Firewall: Optimizing Policies, Testing And Performance Evaluation

By

Vivek N. Changela
(05MCE003)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY OF SCIENCE & TECHNOLOGY

Ahmedabad 382481

May 2007

Firewall: Optimizing Policies, Testing And Performance Evaluation

Major Project

submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

By

Vivek N. Changela
(05MCE003)

Guide
Prof. Jaladhi Joshi

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY OF SCIENCE & TECHNOLOGY

Ahmedabad 382481

May 2007

This is to certify that Dissertation entitled

FIREWALL: Optimizing Policies, Testing and Performance Evaluation

Submitted by

Vivek N.Changela

has been accepted toward fulfillment of the requirement
for the degree of
Master of Technology in Computer Science & Engineering

Dr. S. N. Pradhan	Prof. D. J. Patel
Professor In Charge	Head of The Department

Prof. A. B. Patel
Director, Institute of Technology

CERTIFICATE

This is to certify that the Major Project entitled “FIREWALL: Optimizing Policies, Testing and Performance Evaluation” submitted by Mr. Vivek Changela (05MCE003), towards the partial fulfillment of the requirements for the degree of Master of Technology in Compute Science & Engineering of Nirma University of Science and Technology, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven’t been submitted to any other university or institution for award of any degree or diploma.

Prof. Jaladhi Joshi
Project Guide,
Department of Computer Engineering,
Institute of Technology,
Nirma University,
Ahmedabad

Date: -

Acknowledgements

Every work in this world is done with the help of some or the other intermediaries. These intermediaries are directly or indirectly involved in the work. I am thankful to those persons who helped me a lot in successfully completing my project. Actually those persons are the keys behind our success. According to a famous saying, "***We should thank the one's that have helped you in your work".*** So here in this page I am thanking all those intermediaries who helped in this dissertation work.

First of all I must remember **GOD** who always gives us confidence & positive attitude to perform great work.

I am thankful to **Prof. A. B. Patel**, Director, Institute of Technology for his kind support in all respect during my study. I am grateful to **Prof D. J. Patel**, HOD, Computer Department and **Dr. S. N. Pradhan,** Coordinator of M. Tech. Program for their kind response and co-operation & providing facilities, which are required for our dissertation.

Words would rather fail to express our deep, perennial and everlasting gratitude, indebtedness and reverence for my guide **Prof. Jaladhi Joshi**, who has always been a source of encouragement inspiration and valuable suggestion to me at every step in completion of this dissertation. Their unflinching patronage and affection shall ever remain a treasure in my cherished memory.

Hearty thanks to my **parents and elder brother** who have always encouraged me to get success, without them I could not have reached at this level.

Finally, my special thanks to **all my friends** who helped me at various junctures while completing this dissertation.

Abstract

Firewalls enforce a security policy between two networks by comparing arriving packets against the policy rules to determine whether they should be accepted or denied. As the amount of data being transferred over networks increases over a time, the firewalls used to protect private networks must process traffic both faster and with greater reliability. In order to cope with new application types like multimedia applications and as high-speed networks become more prevalent, delays will become more significant, new firewall architectures are necessary. The performance of these new architectures is a critical factor because Quality of Service (QoS) demands of such applications have to be satisfied.

This thesis covered basics of firewall, which has definitions, types of firewalls, and current firewall approaches. Also as network become complex, managing firewall rules, especially for enterprise networks, has become complex and error-prone. Firewall filtering rules have to be carefully written and organized in order to correctly implement the security policy. In addition, inserting or modifying a filtering rule requires thorough analysis of the relationship between this rule and other rules in order to determine the proper order of this rule and commit the updates for this we presents firewall policies modeling and defined set of anomaly which describes any rules conflicts.

This thesis covered single firewall and disturbed firewall architecture implemented using of-the-shelf components like iptables and iproute2. Iptables is a generic table structure that defines rules and commands as part of the netfilter framework that facilitates packet filtering, Network Address Translation, and packet mangling in the Linux 2.4 and later operating systems. Packet mangling is process of modifying packets TOS bits and marking packets before it goes to routing process.

Finally, thesis explores the firewall security and performance relationship for single firewall and distributed firewalls. We also discuss the tradeoff between security and performance in terms of delay and throughput vs number of rules in single and distributed firewall.

Table of Contents

Acknowledgements IV
Abstract V
Table of Contents VI
List of Figures VIII
List of Tables VIII
Abbreviations IX

Chapter 1 Introduction 1
1.1 Network Security Issues 1
1.2 Current Scenarios 1
1.3 Objective of Study 2
1.4 Scope of Work 3
1.5 Organization of Project 3
Chapter 2 Literature Survey 5
2.1 What Firewall does? 6
2.2 Basic Functions of a Firewall 6
2.3 What a Firewall can't do 8
2.4 General Strategy: Allow-All or Deny-All 9
Chapter 3 Types of Firewall 11
3.1 Packet Filter Firewalls 11
3.1.1 Pros and Cons of Packet Filtering 13
3.1.2 Packet Filter Rule Sets 14
3.2 Stateful Inspection Firewalls 15
3.2.1 Pros and Cons of Stateful Firewall 17
3.3 Application-Proxy Gateway Firewalls 18
3.3.1 Pros and Cons of Application-proxy Gateway Firewall 20
3.4 Network Address Translation 21
3.4.1 Static Network Address Translation 22
3.4.2 Dynamic Network Address Translation 23
3.4.3 Port Address Translation (PAT) 23
3.5 Virtual Private Network 23
Chapter 4 Current Firewall Approaches 24
4.1 Single Firewall Architectures 24
4.1.1 Software Firewalls 24
4.1.2 Hardware Firewalls 25
4.2 Distributed Firewall Architectures 25
4.2.1 Data Parallel Firewall System 26
4.2.2 Functional Parallel Firewall System 28
Chapter 5 Firewall Policy Modeling 31
5.1 Formalization of Firewall Rule Relations 31
5.2 Firewall Policy Anomaly Classification 32
5.2.1 Shadowing Anomaly 32
5.2.2 Correlation Anomaly 33
5.2.3 Generalization Anomaly 33
5.2.4 Redundancy Anomaly 34
Chapter 6 Firewall Policies Optimization 35
6.1 Rule Organization 35
6.2 Use of the State Module 38
6.3 User-Defined Chains 38

Chapter 7 Netfilter/iptables 41
7.1 Architecture of IPTABLES 42
7.2 Packet traversal in IPTABLES 43
7.3 Advanced Features of netfilter/iptables 45
Chapter 8 Performance Evaluations 49
8.1 Tools Used For LAB SETUP And Performance Evaluation 49
8.1.1 Netperf: A Benchmark for Measuring Network Performance 49
8.1.2 Iproute2 50
8.2 Experimental Lab Setup - Single Firewall Design 51
8.2.1 Firewall Configuration for Single Firewall Design 52
8.3 Experimental Lab Setup - Distributed Firewall Design 52
8.3.1 Packet Distributor Configuration 53
8.3.2 Firewalls Configuration for Distributed Firewall Design 53
8.4 Experimental Results 54
8.4.1 Throughput as Function of Number of Rules 54
8.4.2 Request/Response Performance as Function of Number of Rules 55
8.4.3 Delay as Function of Number of Rules - Single Firewall 56
8.4.4 Delay as Function of Number of Rules - Distributed Firewall 57
Chapter 9 Conclusion and Future work 60
References 61

List of Figures

Figure 3.1	OSI Layers Addressed by Packet Filters	12
Figure 3.2	OSI Layers Addressed by Stateful Inspection	16
Figure 3.3	OSI Layers Addressed by Application-Proxy Gateway Firewalls	19
Figure 4.1	Data Parallel, Packets Distributed across an Array of equal Firewall Nodes	27
Figure 4.2	Function Parallel Firewall Design, Rules Distributed across Array of Firewall Nodes	29
Figure 5.1	State diagram for Detecting anomalies for rules Rx and Ry, Ry comes after Rx.	34
Figure 6.1	Standard Chain Traversal	39
Figure 6.2	User-Defined Chains Based on Protocol	39
Figure 7.1	Packets Traversal in Iptables	44
Figure 8.1	LAB Setup for Single Firewall	51
Figure 8.2	Distributed Firewall Design	53
Figure 8.3	Throughput as Function of Number of Rules	54
Figure 8.4	Request/Response Performances as Function of Number of Rules	56
Figure 8.5	Delay as Function of Number of Rules - Single Firewall	57
Figure 8.6	Delay as Function of Number of Rules - Distributed Firewall	58

List of Tables

Table 3.1	Packet Filter Firewall Ruleset	14
Table 3.2	Stateful Firewall Connection State Table	17
Table 3.3	Static Network Address Translation Table	22
Table 5.1	A Firewall Policy example	31
Table 6.1	A Policy Profile	36
Table 8.1	Throughput as Function of Number of Rules	54
Table 8.2	Request/Response Performance as Function of Number of Rules	55
Table 8.3	Delay as Function of Number of Rules - Single Firewall	56
Table 8.4	Delay as Function of Number of Rules - Distributed Firewall	58

Abbreviations

ACL	Access Control List
ASIC	Application Specific Integrated Circuit
CERT	Computer Emergency Response Team
DAG	Directed Acyclical Graph
DoS	Denial of Service
FPGA	Field Programmable Gate Array
Gbps	Gigabits per second
ICMP	Internet Control Message Protocol
IP	Internet Protocol
MAC	Medium Access Control
Mbps	Megabits per second
NAT	Network Address Translation
NIC	Network Interface Card
QoS	Quality of Service.
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

1. INTRODUCTION

With the global Internet connection, network security has gained significant attention in the research and industrial communities. Due to the increasing threat of network attacks, firewalls have become important integrated elements not only in enterprise networks but also in small-size and home networks. Firewalls have been the frontier defense for secure networks against attacks and unauthorized traffic by filtering out unwanted network traffic coming into or going from the secured network. The filtering decision is taken according to a set of ordered filtering rules defined based on predefined security policy requirements [4]. Simply Firewalls are a well-established security mechanism to restrict the traffic exchanged between networks to a certain subset of users and applications.

1.1 NETWORK SECURITY ISSUES

The Internet has become a broad medium of virtually unlimited purposes. However, while some are legitimate or benign, there are a vast number of malicious, or even illegal uses. Many of these malevolent actions are targeted at other users of the Internet, from corporations to individuals, and may not be driven by any particular motivation other than pure malice. It is easy to imagine that as the Internet grows, so will the numbers of abusive users and thus network security issues. In fact, CERT states that since 2000 reported incidents have increased from 21,756 to 137,529 [5]. In order to properly secure an internal network from traffic that could lead to an incident, network infrastructures need to provide control to internal resources. Technology to manage access control has found a place in daily operations with many organizations, including those with strong emphasis on confidential data such as military, government, and corporations with large data stores [3].

