session of said one signaling session report belongs.

14. The method according to one of claims 10 to 13, wherein the service report comprises information on the service and the quality information of the media streams of the service.

15. The method according to one of claims 10 to 13, wherein correlating the media streams and the signaling sessions of one or more services based on the correlation keys comprised within media stream reports of the media streams and signaling session reports of the signaling sessions comprises:

obtaining from a signaling session report correlation keys identifying the media streams belonging to the same service as the signaling session, and

identifying those media stream reports which include one of the correlation keys obtained from the signaling session report.

16. The method according to one of claims 10 to 13, wherein correlating the media streams and the signaling sessions of one or more services based on the correlation keys comprised within media stream reports of the media streams and signaling session reports of the signaling sessions comprises:

obtaining from a media stream report a correlation key identifying the media stream, and

identifying a signaling session report among the signaling session reports which includes the same correlation key obtained from the signaling session report.

17. The method according to one of claims 10 to 16, wherein steps a) and b) are performed by a correlation unit periodically for all signaling session reports and media stream reports generated within a configurable time interval, or in response to a request.

18. The method according to one of claims 10 to 17, further comprising steps a) to d) of claim 1 or the steps of the method according to one of claims 2 to 9.

19. A passive monitoring system for correlating media streams and signaling sessions of services in a packet-switched network, the passive monitoring system comprising:

a signaling plane probe for generating at least one correlation key for each direction of a service based on the information of the signaling session of the respective service monitored at the signaling plane probe, wherein the correlation keys for a respective service are generated using a timestamp indicative of the point in time at which the end of the respective signaling session of the respective service has been detected by the signaling plane probe and using information on the direction of the respective service,

wherein the signaling plane probe is further adapted to store a signaling session report for each signaling session in a database, wherein each signaling session report comprises the at least two correlation keys generated for the respective directions of the service,

a media plane probe for generating a correlation key for each media stream monitored by the media plane probe, wherein the correlation key is generated using a timestamp indicative of the point in time at which the respective media stream has ended and using information on the media data path of the respective media stream,

wherein the media plane probe is adapted to store a media stream report for each media stream in a database, wherein each media stream report comprises at least the correlation key generated for the respective media, and

a correlation unit for correlating the media streams and the signaling sessions based on the correlation keys comprised within media stream reports and signaling session reports.

20. A signaling plane probe comprising:

a network interface card for receiving packets of signaling sessions of multiple media services,

monitoring unit for monitoring the signaling sessions based on the packets of the signaling sessions received by the signaling plane probe to thereby obtain information on a respective one of the signaling sessions, wherein the monitoring unit is further adapted to detect the point in time at which a respective signaling session of the respective service terminates, and

a processing unit for generating at least one correlation key for each direction of a service based on a timestamp indicative of the point in time at which the end of the respective signaling session of the respective service has been detected by the monitoring unit and based on information on the direction of the respective service,

wherein the monitoring unit is adapted to store a signaling session report for each signaling session in a database, wherein each signaling session report comprises at least two correlation keys generated for the respective directions of the service.

21. A media plane probe comprising:

a network interface card for receiving packets of media streams of multiple media services,

monitoring unit for monitoring the media streams based on the packets of the media streams received by the media plane probe to thereby obtain information on a media data path a respective one of the media streams, wherein the monitoring unit is further adapted to detect the point in time at which a respective media stream of the respective service terminates, and

a processing unit for generating a correlation key for each media stream monitored by the monitoring unit, wherein the correlation key is generated using a timestamp indicative of the point in time at which the respective media stream has ended and using information on the media data path of the respective media stream,

wherein the monitoring unit is adapted to store a media stream report for each media stream in a database, wherein each media stream report comprises at least the correlation key generated for the respective media.

22. A correlation unit for correlating media streams and signaling sessions of services in a passive monitoring system of a packet-switched network, the correlation unit comprising:

a database access unit for accessing a database containing signaling session reports for signaling sessions of services and a database comprising media stream reports of media streams of the services, wherein the signaling session reports each contain at least one correlation key for each direction of a respective service, and the media stream reports each contain one correlation key indication the direction of the media stream,

a processing unit for correlating the media streams and the signaling sessions of the services based on the correlation keys comprised within media stream reports and signaling session reports to thereby identify the media stream reports and signaling session report belonging to a respective one of said one or more services, wherein the processing unit is adapted to identify one or more media streams of a respective service and the corresponding signaling session of the respective service based on the correlation keys contained in the media stream reports and signaling session reports, and

a service record generation unit for generating a service record for a respective one of said at least one service based on information contained in the media stream reports and information contained the signaling session report of said at least one service.

23. The correlation unit according to claim 22, further comprising means to perform the steps of the method according to one of claims 10 to 17.

24. A computer-readable medium storing instructions that, when executed by a processor of a signaling plane probe, cause the signaling plane probe:

to receive packets of signaling sessions of multiple media services,

to monitor the signaling sessions based on the packets of the signaling sessions received by the signaling plane probe to thereby obtain information on a respective one of the signaling sessions,

to detect the point in time at which a respective signaling session of the respective service terminates,

to generate at least one correlation key for each direction of a service based a timestamp indicative of the point in time at which the end of the respective signaling session of the respective service has been detected and based on information on the direction of the respective service, and

to store a signaling session report for each signaling session in a database, wherein each signaling session report comprises the at least two correlation keys generated for the respective directions of the service.

25. A computer-readable medium storing instructions that, when executed by a processor of a media plane probe, cause the media plane probe:

to receive packets of media streams of multiple media services,

to monitor the media streams based on the packets of the media streams received by the signaling plane probe to thereby obtain information on a media data path a respective one of the media streams,

to detect the point in time at which a respective media stream of the respective service terminates, and

to generate a correlation key for each media stream monitored by the monitoring unit, wherein the correlation key is generated using a timestamp indicative of the point in time at which the respective media stream has ended and using information on the media data path of the respective media stream,

to store a media stream report for each media stream in a database, wherein each media stream report contains at least the correlation key generated for the respective media.

26. A computer-readable medium storing instructions that, when executed by a correlation unit, cause the correlation unit to correlate media streams and signaling sessions of services in a passive monitoring system of a packet-switched network, by:

accessing a database containing signaling session reports for signaling sessions of services and a database containing media stream reports of media streams of the services, wherein the signaling session reports each contain at least one correlation key for each direction of a respective service, and the media stream reports each contain one correlation key indication the direction of the media stream,

correlating the media streams and the signaling sessions of the services based on the correlation keys comprised within media stream reports and signaling session reports to thereby identify the media stream reports and signaling session report belonging to a respective one of said one or more services,

identifying one or more media streams of a respective service and the corresponding signaling session of the respective service based on the correlation keys comprised in the media stream reports and signaling session reports, and

generating a service record for a respective one of said at least one service based on information contained in the media stream reports and information contained the signaling session report of said at least one service.

27. The computer-readable medium according to claim 26, further storing instruction that, when executed by the processor of the correlation unit, cause the correlation unit to perform the steps of the method according to one of claims 10 to 17.

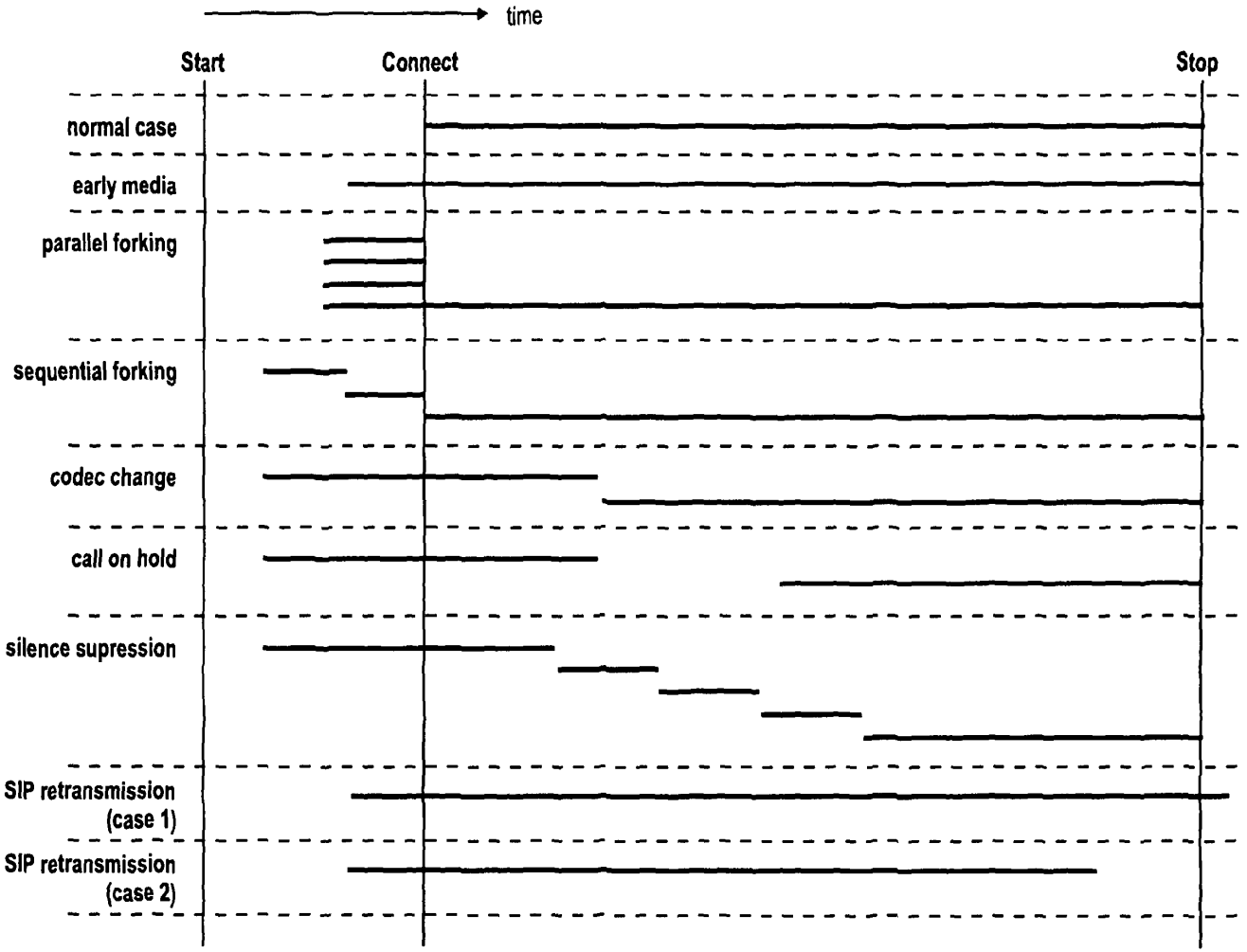

time
Start
Connect
Stop
normal case
early media
parallel forking
sequential forking
codec change
call on hold
silence supression
SIP retransmission (case 1)
SIP retransmission (case 2)

Fig. 1

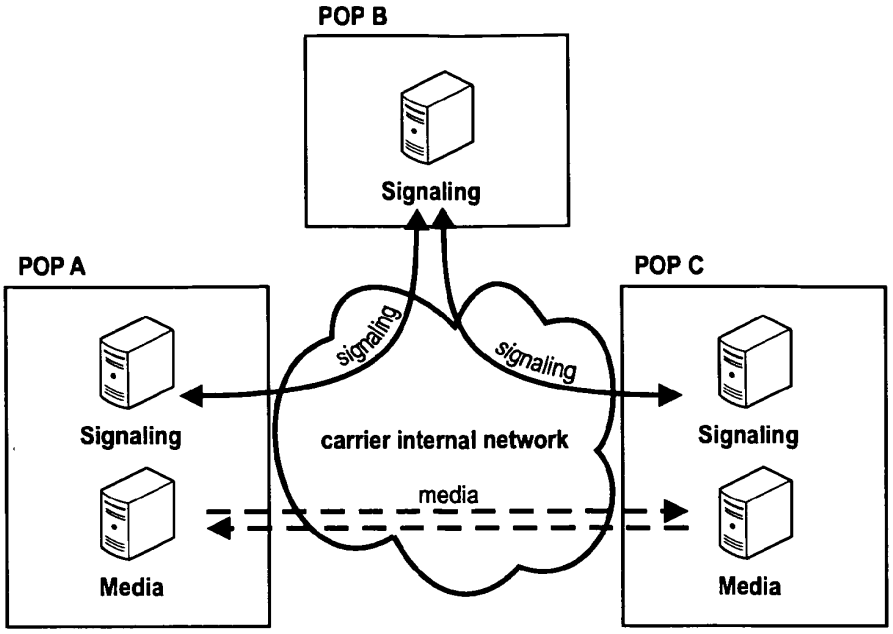

POP B
Signaling
POP A
POP C
signaling
signaling
Signaling
carrier internal network
Signaling
media
Media
Media

Fig. 2

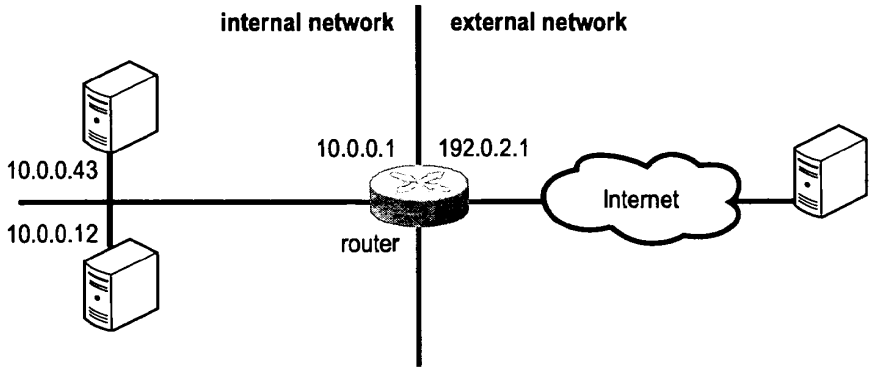

internal network
external network
10.0.0.1
192.0.2.1
10.0.0.43
10.0.0.12
router
Internet

Fig. 3

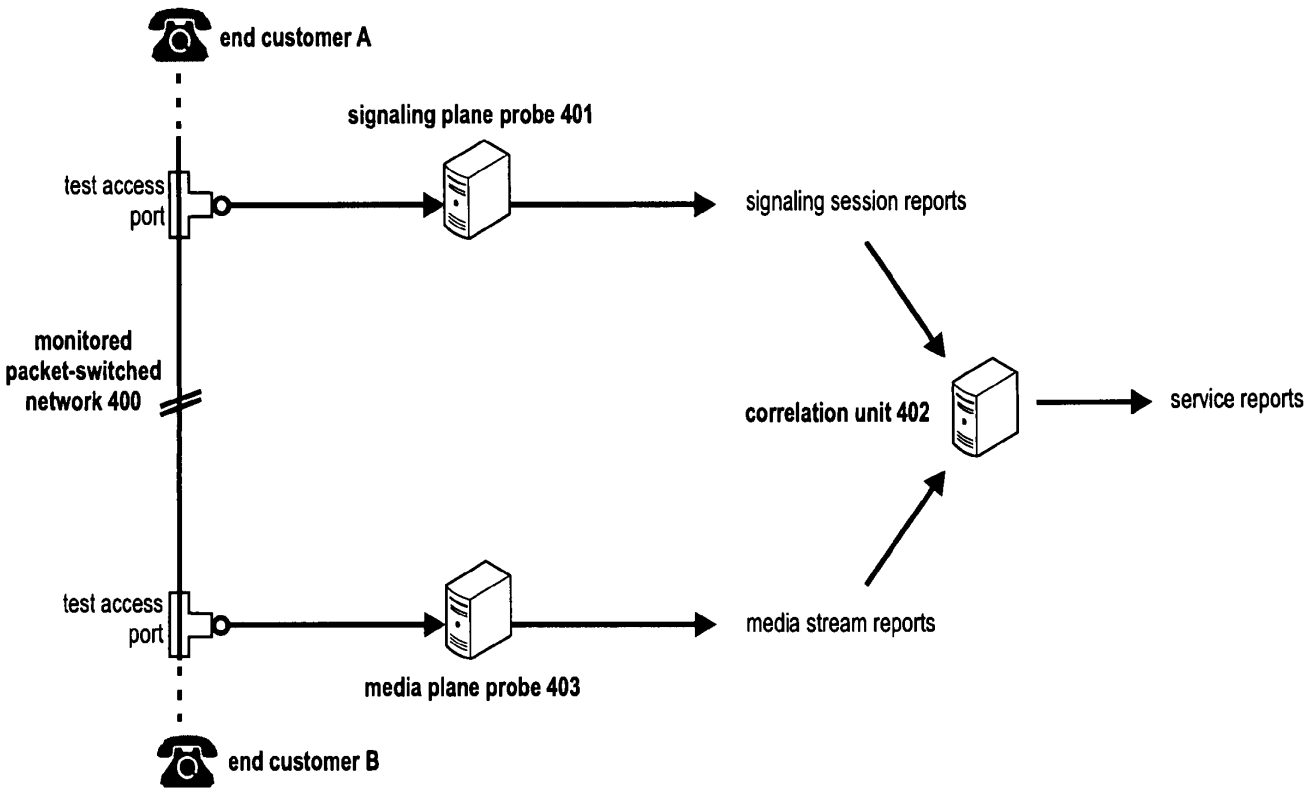

end customer A
signaling plane probe 401
test access port
signaling session reports
monitored packet-switched network 400
correlation unit 402
service reports
test access port
media stream reports
media plane probe 403
end customer B

