**Firewall Design and Algorithm**

***Firewall Basics***

<predicate> → <decision>

***Steps to crate an own firewall***

Step 1. Make sure your firewall is secure. But how? Below are a few steps

1. Locking VPN access.
2. Automatically tracking security policy.
3. Shut off unused network services.
4. Defending critical resources.

Step 2. Plan your firewall zones and IP addresses accordingly

Step 3. Configure access control lists.

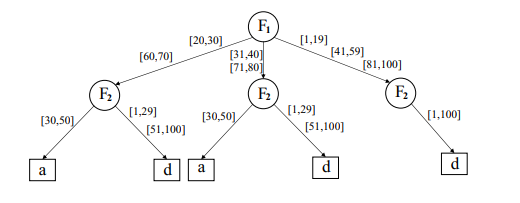
Step 4. Configure your remaining firewall and log your activity.

Step 5. Configure your firewall.

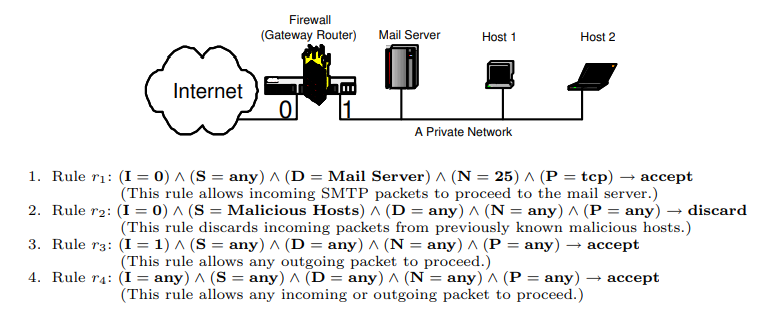
***Firewall Decision Diagram***

F(v) ∈ {F1, ··· , Fd} if v is a nonterminal node

F(v) ∈ DS if v is a terminal node



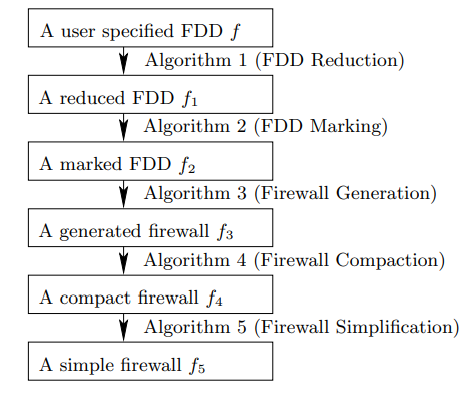
1. Consistency: I(e) ∩ I(e’ ) = ∅ for any two distinct edges e and e’ in E(v).



1. Completeness: e∈ E(v) I(e) = D(F(v)).

(a) (I = 0) ∧ (S = any) ∧ (D = Mail Server) ∧ (N = any) ∧ (P = any) → discard

(b) (I = 0) ∧ (S = any) ∧ (D = any) ∧ (N = 25) ∧ (P = tcp) → discard



***Five steps of our firewall design method (f ≡ f1 ≡ f2 ≡ f3 ≡ f4 ≡ f5)***

***Algorithm 1 (FDD Reduction)***

Input: An FDD f

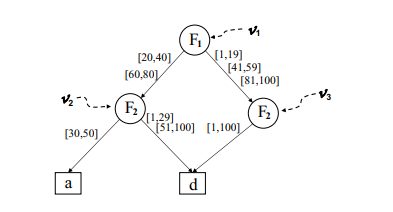
Output: A reduced FDD that is equivalent to f

Steps: Repeatedly apply the following three reductions to f until none of them can be applied any further.

1. If there is a node v that has only one outgoing edge e, assuming e point to node v’ , then remove both node v and edge e, and let all the edges that point to v point to v’ .

2. If there are two nodes v and v’ that are isomorphic, then remove v 0 together with all its outgoing edges, and let all the edges that point to v’ point to v.

3. If there are two edges e and e’ that both are between a pair of two nodes, then remove e 0 and change the label of e from I(e) to I(e) ∪ I(e’ ). (Recall that I(e) denote the label of edge e.)