1.2 CURRENT SCENARIOS

Nowadays, large and small companies are seeking ways of doing business on the Internet for global business. Meanwhile, Internet security issues become a hot topic. Companies accessing the Internet are seeking methods of protecting their

network sites against external attacks and intrusion. Firewall is one of the best solutions. Companies looking for lower cost solution with greater network performance. Setting up a firewall for private network sites in organizations and at home is no longer a too fancy thing. On the other aspect, performance impact may cause major concerns: As significant performance loss will observed while incorporating a secure environment using a firewall for the Internet connection, To what level of security should we expect without sacrificing the network performance? These are the basic questions asked when addressing the design of a secure network. Nowadays Internet connection capacities are become in megabits/sec to gigabits/sec in such high-speed Internet delay become more significant as processing such high-speed data takes time. So we required new firewall solutions, which provide better network performance under high security.

1.3 OBJECTIVE OF STUDY

As mentioned in Section 1.2, Firewall is hot topic in network security and many research are going on in this area of firewall security. Firewall will remain the forefront defense for any computer network. As firewall is the only entry point of any company, so it is easy to block each and every intruder at this place only. Firewall inspects each and every packet sent between networks; also provide access control, auditing, and traffic control facilities.

Similar to router, firewall is single dedicated machine, which inspect packet but generate more delay then routing, as high-speed networks become more prevalent, delay become more significant. This is place where we can protect network from DoS attacks. Also we can mange QoS related policies with help of firewall.

As high-speed networks become more prevalent, traffic increase and required different architecture for handling it. Network becomes more and more complex, therefore it required large number of rules for controlling traffic, and so it required tools for administrators to add/delete/modify existing rules. As adding/deleting/modifying exiting rule set may create negative effects on security. Therefore tools required which handle and shows effects of modifying rules list after and before it.

Because firewall mostly placed at entry point of any network, load balancing, network address translation and virtual private network can be integrated with firewalls.

1.4 SCOPE OF WORK

Firewall is an inevitable security element, which is widely used in most of the organizations that connect to the global network or other branch offices using VPN through Internet. Although, firewall helps in threat management, it adds additional delays in network traffic especially when number of rules increases. So this study mainly focuses on the performance issues. The tools used in the study are open source (e.g. iptables, Linux OS) that increase the probability of industry implementation of the solution derived as conclusions. The results of the study can be evaluated, implemented and further enhanced to optimize the firewall functionality. The basic work covered under the study includes firewall basics, optimization of rules, firewall policy modeling, and firewall designing (single and distributed) and their performance evaluation based on variations of parameters (delay and bandwidth) in proportion with number of rules.

1.5 ORGANIZATION OF PROJECT

Project is organized in main nine chapters chapter 1 contains introduction part followed by objective of project study and scope of work. Chapter 2 contains literature survey, which has basic of firewall that includes definition, functions, etc. Chapter 3 contains types of firewall, in that main four classification of firewall are explained in detail with their pros and cons. Chapter 4 explains current firewall approaches that includes single firewall architecture and distributed firewall architecture. Chapter 5 shows concept of firewall policy modeling that is mainly divided in formalization of firewall rules relations and firewall policies anomaly which describes four type of relation between any two rules that may create problem at time of modifying rule list. Chapter 6 contains firewall policies optimization that is mainly focused on rules organization, state module, and user defined chains for separating rules in different classes. Chapter 7 gives detail of iptables/netfilter firewall, which includes architecture of iptables, packet traversal in iptables chains and finally chapter end with advanced features

of iptables firewall. Chapter 8 mainly deals with experimental lab setup and performance evaluation of firewalls. Performance of firewall is measured in terms of delay and bandwidth and how it is diverge according to number of rules in firewall. For measuring delay performance, ping is used with different size of ICPM packets. And for measuring throughput (Bandwidth) and request response time Netperf tool is used which is freeware. Chapter 8 has conclusion and future work.

2. LITERATURE SURVEY

The term firewall is used to describe an implementation, which manages access controls for network traffic. Also we can say, “A firewall is a piece of software or hardware that filters all network traffic between your computer, home network, or company network and the Internet” [17]. This includes not just packet filtering, but also compatibility with network services such as Quality of Service (QoS) enforcement and auditing. Quality of Service is the idea that transmission rates, loss rates, and other characteristics of network traffic can be measured, improved, and, to some extent, guaranteed in advance. Filtering is accomplished according to an ordered set of rules, known as either a security policy, or Access Control List (ACL), that defines the action to perform on matching packets That is, rules indicate the action to take place for each packet, such as accept, deny, forward, or redirect. Security can be further enhanced with connection state information, allowing dynamic decisions to be made from the knowledge of previous decisions. 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).

A network firewall has to protect against many different kinds of threats. You read about these threats in the papers almost every day: viruses, worms, denial-of-service (DoS) attacks, hacking, and break-ins. Attacks with names like SQL Slammer, Code Red, and NIMDA have even appeared on the every news [17].

Conventionally, incoming and outgoing traffic of a system is managed by a router presiding at the gateway to the external network or internal subnets. Routing is accomplished with basic table information that pairs requested destinations with known routes. Since this point is the possible intersection of different security policies, it is also a prevalent location for firewalls. Thus firewalls are often utilized in place of, or in conjunction with, a router. Traditionally, a firewall is implemented as a single dedicated machine running a service, as either software or hardware that applies the policy to each arriving packet.

As you are designing your protection against attacks from the Internet, never rely on a single form of protection for your network. Doing so can give you a false sense of security. For example, even if you completely disconnect your network from the Internet to prevent a computer virus from entering your network, an employee can still bring to work a floppy disk that has been infected with a virus and inadvertently infect computers in your network.

2.1 WHAT FIREWALL DOES?

So what exactly does a firewall do? As network traffic passes through the firewall, the firewall decides which traffic to forward and which traffic not to forward, based on rules that you have defined. All firewalls screen traffic that comes into your network, but a good firewall should also screen outgoing traffic.

Normally a firewall is installed where your internal network connects to the Internet. Although larger organizations may also place firewalls between different parts of their own network that require different levels of security, most firewalls screen traffic passing between an internal network and the Internet. This internal network may be a single computer or it may contain thousands of computers.
The following list includes the most common features of firewalls

- Block incoming network traffic based on source or destination
- Block outgoing network traffic based on source or destination
- Block network traffic based on content
- Make internal resources available
- Allow connections to internal network (VPN) [17]

2.2 BASIC FUNCTIONS OF A FIREWALL

If you ask several people what constitutes a firewall, you are bound to receive several different answers. Different firewall vendors use the term with different definitions. In its simplest form, a firewall is any device or software product sitting between your network and the Internet that blocks some network traffic. However, most people agree that a true firewall should have at least the following four basic functions:

Packet filtering: The headers of all network packets going through the firewall are inspected. The firewall makes an explicit decision to allow or block each packet.

Network Address Translation (NAT): The outside world sees only one or more outside IP addresses of the firewall. The internal network can use any address in the private IP address range. Source and destination addresses in network packets are automatically changed (or "translated") back and forth by the firewall.

Application proxy: The firewall is capable of inspecting more than just the header of the network packets. This capability requires the firewall to understand the specific application protocol.

Monitoring and logging: Even with a solid set of rules, logging what happens at the firewall is important. Doing so can help you to analyze a possible security breach later and gives feedback on the performance and actual filtering done by the firewall.

Because firewalls are a single point of entry for network traffic entering or leaving your internal network, the firewall is an excellent location to perform additional security tasks. Many firewalls support the following advanced functions:

Data caching: Because the same data or the contents of the same Web site may pass the firewall repeatedly in response to requests from different users, the firewall can cache that data and answer more quickly without getting the data anew from the actual Web site every time.

Content filtering: Firewall rules may be used to restrict access to certain inappropriate Web sites based on URLs, keywords, or content type (video streams, for example, or executable e-mail attachments).

Intrusion detection: Certain patterns of network traffic may indicate an intrusion attempt in progress. Instead of just blocking the suspicious network packets, the firewall may take active steps to further limit the attempt, for example, by disallowing the sender IP address altogether or alerting an administrator.

Load balancing: From a security standpoint, a single point of entry is good. But from an availability standpoint, this single point of entry may lead to a single point of failure as well. Most firewalls allow the incoming and outgoing network request to be distributed among two or more cooperating firewalls [17].

2.3 WHAT A FIREWALL CAN'T DO

A security guard can't prevent all security problems, and neither can firewalls prevent all security problems.

In a way, trying to protect a building is not much different from trying to protect a computer network. Both the building and the network contain employees who are trying to get their work done without interference or hindrance from security measures.

Security threats that a firewall can't protect you from are

Inside attack: Users on the internal network have already passed the firewall. The firewall can do nothing to stop internal network snooping or intrusion attempts from within. Other security measures, such as configuring restricted permissions on workstations and servers, and enabling the auditing of network access, should be implemented to protect against these kinds of attacks. (Although you can deploy firewalls between your corporate servers and your internal users as well.)

Social engineering: This is the term used to describe attacks in which hackers obtain information by calling employees and pretending to be a colleague at the front desk, a member of the security staff, or just somebody from the firm doing routine checks. This person asks for privileged information, such as server names, IP addresses, or passwords. Employees should be aware of these tactics and know that certain information should never be given.

Viruses and Trojan horse programs: Firewalls attempt to scan for viruses in all network traffic, but these wicked programs change constantly. Distinguishing

between acceptable e-mail attachments and malicious content continues to be a problem for computer users. Good precautions should be taken to prevent the spread of viruses and to minimize the damage that a virus can do. Trojan horse programs are perhaps even harder to spot, because they don't attempt to spread to other files or computers like their virus sisters. A very small Trojan horse program that is run once by an unsuspecting user can open up a back door to his computer. A good example of the kind of damage that these programs can do is a Trojan horse program that sends out all collected keystrokes at password prompts once a week.

Poorly trained firewall administrators: The firewall doesn't know what is acceptable and what is not unless an administrator tells it. Competent firewall administrators should correctly specify which network traffic should be blocked. Doorman Sam has the intelligence to understand that a naked man who claims that his clothes and shoes already arrived and he is supposed to join them in the third floor conference room is clearly crazy, even though Sam's security instructions may not have a naked-man-meeting-his-clothes-upstairs clause. Most firewalls, however, can easily be confused by fragmented IP packets and should be explicitly configured to handle such fragments.

2.4 GENERAL STRATEGY: ALLOW-ALL OR DENY-ALL

One of the first things that you must decide when you configure your firewall is the general strategy on how to specify what network packets and protocols you allow inside your network, and which network traffic that you want to block.

The two major possibilities are

Allow-all strategy: Allows all network packets except those that are explicitly denied.

Deny-all strategy: Denies all network packets except those that are explicitly allowed.

At first sight, the Allow-all strategy appears to be the easiest — requiring only that you create an exception list of network protocols or Web site content that is explicitly forbidden. This strategy is also in line with how other components work

on your network, such as non-firewall routers, network cards, and basically all computers that allow all traffic to pass except when explicitly denied.

The Allow-all strategy may sound enticing, but you should always use the second strategy — Deny-all, which is much more secured.

If you use the Allow-all strategy, you have to list every possible method that someone can use to intrude on your network and then come up with the rules to block related network traffic. Doing so results in a lot of rules, and even then you are bound to miss one, two, or several methods that can be used to exploit your network. Clearly, this is not a safe approach.

The Deny-all approach is much easier to administer. No traffic is allowed, except for a small number of explicitly defined protocols and services. The Deny-all approach has two advantages:

- You have to maintain only a small list of allowed network traffic rules. The smaller the list, the easier it is for you to verify that the configuration of the firewall is correct.
- You don’t have to constantly add new rules to exclude newly discovered problems.

