Improving Performance of Cross-Domain Firewalls in Multi-Firewall System

Sonali A. Deo
Dept of Computer Engg,
DYPCOE, Akurdi
Pune, India
deosonalia@gmail.com

Ms. D. B. Gothawal
Dept of Computer Engg,
DYPCOE, Akurdi
Pune, India
Dgohil.1519@gmail.com

Abstract**—**Firewall is used to protect local network from outside untrusted public network or Internet. Every packet coming to and going out from network is inspected at Firewall. Local network policies are converted into rules and stored in firewall. It is used to restrict access of the external network into local network and vice versa. Packets are checked against the rules serially. Therefore increase in the number of rules decreases the firewall performance. The key thing in performance improvement is to reduce number of firewall rules. Optimization helps to reduce number of rules by removing anomalies and redundancies in the rule list. It is observed that only reducing number of rules is not sufficient as the major time is consumed in rule verification. Therefore to reduce time of rule checking fast verification method is used. Prior work focuses on either Intrafirewall optimization or Interfirewall optimization within single administrative domain. In cross-domain firewall optimization key thing is to keep rules secure from others as they contain confidential information which can be exploited by attackers. The proposed system implements cross-domain firewall rule optimization. For optimization multi-firewall environment is considered. Then optimized rule set is converted to Binary Tree Firewall (BTF) so as to reduce packet checking time and improve firewall performance further.

Keywords*- Cross-domain firewalls; Rule optimization; Binary Tree Firewall(BTF); Firewall Decision Diagram(FDD)*

*****

I. Introduction

Firewall plays very important role in securing private network from outside untrusted public network. They are placed at the entrance to the private network and monitor every incoming and outgoing packet. Security policies of the organization are converted into rules and stored in a firewall in the form of ACL (Access Control List). Each rule in a ACL is of the form <predicate> ---> <decision>. Predicate is defined over set of five fields Source IP address, Destination IP address, Source port, Destination port, Protocol and Decision is either Accept or Deny. Every incoming and outgoing packet's faith will be decided at firewall. Packet will be checked against the set of rules in a firewall. If all five field values of rule predicate match with the values in packet header then the rule is considered as a matching rule and corresponding decision of the rule is applied to the packet. It may happen that packet matches with the more than one rules, so generally first match semantics is followed. Each physical interface of a router contains two ACLs, one for filtering incoming packet and another for filtering outgoing packet.

As the firewall rules are checked serially they affect the firewall performance. If the rules are more in number, checking time is more which degrades the firewall performance [1]. Therefore to increase the performance of firewall it is important to reduce number of firewall rules. A study shows that, firewall rules contain anomalies and redundancies [2], removing those results in reduced number of rules. This can be done by "Cross-Domain Firewall Rule Optimization". But this won't help to improve performance much as the main time consuming process is serial checking of the rules. Proposed method adds "Fast Verification" technique to the reduced rule-set obtained in rule optimization so as to improve performance further.

II. Related Work

Existing work focuses on either Intrafirewall optimization [3], [4], [5], [6], [7], [8] or Interfirewall optimization within single administrative domain [9], [10]. Intrafirewall rule optimization is where rules of single firewall are optimized by removing redundancies or by rewriting rules so as to reduce the number of rules. A larger LANs may contain more than one firewall, in such cases optimization can be performed by removal of Interfirewall anomalies. If the firewalls are under single administrative domain optimization is straightforward as firewalls can share their policies with each other as the privacy is not an issue.

In a cross-domain firewall rule optimization [11] rules of both the firewalls should be kept secure from each other. Security of the rules is important for two reasons: 1) Most of the firewalls are misconfigured and contain security holes which may be exploited by an attacker. 2) Policies contain private information of network which can be used by an attacker to host an attack.

Removing anomalies reduces number of rules which results in increase in firewall performance. It is observed that packets are checked against the rules serially which consumes most of the working time of firewall. Solution for this problem is to arrange rules in Binary Tree Firewall (BTF) [12]. In BTF rules are stored in binary tree which helps to reduce packet checking time.

A. *Intrafirewall Rule Optimization*

Most of the firewalls are misconfigured and contain errors [2]. Rules are inserted or removed from firewall rule list as the organization's security policy changes. These changes may lead to rule anomalies. Rule optimization is nothing but removing such anomalies without changing rules logically.