***Algorithm 2 (FDD Marking)***

Input: An FDD f

Output: A marked version f’ of f such that for every marked version f’’ of f, load (f’) ≤ load(f’’)

Steps:

1. Compute the load of each terminal node v in f as follows: load(v) := 1

2. while there is a node v whose load has not yet been computed, suppose v has k outgoing edges e1, · · · , ek and these edges point to nodes v1, · · · , vk respectively, and the loads of these k nodes have been computed

***do***

(a) Among the k edges e1, · · · , ek, choose an edge ej with the largest value of (load(ej ) − 1) × load(vi), and mark edge ej with “all”.

(b) Compute the load of v as follows: load(v) := Pk i=1(load(ei) × load(vi)).

***End***

***Algorithm 3 (Firewall Generation)***

Input: A marked FDD f

Output: A firewall that is equivalent to f. For each rule r, r.mp and r.rp is computed

Steps:

Depth-first traverse f such that for each nonterminal node v, the outgoing edge marked “all” of v is traversed after all other outgoing edges of v have been traversed. Whenever a terminal node is encountered, assuming hv1e1 · · · vkekvk+1i is the decision path where each ei is the most recently traversed outgoing edge of node vi, output a rule r together with its matching predicate r.mp and its resolving predicate r.rp as follows:

r is the rule F1 ∈ S1 ∧ · · · ∧ Fd ∈ Sd → F(vk+1), where

Si = I(ej ) if the decision path has a node vj that is labeled with field Fi and ej is not marked “all”

Si= D(Fi) otherwise

r.mp is the predicate of rule r.

r.rp is the predicate F1 ∈ T1 ∧ · · · ∧ Fd ∈ Td, where

Ti = I(ej ) if the decision path has a node vj that is labeled with field Fi

Ti= D(Fi) otherwise

***Algorithm 4 (Firewall Compaction)***

Input: A firewall (r1, · · · , rn)

Output: An equivalent but more compact firewall

Steps:

1. for i = n ***to*** 1 ***do***

redundant[i] := ***false.***

1. ***for*** i = n ***to*** 1 ***do***

***if*** there exist a rule rk in the firewall, where i < k ≤ n, such that the following four conditions hold

(1) redundant[k] = false.

(2) ri and rk have the same decision.

(3) ri.rp implies rk.mp.

(4) for every rule rj , where i < j < k, at least one of the following three conditions holds:

(a) redundant[j] = true.

(b) ri and rj have the same decision.

(c) no packet satisfies both ri.rp and rj .mp.

***then*** redundant[i] := ***true***.

***else*** redundant[i] := ***false***.

1. ***for*** i = n ***to*** 1

***do*** if redundant[i] = true then remove ri from the firewall.

***Algorithm 5 (Firewall Simplification)***

Input: A firewall f

Output: A simple firewall f’ where f’ is equivalent to f

Steps:

***while*** f has a rule of the form F1 ∈ S1 ∧ · · · ∧ Fi ∈ Si ∧ · · · ∧ Fd ∈ Sd → <decision> where some Si is represented by [a1, b1] ∪ · · · ∪ [ak, bk] where k ≥ 2.

***do***

replace this rule by the following k non-overlapping rules:

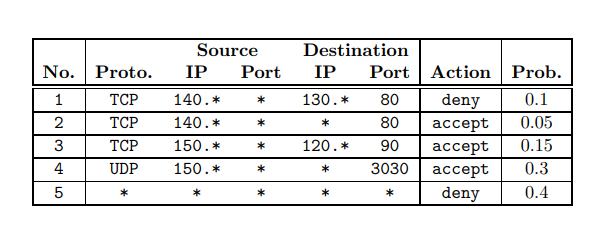
F1 ∈ S1 ∧ · · · ∧ Fi ∈ [a1, b1] ∧ · · · ∧ Fd ∈ Sd → <decision>,