Fig. 4

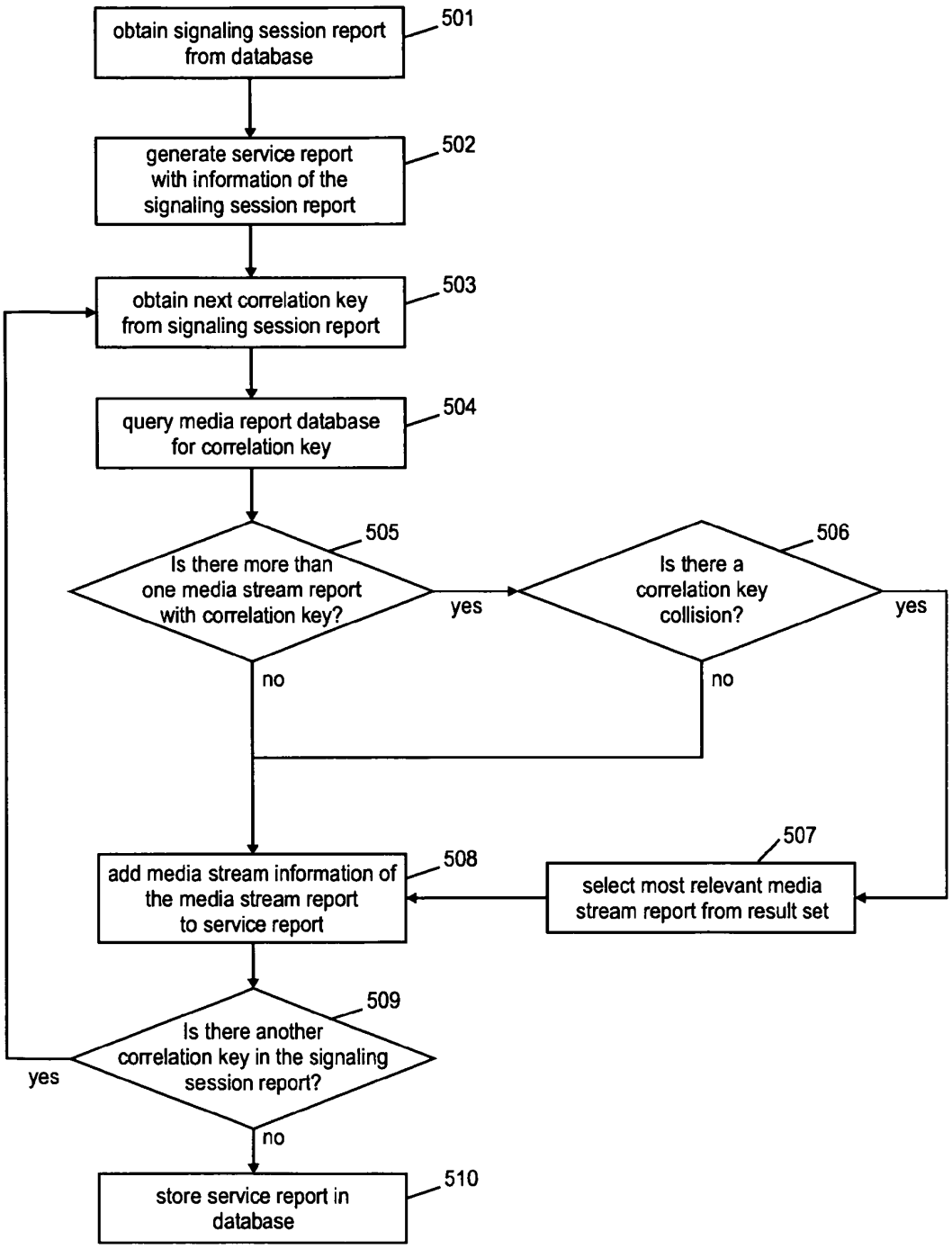

obtain signaling session report from database
501
generate service report with information of the signaling session report
502
obtain next correlation key from signaling session report
503
query media report database for correlation key
504
Is there more than one media stream report with correlation key?
505
yes
Is there a correlation key collision?
506
yes
no
no
add media stream information of the media stream report to service report
508
select most relevant media stream report from result set
507
Is there another correlation key in the signaling session report?
509
yes
no
store service report in database
510

Fig. 5

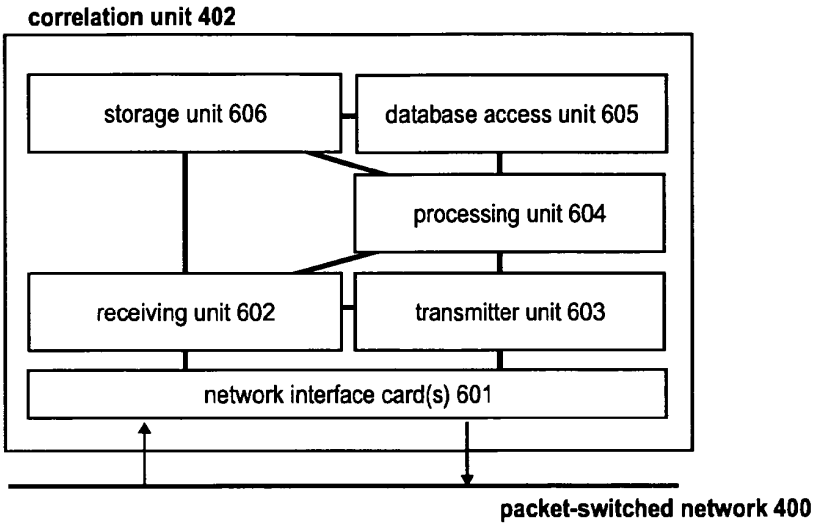

correlation unit 402
storage unit 606
database access unit 605
processing unit 604
receiving unit 602
transmitter unit 603
network interface card(s) 601
packet-switched network 400

Fig. 6

Fig. 7

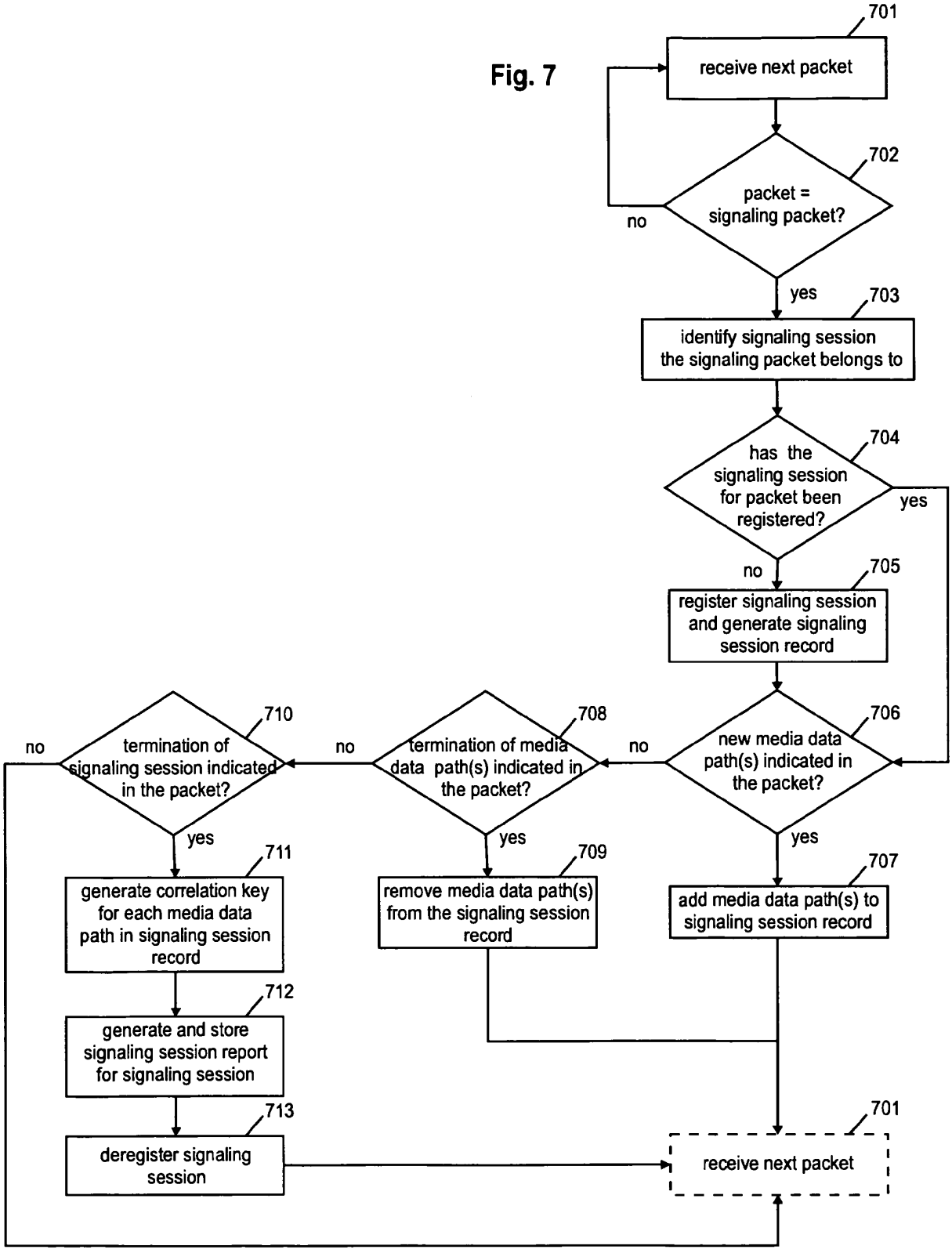
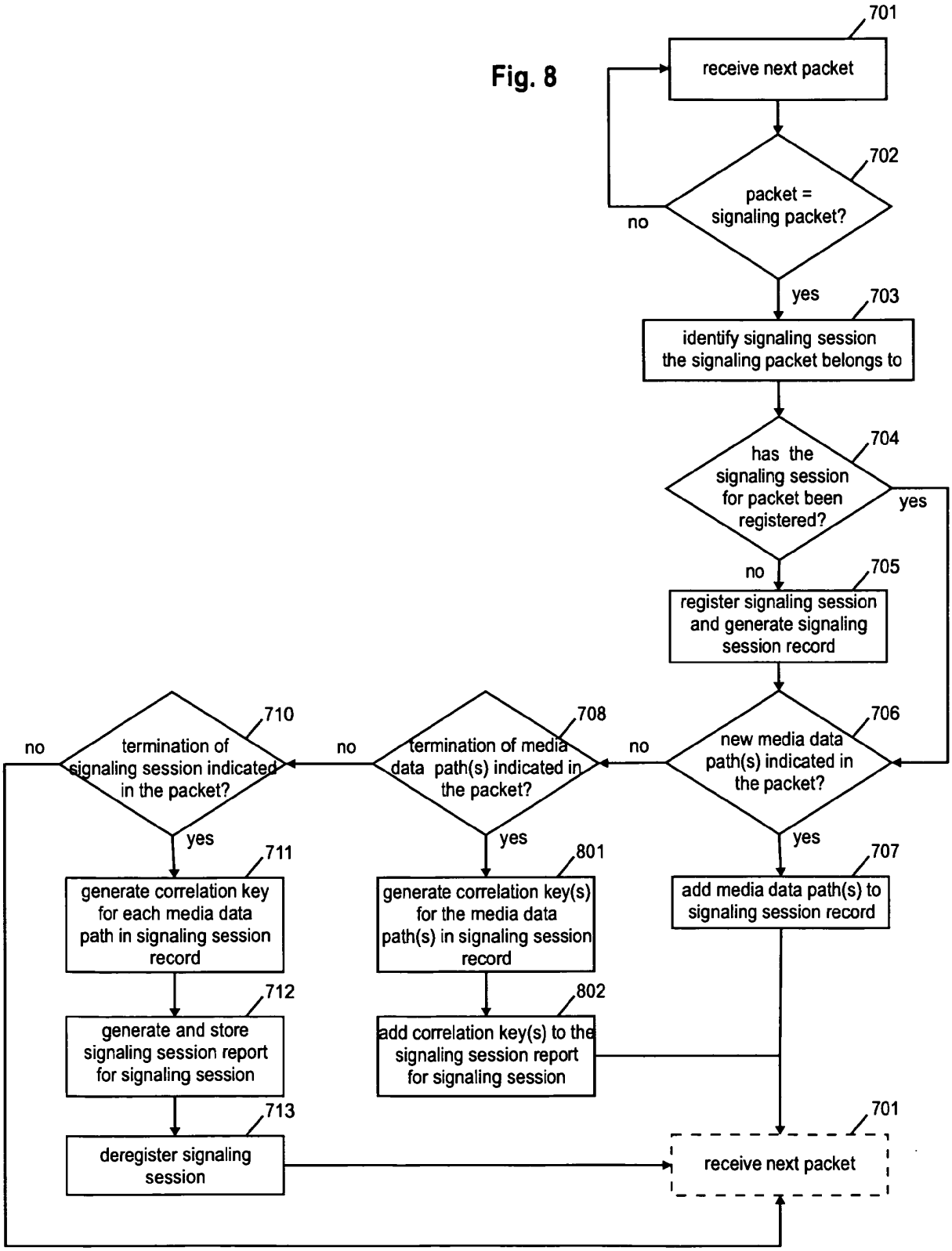

701
receive next packet
702
packet = signaling packet?
no
yes
703
identify signaling session the signaling packet belongs to
704
has the signaling session for packet been registered?
yes
no
705
register signaling session and generate signaling session record
706
new media data path(s) indicated in the packet?
no
yes
707
add media data path(s) to signaling session record
708
termination of media data path(s) indicated in the packet?
no
yes
709
remove media data path(s) from the signaling session record
710
termination of signaling session indicated in the packet?
no
yes
711
generate correlation key for each media data path in signaling session record
712
generate and store signaling session report for signaling session
713
deregister signaling session
701
receive next packet

Fig. 8


701
receive next packet
702
packet = signaling packet?
no
yes
703
identify signaling session the signaling packet belongs to
704
has the signaling session for packet been registered?
yes
no
705
register signaling session and generate signaling session record
706
new media data path(s) indicated in the packet?
no
yes
707
add media data path(s) to signaling session record
708
termination of media data path(s) indicated in the packet?
no
yes
801
generate correlation key(s) for the media data path(s) in signaling session record
802
add correlation key(s) to the signaling session report for signaling session
710
termination of signaling session indicated in the packet?
no
yes
711
generate correlation key for each media data path in signaling session record
712
generate and store signaling session report for signaling session
713
deregister signaling session
701
receive next packet

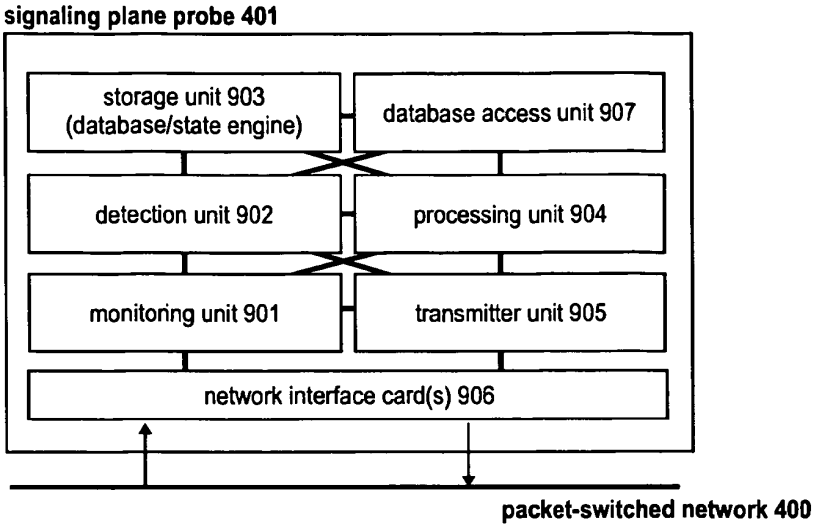

signaling plane probe 401
storage unit 903 (database/state engine)
database access unit 907
detection unit 902
processing unit 904
monitoring unit 901
transmitter unit 905
network interface card(s) 906
packet-switched network 400

Fig. 9

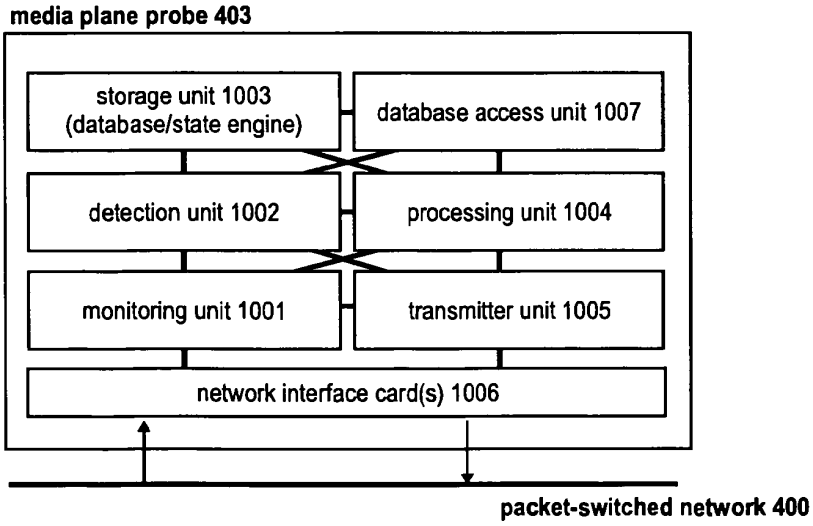

media plane probe 403
storage unit 1003 (database/state engine)
database access unit 1007
detection unit 1002
processing unit 1004
monitoring unit 1001
transmitter unit 1005
network interface card(s) 1006
packet-switched network 400