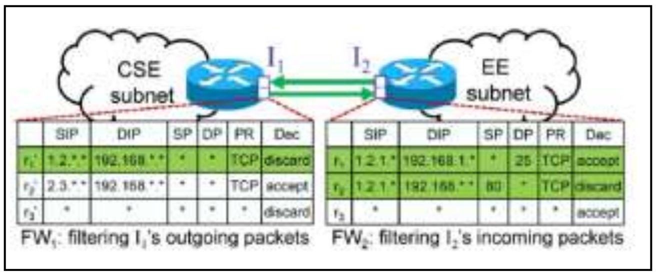
Intrafirewall rule optimization is all about removing anomalies in single firewall. Following are the type of anomalies that can be present in firewall rules [9].

1. Shadowing anomaly
2. Correlation Anomaly
3. Generalization Anomaly
4. Redundancy Anomaly
5. Irrelevance Anomaly

B. *Interfirewall Rule Optimization*

Interfirewall anomalies are the anomalies between two different firewalls of the same network. Intrafirewall rule optimization is not capable of removing Interfirewall anomalies. Interfirewall anomalies can be removed using different Interfirewall rule optimization techniques. If the firewalls are under single administrative domain optimization is straightforward as firewalls can share their policies with each other. In cross-domain firewalls key thing is to keep firewall policies secure from others.

Fig 1 shows example of Interfirewall anomaly. Consider two subnets CSE and EE. They contain firewalls FW1 and FW2 respectively. Let FW1 represents firewall policy on CSE subnet's outgoing interface to EE subnet and FW2 represents firewall policy on EE subnet's incoming interface from CSE subnet.



	SIP	DIP	SP	DP	PR	Dec
r_1'	1.2.*.*	192.168.*.*	*	*	TCP	discard
r_2'	2.3.*.*	192.168.*.*	*	*	TCP	accept
r_3'	*	*	*	*	*	discard

FW_1: filtering I_1's outgoing packets

	SIP	DIP	SP	DP	PR	Dec
r_1	1.2.1.*	192.168.1.*	*	25	TCP	accept
r_2	1.2.1.*	192.168.*.*	80	*	TCP	discard
r_3	*	*	*	*	*	accept

FW_2: filtering I_2's incoming packets

Fig 1: Example Interfirewall Anomaly [11]

Consider a situation where packet p's first matching rule in FW1 says discard the packet and for the same packet p first matching rule r in FW2 says accept/ discard the packet. In such a case rule r from FW2 is said to be redundant as packet will be discarded at FW1 itself and will not reach to FW2 at all. Hence rule can be removed without changing firewall policies logically.

For example consider network shown in Fig 1. FW1 is the list of firewall rules used to filter traffic from CSE subnet to EE subnet. FW2 is the list of rules to filter traffic from EE subnet to CSE subnet. Rules in both the tables are written following the Cisco Access Control Lists format. SIP, DIP, SP, DP, PR, and Dec denote source IP, destination IP, source port, destination port, protocol type, and decision, respectively.

Clearly, all the packets that match r1 and r2 in FW2 are discarded by r1' in FW1. Thus, r1 and r2 of FW2 are redundant rules with respect to r1' in FW1. Therefore removing them will reduce the size of rule list without affecting the network's policy.

Rule optimization removes anomalies in the rule list and reduces number of rules. But this does not improve performance much as more time is consumed in serial verification of the rules. To overcome this problem rules are arranged in BTF.

C. BTF(Binary Tree Firewall)

Binary Tree Firewall rules (BTF) [12] is the binary tree structure used to stored firewall rules. As binary tree sorting and searching, can be applied on rules which consume the time complexity be $L \times O(\log_2 N) \cong O(\log_2 N)$. L means the depth of the BTF tree that is constant number, it is 5 only, and N is the number of members that must be checked. In case of the generic firewall, a time complexity is $O(N^2)$.

Fig 2 illustrates BTF data structure. All the source ip address (SA) ranges are stored at the level 0 i.e. root level. The second level contains destination addresses (DA), the third level contains source ports (SP), the fourth level is of destination ports (DP), and final level is protocol (PRO) respectively.