3. TYPES OF FIREWALL

A true firewall is the hardware and software that intercepts the data between the Internet and your computer. All data traffic must pass through it, and the firewall allows only authorized data to pass into the corporate network. Here we will briefly cover six popular firewalls and combinations thereof. The pros and cons of these various firewall technologies with examples, [18]

- Packet filtering
- Stateful-inspection
- Application-Proxy gateway firewall
- Network address translation
- Virtual private network [1, 2, 4, 18]

3.1 PACKET FILTER FIREWALLS

The most basic, fundamental type of firewall is called a packet filter. Packet filter firewalls are essentially routing devices that include access control functionality for system addresses and communication sessions. The access control functionality of a packet filter firewall is governed by a set of directives collectively referred to as a ruleset.

Packet filtering is the simplest firewall to implement. Routers can be configured to filter packets based on packet content comparison against predefined specification criteria. IP addresses, subnets, TCP or UDP port numbers or a combination of these properties can be used as criteria for access or denial [18].

In their most basic form, packet filters operate at Layer 3 (Network) of the OSI model. This basic functionality is designed to provide network access control based upon several pieces of information contained in a network packet:

- The source address of the packet, i.e., the Layer 3 address of the computer system or device the network packet originated from (an IP address such as 192.168.1.1).

- The destination address of the packet, i.e., the Layer 3 address of the computer system or device the network packet is trying to reach (e.g., 192.168.1.2).
- The type of traffic, that is, the specific network protocol being used to communicate between the source and destination systems or devices (often Ethernet at Layer 2 and IP at Layer 3).
- Possibly some characteristics of the Layer 4 communications sessions, such as the source and destination ports of the sessions (e.g., TCP:80 for the destination port be-longing to a web server, TCP:1320 for the source port belonging to a personal computer accessing the server).
- Sometimes, information pertaining to which interface of the router the packet came from and which interface of the router the packet is destined for; this is useful for routers with 3 or more network interfaces.

Packet filter firewalls are commonly deployed within TCP/IP network infrastructures; however, they can also be deployed in any network infrastructure that relies on Layer 3 addressing, including IPX (Novell NetWare) networks. In the context of modern network infrastructures, firewalling at Layer 2 is used in load balancing and/or high-availability applications in which two or more firewalls are employed to increase throughput or for fail-safe operations.

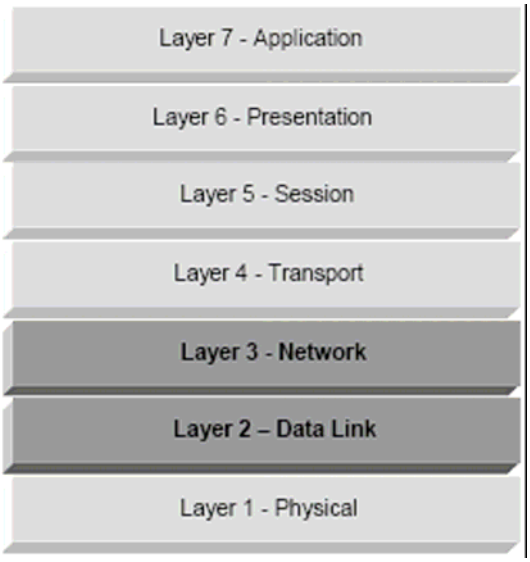

Figure 3.1 OSI Layers Addressed by Packet Filters

Packet filtering firewalls and routers can also filter network traffic based upon certain characteristics of that traffic, such as whether the packet's Layer 3 protocol might be the Internet Control Message Protocol (ICMP) - attackers have used this protocol to flood networks with traffic, thereby creating distributed denial-of-service (DDOS) attacks7. Packet filter firewalls also have the capability to block other attacks that take advantage of weaknesses in the TCP/IP suite.

3.1.1 Pros and Cons of Packet Filtering

Pros:

- The packet filter is the simplest of the firewall technologies to configure.
- Packet filtering capabilities are widely available in many hardware and software routing products, both commercially and they are freely available over the Internet.
- Adding a packet filter to a router produces little or no additional performance overhead.
- The packet filter operates at the network and transport layer, thus working for all applications.
- One screening router can help protect an entire network.
- Packet filter firewalls are very suitable for high-speed environments where logging and user authentication with network resources are not important.

Cons:

- Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application-specific vulnerabilities or functions. For example, a packet filter firewall cannot block specific application commands
- Because of the limited information available to the firewall, the logging functionality present in packet filter firewalls is limited. Packet filter logs normally contain the same information used to make access control decisions (source address, destination address, and traffic type)
- Most packet filter firewalls do not support advanced user authentication schemes. Once again, this limitation is mostly due to the lack of upper-layer functionality by the firewall.
- They are generally vulnerable to attacks and exploits that take advantage of problems within the TCP/IP specification and protocol stack, such as

network layer address spoofing. Many packet filter firewalls cannot detect a network packet in which the OSI Layer 3 addressing information has been altered.

- As the complexity of the packet filtering on a router increases, the router performance will decrease. In some cases, the filtering is incompatible with certain caching strategies commonly used for performance enhancement.
- Some policies cannot readily be enforced by normal packet filtering routers.

Table 3.1 Packet filter firewall ruleset

	Source Address	Source Port	Destination Address	Destination Port	Action	Description
1	Any	Any	192.168.1.0	> 1023	Allow	Rules to allow return TCP Connections to internal subnet
2	192.168.1.1	Any	Any	Any	Deny	Prevent Firewall system itself from directly connecting to anything
3	Any	Any	192.168.1.1	Any	Deny	Prevent External users from directly accessing the Firewall system
4	192.168.1.0	Any	Any	Any	Allow	Internal users can access External servers
5	Any	Any	192.168.1.2	SMTP	Allow	Allow External users to send emails in
6	Any	Any	192.168.1.3	HTTP	Allow	Allow External users to access WWW server
7	Any	Any	Any	Any	Deny	"Catch All" Rule Everything not previously allowed is explicitly denied

3.1.2 Packet Filter Rule Sets

Table 3.1 shows a sample of a packet filter firewall ruleset for an imaginary network of IP address 192.168.1.0, with the “0” indicating that the network has addresses that range from 192.168.1.0 to 192.168.1.254. For most firewalls, the ruleset would be much larger and detailed.

- **Accept**: the firewall passes the packet through the firewall as requested, subject to whatever logging capabilities may or may not be in place.
- **Deny**: the firewall drops the packet, without passing it through the firewall. Once the packet is dropped, an error message is returned to the source system. The "Deny" action may or may not generate log entries depending on the firewall's ruleset configuration.
- **Discard**: the firewall not only drops the packet, but it does not return an error message to the source system. This particular action is used to implement the "black hole" methodology in which a firewall does not reveal its presence to an outsider. As with the other actions, the "Discard" action may or may not generate logs entries.

3.2 STATEFUL INSPECTION FIREWALLS

The most obvious disadvantage of static packet filters is the array of "doors" that must be left open at all times to allow desired traffic. This weakness makes sites with static packet filters open to a wide range of attacks preying on the security of hosts on the internal networks. Since host security is often treated as a lower priority by organizations, the Proxse types of attacks are frequently successful [18].

To address this issue, dynamic packet filtering techniques were developed. Dynamic packet filters open and close "doors" in the firewall based on header information in the data packet as described in the "Packet filtering" box. Once a series of packets has passed through the "door" to it's destination, the firewall closes the door.

Stateful packet filtering is an enhancement to dynamic packet filtering. This technology tries to make sense out of higher-level protocols and adapt filtering rules to accommodate protocol-specific needs (e.g., simulated connections for connectionless protocols such as NFS and RPC services). The stateful packet filter keeps track of state and context information about a session.

This technology can be applied to the UDP protocol as well, setting up a virtual session, giving the illusion of security where no security exists. In Check Point's

implementation, this inspection module sits between the Data Link layer and Network layer.

Adding state tracking to a packet filter certainly may increase the security of the basic filter, but it does not address the content or implications of the traffic being handled. While dynamic packet filtering does well in reducing the amount of exposure, external systems— under the control of the firewall—still are able to make an IP connection with an internal machine as the endpoint. Again, once in, the attacker is pretty much free to roam wherever. Only the personal protection each internal host possesses keeps it safe.

What is commonly known as “spoofing,” pretending to be a trusted IP address, as a method of attacking the network behind that device worked well for many years. Most modern dynamic packet filters include fixes to most known methods of spoofing. But, the trust placed in an external system based on its IP address is still a problem. Even if the incoming traffic is from the proper host, there is no check to confirm that the authorized owners are operating the host. In other words, if a hacker has compromised that external host it can be used as a gateway to your internal network.
Stateful inspection firewalls are packet filters that incorporate added awareness of the OSI model data at Layer 4, as shown in Figure 3.2.

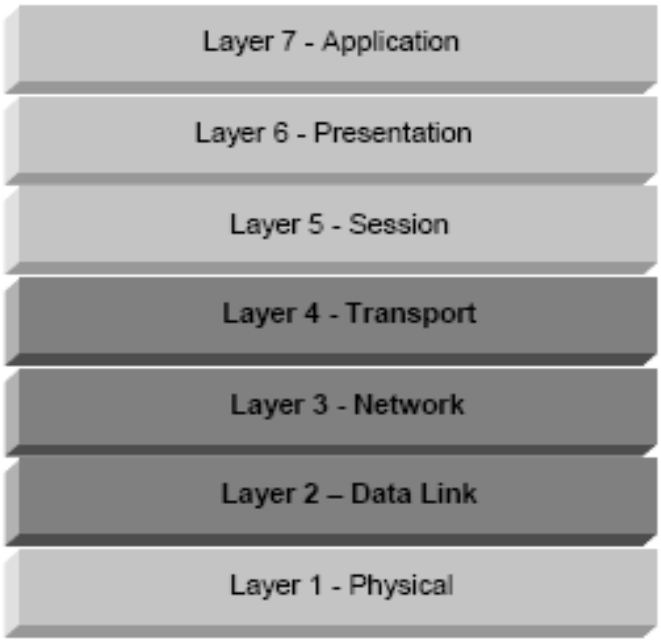

Figure 3.2 OSI Layers Addressed by Stateful Inspection

Stateful inspection firewalls add Layer 4 awareness to the standard packet filter architecture. Stateful inspection firewalls share the strengths and weaknesses of packet filter firewalls, but due to the state table implementation, stateful inspection firewalls are generally considered to be more secure than packet filter firewalls.

Table 3.2 Stateful Firewall Connection State Table

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	176.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.33.212.15	3321	192.168.1.6	80	Established

3.2.1 Pros and Cons of Stateful Firewall

Pros:

- Low overhead/high throughput. The stateful-inspection provides enhanced security over the packet-filtering model without sacrificing notable performance degradation.
- Like the packet filter, it also works at the network and transport layers, thus no special client configuration or client software is required.
- Only temporarily opens holes in the Network Perimeter.

Since the time a hole in the perimeter is open is greatly reduced, many types of attacks that work against static packet filters are more difficult or perhaps impossible to use against a dynamic packet filter. Again, because there is very little work done outside of routing traffic, the overhead is relatively low. Therefore, similar hardware platforms will often produce higher throughput when using dynamic packet filtering techniques than when using application gateways.