Fig. 10

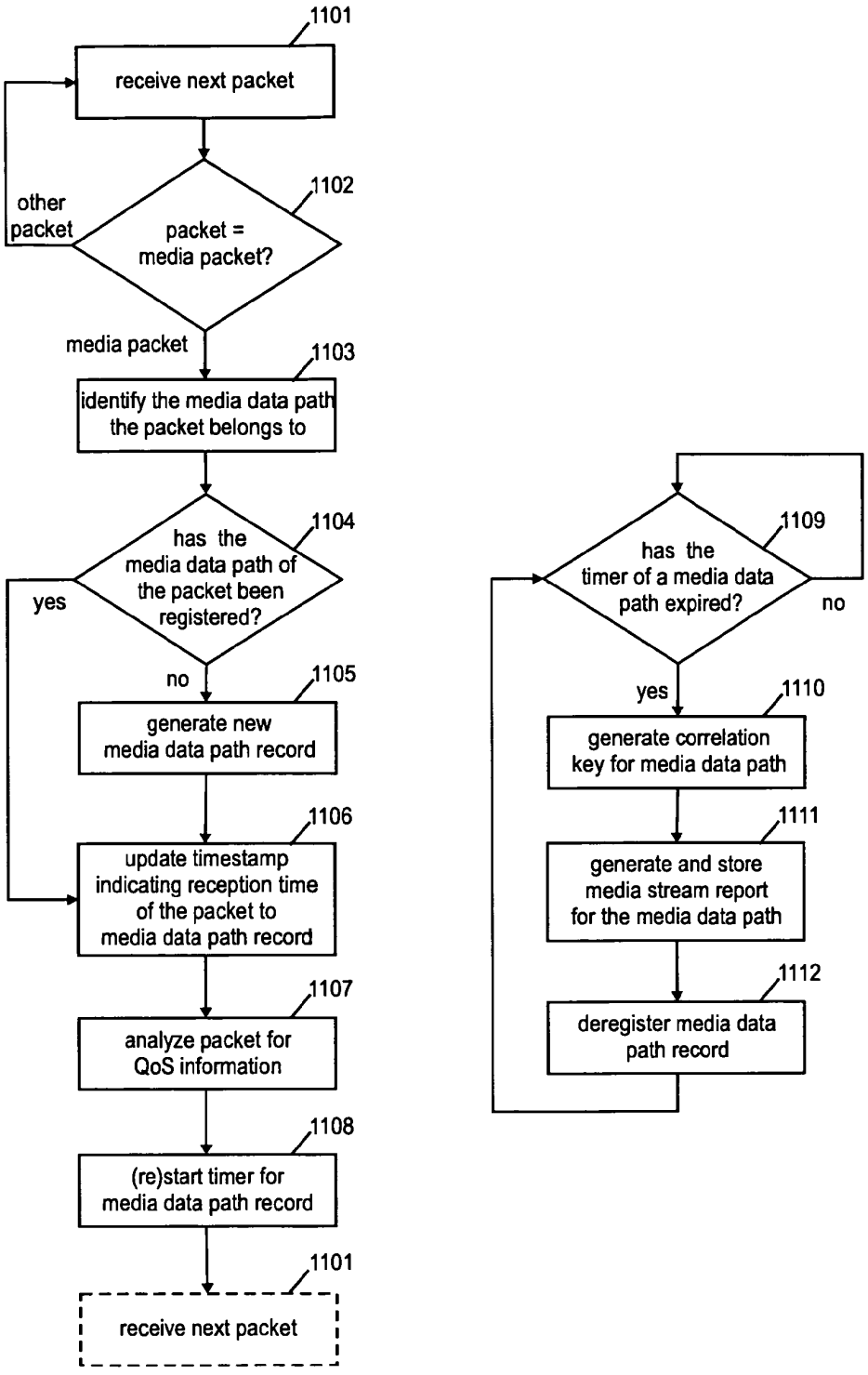

1101
receive next packet
1102
other packet
packet = media packet?
media packet
1103
identify the media data path the packet belongs to
1104
has the media data path of the packet been registered?
yes
no
1105
generate new media data path record
1106
update timestamp indicating reception time of the packet to media data path record
1107
analyze packet for QoS information
1108
(re)start timer for media data path record
1101
receive next packet
1109
has the timer of a media data path expired?
no
yes
1110
generate correlation key for media data path
1111
generate and store media stream report for the media data path
1112
deregister media data path record

Fig. 11

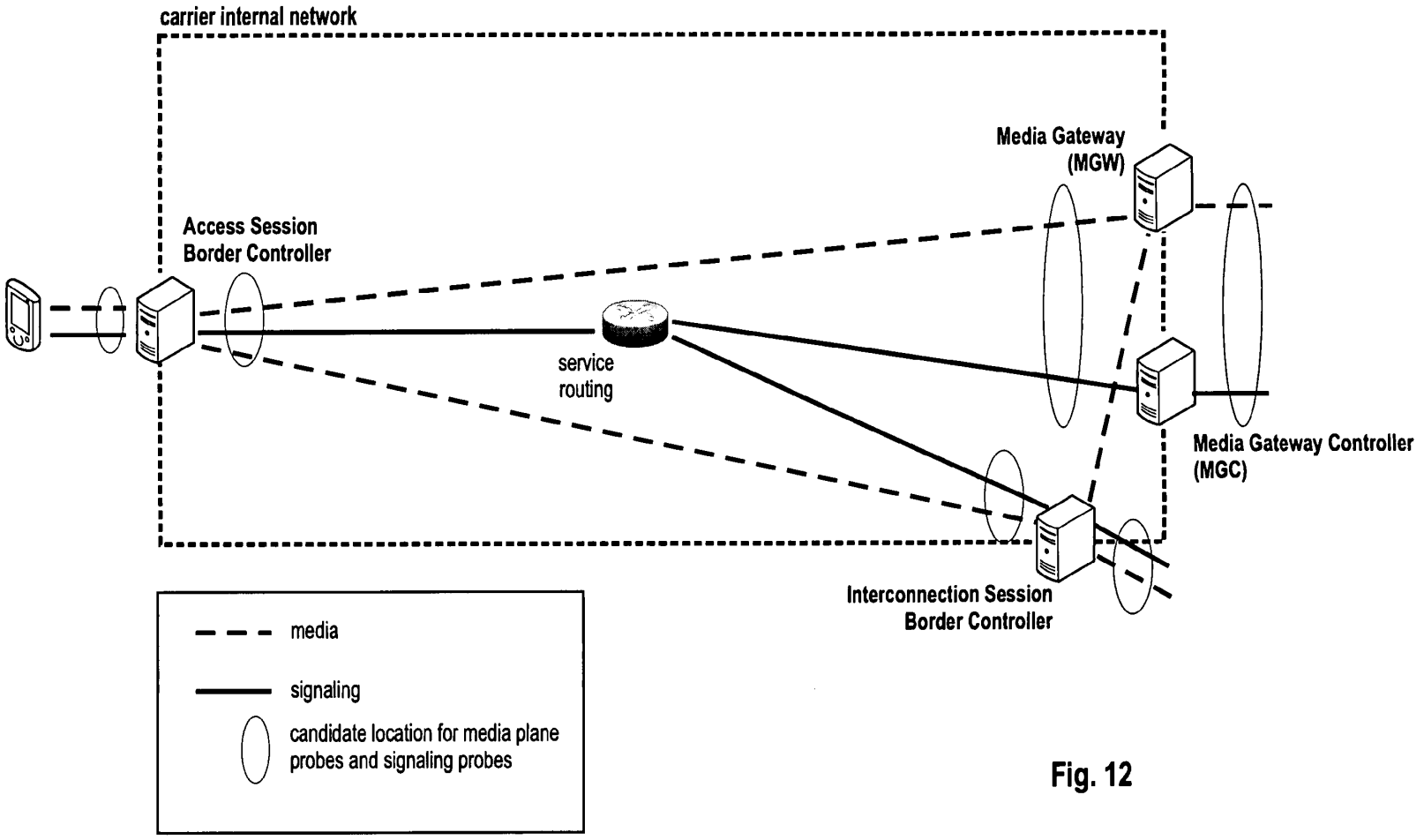

carrier internal network
Media Gateway
(MGW)
Access Session
Border Controller
service
routing
Media Gateway Controller
(MGC)
Interconnection Session
Border Controller
media
signaling
candidate location for media plane
probes and signaling probes

Fig. 12

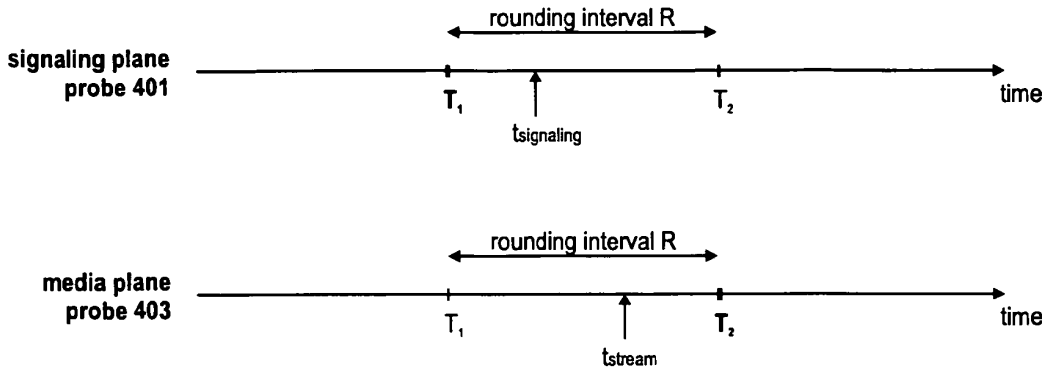

rounding interval R
signaling plane
probe 401
T1
tsignaling
T2
time
rounding interval R
media plane
probe 403
T1
tstream
T2
time

Fig. 13

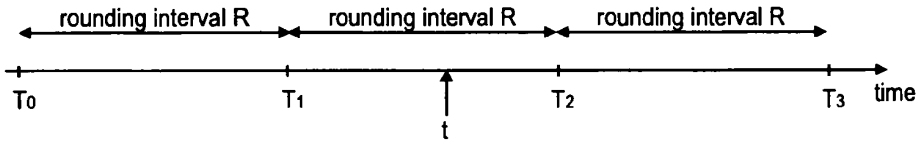

rounding interval R
rounding interval R
rounding interval R
T0
T1
t
T2
T3
time

Fig. 14

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2011/002145

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L29/06 H04L12/24
ADD. H04L12/26

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, COMPENDEX, INSPEC, IBM-TDB, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	ATSUSHI KOBAYASHI ET AL: "VoIP Measurement Architecture Using Data Mediation", 29 October 2009 (2009-10-29), IP OPERATIONS AND MANAGEMENT, SPRINGER BERLIN HEIDELBERG, BERLIN, HEIDELBERG, PAGE(S) 15 - 26, XP019133039, ISBN: 978-3-642-04967-5 page 18, line 5 - page 20, line 35	1-27
A	US 2008/151918 A1 (FOTI GEORGE [CA]) 26 June 2008 (2008-06-26) the whole document	1-27
A	WO 2010/106075 A1 (KONINKL KPN NV [NL]; TNO [NL]; STOKKING HANS MAARTEN [NL]; WALRAVEN FA) 23 September 2010 (2010-09-23) page 4, line 16 - page 8, line 26	1-27

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
10 January 2012	17/01/2012

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Aura Marcos, F

3

Form PCT/ISA/210 (second sheet) (April 2005)

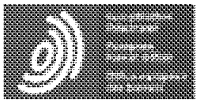
INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2011/002145

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2008151918 A1	26-06-2008	US 2008151918 A1	26-06-2008
		WO 2008078209 A2	03-07-2008
WO 2010106075 A1	23-09-2010	EP 2409432 A1	25-01-2012
		KR 20110125674 A	21-11-2011
		WO 2010106075 A1	23-09-2010

Form PCT/ISA/210 (patent family annex) (April 2005)


Espacenet

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 September 2007 (27.09.2007)

PCT

(10) International Publication Number
WO 2007/109541 A2

(51) International Patent Classification:
G06Q 10/00 (2006.01)

(21) International Application Number:
PCT/US2007/064175

(22) International Filing Date: 16 March 2007 (16.03.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/783,577 16 March 2006 (16.03.2006) US
60/794,006 21 April 2006 (21.04.2006) US

(71) Applicant and
(72) Inventor: **BRUBAKER, Curtis, M.** [US/US]; 46 Monarch Bay Drive, Monarch Beach, CA 92629 (US).

(74) Agent: **O'BANION, John, P.**; O'banion & Ritchey LLP, 400 Capitol Mall, Suite 1550, Sacramento, CA 95814 (US).

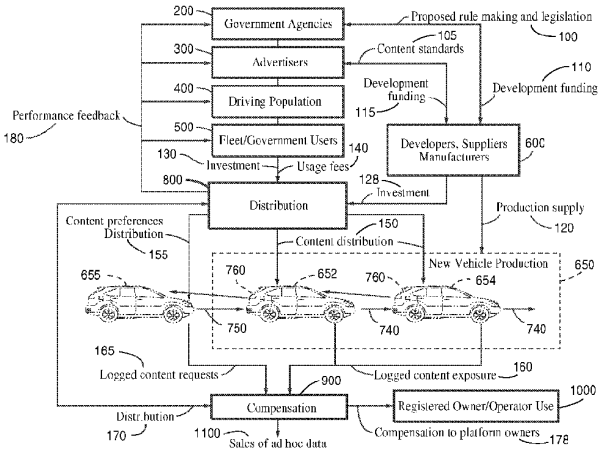
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYSTEM AND METHOD FOR OBTAINING REVENUE THROUGH THE DISPLAY OF HYPER-RELEVANT ADVERTISING ON MOVING OBJECTS



(57) Abstract: A system for obtaining revenue through the display of advertising on fixed or moving objects is disclosed. Such objects receive, store, poll and extract data to present content based on the time of day, locations, and relative movements in a specific environment, where data and information stored in moving or mobile objects within that environment can be transmitted between those objects or to stationary objects, where it is then used to enable and control such displays for which the owners of those objects are then compensated. The system enables the display of hyper-relevant ad content with synchronized remote audio, personal messaging and public service alerts on surrounding objects and provides for the real-time logging and later downloading of data to confirm communications and content deliveries between objects, track and measure consumer engagement, verify consumer's direct responses to ad viewings with accountability systems.

WO 2007/109541 A2

SYSTEM AND METHOD FOR OBTAINING REVENUE THROUGH THE DISPLAY OF HYPER-RELEVANT ADVERTISING ON MOVING OBJECTS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority from U.S. provisional application serial number 60/783,577, filed on March 16, 2006, incorporated herein by reference in its entirety, and from U.S. provisional application serial number 60/794,006, filed on April 21, 2006, incorporated herein by reference in its entirety.

[0002] This application is related to U.S. nonprovisional application serial number 11/552,932, filed October 25, 2006, incorporated herein by reference in its entirety, which claims priority from U.S. provisional application serial number 60/729,919, filed on October 25, 2005, incorporated herein by reference in its entirety.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[0003] Not Applicable

INCORPORATION-BY-REFERENCE OF MATERIAL SUBMITTED ON A COMPACT DISC

[0004] Not Applicable

NOTICE OF MATERIAL SUBJECT TO COPYRIGHT PROTECTION

[0005] A portion of the material in this patent document is subject to copyright protection under the copyright laws of the United States and of other countries. The owner of the copyright rights has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the United States Patent and Trademark Office publicly available file or records, but otherwise reserves all copyright rights whatsoever. The copyright owner does not hereby waive any of its rights to have this patent document maintained in secrecy, including without limitation its rights pursuant

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0006] This invention pertains generally to effectively providing advertising content with respect to a prospective consumer's desires and, more particularly, to an integrated system and method for directing hyper-relevant content to consumers through the use of wireless technologies for the exchange of information, including the compensation of users for providing or viewing content.

2. Description of Related Art

[0007] In the ad business, there are a handful of yet to be realized holy grails. They are: (1) Deliver your ad to the correct demographic (age, gender, income level, etc.); (2) Deliver it at a time its recipient is sure to see it (not when he's out for a soda or in the bathroom); (3) Pitch your ad to consumers who are truly interested in what you are selling (instead of wasting advertising dollars on those who are not); (4) Catch consumers when they're relaxed and open to suggestion (instead of preoccupied with other things); (5) Reach audiences TV can't reach and do it with more targeted spots in a single day than they'd catch on regular TV; (6) Connect with buyers at the right time (i.e.: sell them food at meal times); (7) Get your message to consumers when they are at an opportune location (for example, promote a great restaurant at meal time a couple of blocks from the place); (8) Put ads into the distribution pipeline in just seconds using the Internet; (9) Base billings on the actual content delivered and do it all electronically and automatically; (10) Give advertisers proof their ads actually ran and track them on a minute to minute basis; (11) Provide feedback to the agency or its clients on consumer engagement; (12) When your ad shows, let viewers buy (or at least get more information) at the push of a button. Unlike any existing commercial advertising system, the present invention delivers all of these objectives, and more, by creating a totally new kind of mobile communications platform and

forum.