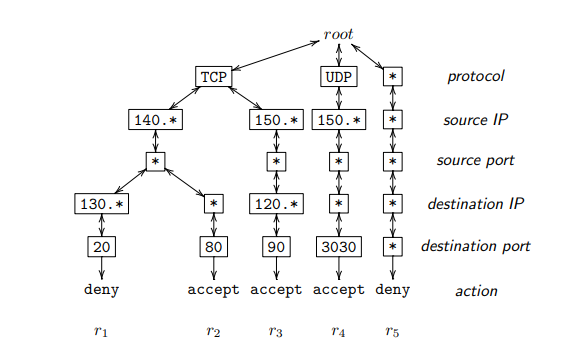
F1 ∈ S1 ∧ · · · ∧ Fi ∈ [a2, b2] ∧ · · · ∧ Fd ∈ Sd → <decision>,

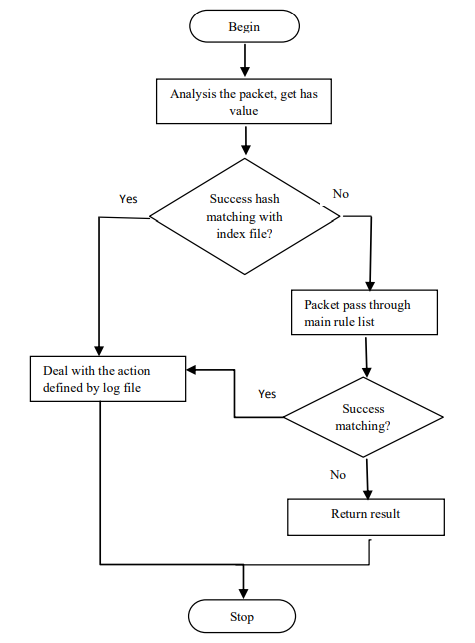
F1 ∈ S1 ∧ · · · ∧ Fi ∈ [ak, bk] ∧ · · · ∧ Fd ∈ Sd → <decision>

***end***

***Hashing Based Packet Matching***







flow chart of proposed matching algorithm

***Firewall Scheduling Algorithm***

**Input** :

(1) array color[1..n] where color[i] is the color of task i;

(2) array cost[1..z] where cost[j] is the cost of executing task j;

(3) array group[1..z] where group[h] is the set of all tasks with color h;

**Output** :

(1) an optimal schedule of the n tasks;

(2) the cost of the optimal schedule;

**Variables**: C, M: array [1..n][1..n] of integer; /\*initial values of C and M are zeros\*/

**Steps**:   
1. FSA-Cost(1, n); /\*compute optimal cost, store trace info in M\*/

2. Print-FSA(1, 1, n); /\*print an optimal schedule using array M\*/

3. print the optimal cost C[1, n];

**End**

**FSA-Cost**(i, j)

if C[i, j]=0 then{

1. min ← cost[color[i]] + FSA-Cost(i + 1, j);

2. M[i, j] ← i;

3. for every element x in group[color[i]] do

if i + 2 ≤ x ≤ j then

if FSA-Cost(i + 1, x − 1) +

FSA-Cost(x, j) < min then{

min ← FSA-Cost(i + 1, x − 1) +

FSA-Cost(x, j);

M[i, j] ← x; }

1. C[i, j] ← min;

}

return C[i, j];

**Print-FSA**(t, i, j)

if i = j then print interval [t, i];

else{

if M[i, j] = i then{

Print-FSA(i + 1, i + 1, j);

print interval [t, i];

}else{

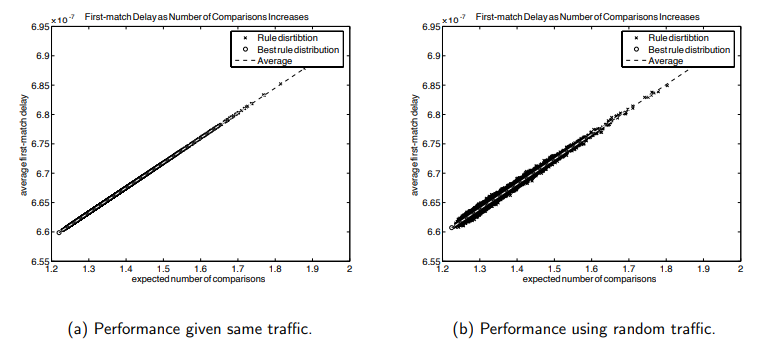
Print-FSA(i + 1, i + 1, M[i, j] − 1);