- Supports almost any service (e.g., back-channel services (life File Transport Protocol (FTP) have to be handled as a special case). Since packet filters are application-unaware, they can be set up to allow any type of IP traffic to pass thru the firewall.

Cons:

- Allows direct IP connections to internal hosts by external clients. While dynamic packet filtering does well in reducing the amount of exposure, external systems—under the control of the firewall—still are able to make an IP connection with an internal machine as the endpoint. The primary disadvantage of any packet-filtering gateway is that once access has been granted by the device to a host on the internal network, the attacker has direct access to any exploitable weaknesses in either the software or the configuration of that host. The ability to jump off to other internal hosts from that point is restrained only by the security present on those hosts.
- Offers no user authentication (if supported, it is supported via an application gateway).
- This type of firewall requires more administrative setup than packet filtering. (The connection table has to be built to track individual packet flows. These flows are then checked against preset policies to determine access or denial action to be taken.)
- A stateful inspection firewall also differs from a packet filter firewall in that stateful inspection is useful or applicable only within TCP/IP network infrastructures. Stateful inspection firewalls can accommodate other network protocols in the same manner as packet filters, but the actual stateful inspection technology is relevant only to TCP/IP. For this reason, many texts classify stateful inspection firewalls as representing a superset of packet filter firewall functionality.

3.3 APPLICATION-PROXY GATEWAY FIREWALLS

Application-Proxy Gateway firewalls are advanced firewalls that combine lower layer access control with upper layer (Layer 7 – Application Layer) functionality.

Application-proxy gateway firewalls do not require a Layer 3 (Network Layer) route between the inside and outside interfaces of the firewall; the firewall software performs the routing. In the event the application-proxy gateway software ceases to function, the firewall system is unable to pass network packets through the firewall system. All network packets that traverse the firewall must do so under software (application-proxy) control.

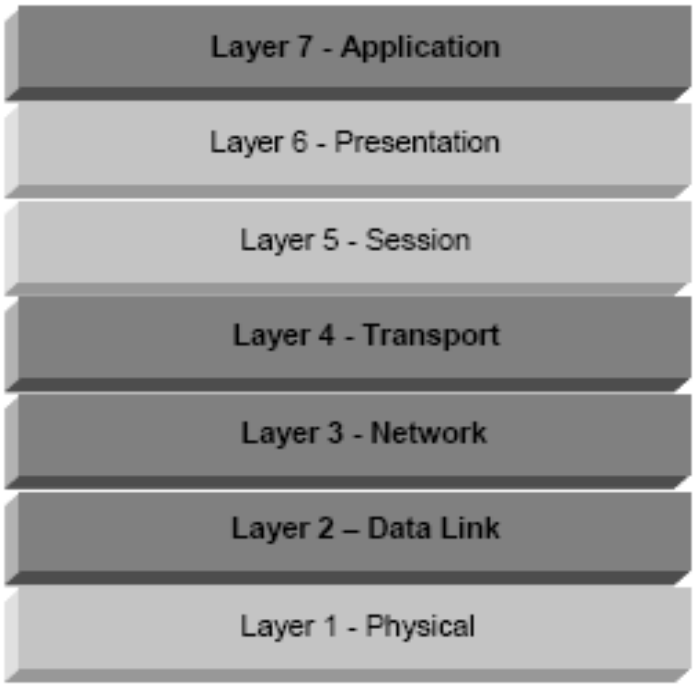

Figure 3.3 OSI Layers Addressed by Application-Proxy Gateway Firewalls

Each individual application-proxy, also referred to as a proxy agent, interfaces directly with the firewall access control ruleset to determine whether a given piece of network. Traffic should be permitted to transit the firewall. In addition to the ruleset, each proxy agent has the ability to require authentication of each individual network user. This user authentication can take many forms, including the following:

- User ID and Password Authentication,
- Hardware or Software Token Authentication,
- Source Address Authentication, and
- Biometric Authentication.

In addition to authentication and logging functionality, dedicated proxy servers are useful for web and email content scanning, including the following:

- Java applet or application filtering (signed versus unsigned or universal),
- ActiveX® control filtering (signed versus unsigned or universal),
- JavaScript filtering,
- Virus scanning and removal,
- Macro virus scanning, filtering, and removal,
- Application-specific commands, for example, blocking the HTTP “delete” command, and
- User-specific controls, including blocking certain content types for certain users

3.3.1 Pros and Cons of Application-proxy gateway firewall

Pros:

- The application proxy operates at the application layer, thus providing the highest level of security and granularity.
- Can be good at logging.
- Can provide caching.
- Can do intelligent filtering.
- Can perform user-level authentication.
- Automatically provide protection for weak or faulty IP implementations.

Cons:

- This firewall is the most complex of the firewalls to configure.
- The proxy acts as relay agent, thus producing a potential
- Performance bottleneck.
- Although proxy software is widely available for the older and
- Simpler services like FTP and telnet, proven software for newer or less widely used services is harder to find.
- May require different servers for each service.
- Usually require modifications to clients, applications or procedures.

3.4 NETWORK ADDRESS TRANSLATION

Network Address Translation (NAT) technology was developed in response to two major issues in network engineering and security. First, network address translation is an effective tool for “hiding” the network-addressing schema present behind a firewall environment. In essence, network address translation allows an organization to deploy an ad-dressing schema of its choosing behind a firewall, while still maintaining the ability to connect to external resources through the firewall. Second, the depletion of the IP address space has caused some organizations to use NAT for mapping non-routable IP addresses to a smaller set of legal addresses.

Like packet filtering, network address translation works by having a router do extra work. Not only does the router send packets on, but it also modifies them. When an internal machine sends a packet to the outside, the network address translation system modifies the source address of the packet to make the packet look as if it is coming from a valid address. When an external machine sends a packet to the inside, the network address translation system modifies the destination address to turn the externally visible address into the correct internal address. The network address translation system can also modify the source and destination port numbers.

Network address translation can use different schemes for translating between internal and external addresses:

- Allocate one external host address for each internal address and always apply the same translation.
- Dynamically allocate an external host address each time an internal host initiates a connection, without modifying port numbers.
- Create fixed mapping from internal addresses to externally visible addresses, but use port mapping so that multiple internal machines use the same external addresses.
- Dynamically allocate an external host address and port pair each time an internal host initiates a connection.

Network address translation is accomplished in main three fashions:

- Static Network Address Translation
- Dynamic Network Address Translation
- Port Address Translation (PAT)

3.4.1 Static Network Address Translation

In static network address translation, each internal system on the private network has a corresponding external, routable IP address associated with it. This particular technique is seldom used, due to the scarcity of available IP address resources. With static network address translation, it is possible to place resources behind (inside) the firewall, while maintaining the ability to provide selective access to external users.

Table 3.3 Static Network Address Translation Table

Internal IP Address	External (Globally Routable) IP Address
192.168.1.100	207.119.32.81
192.168.1.101	207.119.32.82
192.168.1.102	207.119.32.83
192.168.1.103	207.119.32.84
192.168.1.104	207.119.32.85
192.168.1.105	207.119.32.86
192.168.1.106	207.119.32.87

In other words, an external system could access an internal web server whose address has been mapped with static network address translation. The firewall would perform mappings in either direction, outbound or inbound. Table 3.3 shows an example of a static network address translation table that would map internal IP addresses, non-routable according to RFC 1918, to externally routable addresses.

3.4.2 Dynamic Network Address Translation

In Dynamic Network Address Translation, Internal IP Address is converted to the any one of IP address of pull of IP addresses. This is normally using when we have less number of live IP available.

3.4.3 Port Address Translation (PAT)

PAT allows you to translate your local addresses behind a single global address. This is called Port address Translation because firewall uses a single translated source address but changes the source port to allow multiple connections via a single global address. The limitation for PAT is approximately 64000 hosts due to the limitation of available ports (65535) and number of ports already assign to specific service.

3.5 VIRTUAL PRIVATE NETWORK

Another valuable use for firewalls and firewall environments is the construction of Virtual Private Networks (VPNs). A virtual private network is constructed on top of existing network media and protocols by using additional protocols and usually, encryption. If the VPN is encrypted, it can be used as an extension of the inner, protected network.

A virtual private network employs encryption and integrity protection so that you can use a public network (i.e., the internet) as if it were a private network (i.e., a piece of cabling that you control).

A virtual private network is an attempt to combine the advantages of a public network (it's cheap and widely available) with some of the advantages of a private network (it's secure). Fundamentally, all virtual private networks that run over the Internet employ the same principle: traffic is encrypted, integrity protected and encapsulated into new packets. These packets are sent across the Internet to something that undoes the encapsulation, checks the integrity and decrypts the traffic.

4. CURRENT FIREWALL APPROACHES

Current firewall approaches include designs by both commercial entities and open-source projects. They can be implemented as software applications or as dedicated hardware appliances. This chapter will detail some of the options available for current firewall approaches on both single machine and parallel architectures.

4.1 SINGLE FIREWALL ARCHITECTURES

Traditional firewall designs involve a single machine to enforce a security policy. These machines can either: 1. Require a resident operating system to run the firewall as an application; or 2. Run the design on hardware as a custom designed/programmable circuit. This section will introduce some examples and give an overview of their advantages and disadvantages.

4.1.1 Software Firewalls

Generally speaking, the term software firewalls might imply a scope of only userspace programs like Zone Labs Zone Alarm Pro. In fact, software firewalls refer to any number of user-space or kernel level applications. While, due to performance issues, the user-space examples are usually only used on an individual or debugging level, kernel level solutions are commonly employed to protect many production level environments. (Sun's SunScreen, Check Point Firewall 1, Linux's Netfilter, Symantec's Enterprise Firewall,BSD's IPFilter).

The disadvantage of these solutions is the reliance on a resident Operating System (OS). Ultimately this limits speed capabilities to the bounds of the processor, bus, and network interface card. This also limits hardware compatibilities to within the OS's bounds. For instance, iptables can run on Linux compatible systems and Symantec's Enterprise Firewall operates on both Microsoft and Sun OS compatible hardware

4.1.2 Hardware Firewalls

A hardware firewall is a device that has hardware circuitry dedicated to processing network traffic streams.

Popular technologies for hardware packet filtering include:

- Field Programmable Gate Array (FPGA)
- Application Specific Integrated Circuit (ASIC) or Network Processing Unit (NPU)

These devices seek to run packet filtering as close to line speed as possible. In essence they are finite state machines that once given an application, become dedicated to a singular task.

In the case of an FPGA, applications are run by a virtual circuit table to emulate hardware speeds. However, the steep programming learning curve and relative high cost of FPGA's prevent in-house development for most organizations A hardware firewall is a device that has hardware circuitry dedicated to processing network traffic streams.

No matter how much a policy is optimized, and even whether the firewall is executed as software or hardware, policy decision processing speed will still be limited by the capabilities of a single device. The most common solution to this issue is to buy larger and more powerful single point of entry machines, a highly non-modular and cost ineffective approach. In situations where there is a surge of illegitimate traffic, such as Denial of Service (DoS) attacks, a non-scalable firewall solution like a single point of entry can quickly become overwhelmed. In such a scenario legitimate users may notice quickly declining network performance and lower QoS, and may loose all external network access if the firewall fails. If networks are to protect themselves in a cost effective manner while allowing for future growth, more scalable and dynamic solutions for firewall architecture must be devised.