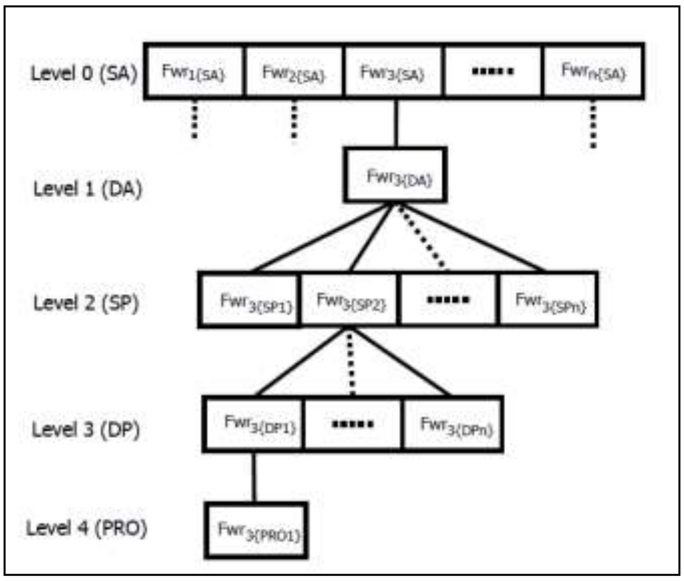


Fig 2: BTF Data Structure [12]

III. PROPOSED SYSTEM

The aim of the proposed system is to enhance firewall throughput by optimizing rules and applying fast verification method for packet checking.

For experiments rules of local firewall are optimized with respect to multiple adjacent cross-domain firewalls. Architecture of the system is shown in Fig 3.

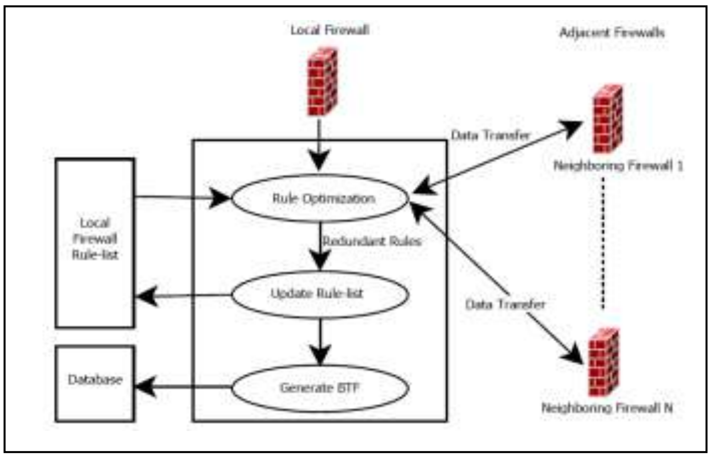


Fig 3: System Architecture

System contains set of firewalls. They belong to single LAN but are under different administrative domains. Therefore they are called as cross-domain firewalls. Local firewall is the firewall whose rules are to be optimized. Adjacent firewalls are the other firewalls in the LAN and here they are called as neighboring firewall. It is considered that local firewall has total N number of neighboring firewalls.

Administrator of local firewall initiates rule optimization process and it sends request for data transfer to all of its neighboring firewalls. During the data transfer privacy of the rules of all firewalls is maintained. Detail procedure of the rule optimization is given next in this section. As an output of rule optimization we get redundant rule-list.

Then existing rule list is updated by removing redundant rules which results in reduced number of rules for local firewall.

After optimization updated rule list will be passed to BTF generation. Here it will be converted into BTF and stored in database. Hereafter packets will be verified against the rules in BTF format rather than traditional time consuming serial fashion. As the time is saved in each packet checking iteration this will result in enhanced firewall throughput.

A. Algorithms

Procedure for the rule optimization is given below

- *Processing at Local Firewall*

1. Local firewall converts its rules to equivalent All-Match FDD. All-Match FDD construction algorithm and All-Match based redundancy removal algorithm is described in [6].

2. Then it extracts all nonoverlapping rules from all-match FDD.

3. Each range is then converted into a set of prefixes. Each prefix is considered only once and duplicate prefixes are discarded, this is done so as to reduce unnecessary traffic and maintain privacy. As other subdomains/ firewalls doesn't know occurrence of prefixes it can't rebuild the rules.