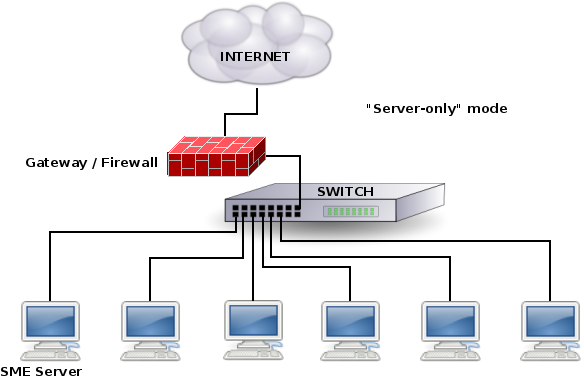
Print-FSA(t, M[i, j], j);

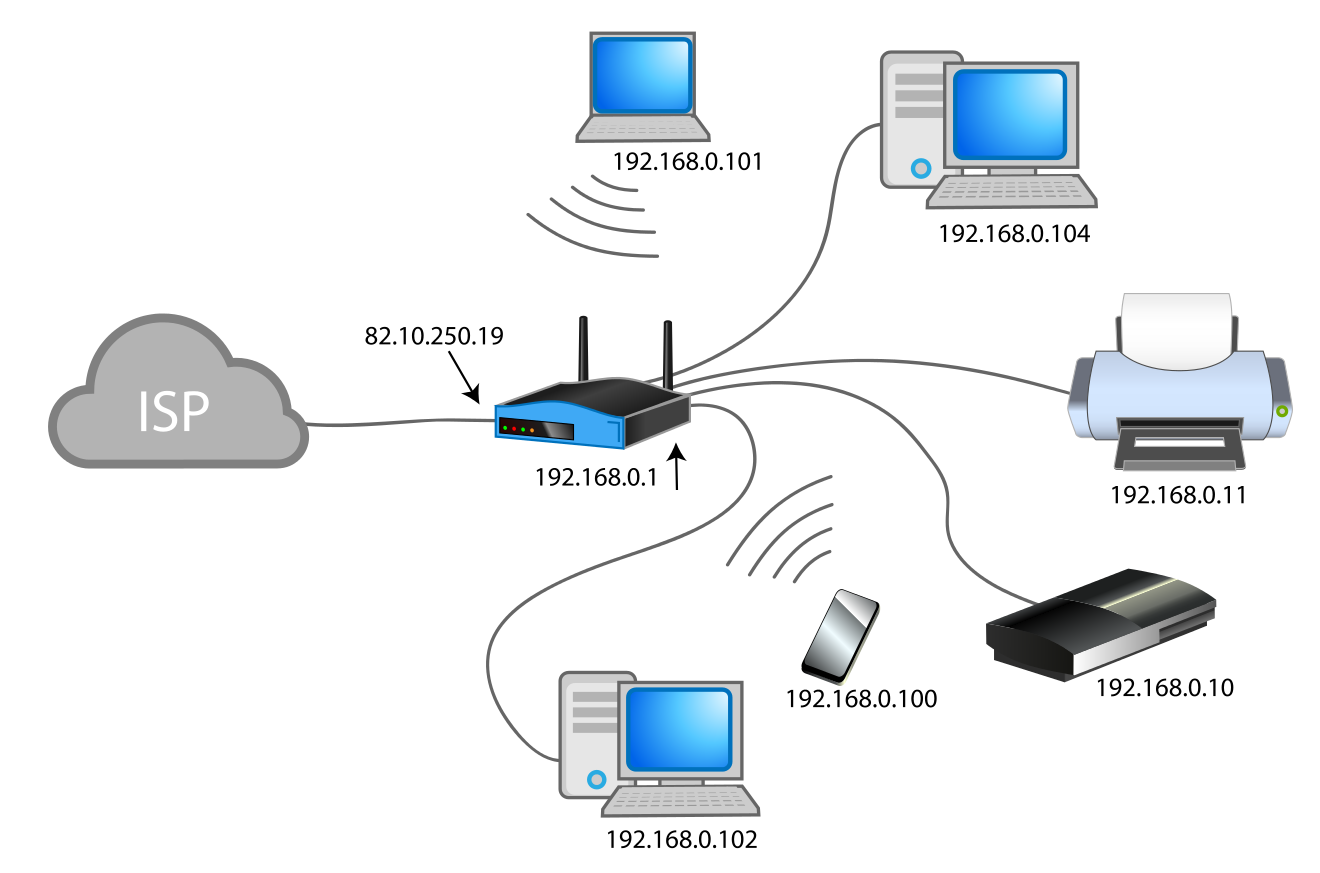
}

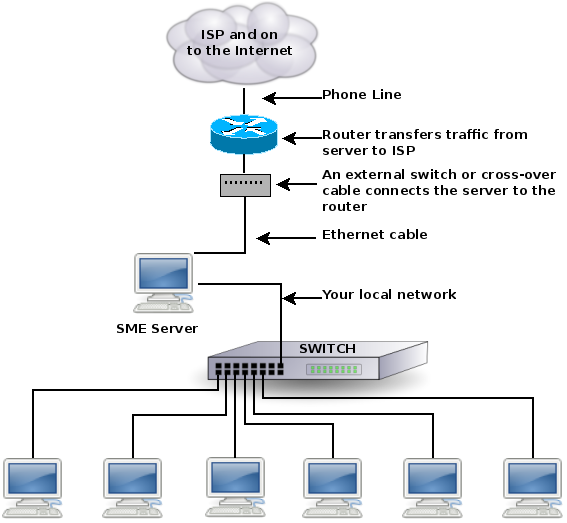
}

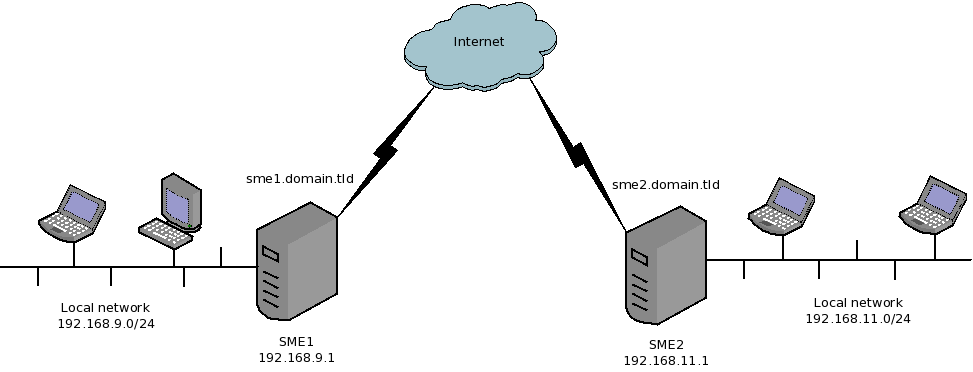
***Distribution Performance***

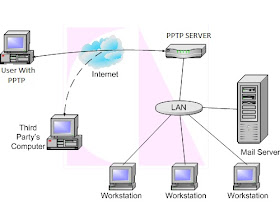
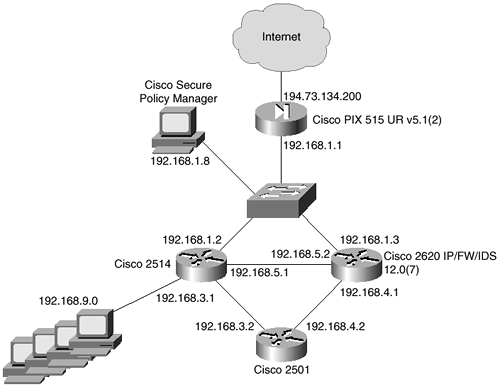
******

***Pictures depicting firewall:***

******

******

******

*** ***

***References***

1. High level firewall language, <http://www.hlfl.org/>.

2. ipchains, <http://www.tldp.org/howto/ipchains-howto.html>.

3. <https://www.osti.gov/servlets/purl/924750>

4. F. Baboescu and G. Varghese. Fast and scalable conflict detection for packet classifiers. In Proceedings of the 10th IEEE International Conference on Network Protocols, 2002.