4.2 DISTRIBUTED FIREWALL ARCHITECTURES

Parallelization can greatly enhance the performance of network firewalls by offering scalability to reduce processing loads. Parallelizing firewalls can be

implemented by either dividing the traffic or the workload across an array firewall nodes [3].

There are two main ways to divide a single processor design with independent discrete computations. The first is to divide the data and the second is to divide the work. Using terminology developed for parallel computing, a design that distributes the data (packets) across the firewall nodes is considered data parallel.

4.2.1 Data Parallel Firewall System

An array of m firewall nodes that enforces a policy R operates in a data parallel fashion if:

- Arriving traffic is divided (distributed) such that each packet is processed by only one firewall node.
- Each firewall node i employs a local policy $R_i = R$, where i = 1, . . .m.

As seen in Figure 4.1, a data parallel firewall system consists of multiple identical firewalls connected in parallel. Each i^{th} firewall node needs a local policy (rule set) R_i that will allow it to act independently. In other words, each local policy is a duplicate of the complete security policy. Arriving packets are then distributed across the firewall nodes such that only one firewall node processes any given packet. Therefore different packets are processed in parallel and all packets are compared to the entire security policy. How the load-balancing algorithm distributes packets is vital to the system and is typically implemented as a high-speed switch in commercial products.

An array of m firewall nodes arranged in a data parallel fashion would always maintain integrity of a policy R.

Although data parallel firewalls have been shown to achieve higher throughput than traditional firewalls and have an innately redundant design, they suffer from major disadvantages.

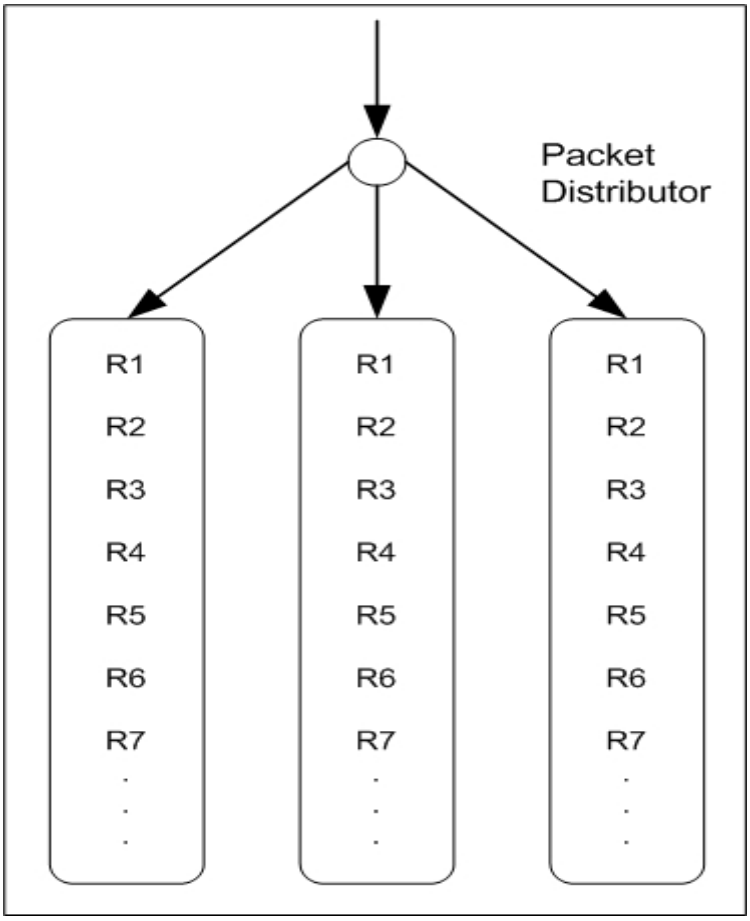

Figure 4.1 Data parallel, packets distributed across an array of equal firewall nodes

First, since the efficiency of scalability is limited to the system input load, the performance benefit over traditional firewall models is only evident under high traffic loads. Since the data parallel firewall system duplicates the policy, it does not reduce the processing time per packet, just the arrival rate into any firewall node.

Second, this design has difficulty enforcing QoS guarantees across networks. For example, since high priority traffic is processed on the same firewall nodes as low priority traffic, the high priority traffic can encounter longer delays if it is queued behind low priority traffic that requires more processing. Under these circumstances, users notice poor network performance, which is a growing concern, as more network applications require QoS assurances.

Third, since stateful inspection maintains tables of previously determined decisions, stateful inspection requires all traffic from a certain connection or

exchange to traverse the same firewall node, which this design does not account for and is difficult to perform at high speeds using the data parallel approach. For instance, assume a parallel firewall system has state filtering enabled to verify traffic by corresponding TCP SYN and ACK flags. If a single firewall node did not receive all packets in that connection stream, the ill matching flags would cause invalid state table entries to be made and intermittent packets could be lost. Therefore all packets belonging to a connection must traverse the same firewall node, or all firewall nodes, specifications not listed in the original design, must share state information. New parallel firewall architectures must solve these problems to meet future demands and increasing security threats.

4.2.2 Functional Parallel Firewall System

Figure 4.2 depicts a function parallel design that consists of an array of firewall nodes and a packet duplicator. The packet duplicator is a device that acts as a mechanism for duplicating packets among the firewall system. In this design, the security policy is distributed across all firewall nodes, while arriving packets are duplicated to all firewall nodes [3].

An array of m firewall nodes that enforces a policy R, with an accept set A, operates in a function parallel with gate fashion if:

1. Arriving traffic is duplicated to all firewall nodes and to an additional gate node.

2. Each firewall node i employs a local policy R_i , i = 1, . . .m, and $\bigcup_{i=1}^{m} Ai = A$

3. After a packet is processed by each firewall node i, the result of only one firewall will be forwarded according to policies.

This design provides a new type of firewall node interaction. Instead of reducing the amount of input on any firewall node, this design reduces the processing time required for any input on any firewall node.

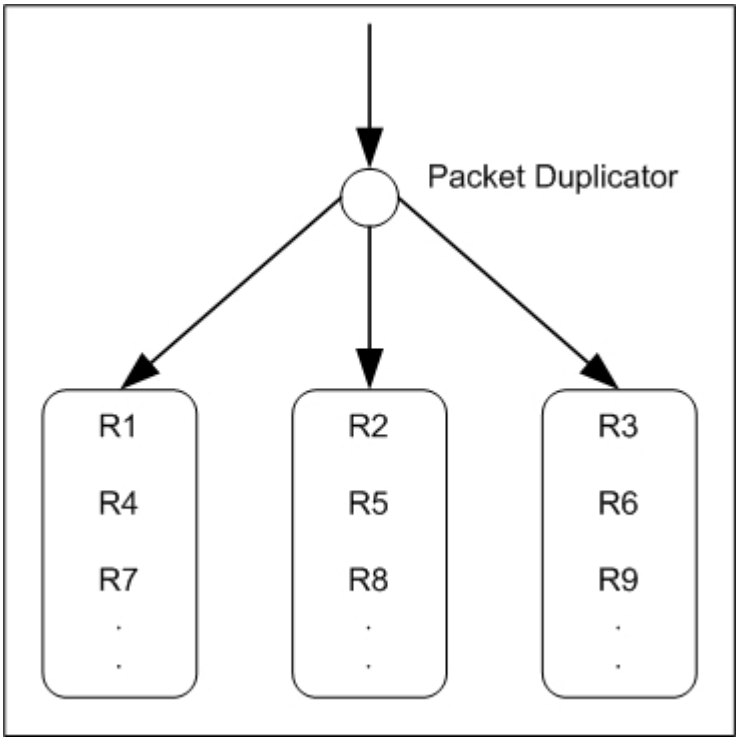

Figure 4.2 Function parallel firewall design, rules distributed across array of firewall nodes

The operation of this function parallel system, as shown in Figure 4.2, can be described as follows. Again, assume a first match policy. When a packet arrives at the function parallel firewall, it is duplicated to each firewall node. To ensure no more than one firewall node routes any packet into the system, policies integrity should be maintains.

Each firewall node processes the duplicated packet using its local policy, including any state information. Note, the rule number may correspond to a state match if the packet belongs to an established connection. In a first match method this number should have a uniformly lower value since state rules are evaluated before policy rules. Instead of a firewall node immediately executing actions on rule matches, it first check in state table, if that is related to established connection then it accept the packets else start checking rules in sequence order.

If all firewall nodes find no-match for that packet in rule set in that, then all executes default policy on that packet. In this design only delay all default policy

will work else if all firewall node haves default accept policy then some packets, which are not matching in any rule, is routed by all firewall nodes.

These Functional parallel firewalls can easily enforcing QoS guarantees across networks. Since high priority traffic is processed on the different firewall nodes. The high priority traffic can encounter lower delays as firewall handling this traffic has lower number of rules to process. In this design, as number of node in distributed firewall increases we can get better performance. Also in this design we can implement stateful impaction as every time traffic of same connection will go through same firewall in distributed firewall.

5. FIREWALL POLICY MODELING

As a basic requirement for any firewall policy management solution, we first model the relations between the rules in a firewall policy. Rule relation modeling is necessary for analyzing the firewall policy and designing management techniques such as anomaly discovery and policy editing. In this section, we formally describe model of firewall rule relations.

Table 5.1 A firewall policy example [9]

ORDER	PROTOCAL	SRC_IP	SRC_P	DST_IP	DST_P	ACTION
1	TCP	140.192.37.20	ANY	*.*.*.*	80	DENY
2	TCP	140.192.37.*	ANY	*.*.*.*	80	ACCEPT
3	TCP	*.*.*.*	ANY	161.120.33.40	80	ACCEPT
4	TCP	140.192.37.*	ANY	161.120.33.40	80	DENY
5	TCP	140.192.37.30	ANY	*.*.*.*	21	DENY
6	TCP	140.192.37.*	ANY	*.*.*.*	21	ACCEPT
7	TCP	140.192.37.*	ANY	161.120.33.40	21	ACCEPT
8	TCP	*.*.*.*	ANY	*.*.*.*	ANY	DENY
9	UDP	140.192.37.*	ANY	161.120.33.40	53	ACCEPT
10	UDP	*.*.*.*	ANY	161.120.33.40	53	ACCEPT
11	UDP	*.*.*.*	ANY	*.*.*.*	ANY	DENY

5.1 FORMALIZATION OF FIREWALL RULE RELATIONS

To be able to build a useful model for filtering rules, we need to determine all the relations that may relate two or more packet filters. In this section we define all the possible relations that may exist between filtering rules, and we show that there is no other relation exists. We determine the relations based on comparing the network fields of filtering rules as follows.

Rules R_x and R_y are ***completely disjoint*** if every field in R_x is not a subset and not a superset and not equal to the corresponding field in R_y.

Rules R_x and R_y are ***exactly matched*** if every field in R_x is equal to the corresponding field in R_y.