[0008] ADVERTISING.

[0009] Madison Avenue is working to outdated media models and needs to figure out emerging media fast or it stands to lose billions of dollars to others who will. Over the next decade, major money will be set in motion to create entirely new ad platforms and new ways to reach consumers in what today is a $250 billion industry. In the process, the ways and means of the contemporary ad business are going to be turned inside out, not so much because of the present invention, but because of the many emerging technologies that make it possible. For years, marketers have taken their best shots at consumers. Much of it is creative guesswork which follows available technology and distribution streams that were invented for other purposes. Television, for example, developed as a communications resource and quickly evolved into a form of free entertainment where marketers could insert the products and services others wished to sell into the paths of consumers while they were having a good time. It was not important if those people were really interested in the products and services; TV was simply the best way to expose products and services to consumers because TV had traffic. However, ad executives are beginning to recognize how much money is being wasted on antiquated media plans. For the first time in history, TV Spot ads were the biggest loser in ad spending for 2005, sinking 9.5% to $15.5 billion, because viewers have 'left the building.' Yet, newspapers, magazines, cable, and syndication all showed gains, with the category of outdoor advertising rising by 9.8% to $3.5 billion.

[0010] While consumers are looking for new things to do and new ways to entertain themselves, marketers are looking for audiences. However, audiences today are increasingly sophisticated about avoiding advertising messages. What is needed isn't just any audience, but an audience that truly cares about what advertisers have to say and sell, and those ad messages have to be bright, quick, and relevant to the new audience's current interests and lifestyle. Today, the gold rush is toward the Internet and to what has been

termed the "3rd screen," those tiny carry-around displays found on cell phones, iPods, and PDAs. Once again, marketers want to insert themselves (and their content) into our everyday business and fun. It is estimated that as much as 25% to 30% of the $100 billion spent each year on brand advertising will soon find its way to such mobile screens, but will users who pay for such services accept advertising? To borrow a statement from a highly-placed marketing industry executive: "If I'm a 22-year old male and I've downloaded four 3D games to my phone in the last 30 days, chances are pretty good I'd be interested in seeing a Sony PS3 commercial on my device." He may be right. Recent studies have determined that kids like watching ads as long as they represent something they're interested in. Look at the Super Bowl, where large numbers of viewers tune in primarily to watch the ads. Consumers like these want to participate, want more to talk about, and want ultimately to have a say, so they can certainly take the time to tell advertisers about the kinds of content they would like to see. Yet, today's marketers have failed to recognize that there are technologies emerging today that can open dialogs between consumers and creatives and still leave the field open for expression and execution by the professionals in it.

[0011] New canvases are needed along with the audiences to go with them, but to secure them will require more than repurposed TV spots for tiny screens on the go. What is required is a systemic rebirth of the ad game including novel applications and pragmatic distribution. But more than anything, it requires marketers to get more in touch with consumers who are more in touch with themselves, with consumers who are articulate enough to tell us at least two things: 1) who they are, and 2) what they want. Technology can enable this, but there needs to be an economic reason for consumers to opt into the game. The present invention has been conceived to provide just that.

[0012] VISUAL CONGESTION

[0013] American cities are converging on BladeRunner-like visual clutter, and few regulatory brakes have been applied to this accelerating buildup. Electronic displays on taxi tops vie with sign-wrapped busses and trailers, bus

shelters plastered with ads, lighted store signs, and a profusion of billboards, many with revolving panels and flashing lights, and now many with radio-enabled audio. The situation promises to worsen. Nanoscale components promise larger, cheaper, high-resolution video screens that are bound to find their way into signs, surfaces, and products of all kinds. We are already inundated with visual ads: our daily sensory diet includes thousands of commercial impressions, from pop-ups and animations in Web content to TV's full arsenal of 30-second spots, now with product placements embedded in what remains of the entertainment. With increasingly new technologies and eye-catching designs competing for drivers' attention, regulatory agencies, traffic engineers, and advocates are beginning to question which of these potential distractions may also pose a risk to driving. To what extent can commercial signage be blamed for accidents? What do we know about the nature of attention and distraction? Can this proliferation be rationally controlled?

[0014] Surveys show that signs and billboards indeed compromise traffic safety, especially at intersections and on curves. Flashing lights, motion, visual clutter, and novelty (e.g., the Times Square / Las Vegas Strip effect that is replicating so quickly in major cities) are implicated as part of the problem. Additionally, research on driver distractions confirms that visual complexity compromises safety by forcing drivers to scan the environment longer for street signs, turns, or critical landmarks. Researchers using eye-tracking devices to monitor drivers in traffic are finding that video signs are more distracting than static signs and can act as catalysts, increasing ad gazing of all types, even in unsafe situations. Similar studies are documenting the effects of visual distractions on both young and old drivers to determine what constitutes safe, manageable amounts of information, and the effects of animation, video, and moving panels on drivers, in efforts to set standards so that electronic signs and billboards are legible and non-distracting. To be non-distracting, however, is to contradict the entire purpose of outdoor advertising, and therein lies the problem.

[0015] It is undisputed that people are easily distracted. Theories of attention point out relationships between mental excitation/interest and the ability to perform tasks. We function best somewhere between boredom and over-stimulation. What researchers have found is that new or unexpected stimuli trigger involuntary responses and the more bored or unfocused we are, the more susceptible we are to these surprises. Studies have also identified modes of visual perception: a focal, or search, mode that is narrow and specific, and an ambient mode that is a sort of default, which is not focused on anything in particular, but having better peripheral awareness. We've learned that computer users performing search tasks in the center of the screen are slowed when objects appear on the perimeter, even when they aren't consciously aware of them. The more peripheral objects, the greater the distraction. Thus, the more we direct our attention towards that periphery, the less we focus on what's in the middle of our screens, or in the middle of our roads. Researchers have also found that new, moving and looming objects command attention; that the onset of motion triggers overriding or urgent attention, possibly tapping the survival instinct; and that changes in color can capture attention. None of this proves that outdoor ads cause accidents, but behavioral mechanisms clearly come into play when drivers encounter roadside signage. The notion that content is a major factor in drawing attention is starting to be examined. Because content and distractions are inevitable, a solution may lie in controlling them and in determining exactly how, when and where they should occur. That's a tall order, but it is also a primary objective of the present invention.

[0016] PHYSICAL CONGESTION

[0017] Just as advertising seems to clutter our landscape, so do moving objects. Traffic congestion in our major cities has reached epidemic proportions, prompting nationwide studies to get a handle on the problem. In places like Los Angeles, where, in the late fifties, one could drive anywhere to its outskirts in 20 minutes, it now takes hours, regardless of the time of day. According to the 2005 Urban Mobility Study Report, released in May by the

Texas Transportation Institute of Texas A&M University, sixty-seven percent of the peak period travel was congested in 2003, compared to only 32 percent in 1982; fifty-nine percent of the major road system was congested in 2003 compared to 34 percent in 1982. The number of hours of the day when congestion is encountered has grown from about 4.5 hours in 1982 to more than 7.1 hours today. Traffic congestion is worse in the large urban areas than in the smaller ones, but even the smaller areas are unable to keep pace with rising demand. According to the study, 10 years is not an unrealistic timeframe to go from an idea to a completed project or to an accepted program due to the significant planning and development required. But in ten years at current growth rates, the urban area average congestion values will jump to the next highest classification—i.e.: medium areas in 2013 will have the congestion problems large areas had in 2003.

[0018] What does this mean in cost to consumers? In 2003, congestion (based on wasted time and fuel) cost drivers about $63.1 billion in the 85 urban areas studied. The average cost per traveler in those 85 urban areas was $794. Costs ranged from $1,038 per traveler in "very large" urban areas down to $222 per traveler in the "small" areas—and all this with fuel costing far less than it does today. Of the 85 urban areas studied, 2.3 billion gallons of fuel were wasted, an amount that would fill 46 supertankers or 230,000 gasoline tank trucks. In urban areas with populations greater than 3 million, 1.5 billion gallons of fuel (more than two-thirds of the total!) was wasted. At $2.00 per gallon, that's $3 billion tossed out by drivers in stop and go traffic. During that same period the Outdoor Advertising Association of America (OAAA) reports advertisers spent $5.5 billion on out-of-home advertising. If outdoor advertisers had instead placed that $5.5 billion in ads using the present invention, they'd not only have put their money into a superior ad platform, they'd have been able to reimburse drivers for their wasted $3 billion and still have $2.5 billion left over to spend elsewhere.

[0019] Plenty of studies have been launched to figure out what to do about our worsening traffic, but few concrete proposals have surfaced. Certainly, no

proposals have emerged suggesting that we compensate drivers for their daily delays. Just the same, until we have a meaningful cure for congestion, the present invention can, to a degree, offset owners for their wasted time and money while they experience it.

[0020] MOVING OBJECTS

[0021] For as long as there have been moving objects, people have advertised on them. From early buckboards and stagecoaches to modern trains, subways, busses and aircraft, moving objects are the basis for corporate and self promotion, personal expression, identification and advertising. Today, it is not unusual to see cars, trucks, entire busses, even independently-towed trailers adorned with wraps or ads for products and services—many of these are backlighted to draw the attention of drivers and pedestrians while they parade down city streets nationwide. One of the world's largest advertisers, ClearChannel Outdoor, markets a variety of Taxi Tops for use in city traffic atop cabs. These come in various sizes and in two, three or four sided models which are typically backlighted to display graphics at eye level to passing sidewalk or pedestrian foot traffic. Some of these have lenticular screens to create movement, built-in LED readout pads, or larger LED panels to produce scrolling text, or more elaborate displays with flash type animation. The latter can be optionally linked to an onboard GPS system so it can be programmed to show only on specific streets or in special areas of the city. On these displays, ClearChannel's clients control the content, not in real time, but well in advance of street exposure. ClearChannel also offers a 14" x 36" flat, bolt-on trunk mounted sign which is positioned more or less at eye level to following motorists. ClearChannel's Taxi Media division refers to these as advertising "tonnage" since the car behind will view the same static, print ad for 5 to 20 minutes (the time a vehicle spends behind it in traffic in such places as New York City).

[0022] Another company, MotionLED, sells a long, slim LED panel which can be attached with suction cups to the interior of a vehicle's windows where it can scroll text messages horizontally, similar to those present in retail stores.

These units are available in four lengths, from 8 to 40 inches, and in a choice of lettering in one of three colors. All of these are piecemeal efforts toward the presentation of advertising messages in public spaces and represent little more than a "shotgun" approach to demographic targeting. Advertisers using these techniques can only hope they will get exposure in the right part of town or at the right time of day and to people who may be inclined towards their product.

[0023] DISTRIBUTION

[0024] Madison Avenue is working aggressively to convert its costly, labor-intensive and still largely paper-based process of buying and billing media. Even the Internet has yet to develop the crucial rules of business or programming schemas for buying and selling online that Madison Avenue truly needs. A recently released status report compiled by the American Association of Advertising Agencies (AAAA) shows that the Internet has achieved only one out of nine essential steps for conducting all electronic advertising, giving it the worst "discrepancy" rates (the percentage of advertising buys that don't run as originally ordered) of any major medium. That puts the Internet on par with out-of-home, radio, and network TV as the least progressive electronic business trading partners. Even the most analog of all media—magazines and newspapers—have made greater electronic business strides than the Internet and are today capable of at least taking orders electronically. The advertising community needs to engage in a massive combined effort before it can claim independence from the labor-intensive manual steps of paper processing and human input which drives up the cost of media buys and produces accounting inaccuracies. The present invention represents not merely an ad platform to deliver hyper-relevant content, but is one which is totally electronic from the ground up, integrating creativity, distribution, tracking, measurement, engagement, accountability and compensation into a single, verifiable, automated closed-loop network. As comprehensive and seamless a system has been previously unheard of in the world of contemporary advertising.

[0025] TELEVISION

[0026] Advertisers place ads everywhere from skywriting to urinals and from grocery store check-out dividers to the fruits and veggies themselves. However, it has always been the full motion video display that delivers the goods. Advancements in flat screen technology have allowed displays to creep into elevators, gas pumps and restrooms—wherever someone with a need to pause for a few seconds is present. Just as relentlessly, marketers are pursuing consumers who are on the move. With nearly 200 million U.S. subscribers to wireless services, marketers are wondering if ads beamed to wireless devices such as cell phones, Blackberries and hybrid devices will evolve into a viable media platform without upsetting their users. Some have referred to GPS cell phones as "the silver bullet" since they promise to let marketers send retail pitches to mobile users who might be in the vicinity of their stores. Chrysler Corporation is launching its own Mobile Phone TV Channel called "The Jeep Channel" which will actually run TV style commercials on cell phones.

[0027] From tiny screens to huge ones: ClearChannel's Outdoor Division and Digital Advertising Network, a Montreal provider of digital screen networks, have teamed to place hundreds of 4 x 16 foot video displays in shopping mall food courts across the country. Among other things, these will show the same kinds of 30-second TV spots you can see in your living room. And despite declines in viewership due to ad-skipping Digital Video Recorders (DVRs) and media alternatives such as the Internet, American viewers still spend between four and eight hours per day, per person, glued to their TVs—more than any other nation—but with Japan a close second. If these hours are spent watching commercial television, it's a fair guess they're watching about 15 minutes of commercials as well. Look again at the 2005 Urban Mobility Study Report. In the "very large" areas studied, there were 61 hours of delay per traveler per year, which is 219,600 seconds a year. In television ad terms that's nearly thirty 30-second commercial spots per day, and nearly sixty 15-second commercial spots each day. In Los Angeles, the numbers are

even better. Angelinos experience 93 hours of delay per traveler per year or 334,800 seconds a year. That equals 44.64 30-second TV spots per average day, or nearly 90 15-second spots per day.

[0028] MEASUREMENT

[0029] Getting a handle on who's watching what in terms of advertising is critical to proving the worth of a media platform. This need for proof has spawned a variety of devices, programs and companies—some are now household words—whose job it is to measure the existing ad viewer base. ClearChannel is evaluating Requests for Proposals (RFPs) looking for new electronic devices to measure its huge radio audiences. Arbitron, a firm specializing in ad measurement, has developed a device called a portable people meter (PPM), a passive electronic device intended to replace the old-fashioned paper diary methods of documentation. Nielsen Media Research, meanwhile, plans a "portfolio" strategy toward adopting devices to improve the accuracy of its TV ratings services, which today also measure the time-shifted viewing of DVRs, video-on-demand and other competing platforms such as the Internet and mobile media. In December 2005, Nielsen Outdoor delivered its first wave of out-of-home demographic data to reveal the most likely people to see advertising on billboards. Nielsen convinced respondents to wear GPS-equipped pager size devices called Npods, then combined their respondents' traffic patterns with a map of outdoor advertising sites to determine who passes what kinds of outdoor ads, and when. Nielsen's data does not provide demographics on a per user basis for individual outdoor sites; it only gives advertisers a sense of what consumers could have been exposed to and roughly how often. The methods for gathering and applying ad measurements are woefully lacking in an industry that, in other respects, is mature and widespread.

[0030] Fundamental to ad measurement is defining what constitutes an "impression" or a "view." Special software has been created to measure billboard views. Systems for television identify the number of household impressions, the time of day video-on-demand (VOD) programs are viewed

[0031] The present invention does not have these problems. First, the present invention is not inserting commercials into an entertainment stream, it is showing ads exclusively. Secondly, since every showing is essentially "requested" by an electronic device on one moving object or another, all such requests and completed deliveries are confirmed as they occur and are logged for later downloading. Such integral measurements reveal what content was viewed, precisely where it was viewed (based on GPS data), when and by whom it was viewed (in significant demographic detail), and for exactly how long. The system also documents whether or not any impulse information was requested or if any purchases were made as a result of the viewing. Finally, the system can the track geographic movements on the part of moving objects after content viewings to confirm local area effectiveness—for example driving to a specific hotel or restaurant immediately after seeing an ad for it. Furthermore, every content request—whether or not it resulted in a confirmed delivery—is recorded on the hard drive of every involved moving object. This means that a detailed 24-hour map—a "data snapshot," if you will—can be

reconstructed to tell marketers where consumers went, when they went, who may have been with them at the time, what they were looking for as they did and whether or not in some cases they found it. To the best knowledge of the Applicant, to date, no other organization, system, means or technology has conceived or proposed offering such an exacting and automated capacity to deliver, measure, collect, process, assess, re-create, distribute, confirm, and then measure again the absolute effectiveness of conventional or hyper-relevant advertising and other types of consumer content on such a widespread scale and, in addition, to reward the average consumer for so doing.

[0032] TARGETING

[0033] Geo-destination targeting is a process of combining IP-based targeting—a very common technique that allows advertisers to target ads based on the location of a user—with, for example, information about the city a user is searching for. For example, if a user in San Francisco is searching for an address in Austin, Texas, online marketers can target airline ads advertising cheap fares from San Francisco to Austin.