4. Local LAN encrypts these prefixes with its private key and sends to all neighboring firewalls.

5. Neighboring firewall further encrypts these prefixes with their private key for further processing.

- *Processing at Neighboring Firewall*

1. Each neighboring firewall converts its rule-list to equivalent Firewall Decision Diagram (FDD) [13] [14].
2. Then all nonoverlapping rules with discard decision are extracted from FDD.
3. Each range is converted into a set of prefixes. Each prefix is considered only once and duplicate prefixes are discarded, for the same reason stated above in processing at local firewall.
4. The prefixes are then encrypted with its private key and send to local firewall. For encryption only commutative encryption methods can be used.
5. Local firewall double encrypts already encrypted prefixes with its key and sends them back.
6. Neighboring firewall reconstructs nonoverlapping rule with discard decision from double encrypted prefixes.
7. A distinct random index is assigned to each rule. These indices are used to identify redundant rules from local firewall.

- *Comparison of Rules of Local and Neighboring Firewall*

After processing and encryption at local and neighboring firewalls, neighboring firewall has double encrypted nonoverlapping rules from neighboring firewall and double encrypted number prefixes from local firewall. The steps of comparison are as follows.

1. Each number prefix α from local firewall is checked with prefixes in a nonoverlapping rules from neighboring firewall.

2. If the number prefix matches with any of the prefix in the rule, then index of the rule is associated with the number α.

3. A single prefix number may found matching prefixes in more than one rules. Therefore neighboring firewall assigns a set of indices with the number.

4. If the matching prefix is not found in any of the rule then empty set is assigned with the number.

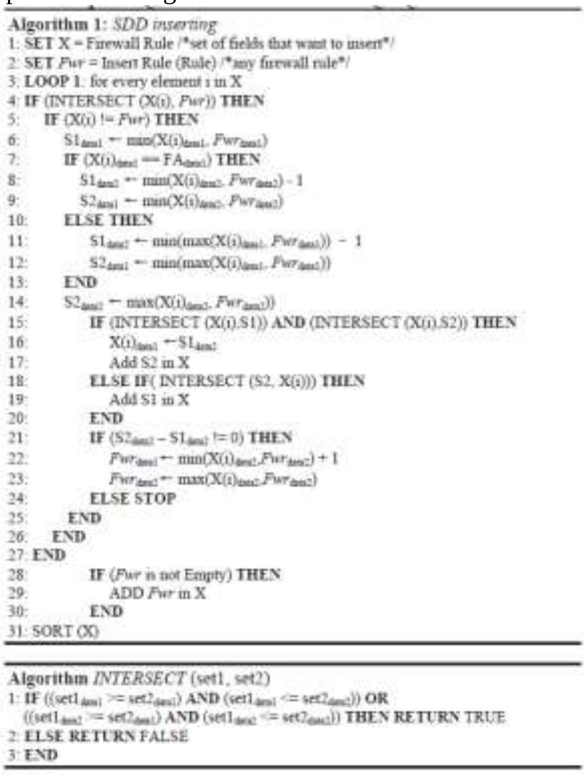
5. After receiving sets from neighboring firewall, local firewall finds index of the rule that overlaps with prefix family.

6. For a nonoverlapping rule in local firewall, if all of it's prefixes have a common index then the rule is redundant.

7. After finding redudant rules with respect to all of the neighboring firewall comman redundant rules are considered as final redundant rules of local firewall.