5. Y. Bartal, A. J. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. In Proceeding of the IEEE Symposium on Security and Privacy, pages 17–31, 1999.

6. Y. Bartal, A. J. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. Technical Report EES2003-1, Dept. of Electrical Engineering Systems, Tel Aviv University, 2003.

7. A. Begel, S. McCanne, and S. L. Graham. BPF+: Exploiting global data-flow optimization in a generalized packet filter architecture. In Proceedings of ACM SIGCOMM ’99, 1999.

8. R. E. Bryant. Graph-based algorithms for boolean function manipulation. IEEE Trans. on Computers, 35(8):677–691, 1986.

9. CERT. Test the firewall system. http://www.cert.org/ security-improvement/practices/p060.html.

10. E. W. Dijkstra. Goto statement considered harmful. Communications of the ACM, 11(3):147–148, March 1968.

11. D. Eppstein and S. Muthukrishnan. Internet packet filter management and rectangle geometry. In Symp. on Discrete Algorithms, pages 827–835, 2001.

12. M. Frantzen, F. Kerschbaum, E. Schultz, and S. Fahmy. A framework for understanding vulnerabilities in firewalls using a dataflow model of firewall internals. Computers and Security, 20(3):263–270, 2001.

13. M. G. Gouda and A. X. Liu. A model of stateful firewalls and its properties. In Proceedings of the IEEE International Conference on Dependable Systems and Networks (DSN-05), pages 320–327, June 2005.

14. P. Gupta and N. McKeown. Algorithms for packet classification. IEEE Network, 15(2):24–32, 2001.

15. J. D. Guttman. Filtering postures: Local enforcement for global policies. In Proceedings of IEEE Symp. on Security and Privacy, pages 120–129, 1997.

16. A. Hari, S. Suri, and G. M. Parulkar. Detecting and resolving packet filter conflicts. In Proceedings of IEEE INFOCOM, pages 1203–1212, 2000.

17. S. Hazelhurst. Algorithms for analyzing firewall and router access lists. Technical Report TR-Wits-CS-1999-5, Department of Computer Science, University of the Witwatersrand, South Africa, July 1999.

18. S. Hazelhurst, A. Attar, and R. Sinnappan. Algorithms for improving the dependability of firewall and filter rule lists. In Proceedings of the Workshop on Dependability of IP Applications, Platforms and Networks, 2000.

19. S. Kamara, S. Fahmy, E. Schultz, F. Kerschbaum, and M. Frantzen. Analysis of vulnerabilities in internet firewalls. Computers and Security, 22(3):214–232, 2003.

20. A. X. Liu, M. G. Gouda, H. H. Ma, and A. H. Ngu. Firewall queries. In Proceedings of the 8th International Conference on Principles of Distributed Systems, LNCS 3544, T. Higashino Ed., Springer-Verlag, pages 124–139, December 2004

21. A. Mayer, A. Wool, and E. Ziskind. Fang: A firewall analysis engine. In Proceedings of IEEE Symp. on Security and Privacy, pages 177–187, 2000.

22. J. D. Moffett and M. S. Sloman. Policy conflict analysis in distributed system management. Journal of Organizational Computing, 4(1):1–22, 1994.

23. G. Patz, M. Condell, R. Krishnan, and L. Sanchez. Multidimensional security policy management for dynamic coalitions. In Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX II), June 2001.

24. J. Quinlan. Induction of decision trees. Machine Learning, 1(1):81–106, 1986.

25. K. Strehl and L. Thiele. Interval diagrams for efficient symbolic verification of process networks. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 19(8):939–956, 2000.

26. A. Wool. Architecting the lumeta firewall analyzer. In Proceedings of the 10th USENIX Security Symposium, pages 85–97, August 2001.

27. A. Wool. A quantitative study of firewall configuration errors. IEEE Computer, 37(6):62–67, 2004.

28. Discovery of policy anomalies in distributed firewalls. In IEEE INFOCOM’04, pages 2605–2616, March 2004.

29. M G. Gouda and Alex X. Liu? Department of Computer Sciences The University of Texas at Austin Austin, Texas 78712-0233, U.S.A