[0034] In the same way, the present invention can poll information from a moving object. For example, by using its onboard GPS Navigation system with its selected destination and route and then coupling that information with an owner's stored profile, a wide variety of highly directed ads can immediately be triggered for presentation on any of the moving objects it might encounter en route: retail stores of high interest, restaurants with favorite foods, hotels, trip stops or other points of interest that are known to meet his or his family's preprogrammed interests. Additionally, if onboard data indicates a history in searching out or purchasing certain kinds of items, such as men's leather boots, for example, and the traveler is passing through a town where there is a manufacturer specializing in fine leather boots, the present invention can "borrow" the displays of surrounding objects as the traveler nears town or passes through it, and call that fact to his attention. The providers of those borrowed displays will be compensated in kind for their use.

[0035] BEHAVIORAL TARGETING

[0036] Behaviorally-targeted advertising is a way for marketers to get better aquatinted with their buyers. The Internet allows researchers to track buyers after the fact, to measure the things they look at, browse through or order. When they go back for more they can be "retargeted" with relevant advertising, relevant because they've already expressed significant interest in it, or perhaps in something related to it. Retargeting is used to cross-sell customers who previously bought something, or to contact prospects again in an attempt to lead them to a conversion event, such as a sale, a download or a newsletter signup. This works better with some groups than with others but it's another way for potential buyers to tell potential sellers who they are and what they want through their actions. Why not take this idea to the next level—as with the present invention—where buyers and sellers actually cooperate in such an exchange and where each side benefits greatly from the effort?

[0037] BUZZ TARGETING

[0038] Nielsen BuzzMetrics, a company which specializes in the measurement of consumer-generated media and word of mouth, claims, "In a world where word of mouth is on steroids, marketers need to focus just as much on the negative buzz as on the positive," then goes on to characterize most advertisers as "ill-equipped" to respond to the quick-acting blogospheres so famous for churning out bad news at prodigious rates. The real 'read-between-the-lines' news here is that today's consumers are quite willing to speak out, and speak out loudly, about having better products and services. In other words, when it comes to understanding their customers, advertisers no longer have to be reactive, they can be proactive. They can ask. This is something marketers have not often bothered to do. Learning what buyers really want and really care about—what truly turns them on—and using that to mutual advantage is a critical part of the present invention.

[0039] INTERACTIVITY

[0040] Remember the last item on the list of advertising holy grails? Buy at the

push of a button? XM Satellite Radio is preparing to ship portable MP3 players that will do just that, at least for music. A deal with Napster lets users push a button to "bookmark" songs they hear, and then the next time they dock their player with their computer it automatically buys those songs over the Internet. This is possibly the start of a trend. The present invention, however, goes a few steps further, thanks to a totally integrated advertising and distribution infrastructure. When users in moving objects in the field see an ad that intrigues them, they can push a button to get immediate information (such as pricing and availability), or they can request other things such as having a brochure sent to their computer. If they really like what they see and hear, however, they can buy the item on the spot. No need to dock with the Internet to complete a transaction, because credit and shipping have all been set up in advance, as will be explained later. You can even request a day or two to change your mind before the product ships. Imagine you're major film studio releasing a big movie for the weekend. You've budgeted a substantial ad spend to promote the grand opening with significant print and TV spots to generate excitement. How would you like to be able to deliver a mind-blowing 30-second trailer with surround sound to 100,000 teenage boys driving at 6:00 PM on a Friday night just as they're trying to figure out what they're doing that weekend? Then follow it up with a push button opportunity to have tickets waiting at a theater just minutes from their precise locations along with an electronic discount at a nearby Burger King? The present invention allows exactly such a promotion to be executed, both instantly and seamlessly.

[0041] THE NEW CANVAS

[0042] Computer and display technologies are considerably more advanced than the world of advertising which uses them. Liquid crystal displays (LCDs), Plasma and Digital Light Processing (DLP) have hit their stride and, although they are still expensive for many consumers, they are today produced in volume and their costs are falling. LCDs were invented around 1963 and were first intended as slimmed-down replacements for bulky CRTs or as screens for wall mounted TVs. Scaling up to large surfaces, unfortunately, was a problem.

Instead, LCDs became the standard display for everything from watches to laptop computers and, as is commonly known, are rapidly replacing conventional lighting including brake lights and tail lamps on production automobiles.

[0043] A newer product, Organic Light Emitting Diodes (OLEDs), pioneered and patented by Kodak/Sanyo, now promise that original vision with higher levels of brightness and sharpness not possible with previous technologies. OLEDs are self-luminous but don't require the backlighting, diffusers or polarizers required by LCDs. OLEDs consist of two charged electrodes sandwiched on top of organic light emitting material, which eliminates the need for mercury lamps and yields a thinner, lighter display having very low power consumption. These displays are tough enough to use in portable devices and automobiles, can be viewed at angles up to 160 degrees and are able to produce clear, distinct images, even in bright light. OLEDs also produce high image resolutions, and because each pixel can be turned on or off independently, they can create multiple colors in a very fluid, smooth edged display. They are 20% to 50% cheaper than LCD processes yet utilize plastics that make them tougher and more rugged. Manufacturers claim processes that are akin to the "printing" of newspapers and anticipate panels that are bendable and potentially formable with no inherent barrier to large size glass and displays. In other words, these should be very cheap in the future.

[0044] Meanwhile, scientists at the Fraunhofer Institute for Applied Polymer Research (IAP) in Potsdam are making OLEDs transparent. While metal oxide coatings in earlier OLEDs made them opaque, researchers are investigating transparent physical properties. Such displays can be switched on and off to create graphics or video on demand, embedded in normal glazing. Because these new panels interfere little with visible light or views, their development is ideal for such applications as heads-up displays in car windshields or for the kinds of displays anticipated by the present invention. Of significance is that under the proposed business model, capital investment

will become available to refine processes and develop the necessary production equipment for the earlier supply of OLED displays on a cost-effective basis, volume-matched to an established and mature transportation industry.

[0045] GOVERNMENT AGENCIES

[0046] It may be beyond the practical scope of this document to assess all of the potential areas of interface with respect to the U.S. government and the present invention. Although the U.S. government has many ongoing programs relating to moving object telecommunications, the applicant is unaware of any specific government prior art that would conflict with this invention. One envisions the government's role as essentially responsive and supportive towards the present invention by providing or assisting in the aspects of technical assessment, rulemaking, the creation of new standards and legislation. There are areas, however, where the U.S. government is independently or with private contractors exploring programs where the intent or results could overlap and lead to similar conclusions or cooperative implementations. Here are three such examples:

[0047] 1) The U.S. Department of Transportation has launched its Intelligent Transportation System (ITS) program, designed to improve transportation safety and mobility and enhance productivity through the use of advanced communications technologies. ITS encompasses a broad range of wireless and wire line communications-based information and electronics technologies which, when integrated into the U.S. transportation infrastructure and into vehicles themselves, should relieve congestion, improve safety and enhance American productivity. One such ITS study involves the creation of a LIGHT VEHICLE FORWARD-LOOKING, REAR-END COLLISION WARNING SYSTEM for eventual integration into passenger cars worldwide. This study attempts to establish guidelines for the creation of a driver warning system which can prevent or reduce rear end collisions.

[0048] In some respects the proposed system is similar to the present invention in the sense that it proposes the use of a forward looking proximity

sensing technology to read the speed of a vehicle in its forward path, calculate the range and closing speeds and then issue an 'impact warning' or a 'following too close warning' if conditions warrant, one that will be noninvasive to the driver of the following (host) vehicle. ITS properly assesses the available data on driver inattention (some of which was previously taught) but then fails to make recommendations consistent with its own and common findings. For example, Section 3.2.1, Background for the ITS guidelines, states: "The primary rear-end collision causal factor is driver inattention to the driving task (approximately 66% to 77%). Driver inattention in this context includes both inattention and distraction. Drivers are constantly scanning the roadway environment (i.e., looking down the road, left side, right side, scanning the mirrors and attending to internal and external stimuli). Drivers can only focus well on one thing at a time because the eyes focus together. So, for example, when drivers are attending to other stimuli they may not be able to adequately perceive the roadway in front of the vehicle. It is often necessary for drivers to take their eyes off the roadway and "attend" to other stimuli when operating in-vehicle controls and possibly carrying on conversations. Often drivers use multiple glances to attend to other stimuli thus taking their eyes off the roadway. These glances distract the driver and lead to inattention to the driving task. The glances may be momentary or of extended duration. Driver distraction accounts for approximately 11% to 24% of driver inattention. Drivers can also exhibit a behavior where they are focused on the roadway ahead but don't perceive the changes that are occurring. This "looked but didn't see" phenomenon is also indicative of driver inattention. This is similar to the previously discussed problems resulting from distractive advertising.

[0049] Unfortunately, ITS, in its recommendations, goes on to assume a very common solution found in various moving objects such as aircraft, trains or other types of machinery: that of installing a prominent visual warning indicator on the instrument panel of the vehicle, possibly augmented by an audio and a haptic (force feedback) warning, or a thumping on the throttle pedal. In other

words, at a time when an inattentive driver is supposed to be re-engaging his thoughts and responding to an intense vehicular situation on the road ahead of him, his attention, based on ITS recommendations, is being diverted by yet more visual, audible and physical stimuli. As noted hereinabove, researchers have determined that new or unexpected stimuli trigger involuntary responses, and the more bored or unfocused we are, the more susceptible we are to these surprises. Also discussed above is the phenomenon of computer users performing search tasks, such that the more their attention is directed towards the periphery of their desired focus, the less they focus on the middle of their screens or the meaningful target. Imagine a warning light goes off on the on the instrument panel accompanied by an audible warning and a thumping on your right foot. Your natural impulse may be to look first at the light then possibly towards the floor until you collect your thoughts and by then, based on studies, the prospect of overreacting is high. All this when your attention should be outward on the car in front of you which represents an imminent collision. We've also learned that new, moving and looming objects command attention and that the onset of motion triggers overriding or urgent attention, possibly tapping the survival instinct, and also that changes in color can capture attention. But interpreting this as a potential solution is overlooked by ITS' engineers. We have a long and effective history of putting brake lights in prominent locations on vehicles that might stop quickly in front of us. It works. Perhaps a more natural and less distractive solution would be to utilize rear-facing video displays on the vehicles in front of us—as taught by the present invention—to provide color and graphic transitions for 'following too close' with more intense warnings or warnings with more impact (supported by the audible cues) upon more imminent potential collisions. Such a warning would audibly alert a driver to a situation which would be supported by directing his visual attention precisely where it should be to avoid an accident. This would be enabled by a similar proximity technology that ITS is proposing, then adding a returning wireless signal to the leading vehicle.

[0050] Upon full implementation, the present invention represents a more

effective solution to the ITS proposed collision avoidance and 'following too close' visual warning since it provides a means to trigger and present attention-grabbing color and light changes prominently on the aft surfaces of moving objects which are in certain modes designed to capture the attention of the operators of following moving objects. It is the intent of the present invention to facilitate numerous such novel safety solutions and these are further explained in the subsequent text.

[0051] 2) The U.S. Emergency Alert System (EAS) was designed by the Federal Communications Commission so that important emergency information could be sent quickly to targeted specific areas. EAS alerts not only broadcast media but also cable television, satellite, pagers and new forms of digital technology such as Direct Broadcast Satellite, High Definition Television, and Video Dial Tone. FCC rules have required broadcasters to monitor at least two independent sources for emergency information, ensuring that it is received and delivered to viewers and listeners in a timely manner. However, technology has moved on. Today, significant new methods are available to improve the capabilities of the present EAS. Satellites can be used to deliver EAS messages within seconds with high available levels of security without the geographical limitations of today's EAS. The Internet is a further impact-worthy technology since it offers redundant or back-up path communication with valuable follow-up capabilities. However, a private study on EAS effectiveness, completed in 2002, identifies three main concerns preventing EAS from becoming a truly effective system. One is that there is no concerted government or industry effort that combines EAS and other alerting techniques with existing and new technologies such as wired and wireless Internet, cell phones, PDAs and pagers to form a combined and seamless warning system. An FCC Report from the same period allocated increased numbers of EAS event and location codes so that local emergency managers and law enforcement officials could plan vastly improved local emergency public information warnings. Those new codes, according to the report, could lead to valuable information displays that are not currently

connected to the EAS, such as changeable highway message signs. Every state, county, part of a county, and offshore (marine) area, has a specific number and hundreds of allocated numbers are unused. These could effectively identify unique areas or zones for hazardous weather conditions, nuclear power plants, military bases, neighborhoods or even groups of individuals such as police, firefighters, FEMA or other personnel, provided that the equipment is in place to facilitate those communications.

[0052] One aspect of the current invention is that any moving object so equipped becomes, in fact, a rolling billboard, well suited to delivering location-sensitive, real time information to other moving objects within the area. Weather information such as flash flood, tornado or hurricane warnings or on the spot highway information such as traffic conditions, detours, road-out, or turn-back situations.

[0053] 3) In a similar effort, Motorola recently signed a contract with The Michigan Department of Transportation (MDOT) for what Motorola calls a "Wireless Super Highway" program. The project represents an initial deployment of a wireless network aimed at supporting the previously mentioned U.S. Government Vehicle Infrastructure Integration (VII) initiative. Motorola and the State of Michigan are exploring ways to establish a roadside network that can reduce accidents and road congestion by wirelessly connecting vehicles to the roadside and to other vehicles. MDOT is helping to fund the deployment of Motorola's system in Southfield, Michigan, as part of the State's ongoing investment in real-world testing of practical Vehicle Infrastructure Integration applications. MDOT is also supporting the research and development efforts of private sector firms as well as Original Equipment Manufacturers. Motorola's vision of wireless future roadways includes: vehicles that could detect potholes, ice-patches or other road hazards, then wirelessly transmit that information to other nearby vehicles so that they can avoid the hazard; and a panic brake alert application that can operate in foggy conditions and in which trailing vehicles could receive an in-vehicle alert recommending they brake immediately. Government and private industry

IPR2023-01353 Page 01186

projects like these could share, utilize and benefit from the present invention because its parallel development and mass-market deployment will be driven by private investment in commercial advertising and in the development of personal media platforms that promise new incomes to consumers or the equivalent of new production vehicles that are less costly to operate.

[0054] INTEGRATION

[0055] We see an advertising industry in relative disarray, desperate to develop new platforms in the aftermath of falling numbers for old media (like TV) while chasing after the new ones (cell phones and iPods) like gadget freaks at the local Best Buy. All the while the solution has been staring them in the face: capture consumers on their drives to and from work each day. This is perhaps the "4th screen": a unique media platform totally dedicated to the display of ads. For a truly dedicated platform, however, integration must occur at a variety of levels:

[0056] 1) Advertising. Advertisers can start by cleaning up their own act by a) electronically developing a new creative alliance with the public and by connecting their creatives and managers to that pipeline, and b) incorporating distribution, measurement, consumer feedback, tracking consumer engagement, content delivery accountability and compensation into a single automated system.

[0057] 2) Manufacturing. As previously explained, the technology exists today to fully implement the present invention. Making it cost effective requires the acceleration and integration of manufacturing processes appropriate to the higher volume production of OLED displays to ensure even lower costs and permit mass-market installations on moving objects across the board, industry wide. Even here, proper integration must be thoughtful and paramount. In terms of traditional components, the present invention has the capacity either to replace or to incorporate into a single revenue-producing part for automated manufacture the following items from standard automotive production: tail lights, turn signals, brake lights, back up lights, license plate light and side marker lamps including their molded/metalized plastic housings, reflectors,

lenses, bulbs (or LEDs), gaskets and connector seals (2 per side), center mount stop lights, vehicle brand and model logos or graphics with attach hardware; panel finishes with decorative trims including primers, surface preparation, finish painting, molding or plating; license plate recess structures and lamp mount assemblies, attach points and frames; and, finally, the independent stamping, painting and distribution of license plates and plate frames together with annual DMV registration sticker production, distribution and consumer attachment, together with all overhead, shipping, assembly and direct labor related thereto. How nice to be able replace all that with a single part that not only launches an industry, but puts money back into a new car owner's pockets in an age of skyrocketing fuel costs.

[0058] 3) Government. As previously discussed, there are numerous programs where development of the present invention can enhance or enable better solutions to current government programs. Of greater importance is government participation on proposed standards, rulemaking, legislation and compliance. It is in the best interests of all concerned that the use of the present invention as a rich, mobile communications resource is exploited beyond mere marketing. It should be developed, from the outset, as a highway safety system in its own right by coordinating the interests of such government agencies as the National Highway Traffic Safety Administration (NHTSA), the Federal Highway Administration (FHWA), the National Telecommunications and Information Administration (NTIA), the Research and Innovative Technology Administration (RITA), the National Oceanic and Atmospheric Administration (NOAA) and National Weather Service (NWS), and the Bureau of Transportation Statistics (BTS), as well as other international agencies and commissions. This ensures commerce and system compatibility across the board and across borders, in terms of highway signage, hazard and weather warnings, vehicle identification and signage, vehicle lighting and safety systems and public service announcement and alert networks such as the EAS, to name but a few. For the same reasons, the U.S. Armed Forces may need to be peripherally involved. The coordination of state and local agencies

to incorporate standards for police, firefighters and similar first responders as part of a versatile and failsafe mobile communications system is just as important as the involvement of the Federal Government. A tall order, but considerably simpler than reaching the same levels of consensus and production readiness as were necessary in areas such as vehicle standards for crashworthiness, fuel economy and emissions.