- *BTF Generation*

Rule optimization module optimization algorithm discussed above optimizes ruleset with respect to other firewalls in the network. This results in reduced number of rules. But only reducing rules is not sufficient for improving firewall performance. As the rules are checked serially to decide faith of the packet. So it is necessary to modify the rule checking strategy to reduce time. One of the solutions is to arrange rules in a tree. So at the time of verification rules are compared by searching the tree. This will definitely take less time than serial checking. Algorithm for BTF generation algorithm is presented in Fig 4.



```
Algorithm 1: SDD inserting
1: SET X = Firewall Rule /*set of fields that want to insert*/
2: SET Fwr = Insert Rule (Rule) /*any firewall rule*/
3: LOOP 1: for every element i in X
4: IF (INTERSECT (X(i), Fwr)) THEN
5:     IF (X(i) != Fwr) THEN
6:         $S1_{dim1} \leftarrow \min(X(i)_{dim1}, Fwr_{dim1})$
7:         IF ($X(i)_{dim1} == FA_{dim1}$) THEN
8:             $S1_{dim2} \leftarrow \min(X(i)_{dim2}, Fwr_{dim2}) - 1$
9:             $S2_{dim1} \leftarrow \min(X(i)_{dim2}, Fwr_{dim2})$
10:        ELSE THEN
11:            $S1_{dim2} \leftarrow \min(\max(X(i)_{dim1}, Fwr_{dim1})) - 1$
12:            $S2_{dim1} \leftarrow \min(\max(X(i)_{dim1}, Fwr_{dim1}))$
13:        END
14:        $S2_{dim2} \leftarrow \max(X(i)_{dim2}, Fwr_{dim2}))$
15:            IF (INTERSECT (X(i),S1)) AND (INTERSECT (X(i),S2)) THEN
16:                $X(i)_{dim2} \leftarrow S1_{dim2}$
17:                Add S2 in X
18:            ELSE IF( INTERSECT (S2, X(i))) THEN
19:                Add S1 in X
20:            END
21:            IF ($S2_{dim2} - S1_{dim2}$ != 0) THEN
22:                $Fwr_{dim1} \leftarrow \min(X(i)_{dim2}, Fwr_{dim2}) + 1$
23:                $Fwr_{dim2} \leftarrow \max(X(i)_{dim2}, Fwr_{dim2})$
24:            ELSE STOP
25:        END
26:    END
27: END
28:            IF (Fwr is not Empty) THEN
29:                ADD Fwr in X
30:            END
31: SORT (X)
```

```
Algorithm INTERSECT (set1, set2)
1: IF (($set1_{dim1}$ >= $set2_{dim1}$) AND ($set1_{dim1}$ <= $set2_{dim2}$)) OR
   (($set1_{dim2}$ >= $set2_{dim1}$) AND ($set1_{dim2}$ <= $set2_{dim2}$)) THEN RETURN TRUE
2: ELSE RETURN FALSE
3: END
```

Fig 4: BTF Generation algorithm [12]

IV. EVALUATION SETUP

The system is implemented using JDK 1.7 on Windows 7. For security reasons it is not possible to obtain real firewall/ firewall rules for processing. Therefore experiments are conducted using three synthetic firewalls. Each firewall examines five fields, source IP, destination IP, source port, destination port, and protocol.

V. RESULTS

The aim is to enhance firewall performance by reducing rule list which ultimately results in reduced packets checking time. System is tested using synthetic firewalls and sample rules were created using method specified in [15]. The following **Error! Reference source not found.** shows actual outputs obtained in system testing. For different number of rules time required for packet checking using different methods is considered. First column shows number of rules, second column shows the time required to check packets using original rule list, third column is about time required to check packets using optimized rule list and last column shows packet matching time after construction of BTF. All times are in milliseconds.

The same results are shown using graph in **Error! Reference source not found.**. Each line in a graph represents different method used for packet matching. Blue line represents time required using original rule list. Green line shown time taken after rule list is optimized. Red line represents the time after BTF construction. It can be observed that in the first two methods packet matching time increases with increase in number of rules. After BTF construction time remains steady though the number of rules is increased.

TABLE I. RESULTS

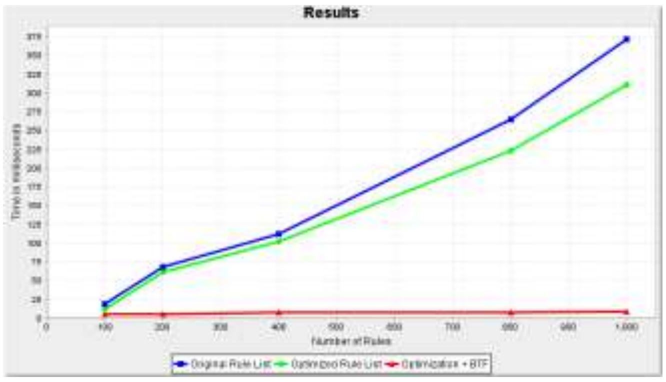


Fig 5: Result Graph

A. Performance Evaluation

Here performance measure used is time. Therefore for evaluation time complexities are considered. Time required with the original rule list is $O(n)$ where n is number of rules in the rule-list. For optimized rule-list required time is $O(n')$ where n' is number of rules after optimization and therefore $n' \leq n$. Complexity of BTF operations is $O(l \log n') + c$ where $O(l \log n')$ is the complexity of binary tree searching/ sorting operations is and c is the constant time required to fetch decision for the packet from database. Therefore evaluation time is constant i.e. c. This shows that packets matching time is constant and is independent of the number of rules in the list.