Rules R_x and R_y are ***inclusively matched*** if they do not exactly match and if every field in R_x is a subset or equal to the corresponding field in R_y. R_x is called the subset match while R_y is called the superset match.
For example, rule 1 inclusively matches rule 2 in Table 5.1. Rule 1 is the subset match of the relation while rule 2 is the superset match.

Rules R_x and R_y are ***partially disjoint*** (or partially matched) if there is at least one field in R_x that is a subset or a superset or equal to the corresponding field in R_y, and there is at least one field in R_x that is not a subset and not a superset and not equal to the corresponding field in R_y.
For example, rule 2 and rule 6 in table 5.1 is partially disjoint. (or partially matched).

Rules R_x and R_y are ***correlated*** if some fields in R_x are subsets or equal to the corresponding fields in R_y, and the rest of the fields in R_x are supersets of the corresponding fields in R_y.
For example, rule 1 and rule 3 in Table 5.1 are correlated.

5.2 FIREWALL POLICY ANOMALY CLASSIFICATION

Here, we describe and then define a number of possible firewall policy anomalies. These include errors for definite conflicts that cause some rules to be always suppressed by other rules, or warnings for potential conflicts that may be implied in related rules [9].

5.2.1 Shadowing Anomaly

A rule is shadowed when a previous rule matches all the packets that match this rule, such that the shadowed rule will never be activated. Rule Ry is shadowed by rule Rx if Ry follows Rx in the order, and Ry is a subset match of Rx, and the actions of Rx and Ry are different. As illustrated in the rules in Table 5.1, rule 4 is a subset match of rule 3 with a different action. We say that rule 4 is shadowed by rule 3, as rule 4 will never get activated.

Shadowing *is a critical error* in the policy, as the shadowed rule never takes effect. This might cause a permitted traffic to be blocked and vice versa. It is important to discover shadowed rules and alert the administrator who might correct this error by reordering or removing the shadowed rule.

5.2.2 Correlation Anomaly

Two rules are correlated if the first rule in order matches some packets that match the second rule and the second rule matches some packets that match the first rule. Rule Rx and rule Ry have a correlation anomaly if Rx and Ry are correlated, and the actions of Rx and Ry are different. As illustrated in the rules in table 5.1, rule 1 is in correlation with rule 3; if the order of the two rules is reversed, the effect of the resulting policy will be different.

Correlation is considered an *anomaly warning* because the correlated rules imply an action that is not explicitly handled by the filtering rules. Consider rules 1 and 3 in table 5.1. The two rules with this ordering imply that all HTTP traffic coming from address 140.192.37.20 and going to address 161.120.33.40 is denied. However, if their order is reversed, the same traffic will be accepted. Therefore, in order to resolve this conflict, we point out the correlation between the rules and prompt the user to choose the proper order that complies with the security policy requirements.

5.2.3 Generalization Anomaly

A rule is a generalization of another rule if this general rule can match all the packets that match a specific rule that precedes it. Rule Ry is a generalization of rule Rx if Ry follows Rx in the order, and Ry is a superset match of Rx, and the actions of Ry and Rx are different. As illustrated in the rules in Table 5.1, rule 2 is a generalization of rule 1; if the orders of the two rules are reversed, the effect of the resulting policy will be changed, and rule 1 will not be effective anymore, as it will be shadowed by rule 2. Therefore, as a general guideline, if there is an inclusive match relationship between two rules, the superset (or general) rule should come after the subset (or specific) rule.

Generalization is considered only an *anomaly warning* because the specific rule makes an exception of the general rule, and thus it is important to highlight its action to the administrator for confirmation.

5.2.4 Redundancy Anomaly

A redundant rule performs the same action on the same packets as another rule such that if the redundant rule is removed, the security policy will not be affected. Rule Ry is redundant to rule Rx if Rx precedes Ry in the order, and Ry is a subset or exact match of Rx, and the actions of Rx and Ry are similar. If Rx precedes Ry in the order, and Rx is a subset match of Ry, and the actions of Rx and Ry are similar, then Rule Rx is redundant to rule Ry provided that Rx is not involved in any generalization or correlation anomalies with other rules preceding Ry. As illustrated in the rules in Table 5.1, rule 7 is redundant to rule 6, and rule 9 is redundant to rule 10, so if rule 7 and rule 9 are removed, the effect of the resulting policy will not be changed.

Redundancy is *considered an error*. A redundant rule may not contribute in making the filtering decision, however, it adds to the size of the filtering rule table, and might increase the search time and space requirements. It is important t discover redundant rules so that the administrator may modify its filtering action or remove it altogether.

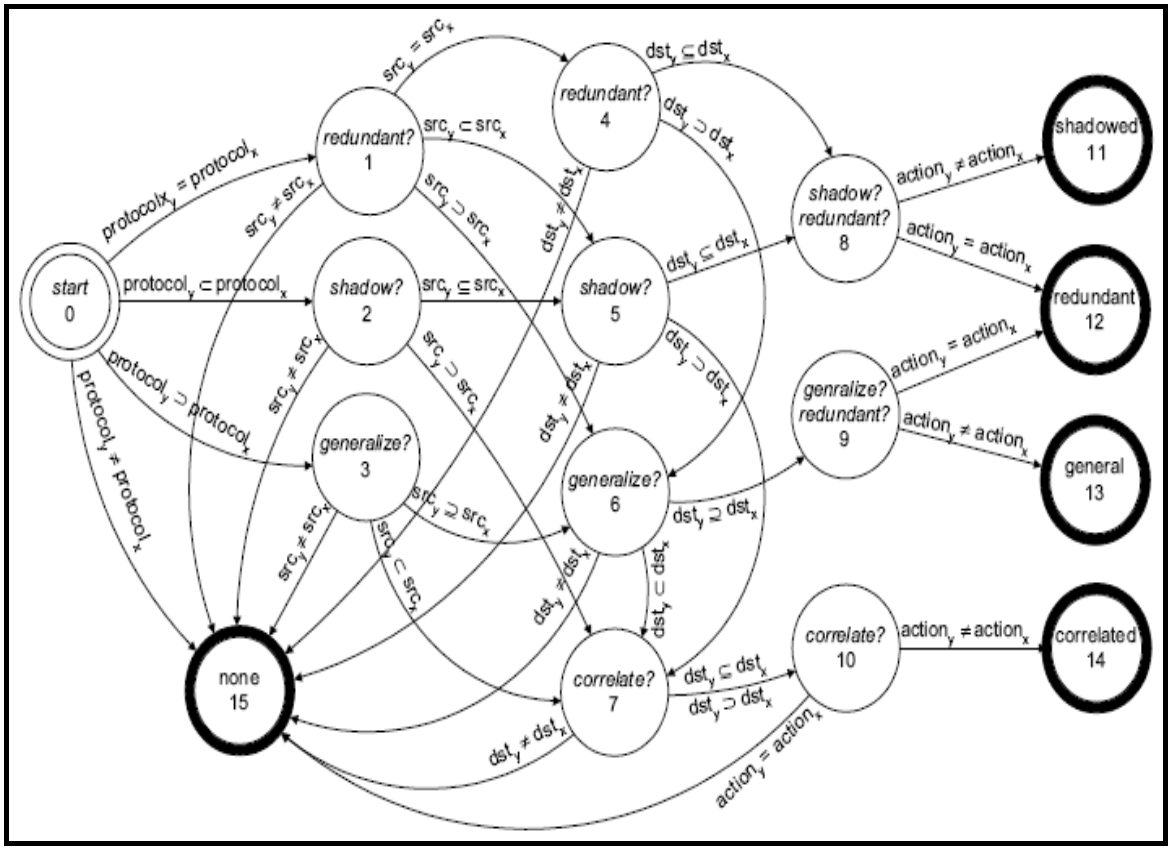

Figure 5.1 State diagram for detecting anomalies for rules Rx and Ry, Ry comes after Rx. [9]

6. FIREWALL POLICIES OPTIMIZATION

As we now that, Rule traversal is top to bottom, one rule at a time, until the packet matches a rule. The rules on a chain must be ordered hierarchically, from most general to most specific. As network complexity increases we have to concentrate on organization of rules list, as each and every packet check against each rules. As high-speed network become more prevalent, delays will become more significant. This chapter introduces firewall optimization. Optimization can be divided into three major categories: rule organization, use of the state module, and user-defined chains [2].

6.1 RULE ORGANIZATION

There is no hard-and-fast formula for rule organization. The two main underlying factors are which services are hosted on the machine and the machine's primary purpose, noting especially the heaviest traffic services on the machine. The third underlying factor to consider when preparing to organize rules for firewall optimization is the available network bandwidth, the speed of the Internet connection.

A policy DAG is used to optimize a policy in order to reduce the time [13] required to find a match while retaining integrity. This section will introduce the concept of a policy profile, then use the policy DAG to reorder the rules for optimization.

Statistically, in any firewall policy model, given a varying traffic demographic, it is probable that particular rules in the set will have a higher frequency of first matches than others. Matches are also known as 'hits,' denoted as h_i, where i is the i^{th} rule in the policy. The knowledge of this distribution gathered over time is the policy profile.

A policy profile, example is seen in Table 6.1, brings to light an optimization consideration when using a first match policy. Since traffic is processed by comparing each packet to the policy until a match is found, if rules with higher hit ratios are at the bottom of the policy, then the average comparisons per

packet is non-optimal. Let P = $\{p_1, p_2, \ldots p_n\}$ be the policy profile, where $p_i = h_i / \sum_{j=1}^{n} hj = 1$, h_j is the probability, or hit ratio, that a packet will first match the i^{th} rule in the policy.

Table 6.1 A policy profile

Rule No.	Probability
1	0.01
2	0.02
3	0.17
4	0.1
5	0.2
6	0.5

In a comprehensive policy, every packet will find a match, thus $\sum_{i=1}^{n} pi = 1$. To calculate the average comparisons before a match is found per packet, E [n], on a first match policy, we use

$$E[n] = \sum_{i=1}^{n} i * pi \quad [9]$$

If a policy were reordered such that the hit ratios were in decreasing order, then the most common traffic would be handled in the quickest manner, reducing the average comparison per packet.

Unfortunately, precedence imposes a constraint when reordering a policy, preventing an ideal highest to lowest hit ratio order. The goal then is to rearrange a policy without violating precedence issues. DAG's provide a method of maintaining relationships between rules while still allowing reordering.

While it is clear that DAG's provide a way to ensure integrity maintenance, providing a sorting algorithm is the difficult part and advanced methods remain an open question. This is primarily due to the fact that any reorder of the policy would change the initial variables required to sort the policy (e.g. hit ratio),

creating a highly recursive situation. When seeking the optimal rule order, the problem escalates to NP-hard [10].

Other then this DAG's rules presentation, Network administrators has to follows, following tips for better firewall performance. Also this depends on type of services running and amount of traffic for different services passing.

1) Begin with Rules That Block Traffic on High Ports

These types of rules must come before the rules allowing traffic to specific services. Obviously, the FTP data channel rules must come near the end of the rule list, even though you'd want the rules to be near the top of the list because FTP transfers tend to be large.

2) TCP versus UDP services: Place UDP rules after TCP rules

Overall, UDP rules should be placed later in the firewall chains, after any TCP rules. This is because most Internet services run over TCP, and connectionless UDP services are typically simple, single-packet query-and-response services. Testing the single or UDP packet or a handful of them against the preceding rules for ongoing TCP connections doesn't add noticeable drag to a UDP query and response. A notable exception is streaming media, such as the RealAudio data stream.