[0059] 4) Financial. The integration of financial interests is critical to getting a project of this magnitude launched. Commercial advertising is clearly the financial driver so it is reasonable to expect significant investment, directly or indirectly from the world of large corporate advertisers and their agencies along with companies having core stakes in communications, transportation or media. As is later explained, governments are likely to contribute to the development of segments they feel is in the best interest of the public or in areas that are in line with their agency's charter, programs or objectives. A third leg in investment may come from the individual motor companies, who will appreciate that the sale of moving objects which can also generate incomes for their owners (and is therefore tantamount a product having a lower cost of operations) is likely to gain favor with the consumer and that competition could quickly drive acceptance of products incorporating the present invention industry wide. A final leg of support will come from one or more of the major communications or outdoor advertising companies that will emerge as gatekeeper for this new platform. The real money will be in content Distribution and Compensation.

[0060] 5) Compensation. The present invention introduces a novel and powerful means to reward the entire driving population, from the beginning student driver to daily commuters; from carpools to commercial fleet operators; and from the lowest income drivers to the wealthiest, delivering to all an ability to earn income as they sit in traffic, or alternatively, to lower their costs of transportation by allowing the use of their personal assets in a profitable public forum. In terms of market size for the present invention, the applicant envisions an orderly phase-in of video-equipped moving objects, with

ultimate market penetration limited only by the total numbers of moving objects produced worldwide.

[0061] COMPENSATION

[0062] There are certainly instances where the driving population has been enlisted to promote or advertise products and services using their personal vehicles for profit. These have been limited in scope and technology, are costly, relatively inconvenient and generally inconsistent with average consumer lifestyles. The method involves paying drivers to wrap their personal cars in ads touting products such as ice cream, juice bars or Internet services. The driver's job is simple: drive to and from work, pick up the kids, run errands—and be willing to operate as a traveling billboard. Some companies offer drivers the free use of a new ad-wrapped car, while other drivers are paid $300 to $400 a month to allow the wrapping on their own vehicles. Critics call it another example of creeping—and creepy—commercialism. As mentioned previously, taxi owners in some major cities are being allowed to place ads on the outside of their cabs. Leasing ad space in this way provides taxi owners and drivers with additional revenue which helps them offset the costs of adding wheelchair-accessible vehicles and similar mandated requirements. Cab owners and the drivers who lease them split revenues from the advertising, which has been estimated at around $400 per car per month.

[0063] As will be seen in a later description of the present invention, registered owners of appropriately equipped moving objects may be similarly compensated, but through a novel combination of private and public entities which control content and monitor compensation, operate automated accountability systems and integrate multiple media resources, financial partnerships and transportation support industries. Ironically, the huge revenues and personal incomes promised by the present invention are made possible by the very thing that causes millions of drivers each day to waste their precious time and money: traffic congestion.

[0064] CONCLUSION

[0065] Madison Avenue, up to now, has been reactive instead of proactive. It hires other entities to follow up after the fact to determine whether or not their content was actually delivered, and if it was, whether or not it went to the right parties. It has no effective procedures in place for judging ad "performance" (other than post analysis assumptions by creatives and the evaluation of sales) to determine what, if anything, the public thinks about the content they believe was delivered. Madison Avenue is about creative guesswork. It's about "following" media technologies rather than leading them. It would be extraordinarily advantageous for marketers to have a channel through which consumers could actively or even passively inform advertisers about the things they truly want to see and hear, for advertisers to fill those channels with content about products and services consumers really want and need, and to provide this content at a time and place of the consumer's choosing. For the first time in advertising history, the present invention enables a self-monitoring mobile network in which consumers can request specific content; view it, hear it, accept, reject or praise it; provide immediate and direct feedback on that content to the creatives and managers involved; and allow those creatives to modify and resubmit that content for distribution. Marketers are able to receive real time confirmation on that distribution along with priceless data on its engagement, effectiveness and acceptance, and the presenters of that content are financially rewarded on a per ad, per screen basis for the use of their platforms just as though they were partners in the same industry, which they are.

[0066] The applicant is unaware of the existence of any commercially-viable system capable of generating revenue for the general driving population through the display of targeted or hyper-relevant video advertising on moving objects which is governed and controlled by stored database profiles in other moving objects. Nor is the applicant aware of any single, closed-loop ad platform which allows consumers to request specific kinds of content, to distribute that content, to confirm delivery of that content, to measure its effectiveness, to account for and compensate such deliveries, and to create

new content based upon direct consumer feedback and confirmed in-field performance. The present invention offers a unique opportunity to turn around the status quo in the world of advertising. It promises new channels of communication with consumers, new sources of public and private capital for investment plus significant ROI with shared revenue to the driving population. It further promises the alignment of multiple essential industries for the manufacture of unique video displays, the improvement of moving object safety and cost effectiveness, the environmental clean up of transportation corridors and the sharing of just compensation with the owners and operators of a new kind of mobile communications and advertising platform.

SUMMARY OF THE INVENTION

[0067] The present invention describes a system, apparatus and method for obtaining revenue through the display of either conventional, targeted or hyper-relevant advertising on moving objects which receive, store, poll and extract data to present ads and other types of information based on the time of day, their location and their movements relative to other objects in a specific environment, and where data and information stored in moving or mobile objects within that environment can be transmitted between said objects, or to stationary objects where it is then received, polled, extracted and used to enable and control such displays for which the owners of those objects are then compensated. A proposed embodiment describes the development, production, deployment and operation of an integrated audio/video and data communications system capable of presenting hyper-relevant commercial advertising with synchronized audio, real time road, weather and emergency alerts, personal, fleet or government communications, mobile signage as well as vehicle branding, illumination and safety systems on and between moving objects. The business model portion describes a means for business development, technological adoption, funding, rulemaking, certification, media creation, content distribution, consumer utilization and user/owner operations and compensation. The apparatus section defines the placement of lightweight video displays on or integral to the body panels, exterior finishes

and/or glazing of both moving and stationary objects with an electronic means for the wireless transmission of stored, encoded user profiles containing the ad viewing preferences for the occupants of one moving object, to the receiving system in another moving or fixed object. The system enables the display of hyper-relevant ad content with synchronized remote audio, personal messaging and public service alerts on surrounding objects and provides for the real-time logging and later downloading of data to confirm communications and content deliveries between objects, track and measure consumer engagement, verify consumer's direct responses to ad viewings with accountability systems for moving object owner/operator Just Compensation. The invention further enables the widespread, non-invasive, profile-driven, ad hoc collection of privacy-compliant data for research purposes relating to consumer movements, travel patterns, buying habits, interests, needs and preferences.

[0068] The present invention also defines a high-volume production solution in which data and information contained in objects moving within the environment can be transmitted, received, polled, extracted and used to control or modify the audio and video output on viewed moving objects based on the viewer's specific interests, preferences, habits and needs, the needs or conditions of the environment or the requirements of those who are responsible to monitor safe and secure movements within it. It further describes a means for the low cost conversion of production vehicles so they can display video and transmit synchronized audio to other vehicles. Considerations regarding the apparatus involve the placement of video displays on or integral to the body panels of existing production vehicles, exterior finishes and/or glazing with an electronic means for receiving, storing, queuing and displaying ad content, and for the wireless transmission of stored audio or audio codes to a viewing vehicle's internal sound system. The present invention also includes a method for compensating the owners of moving objects for the use of their assets as mobile advertising and communications platforms and further defines a means to generate revenue

from the ad hoc collection of hyper-relevant data through the deployment and use of those platforms.

[0069] By allowing an object to learn a little about your personal history, background and experiences, together with a little information about your short or long term goals, needs or desires and by then granting that object—whether the object is small enough to carry in a pocket or purse or large enough to ride in—permission to passively convey that information to other nearby objects at certain times and places in your life, you permit those other objects to interact with you or with your own object and your own life in a way in which the net experience can enhance, enrich or inform. Focusing for the moment on a single embodiment of the invention—and on a moving object which is large enough for people to ride in—one can envision a scenario in which an individual (or a group of individuals) riding in such a moving object which, as just suggested, possesses some information about the people it is carrying—for example, a little about "who they are" and "what they want"—and in which that object can transmit that information to other nearby objects (whether those objects are mobile or fixed). Once those nearby objects receive and process that transmission, they can respond by presenting very special information of significant value to the individuals riding in the first object. Imagine, in the above scenario, an individual seated in such a land-based moving object who is viewing the road ahead. Let's assume he is also controlling or guiding his moving object so his eyes are where they are supposed to be: on the road and on any moving objects in his path. Assume the object in which he is riding now transmits unique information concerning his personal interests, needs, habits, or experiences, then passively, but without disclosing any confidential information, additionally transmits general data about who he is (age, sex, marital status, educational level, occupation, number of children, area of residence, etc.) combined with a little information about the moving object in which he is riding and its recent movements, to another object or objects nearby. If the information is received by a similarly equipped moving object, that object can now process that information and

draw from an internal library of stored data, data that is of unique interest or of significant value to the individual who is in another object which is following and watching. For example, if data stored in the library of a moving object ahead is converted into a digital signal containing audio information and is transmitted wirelessly to a moving object that is following, then the individual who is inside the following object can be immersed in audible sounds that are of genuine interest and value to the individual(s) inside. If that library data is processed into video information and is then delivered to a video display positioned prominently on rear portions of the moving object ahead, then the individual seated at the controls of the following object can also see images displayed on the object ahead of him which can be further synchronized to the audio information, all of which is derived from the information transmitted from his own moving object, and which is therefore of great personal interest or value. Thus, a set of carefully prepared preferences in the form of personal information transmitted from one moving object to another, or to fixed objects nearby, can—when combined with data relating to the environment and the movement of the objects within it—trigger or enable many forms of audio visual imagery that can address the previously mentioned twelve objectives in commercial advertising. What if such transmitted and displayed audio and visual information is not merely based on a person's preferences and moving objects, but on the conditions of the surrounding environment itself? Once again, in an independently controlled, land-based moving object—one which might be negotiating a highway or following electronic guideways or tracks—if there were a serious slowing or blockage of other moving objects or a hazardous condition ahead, that information could be received by one or more such moving objects and could then be simultaneously and prominently displayed on those other moving objects or on nearby stationary objects, thereby alerting all occupants and observers to any local and potentially serious conditions. Such displays might simultaneously alert the operators of moving objects in foggy conditions to the slowing of multiple vehicles some distance ahead, to the presence of detours, one-way traffic, icy surfaces, flash

IPR2023-01353 Page 01195

flooding or tornados, or to conditions such as landslides, a bridge out or extended construction. One way to do this is by displaying a graphic and pictorial message on the display surfaces of the moving object immediately ahead. In one such example, information triggering such an audio or visual presentation might come from a wireless signal targeting a specific geographical point or route, or from a broader signal that might blanket an entire town or part of a city. Triggering information could also come from previously downloaded instructions or from a preprogrammed internal resource library able to interact with the Internet or with the navigational system of an individual moving object. Thus, at a predetermined time and place during its travels, moving objects could display critical information, alerts or warnings either to or upon other moving or fixed objects in the environment which could then be viewed by the occupants of those moving objects or by individuals in the immediate area (such as pedestrians or foot traffic) or by other means (such as traffic monitoring cameras) capable of viewing such moving objects from remote locations. Although this latter kind of audio visual presentation might at first seem similar to on-board GPS navigation systems, which place prerecorded audio/visual content on displays inside a moving object, the visual content of the present invention is displayed on the outer surfaces of moving objects or on nearby fixed objects—generally at eye level or within clear line of sight of the moving object's operator as in the case of a land-based vehicle—and is therefore generally intended for viewing through the windows of another moving object or from outside of a moving object. Thus, the location of the display of the present invention can be considerably more effective and a great deal safer than displays located along a roadside or on the interior of a moving object, because operators do not need to take their eyes off the other objects moving in their paths to focus on the interior of their conveyance or to focus and track signage, illumination or warnings that are stationary at roadside. This is the reason we put brake lights on the moving vehicles in front of us, and not on a warning light on our vehicle's instrument panel. Moreover, such video presentations are timely and reflective of

IPR2023-01353 Page 01196

immediate conditions since they can be electronically updated, deployed, triggered and displayed instantly and in real time wherever there are appropriately equipped moving objects. Further, when such images or sounds are generated using interactive onboard database libraries they can be modified in real time to incorporate the needs and characteristics of both the moving objects and their operators. Importantly, all such content is monitored for safe and timely presentation because all delivery is ultimately controlled by the speeds and relative motions between moving objects themselves, the specific locations of those objects in the environment and the condition of the environment itself.

[0070] If moving objects are able to exchange data sufficient to enable conventional, targeted or hyper-relevant audio/video commercial advertising and time-sensitive, location-specific communications, then they can also display highly personalized messages and personal advertising, enable personal sponsorships, assist in law enforcement or emergency operations, produce a unique means for the branding of moving objects and offer many exciting opportunities for the lighting, operating and safety systems of moving objects and for their basic registration and identification. For example, a land-based moving object might display the name or logo of its owner's favorite sports team or political candidate, as a kind of "electronic bumper sticker". Another owner might display a promotion or an ad for his own small business and do it at precise times of the day or night and in meaningful venues, such as his home neighborhood or business or university campuses. By adding a means for law enforcement to communicate with moving objects and a method for the owners of mobile objects to communicate with their own property, lost, stolen or speeding moving objects would be able to announce their presence and alert others in the vicinity. Importantly, such displays will enable the manufacturers of moving objects to incorporate and periodically change the name, branding, functional designs or styling of their moving objects simply by modifying the electronic data controlling the content of its default display. Another significant improvement will come with the

introduction of an all-electronic system for the registration and licensing of moving objects. The display's ability to present powerful, well illuminated external graphics and nomenclature on moving objects and to further encrypt and link such data electronically to the moving object itself will result in a system where all moving objects, upon acquisition by an owner, will immediately be programmed to display identifying numbers (i.e., license plates) properly and prominently in accordance with set standards and such content would be updated each year if annual registrations are required. All such electronic procedures would be encrypted and documented to prevent tampering or hacking without otherwise triggering companion regulatory monitoring systems. One can see that such a system, applied again in the area of land-based road vehicles, could eventually eliminate the need for the manufacture, distribution, installation, maintenance, changing and/or replacement of vehicle license plates altogether. If such a system were properly and securely designed it could also prevent the theft and false use of such vehicle identification. Just as significantly, annual DMV registrations could be conducted solely online and paid registrations would be sent electronically via the Internet directly to vehicles themselves, eliminating the cumbersome process of printing, distributing and installing new registration stickers each year. However, by far the greatest benefit to the use of this invention is that it allows the operators of moving objects to keep their eyes, their focus and their attention on the other moving objects in play around them, and not on the ever-increasing array of brightly-animated billboards, roadside signage and other distractions competing for their visual attention. In fact, the present invention offers the distinct possibility of cleaning up the landscape by enabling significant reductions—and quite possibly the eventual elimination altogether of conventional signage and advertising in high traffic environments.

[0071] Using today's off-the-shelf technology, one could easily assemble an operating demonstrator of the described invention. However, it might be more expensive than necessary and would certainly lack the integration, legislation

and marketability vital to inaugurating such an ambitious and widespread program. To succeed, such an effort is likely to require an international consensus in, but not limited to, such areas as proposed rulemaking, legislation, testing, compliance, design, production and performance, in the realm of product development; the coordination of multiple national interests in terms of highway, airway or waterway traffic management, safety and infrastructures; general telecommunications, telematics and technology adoption as well as the integration of national, state and local environmental and weather services, to name but a few. Of equal importance are standards and practices for content, content display, delivery, distribution, engagement tracking, accountability and finance in the areas of advertising, and also coordinated developments in data processing, displays, telematics, body panels and glazing for the mass market production of moving objects on the international manufacturing side. None of this will happen without money and clear incentives to generate appropriate capital and cash flows to make this invention a commercial reality. It is therefore another prime objective of this invention to demonstrate not merely a capable and operable systems technology, but additionally a general business model designed to stimulate capital investment, the application of development funds, appropriate investment in R&D and the capital equipment necessary to produce cost-effective end products, the proper monitoring and effective use of those products or services and finally to foster a means and enable a system to financially reward all who participate, from the advertisers and agencies who will pay to use the invention's ad platform to the manufacturers of moving objects that will introduce what will be the equivalent of more cost-efficient moving objects as the result of incorporating this invention to the governments who will assure better and safer public services through the adoption of this invention to the new entity(ies) that will profit through the distribution of content through this invention to the many transportation support industries which, through partnerships, will enable fuel, insurance and other incentives to stimulate acceptance of this invention and, perhaps most importantly, to the

IPR2023-01353 Page 01199

numerous fleet, business and general mobile population users who will ultimately offer up their private and personal moving objects to be used as public communications and advertising platforms for profit.

[0072] Commercial advertisers will continue to deal with selected ad agencies and will develop creative content in the same ways as they do today, but they will do it under the new industry standards and practices defined under the present invention. Content producers will continue to work with their ad agencies, creating end products appropriate to the selected media, then media buyers will purchase ad slots and schedule content, but it will be done over the Internet. Submitted content will be first reviewed for conformance to standards and practices and technical requirements, and then encoded for time, venue, and relative priority over other scheduled ads or content types. Once content is encoded, the best method for distribution is determined (cable, satellite, wireless), fees are attached, and the scheduled content is uploaded to the network. Distribution methods are discussed in the drawings, but in practice, final method for delivery may depend on the infrastructure available in the various markets.