VI. CONCLUSION AND FUTURE SCOPE

The goal of the system is to improve performance of a firewall by reducing packet checking time in firewall. Packet checking time is directly proportional to the number of rules. Therefore to increase performance basic need is to decrease the number of rules in a firewall. Quantitative studies shows that most firewalls are misconfigured and contains redundancies and anomalies. Removing these anomalies reduces the number of rules. Intrafirewall optimization and Interfirewall optimization within single administrative domain is straightforward as firewalls have direct access to each other's rules. In a cross-domain firewall optimization firewalls can not reveal each other's rules, so while sharing rules privacy should be maintained. After optimization rules are stored in BTF so as to reduce packet matching time further, and improve firewall performance. In this paper it is considered that firewalls are directly connected to each other and there is no any host or NAT device in between. So protocol is only applied for such cases. Procedure of the rule optimization is somewhat complex and reducing complexity of the algorithm should be further studied.

ACKNOWLEDGMENT

The authors would like to thank the publishers, researchers for making their resources available and teachers for their guidance. We also thank the college authority for providing the required infrastructure and support. Finally, we would like to extend a heartfelt gratitude to friends and family members.

Number of Rules	**Time required for packet checking in milliseconds**		
	Original Rule List	***Optimized Rule List***	***Optimization+BTF***
100	19	12	5
200	68	62	5
400	112	102	8
800	265	223	8
1000	371	311	9

REFERENCES

[1] nf-HiPAC, "Firewall throughput test," 2012. [Online]. Available: http://www.hipac.org/performance_tests/results.html.

[2] Wool, "A quantitative study of firewall configuration errors," Computer, vol. 37, pp. 62-67, 2004.

[3] X. Liu, E. Torng and C. Meiners, "Firewall Compressor: An Algorithm for Minimizing Firewall Policies," IEEE INFOCOM, 2008.

[4] Q. Dong, S. Banerjee, J. Wang, D. Agrawal and A. Shukla, "Packet Classifiers In Ternary CAMs Can Be Smaller," in ACM SIGMETRICS, 2006.

[5] R. Meiners, A. X. Liu and E. Torng, "BitWeaving: A Non-Prefix Approach to Compressing Packet Classifiers in TCAMs," in IEEE/ACM TRANSACTIONS ON NETWORKING, 2012.

[6] X. Liu, C. R. Meiners and Y. Zhou, "All-Match Based Complete Redundancy Removal for Packet Classifiers in TCAMs," in IEEE INFOCOM, 2008.

[7] R. Meiners, A. X. Liu and E. Torng, "Topological Transformation Approaches to Optimizing TCAM-Based Packet Classification Systems," in ACM SIGMETRICS, 2009.
[8] R. Meiners, A. E. Liu and E. Torng, "TCAM Razor: A systematic Approach Towards Minimizing Packet Classifiers in TCAMs," IEEE/ACM Trans. Netw, vol. 18, pp. 490-500, Apr.2010.
[9] S. Al-Shaer and H. H. Hamed, "Discovery of Policy Anomalies in Distributed Firewalls," in IEEE INFOCOM, 2004.
[10] M. Yoon, S. Chen and Z. Zhang, "Minimizing the Maximum Firewall Rule Set in a Network with Multiple Firewalls," in IEEE TRANSACTIONS ON COMPUTERS, 2010.
[11] F. Chen, B. Bruhadeshwar and A. X. Liu, "Cross-Domain Privacy-Preserving Cooperative Firewall Optimization," IEEE/ACM TRANSACTIONS ON NETWORKING, vol. 21, pp. 857-868, 2013.
[12] S. Khummanee, A. Khumseela and S. Puangpronpitag, "Towards a New Design of Firewall: Anomaly Elimination and Fast Verifying of Firewall Rules," in 10th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2013.
[13] M. G. Gouda and X.-Y. A. Liu, "Firewall Design:Consistency, Completeness, and Compactness," in IEEE ICDCS, FirewallDesign, 2004.
[14] M. G. Gouda and A. X. Liu, "Structured firewall design," Computer Networking, vol. 51, pp. 1106-1120, 2007.
[15] S. Singh, F. Baboescu, G. Varghese and J. Wang, "Packet classification using multidimensional cutting," in ACM SIGCOMM, 213-224, 2003.