3) ICMP services: Place their rules late in the rule chain

ICMP is another protocol whose firewall rules can be placed late in the rule chain. ICMP packets are small control and status messages. As such, they are sent relatively infrequently. Legitimate ICMP packets usually consist of a single, no fragmented packet. With the exception of echo-request, ICMP packets are almost always sent as a control or status message in response to an exceptional outgoing packet of some kind.

4) Place Firewall Rules for Heavily Used Services as Early as Possible

Generally, there are no hard-and-fast rules for firewall rule placement in a list. Rules for heavily used services, such as the HTTP-related rules for a dedicated web server, should be placed as early as possible. Rules for applications that involve high, ongoing packet counts also should be placed as early as possible.

However, as mentioned earlier, the data stream protocols for applications such as FTP and RealAudio require the rules to be placed near the end of the chain, after any other application rules.

6.2 USE OF THE STATE MODULE

Using the state module's ESTABLISHED and RELATED matches essentially allows for moving all rules for ongoing exchanges to the head of the chains, as well as eliminating the need for specific rules for the server half of a connection. In fact, bypassing filter matching for ongoing, recognized, previously accepted exchanges are one of the two primary purposes of the state module.

The state module's second primary purpose is to serve a firewall-filtering function. Connection-state tracking allows the firewall to associate packets with ongoing exchanges. This is particularly useful for connectionless, stateless UDP exchanges.

6.3 USER-DEFINED CHAINS

As described in Section 7.1, the filter table in iptables has three permanent, built-in chains: INPUT, OUTPUT, and FORWARD (discuses about same later in iptables chapter). Iptables enables you to define chains of your own, called user-defined chains. These user-defined chains are treated as rule targets that is, based on the set of matches specified in a rule, the target can branch off or jump to a user-defined chain. Rather than the packet being accepted or dropped, control is passed to the user-defined chain to perform more specific match tests relative to packets matching the branch rule. After the user-defined chain is traversed, control returns to the calling chain, and matching continues from the next rule in the calling chain unless the user-defined chain matched and took action on the packet.

Figure 6.1 shows the standard, top-down rule traversal using the built-in chains.

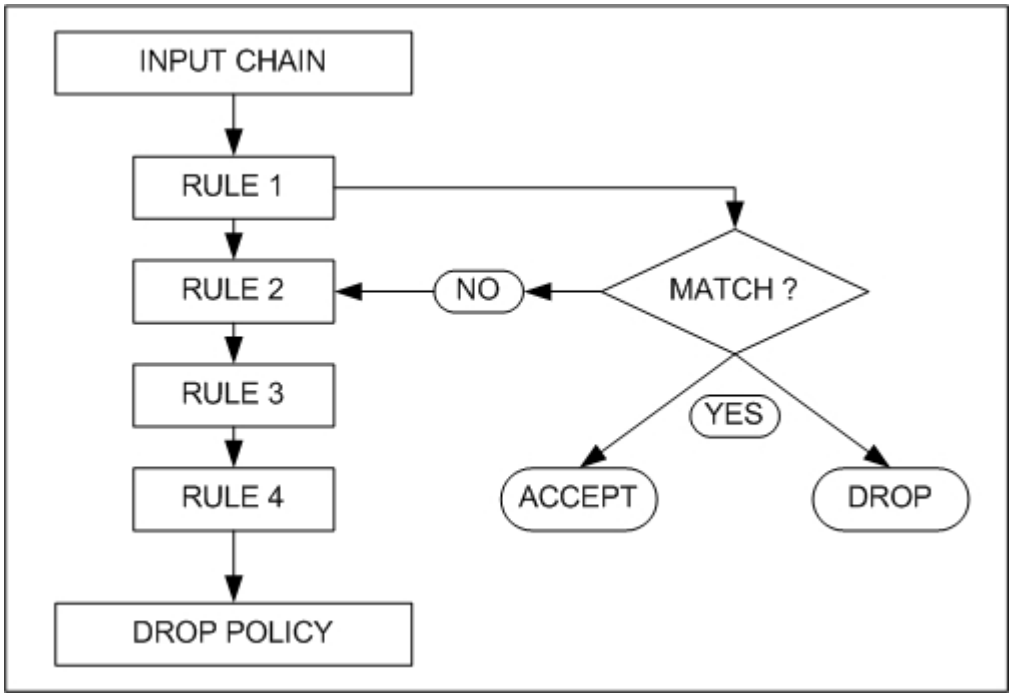

Figure 6.1 Standard chains traversal

User-defined chains are useful in optimizing the ruleset and therefore are often used. They allow the rules to be organized into categorical trees. Rather than relying on the straight-through, top-down check-off list type of matching inherent in the standard chain traversal, packet match tests can be selectively narrowed down based on the characteristics of the packet.

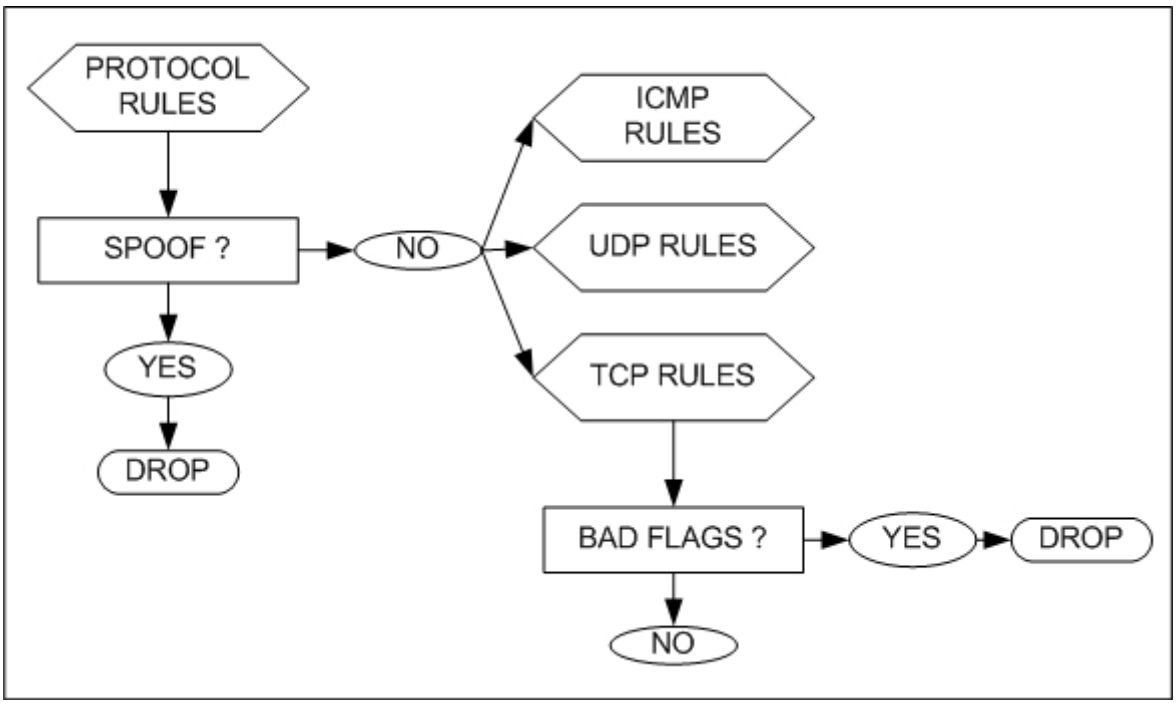

Figure 6.2 User-defined chains based on protocol

Figure 6.2 details the user-defined chain for the protocol rules for packets specifically addressed to this host. As shown, user-defined chains can jump to other user-defined chains containing even more specific tests. In this figure we can see we dividing rules for TCP, UDP and ICMP in separate chains. So at first place whenever any packet come it check's for protocol type and according to that packet is targeted to appropriate chains. Which reduces average number of rules required for processing packets in firewall node. So we get higher firewall performance.

7. NETFILTER / IPTABLES

Netfilter is a framework that provides a set of hooks within the Linux kernel for intercepting and manipulating network packets. The best-known component on top of netfilter is the firewall, which filters packets, but the hooks are also used by other components, which perform network address translation, stateful tracking and packet enqueueing to userspace. The name Netfilter also refers to the name of the project that provides a set of firewalling tools for Linux. These components are usually Loadable Kernel Modules, although the project also offers a set of userspace tools and libraries [19].

Iptables is the name of the user space tool by which administrators create rules for the packet filtering and NAT modules. While technically iptables is merely the tool which controls the packet filtering and NAT components within the kernel, the name iptables is often used to refer to the entire infrastructure, including netfilter, connection tracking and NAT, as well as the tool itself. Iptables is a standard part of all modern Linux distributions [19].

So simply we can say, “There is indeed a difference between iptables and Netfilter, though you'll often hear the terms used interchangeably. Netfilter is the Linux kernel-space program code to implement a firewall within the Linux kernel, either compiled directly into the kernel or included as a set of modules. On the other hand, iptables is the userland program used for administration of the Netfilter firewall” [2].

Rules in iptables are grouped into chains. A chain is a set of rules for IP packets, determining what to do with them. Each rule can possibly dump the packet out of the chain (short-circuit), and further chains are not considered. A chain may contain a link to another chain - if either the packet passes through that entire chain or matches a RETURN target rule it will continue in the first chain. There is no limit to how nested chains can be. There are three basic chains (INPUT, OUTPUT, and FORWARD), and the user can create as many as desired. A rule can merely be a pointer to a chain also.

7.1 ARCHITECTURE OF IPTABLES

There are three built-in tables in iptables, each of which contains some predefined chains. It is possible for extension modules to create new tables. The administrator can create and delete user-defined chains within any table. Initially, all chains are empty and have a policy target that allows all packets to pass without being blocked or altered in any fashion [2].

- Filter table — This table is responsible for filtering (blocking or permitting a packet to proceed). Every packet passes through the filter table. It contains the following predefined chains, and any packet will pass through one of them:
    - INPUT chain — All packets destined for this system go through this chain (hence sometimes referred to as LOCAL_INPUT)
    - OUTPUT chain — All packets created by this system go through this chain (aka. LOCAL_OUTPUT)
    - FORWARD chain — All packets merely passing through the system (being routed) go through this chain.

- NAT table — This table is responsible for setting up the rules for rewriting packet addresses or ports. The first packet in any connection passes through this table: any verdicts here determine how all packets in that connection will be rewritten. It contains the following predefined chains:
    - PREROUTING chain — Incoming packets pass through this chain before the local routing table is consulted, primarily for DNAT (destination-NAT).
    - POSTROUTING chain — Outgoing packets pass through this chain after the routing decision has been made, primarily for SNAT (source-NAT).
    - OUTPUT chain — Allows limited DNAT on locally-generated packets

- Mangle table — This table is responsible for adjusting packet options, such as quality of service, TOS modifications, marking packets and so on before routing process take place. All packets pass through this table. Because it

is designed for advanced effects, it contains all the possible predefined chains:

- PREROUTING chain — All packets entering the system in any way, before routing decides whether the packet is to be forwarded (FORWARD chain) or is destined locally (INPUT chain).
- INPUT chain — All packets destined for this system go through this chain
- FORWARD chain — All packets merely passing through the system go through this chain.
- OUTPUT chain — All packets created by this system go through this chain
- POSTROUTING chain — All packets leaving the system go through this chain.