[0073] Law enforcement and military and government-related public service content is periodically uploaded in a similar fashion, starting with quality and conformance standards for graphic content, followed by similar time, venue, and priority encoding over other types of content. Typically, government or public service content will have a display priority over ads. The triggering of public service alerts is critical and is sometimes accomplished through external systems. In the case of the roadwork alert which saved mom from a delay in her trip, generic audio and visual content was pre-loaded into her hard drive (perhaps at the factory) while actual triggering for the event was accomplished by an on-board GPS input having time and location constraints (a moving object entering a defined area at a certain time of day) or through a localized broadcast radio signal.

[0074] Fleet and business users such as taxicabs, delivery vans, or car rental agencies, or even large commercial carriers and government-owned and

operated fleets such as U.S. Postal Service vehicles or GSA carpools will submit specialized content under their fleet contracts. After similar approvals and scheduling, fleet and government content is uploaded via the Internet and is distributed to fleet parking or maintenance garages and storage areas in the same way as family, consumer, and commercial advertiser content is distributed in business, university, or mall parking garages.

[0075] It is anticipated under the present invention that law enforcement, fire departments, and perhaps even military users would be granted unique access, enabling these users to override any ongoing, in-field content displays in the interests of public safety. For example, if police in a land-based or airborne moving object were able to "take possession" of the video display on the exterior of a land-based moving object which is presumed to be lost or stolen, is involved in a car chase, or is posing some immediate hazard or threat to others or to the environment, such possession could release prerecorded data from the hard drive in that moving object—and in other nearby moving objects—so that visual warnings and other types of displays could appear prominently and simultaneously on a multitude of moving or stationary objects within the area.

[0076] The applicant envisions that, except in rare cases, all users would pay for the right, privilege, and opportunity to distribute content using the present invention and that a portion of these payments, after expenses, will be paid to the individual owners or operators of the various moving or stationary objects that comprise the ad platform and were involved in the delivery of that particular content. The present invention introduces an integral system of electronic accountability to assure this.

[0077] General consumers will pay for the distribution of their own ads and personal messages, even when such displays are on their own moving objects, although such personal use will likely be priced at a discount. Of course, such personal use reduces the opportunity for these users to earn greater revenue by displaying content for big brand advertisers over their own.

[0078] Professional advertisers will pay a premium to use the system,

especially during prime hours and venues. This parallels established norms in the industry which are based upon reaching specific markets at certain times of the day and in specific locations delivering a certain number of impressions to a preferred demographic. With the present invention, fees charged will be based on the going rates for the media type, content, and application, but with payments based only on deliveries that are electronically confirmed.

[0079] Government and public service users may have content distributed under special contracts or circumstances and such contracts might possibly use public funds.

[0080] Financially, the present invention is a little like offering stock in a public company. The company's shareholders own the moving objects—its communications platform—which is used from time to time to distribute the company's products. A shareholder's financial investment comes through his purchase, maintenance, and operation of his own transportation. His return on investment (ROI) comes in the form of a payment for services rendered or as a dividend based on the company's performance. The more a shareholder invests in the company—buying and using more moving objects or by using them more often—the greater his ROI. The larger the number of shareholders, the bigger the company's distribution network and the greater its potential for revenue.

[0081] However, earlier investment will be required from a number of resources. Seed money from a coalition of key entities who are those most likely to benefit from the new platform is likely to occur first. The advertising industry has the driving incentive and is the long term beneficiary; thus, early investment may be from the more entrepreneurial content producers, media buyers, and ad agencies, along with their large corporate clients, and finally the traditional competing major media platforms (television, Internet and outdoor). Next, the Federal and local governments are investment candidates, principally in supporting R&D, legislation, and in the operational areas of highway traffic safety, and public communications. Finally, the manufacturers of moving objects will invest in their own competitive design and development

efforts together with investment from government and transportation support industries such as petroleum, insurance, banking, computing, electronics, and telecommunications, along with State Department of Transportation Agencies and others.

[0082] The kinds of moving objects that will launch the system in a land-based embodiment are likely to fall into roughly four initial categories: 1) New Vehicle Production; 2) Pre Existing Vehicles (or aftermarket installations); 3) Vehicle Conversions (the replacement of existing body panels such as rear deck lids or doors on SUVs, minivans, or station wagons and truck tailgates) with their original body panels converted to accept the appropriate display technologies; and 4) Modular "add-on" packages suitable for use on trucks, trailers, RV's, motor homes, and on very old or unusual types of vehicles.

[0083] 1) New Vehicle Production in a preferred embodiment, a land-based moving object such as an automobile will contain a small electronics package that will interface with other on-board equipment such as GPS navigation, audio entertainment, and perhaps newer telematics systems, if available. This package will include a transmitter, which sends a wireless encoded signal with information on the vehicle type and VIN (or comparable identifying information) along with encoded user profiles or user codes which define the preferences and personal interests for each occupant selected, to moving object receivers or fixed object receivers within range. Signals may be broadcast in omnidirectional patterns, depending on the application, but in cases involving motor vehicle applications in traffic, such signals are most likely to be transmitted in a generally forward-looking pattern having a horizontal and vertical spread of from 15 to 30 degrees or that typically illuminated horizontally by a motor vehicle's headlamps. This spread may ultimately be governed by rulemaking, and could further be automatically adjusted by local road, personal preferences, or traffic conditions. The electronics package will include, on each vehicle, an antenna to receive such signals, a motion sensor to measure the speed of the receiving vehicle, a rear-looking proximity sensor to measure the closing speeds of any following

vehicle(s), and a decoder to process the signals transmitted from such vehicles. The package will contain antennas for receiving and transmitting information wirelessly from network access points (NAPs), an upload/ download transceiver, a central processor, and a hard drive for storing all information received that is downloaded from a network or is logged in the field as a result of intercommunication between various moving objects. When signals are received from similarly equipped nearby vehicles, they are typically in the form of user codes which define the nature of that vehicle's occupants along with their content preferences (as in the case of hyper-relevant advertising) together with information on the vehicle itself relating to a specific VIN and including specifications on the physical characteristics of that particular moving object. In most cases, these requests are used to poll the onboard hard drive in a viewed (or displaying) vehicle and to then queue the necessary content for a video display based on predetermined delivery protocols. Specific instructions governing the delivery of content are encoded with that content. Content can be stored on board a viewed vehicle, a viewing vehicle, or on both. Video content stored on a viewed (display) vehicle is normally queued and displayed from that vehicle's self contained hard drive storage medium. However, video content that is queued from storage in a viewing vehicle must be wirelessly transmitted to the viewed (display) vehicle for presentation on that vehicle's external display means (unless it is intended for an in-vehicle display). By contrast, audio content queued from storage in a viewed (display) vehicle must be transmitted wirelessly to the viewing vehicle for reproduction and playback over its internal sound system. Thus, in practice, and to avoid unnecessary short range transmissions of audio and video content, the system will first choose to store all content in all moving objects so that only user/content codes are transmitted between vehicles. In this way the video content itself is never transmitted to another vehicle, but is queued and displayed solely within the viewed (display) vehicle's system. At the same time, any audio content related to that display of video is queued, synchronized and released for audio reproduction over the internal sound

system of the viewing vehicle. All such presentations are controlled by the motion and proximities of the moving vehicles involved, their relative speeds, their locations and relative positions, the time of day, and a variety of other factors bearing on subsequent signals received from the viewing vehicles, from other vehicles, and from the environment, and those generated by the presenting vehicle itself.

[0084] 2) Pre Existing Vehicles are those that have been manufactured prior to availability of the present invention, or simply those without the invention; however, owners or users should be able to purchase, lease, or otherwise add versions of the New Vehicle Production package to such moving objects. Aftermarket packages do not need to add the video display portion of the technology unless they wish to earn money from the display of content. If they wish to view hyper-relevant content and/or control what they see and hear from moving objects using the full technology, they can buy and install a stripped-down version of the full production system. Such an aftermarket version would have the ability to receive, process, store, encode, and transmit data including content requests, VIN data, and user/occupant profiles or user codes. It would also have the ability to log content requests, information relating to the content presented by other vehicles, and its owner's responses to viewed content. All of this information remains stored on the hard drive of any Pre Existing Vehicle for later downloading to the network.

[0085] 3) Vehicle Conversions will involve installations of a New Vehicle Production package, but on a panel by panel basis in terms of the video display. Such installations are perfect for fleet or limited production applications like taxis or commercial vans, particularly those which operate in high-density traffic areas. The most cost effective way to implement such conversions is to isolate the display portion of the technology to body components which are easily removable or replaceable, and often set into their own easily serviceable shock mitigation mountings such as the rear hatches of SUVs or minivans, truck tailgates, and doors or trunk lids of passenger cars. The electronics package, proximity sensors, antennas,

receivers, and transmitters can be easily integrated into existing body structures. Indeed, such conversions for city transit and taxi fleets are anticipated as a form of beta-test and preproduction ramp-up and as market test deployments of the present invention.

[0086] 4) Modular "add-on" packages which offer full system capability (proximity and motion sensing, data transmission, processing and storage as well as displays) that can be "bolted-on" to large, unusual or older types of vehicles to allow their owners revenue generating potential are easily managed. Such applications would be for trucks and busses, or even large or medium sized fixed roadside signage or billboards under the right circumstances. Imagine a full-sized SEARS or WALMART truck-trailer rig carrying large panel displays with the company logo, which is also able to read the specific preferences and shopping interests of passing or surrounding motorists and then instantly queue and display products of genuine interest to them.

[0087] It is therefore an object of the present invention to provide a system and method for obtaining revenue through the display of advertising content and other types of audio/video information on the exterior surfaces of mobile objects which can receive, store, contain, poll, extract and use information for this purpose which has been transmitted from other fixed or mobile objects within its vicinity.

[0088] It is another object of the present invention to allow data and information which is stored in one mobile object to be transmitted to another mobile or fixed object in the environment where that data is then received, polled, extracted and used to control the display of ad content or other types of information on the surfaces of such other objects.

[0089] It is a further object of the present invention to display and permit the viewing of such displays at precise times of the day and in exacting geo-locations, further governed by the relative speeds, angles, proximities or movements of other moving or stationary objects within the environment.

[0090] It is another object of the present invention to control the visual displays

on moving objects which are in motion so that such displays are not distractive to the process of operating moving objects within the immediate vicinity.

[0091] It is yet another general object of the present invention to provide a means for the owners of such mobile or fixed objects to be compensated for the use of their property as it relates to the above referenced system.

[0092] It is yet another general object of the present invention to permit the widespread use of personal vehicles for revenue producing activity.

[0093] It is yet another general object of the present invention to compensate owners and operators of moving objects for the confirmed use of their assets.

[0094] It is yet another general object of the present invention to convert drivers' wasted time and money in traffic into incomes, thereby increasing the tax base.

[0095] It is yet another general object of the present invention to make new vehicle ownership and/or operation virtually cost free.

[0096] It is yet another general object of the present invention to stimulate the earlier consumer adoption of alternate energy vehicles by incorporating into them the revenue-generating advantages of the present invention.

[0097] It is a further general object of the present invention to allow government agencies, or those responsible for the safe and proper use of such moving objects, to override the display of advertising or lower priority utilization of the invention to facilitate communications which may be of greater importance in terms of public safety and well being.

[0098] It is another object of the present invention to allow advertisers to submit ad content together with delivery information for approval and subsequent distribution to moving objects.

[0099] It is a further object of the present invention to allow the general driving population to select the kinds of advertising content it would like to view and display, and to further enter into contracts defining their rights and responsibilities to use that content, and to receive compensation for such use.

[00100] It is another general object of the present invention to facilitate an online community through which users of the present invention can register

their moving objects, establish memberships and submit user profiles containing their personal preferences and user criteria.

[00101] It is yet another object of the present invention to allow fleet and government users of moving objects to use the present invention to determine the kinds of advertising content or other types of information they would like to view and display, and to enter into contracts defining their fees and compensation for such use.

[00102] It is a another object of the present invention to facilitate a system in which users can identify "who they are" and "what they want" and then be rewarded for providing that information by being able to control what they see and hear with respect to ad content that is presented to them.

[00103] It is another object of the present invention to permit the distribution of advertising or sponsorship content, private or personal content and public service or emergency content directly and simultaneously to multiple moving objects where it can be stored, retrieved and displayed upon demand.

[00104] It is a further object of the present invention to allow content to be encoded for unique deliveries (such as for time and place, under specific conditions and circumstances or to specific demographics) and to enable contracts with clients, customers and entities who wish to pay for such encoding.

[00105] It is a further object of the present invention to allow users to request that specific kinds of content be displayed on their own or on other moving objects on a predetermined or spontaneous basis, and to optionally pay for that service.

[00106] It is still another object of the present invention to permit users to delete, or prevent the display of, certain kinds of content on moving objects, on a predetermined or spontaneous, real-time basis, and to optionally pay for that service.

[00107] It is another object of the present invention to enable the display of advertising and sponsored content on the exterior surfaces of moving objects when such objects are parked or are non-moving and that such content be

related to the immediate surroundings and/or be controlled by the presence of both pedestrians and handheld electronic devices.

[00108] It is a further object of the present invention to provide a means to store information and data relating to any delivered content requests, any "delete" requests or any "prevent" requests on board a moving object together with a means for subsequently uploading that information for the purpose of tracking, measurement, accounting and compensation.

[00109] It is yet another object of the present invention to enable the wireless, automated downloading of encoded user codes or profiles to moving objects.

[00110] It is a further object of the present invention to enable the wireless, automated uploading of encoded public service content (such as public alerts and weather warnings) together with venue-specific or time-specific triggering criteria to the on-board storage devices of moving objects.

[00111] It is another object of the present invention to enable the wireless, automated downloading of encoded commercial ad content, together with encoded delivery information, to the on-board storage devices of moving objects.

[00112] It is another object of the present invention to enable the wireless, manual uploading of encoded personal content by private citizens, together with encoded delivery information, directly to the on-board storage devices of moving objects from home or personal networks.

[00113] It is another object of the present invention to allow users to spontaneously request additional information, to purchase merchandise or otherwise respond to electronic offers and advertisements as they view such content from or while operating moving objects.

[00114] It is a further object of the present invention to provide a means for logging and storing information on board a moving object relating to the delivery of audio or video public service content, including content requests, and to provide for the later uploading of such information for tracking, engagement measurement, accounting, and compensation purposes.

[00115] It is a further object of the present invention to provide a means for

logging, documenting and storing information on board a moving object relating to the delivery of audio or video advertising or sponsorship content, or content requests, and to provide for the later uploading of that information to a network for tracking, engagement measurement, accounting, and compensation purposes.

[00116] It is a further general object of the present invention to provide a means for logging, documenting and storing information on board a moving object relating to the delivery of audio or video personal use content, and for the later uploading of that information to a network for the purpose of tracking, engagement measurement, accounting, and compensation.

[00117] It is a further object of the present invention to enable the display of video content on the exterior surfaces of one's own moving object, such content generated previously and stored, or created spontaneously in real time (such as an instant personal text message), and optionally, to enable and cause the audio portion of that content to be synchronized to said video and wirelessly transmitted to another moving object for its reception and listening.

[00118] It is another object of the present invention to enable a means for the owners of current production motor vehicles to easily and cost-effectively exchange certain of their motor vehicle's body panels for revenue-producing video display body panels which then permit them to be compensated for the use of their property as it relates to the above referenced system.

[00119] It is another object of the present invention to enable the cost-effective fleet conversions of personal and commercial motor vehicles through the exchange of vehicle body panels and the addition of electronics to allow individuals or small business fleet owners to profit from the use of their vehicles as multi-media communications platforms.

[00120] It is a further object of the present invention to enable the simultaneous uploading and downloading of information and content from a network to large numbers of parked or moving objects for the purpose of updating on-board libraries, or for the purposes of tracking, engagement measurement, accounting and compensation.

[00121] It is a another object of the present invention to enable the uploading of information stored in moving objects relating to content, time, place, nature and trigger for any logged delivery, and to then process and to feedback that information to authorized government users, advertising agencies, advertisers or registered fleet, business or personal users for the purposes of tracking, engagement measurement, accounting, compensation or billing.

[00122] It is a further general object of the present invention to display conventional, targeted or hyper-relevant advertising, public service content and other types of information transmitted to moving objects on the internal video displays of such moving objects.

[00123] It is a further general object of the present invention to display conventional, targeted or hyper-relevant advertising content, public service content and other types of information on the internal video displays of moving objects with the presentation of said displays governed and controlled by the speed, location, contents, time of day, proximity and nature of the moving object and of other nearby moving or stationary objects within an environment.

[00124] It is another general object of the present invention to permit the selling, scheduling and display of hyper-relevant advertising content, together with public service content, personal content and other types of information, on video displays which are located on the interior of moving objects.

[00125] It is another object of the present invention to permit the distribution, display and documentation of conventional, targeted or hyper-relevant advertising, public service content, personal content and other types of information, on video displays located on the interior of moving objects, and to financially compensate the owners or operators for such display.

[00126] It is another object of the present invention to upload user codes which have been transmitted on an ad hoc basis between moving objects in the field for the purpose of tracking consumer movements by VIN, demographics, time, place, interests and other factors, for the purpose of research, marketing and planning.

[00127] It is another object of the present invention to enable the sale of user

codes and information relating to the movements of consumers in the field by VIN, demographics, time, place, interests and other factors for the purpose of research, marketing and planning.

[00128] It is another object of the present invention to convert the ad industry's enthusiasm and its need for unique advertising platforms into capital investment in said invention, thereby stimulating companion investment from private industry and government.

[00129] It is a further object of the present invention to use its ability to generate incomes for the general driving population to stimulate further investment from industry and government.

[00130] It is another object of the present invention, through its ability to generate incomes for the driving population, to leverage the competitive interests of manufacturers causing them to incorporate the invention into their high volume production of moving vehicles, said higher volumes reducing the costs for manufacture and making the product more accessible to the driving population.

[00131] It is a further object of the present invention to enable a system wherein a moving object can detect and learn the traffic patterns and signal sequences at highway intersections, convey that information to other moving objects, synchronize that information with said objects and then calculate the time remaining at such locations within which content can be safely displayed.

[00132] It is a further object of the present invention to enable a system for the geo-tracking of moving objects in a manner that will not violate civil liberties, since tracking data cannot be lawfully generated in the absence of a subpoena.

[00133] It is a further general object of the present invention to permit the use of its video display(s) for entertainment purposes or for static presentations on moving objects when enabled by appropriate audio and video devices.

[00134] It is a further object of the present invention to provide a novel system for reducing the spacing between moving objects when such objects are in stopped, or near-stopped modes, thereby conserving significant amounts of

wasted city traffic lanes especially at traffic signals, intersections, on ramps and turning lanes.

[00135] It is a further general object of the present invention to allow law enforcement to remotely override or take command of a moving object's active display to confirm moving object identification, highlight lost or stolen moving objects or to use a moving object as a mobile hazard, lighting or communications platform.

[00136] It is a another object of the present invention to electronically display moving object identification, licensing and registration at all appropriate times and to further allow the driving population, including law enforcement, to electronically capture and record such identification upon demand.

[00137] It is a further object of the present invention to incorporate all of the normally required lighting, safety, branding, licensing and registration systems, to the greatest extent possible, into the digital video display(s) of future moving objects.

[00138] An aspect of the invention is a system, comprising: a first object; a transmitter in the first object; a second object; a receiver in the second object; and a profile comprising data regarding a user associated with the first object; wherein the profile or a subset of the profile is transmitted from the first object to the second object; wherein content is presented to the first object; and wherein at least a portion of the content is based on the profile or subset of the profile transmitted from the first object to the second object.

[00139] In one mode, the profile is stored in the first object. Another embodiment of this aspect further comprises compensation to a user associated with the second object for presenting content. In another mode, the first object is capable of motion. In other modes, the first object is a vehicle, a device in a vehicle, or a handheld device. In other modes, the second object is a vehicle or is fixed or movable signage.

[00140] In another mode, the first object is a vehicle; and the profile further comprises data regarding other users or potential users associated with the first object. In another mode, the first object and the second object exchange

information relating to relative positions, speeds, and motion of the first and second objects; information relating to the character, configuration, and properties of the first and second objects; information relating to surrounding objects; and information relating to surrounding locations; or any combination thereof. In another mode, presentation of the content is influenced by information relating to scheduling patterns of nearby traffic signals. In still another mode, the content further includes content not based on said profile.

[00141] Another embodiment of this aspect further comprises a priority system relating to presentation of the content based on a nature of the content. In one mode of this embodiment, a nature of the content is classified as emergency content; content relating to a surrounding environmental condition; registration and identification content relating to the second object; public safety content; or a combination thereof.

[00142] In another mode, relative motion data from the first and second objects determines whether content is presented and whether presented content is presented as animated video, static video, or a combination of both. In another mode, the content includes audio content, video content, or a combination of both; wherein the audio content is presented by the first object; and wherein the video content is presented by either the first object or the second object.

[00143] In another mode, the first and second objects are vehicles; the content comprises positioning information relating to the first and second objects; and the content indicates to an operator of the first object an optimum stopping position. In another mode, speed and positioning information is exchanged between all objects within transmission range; and the content presented to the first object indicates to an operator of the first object relative braking conditions of any objects in front of the first object.

[00144] In another mode, the second object has a user associated with the second object; and at least a portion of the content presented to the first object is created by the user associated with the second object. In another mode, a third party purchases a right to present content on the second object; and the second object presents content based on preferences of the third party. In a

further mode, the right is based on a fixed or relative geographical area; a time of day; a period of available display time; or any combination thereof.

[00145] In another mode, the profile is transferable to other objects at an option of any user associated with the first object. In another mode, the content is viewable by persons other than those associated with the first object. In another mode, the content is presented in a manner to render the content not viewable by individuals other than those associated with the first object.

[00146] Another embodiment of this aspect further comprises a display integrated with the second object; wherein at least a portion of the content presented to the first object occurs on the display. In another mode, presentation of the content occurs on an exterior portion of the second object.

[00147] Another embodiment of this aspect further comprises a first storage medium associated with the first object; and a second storage medium associated with the second object; wherein data related to presentation of the content is stored in the first and second storage media. Another mode of this embodiment further comprises a central server wirelessly connected to the first and second objects; wherein the data related to presentation of the content is uploaded to the central server. In a further mode of this embodiment, a compilation of uploaded data in the central server is offered for sale. In one mode of this embodiment, the data related to presentation of the content includes time of display; subject matter of said content; duration of display of the content; a subset of the profile; or any combination thereof.

[00148] In one mode, presentation of the content occurs in an interior portion of the first object. Another embodiment of this aspect further comprises compensation to the user associated with the first object for receiving content, viewing content, or any combination thereof.

[00149] In another mode, information related to identities of all adjacent and nearby objects is exchanged between and stored in the first object and any adjacent or nearby objects. In another mode, the first and second objects exchange information related to audio playing in the first object; and the content presented to the first object comprises video content synchronized to

the audio. In another mode, the content presented to the first object is limited by a user associated with the second object.

[00150] Another aspect of the invention is a computer implemented method, comprising: compiling a unique profile of a first user based on analysis of data provided by the first user; associating the profile with a first object; transmitting the profile or a subset of the profile to a second object; and presenting content to the first user; wherein the first object is associated with the first user; wherein the content originates from the second object; and wherein at least a portion of the content is based on the profile or subset of the profile transmitted to the second object.

[00151] An embodiment of this aspect further comprises compensating a user associated with the second object for presenting content to the first user.

[00152] Another embodiment of this aspect further comprises prioritizing display of the content based on purpose of the content; wherein purpose of the content is advertising; informational; traffic related; safety related; news related; region specific; political; mere frivolity; or any combination thereof. One mode of this embodiment further comprises pre-empting presentation of the content based on surrounding environment of the first and second objects; presence of emergency circumstances; public interest; or any combination thereof.

[00153] Other embodiments of this aspect further comprise prioritizing display of content based on preferences of the first user; transferring the profile to other objects at an option of the first user; displaying registration and identification information for the second object on the second object; or displaying content created by a user associated with the second object on the second object.

[00154] Another embodiment of this aspect further comprises displaying indicators of optimal traffic and view positioning on the second object to the first user; wherein the first objects and the second objects are vehicles.

[00155] Other embodiments of this aspect further comprise pre-empting display of the content in the presence of external wireless signals for emergency or traffic direction purposes; presenting audio content to the first user using the

first object; presenting video content to the first user using the first object, the second object, or a combination of the first and second objects; or exchanging information relating to relative positions, speeds, and motion of the first and second objects; information relating to the character, configuration, and properties of the first and second objects; information relating to surrounding objects; and information relating to surrounding locations; or any combination thereof, between the first object and the second object.

[00156] Another embodiment of this aspect further comprises determining relative motions of the first and second objects; and presenting no content or content as animated video, static video, or a combination of animated and static video, based on the relative motions of the first and second objects.

[00157] In other modes, the content is presented on a display integrated with the second object, or the content is presented on an exterior portion of the second object.

[00158] Other embodiments of this aspect further comprise storing data related to presenting the content in the first and second objects; uploading data related to the presenting the content to a central server wirelessly connected to the first and second objects; or offering a compilation of uploaded data from the central server for sale. In another embodiment, the data related to presenting the content includes time of display; subject matter of the content; duration of display of the content; a subset of the profile; or any combination thereof.

[00159] A still further aspect of the invention is a method, comprising: populating a source object with content; and presenting the content to a recipient object in response to a unique profile or portion thereof associated with the recipient object that is provided by the recipient object to the source object.

[00160] Yet another aspect of the invention is a system, comprising: a content storage device; and means for presenting content stored on the content storage device to a recipient object in response to a unique profile or portion thereof associated with the recipient object that is provided by the recipient

object to the source object.

[00161] Further aspects of the invention will be brought out in the following portions of the specification, wherein the detailed description is for the purpose of fully disclosing preferred embodiments of the invention without placing limitations thereon.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING(S)

[00162] The invention will be more fully understood by reference to the following drawings which are for illustrative purposes only:

[00163] FIG. 1 illustrates a general embodiment and overview of the present invention.

[00164] FIG. 2 is a diagram showing the relationship of governments and manufacturers to the present invention.

[00165] FIG. 3 is a diagram showing the relationship of advertisers and manufacturers to the present invention.

[00166] FIG. 4 is a diagram showing the relationship of the driving population and fleet or government users to the present invention.

[00167] FIG. 5 is a block diagram depicting an owner registration process for the present invention.

[00168] FIG. 6 illustrates a first embodiment of a land-based moving object using the invention.

[00169] FIGS. 7-10 illustrate an aftermarket body panel conversion for a sport utility vehicle (SUV).

[00170] FIGS. 11-13 illustrate an aftermarket body panel conversion for a passenger sedan or Taxi.

[00171] FIGS. 14-16 illustrate an aftermarket body panel conversion for a station wagon.

[00172] FIG. 17 is a block diagram depicting the electronic vehicle packages for the present invention.

[00173] FIGS. 18A-18B are a block diagram depicting the Distribution model of the present invention.

[00174] FIG. 19 is a block diagram depicting the Compensation model of the present invention.

[00175] FIGS. 20-23 are illustrations showing a land-based vehicle and three basic types of displays.

[00176] FIGS. 24-27 are illustrations showing four different types of content display.

[00177] FIG. 28 is a diagram showing geo-specific content distribution.

[00178] FIG. 29 is a diagram showing content purchase and encoding by content type.

[00179] FIG. 30 is a diagram showing content viewing based on FIG. 28.

[00180] FIG. 31 is a block diagram showing content protocols and queuing in a viewed vehicle.

[00181] FIG. 32 is an illustration showing geo-specific content distribution in a rural area.

[00182] FIGS. 33-34 are illustrations showing the display interactions between moving objects in a city and in a freeway environment.

[00183] FIG. 35 is a diagram of the Intersection Clock for measuring the time available for display.

DETAILED DESCRIPTION OF THE INVENTION

[00184] Referring more specifically to the drawings, for illustrative purposes, the present invention is embodied in the system and method generally shown in FIG. 1 through FIG. 35. It will be appreciated that the system and apparatus may vary as to configuration and as to details of the parts, and that the method may vary as to the specific steps and sequence, without departing from the basic concepts as disclosed herein.

[00185] FIG. 1 illustrates a general embodiment and overview of the present invention by showing some of the essential elements which must interact to launch such a unique advertising and communications platform: first, the user base (comprised here of Government Agencies 200, Advertisers 300, the Driving Population 400, and Fleet/Government Users 500); the Developers, Suppliers, and Manufacturers of moving objects 600, (in this embodiment,

production automobiles); the Distribution entity (or entities) 800, the Compensation entity 900 (shown separate here for clarity; in practice, this entity would likely be combined with the Distribution entity), and the body of Registered Owners and Operators, represented at 1000. Although a given embodiment might include a large number of such groups with wide variations of interactions between them, the groups listed herein are some of the most basic and essential to a widespread deployment of the present invention. Such relationships will range from development funding and investment to the manufacture, implementation, operation, and maintenance of the system. A fundamental element to early planning is the proposed rulemaking shown at 100, followed by the legislation of design, performance, and compliance standards for the operating aspects of the emerging system. Such effort represents a give-and-take objective and creative exchange between government and industry following which policies will emerge that all the parties can live with. Precedents and parallels exist for such efforts between government and transportation, and are well known with respect to establishing even more stringent standards for automotive performance, such as, for example, The Department of Transportation's Federal Motor Vehicle Safety Standards (FMVSS) concerning vehicle operating standards and crashworthiness; the EPA's Clean Air Act Amendments (CAAA) or National Low Emission Vehicle (NLEV) Standards for vehicle emissions or NHTSA's Corporate Average Fuel Economy (CAFE) Standards for fuel economy. Such standards will be negotiated essentially between the transportation industry suppliers and automakers 600 and appropriate Government Agencies 200, with input by any other interested parties. Content standards 105 defining how conventional, targeted and hyper-relevant advertising is ultimately presented to consumers will most likely be established concurrently with the operating and distribution portions of the system. This interaction will be between the advertising community 300, the manufacturing body 600, the distribution entity 800, and its compensation unit 900, with input from concerned government entities, fleet and government operators 600 and from the driving population.

[00186] It is envisioned that very early funding for R&D and technology development may come from the investment arms 110, 115 of the advertising and media community on the commercial side, and from the government for exploration on the highway/traffic safety and public service side. This early development funding would be combined with funding provided by the major automakers and their suppliers 600, resulting in prototype operating systems, and leading eventually to full production supply 120, with such production to be integrated with regularly scheduled vehicle production. There are also specific, highly targeted operating niche markets ideal for introducing this platform on a controlled basis, such as fleet and taxi operations, which will be explained later. It should be noted that there are also as many different potential scenarios and models for program funding since interest, participation, and investment is expected from many private sector entities, foundations, institutions, or government agencies. Another likely investment scenario is reflected at 128, in the form of potential investment in the Distribution entity by the transportation manufacturers themselves, because their end products are essential components to this new advertising and communications platform.

[00187] Private investment 130 from the user body can assume many forms, such as state and local government funding in support of innovative law enforcement applications (some of which have been previously described); direct equity investment from advertisers 300 or from big media owners; private investment from the general driving population 400 or from fleet and business users 500 (other than their investment in private vehicle ownership), such investments coming perhaps in the form of equity in the Distribution entity 800, perhaps through an IPO. Once Distribution is operational, however, the user body will commence paying Usage Fees 140 to place content on the new platform.

[00188] The output side of the Distribution unit 800 is the delivery of content (as discussed in the preceding section) distributed directly to the vehicles registered in the network through channels 150, 155, and are explained later.

Content 150 represents conventional, targeted and/or hyper-relevant audio and video ad content, public service content, personal content and encoded personal profiles from fleet or government users and the general driving population. Content 155 represents the delivery of personal preferences to Pre Existing Vehicles 655 which use aftermarket systems sold through retailers or installed as upgrades at automobile dealerships. Pre Existing Vehicles 655 are able to wirelessly transmit VIN, model data, content requests and ad control signals 750 to New Production Vehicles 652, 654, and then receive audio and view video content derived from that data on moving object video displays 760. Fully equipped New Production Vehicles 652, 654 will send similar VIN, content and control requests 740, receive audio content internally and view video content on displays 760 which are integral to those vehicles, exterior surfaces or glazing. As previously explained, all such content and control requests are logged on both new and aftermarket platforms for later wireless downloading as shown by 160, 165 via network access points (NAPs) to be discussed in a later section.

[00189] The Compensation unit 900, collects all data downloaded from registered moving objects and compares content requests with content deliveries (discussed in greater depth later), and confirms actual content deliveries by matching these data. Confirmed distribution information is subsequently returned to the Distribution unit 800 (which may be the same as Compensation unit 900) as shown at 170. A critical step in content delivery measurement, accounting and compensation occurs at this stage. The applicant believes that it is important for any such system to be able to identify and prevent what has become known in the Internet advertising industry as "click-fraud." This is a process in which users, entities or software create fraudulent requests for content deliveries (viewings) in an effort to demonstrate greater interest in a specific ad or site than really exists. In the case of the present invention, this could be attempted by individuals to get paid for presenting ads that were not actually viewed. At this stage, an exchange of data 170 between the Compensation unit 900 (which has

downloaded logged vehicle data) and the Distribution unit 800 takes place which involves the matching of specific VINs on moving objects together with the time, place and the physical movements of those objects to determine the likelihood of fraud. This process is explained in greater detail in a following section. Once in-field content deliveries have been authenticated, the Compensation unit 900 can approve payments to the registered vehicles that have had their display of content confirmed via the mechanism at 170. This applies to all registered production vehicles 652, 654, and in some instances to the vehicles which only view content such as vehicles having aftermarket installations 655. The Distribution unit 800 can then feed back in-field performance measurements to the general user body as indicated at 180. The Compensation unit 900 has, among numerous tasks discussed in a later section, two key functions: the payment of financial compensation to Registered Platform Owner/Operators 1000, as shown at 178; and the sale of ad hoc collected data to marketers and researchers, as indicated at 1100.

[00190] FIG. 2 shows potential relationships between Government Agencies 200 and Developers, Suppliers and Manufacturers 600 relative to the funding, regulation and use of the present invention. Although there are many such possible combinations, named are a few of those agencies that may have a bearing on some of the Regulatory issues 220 and public service usage issues 240. The National Highway Traffic Safety Administration (NHTSA) 202, for example, would have a clear role in establishing vehicle standards, including testing and compliance procedures, just as it has had in other Department of Transportation (DOT) programs, some of which have been previously mentioned. The Federal Highway Administration (FHWA) 204 has a vision to create the best transportation system in the world. In partnership with the overall highway and transportation community, its charter is to assure the safest and most efficient highway and intermodal transportation systems possible, in which crashes, delays and congestion are significantly reduced, where roads protect ecosystems, where pedestrians and bicyclists are accommodated and where transportation services are restored immediately