In addition to the built-in chains, the user can create any number of user-defined chains within each table, which allows them to group rules logically.

7.2 PACKET TRAVERSAL IN IPTABLES

As described in Section 7.1, there are main three tables in iptables mangle, NAT and filter table and each table has predefined chains. Each chain contains a list of rules. When a packet is sent to a chain, it is compared against each rule in the chain in order. The rule specifies what properties the packet must have for the rule to match, such as the port number or IP address. If the rule does not match then processing continues with the next rule. If, however, the rule does match the packet, then the rule's target instructions are followed (and further processing of the chain is usually aborted). Some packet properties can only be examined in certain chains (for example, the outgoing network interface is not valid in the INPUT chain). Some targets can only be used in certain chains, and/or certain tables (for example, the SNAT target can only be used in the POSTROUTING chain of the NAT table).

Figure 7.1 shows how packets travel the tables and there chains.

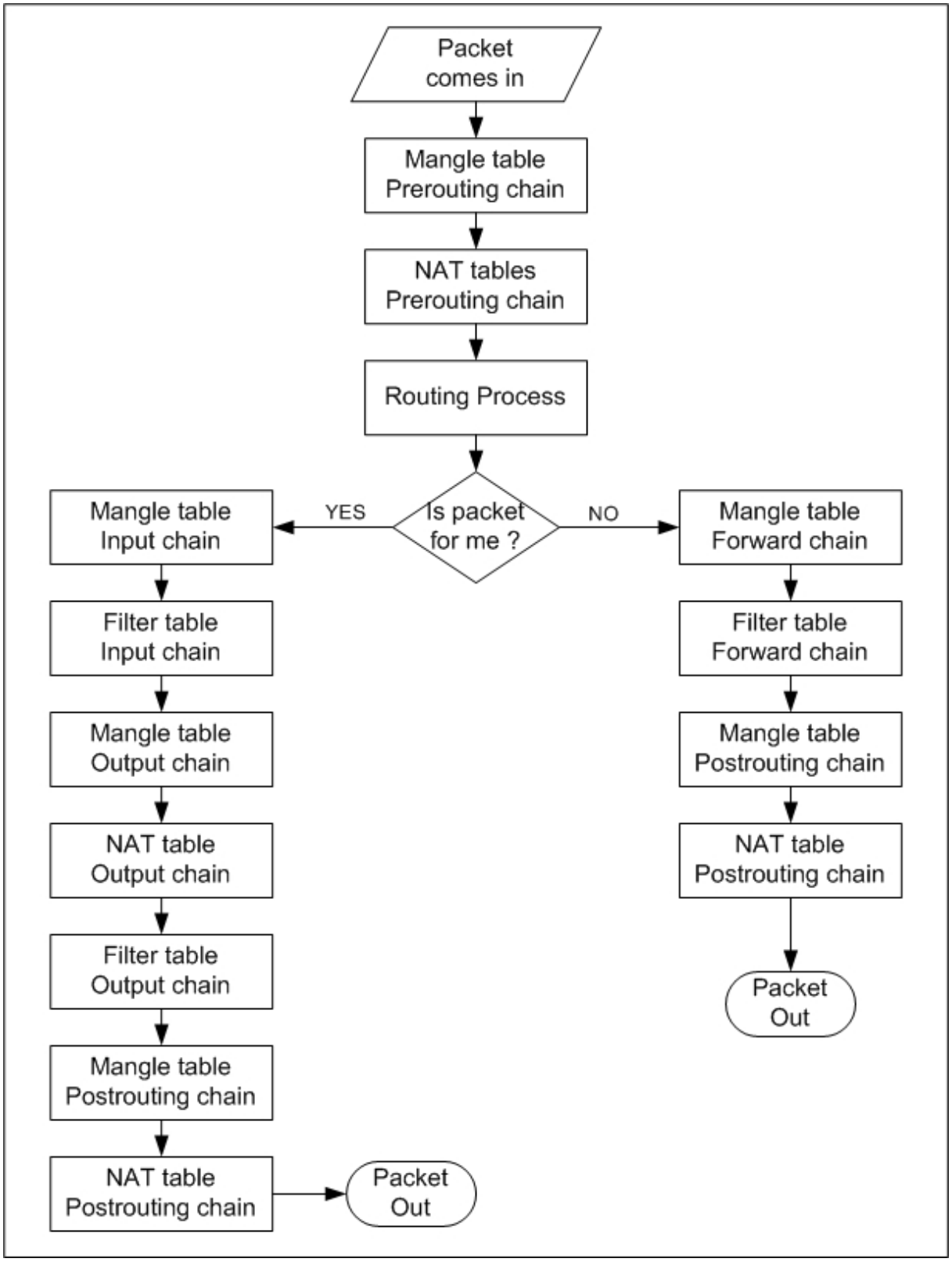

Figure 7.1 Packet traversal in iptables

7.3 ADVANCED FEATURES OF NETFILTER/IPTABLES

It is commonly known that netfilter/iptables is the firewall of the Linux operating system. What is not commonly known is that iptables has many hidden gems that can allow you do things with your firewall that you might never have even imagined [20].

Following are the advanced features of iptables firewall [21]:

1. Specifying multiple ports in one rule

The multiport module allows one to specify a number of different ports in one rule. This allows for fewer rules and easier maintenance of iptables configuration files.

For example, if we wanted to allow global access to the SMTP, HTTP, HTTPS and SSH ports on our server we would normally use something like the following:

```
-A INPUT -i eth0 -p tcp -m state --state NEW --dport ssh   -j ACCEPT
-A INPUT -i eth0 -p tcp -m state --state NEW --dport smtp  -j ACCEPT
-A INPUT -i eth0 -p tcp -m state --state NEW --dport http  -j ACCEPT
-A INPUT -i eth0 -p tcp -m state --state NEW --dport https -j ACCEPT
```

Using the multiport matching module, we can now write:

```
-A INPUT -i eth0 -p tcp -m state --state NEW -m multiport --dports ssh,smtp,http,https -j ACCEPT
```

2. Load balancing with random and nth

Both the random and nth extensions can be used for load balancing. If, for example, you wished to balance incoming traffic between two mirrored web servers then you could add either of the following rule sets to your nat table:

```
-A PREROUTING -i eth0 -p tcp --dport 80 -m state --state NEW -m nth --counter 0 --every 2 --packet 0 -j DNAT --to-destination 192.168.0.5:80
-A PREROUTING -i eth0 -p tcp --dport 80 -m state --state NEW -m nth --counter 0 --every 2 --packet 1 -j DNAT --to-destination 192.168.0.6:80
```

or

```
-A PREROUTING -i eth0 -p tcp --dport 80 -m state --state NEW -m random --average 50
-j DNAT --to-destination 192.168.0.5:80
-A PREROUTING -i eth0 -p tcp --dport 80 -m state --state NEW -m random --average 50
-j DNAT --to-destination 192.168.0.6:80
```

The nth matching extension allows you to match the nth packet received by the rule. There are up to 16 (0...15) counters for matching the nth packets. The above four (nth) rules use counter 0 to count every 4th packet. Once the 4th packet is received, the counter is reset to zero. The first rule matches the 1st packet (--packet 0) of every four counted, the second rule matches the 2nd packet (--packet 1).

The random matching extension allows you to match packets based on a given probability. The first rule from the set of random rules above matches 50% (--average 50) of the TCP connections to port 80 and redirects these to the first mirrored server. Of the 50% of connections not matching on the first rule, 50% will match the second rule.

3. Restricting the number of connections with limit and iplimit

The limit matching extension can be used to limit the number of times a rule matches in a given time period while the iplimit extension can restrict the number of parallel TCP connections from a particular host or network. These extensions can be used for a variety of purposes:

- to protect against DOS (denial of service) attacks such as preventing a flood of HTTP requests to your web server while ensuring all your customers have unlimited access
- to prevent a brute-force attack to guess passwords
- to limit Internet usage by staff during working hours

4. Matching against a string in a packet's data payload

The string extension allows one to match a string anywhere in a packet's data payload. Although this extension does have many valid uses, I would strongly advise caution. Let's say, for example, that our Linux firewall is protecting an

internal network with some computers running Microsoft Windows® and we would like to block all executable files. We might try something like

```
-A FORWARD -m string --string '.com' -j DROP
-A FORWARD -m string --string '.exe' -j DROP
```

5. Time-based rules

We can match rules based on the time of day and the day of the week using the time module. This could be used to limit staff web usage to lunch-times, to take each of a set of mirrored web servers out of action for automated backups or system maintenance, etc. The following example allows web access during lunch hour:

```
-A FORWARD -p tcp -m multiport --dport http,https -o eth0 -i eth1 -m time --timestart 12:30 --timestop 13:30 --days Mon,Tue,Wed,Thu,Fri -j ACCEPT
```

Clearly the start and stop times are 24-hour with the format HH:MM. The day is a comma-separated list that is case sensitive and made up of Mon, Tue, Wed, Thu, Fri, Sat and/or Sun.

6. Packet matching based on TTL values

The TTL (Time-To-Live) value of a packet is an 8-bit number that is decremented by one each time the packet is processed by an intermediate host between its source and destination. The default value is operating system dependant and usually ranges from 32 to 128. Its purpose includes ensuring that no packet stays in the network for an unreasonable length of time, gets stuck in an endless loop because of bad routing tables, etc. Once the TTL value of a packet reaches 0 it is discarded and a message is sent to its source, which can decide whether or not to resend it.

The usefulness of packet matching based on TTL value depends on your imagination. One possible use is to identify "man-in-the-middle" attacks. If you regularly connect from home to work you could monitor your TTL values and establish a reasonable maximum value at the receiving end. You can the use this to deny any packets that arrive with a higher TTL value as it may indicate a

possible "man-in-the-middle" attack; someone intercepting your packets, reading/storing them and resending them onto the destination.

As a simple example, let's reject all packets from a specific IP with a TTL of less than 40:

```
-A INPUT -s 1.2.3.4 -m ttl --ttl-lt 40 -j REJECT
```

You can also check for TTL values that are less than (--ttl-gt) or equal to (--ttl-eq) a particular value.

7. MARK mangle tables target extension

The MARK target is used to set Netfilter mark values that are associated with specific packets. This target is only valid in the mangle table, and will not work outside there. The MARK values may be used in conjunction with the advanced routing capabilities in Linux to send different packets through different routes and to tell them to use different queue disciplines (qdisc), etc. Note that the mark value is not set within the actual package, but is an value that is associated within the kernel with the packet. In other words, you cannot set a MARK for a packet and then expect the MARK still to be there on another host. If this is what you want, you will be better off with the TOS target, which will mangle the TOS value in the IP header.

For example,

```
iptables -t mangle -A PREROUTING -p tcp --dport 22 -j MARK --set-mark 2
```

The --set-mark option is required to set a mark. The --set-mark match takes an integer value. For example, we may set mark 2 on a specific stream of packets, or on all packets from a specific host and then do advanced routing on that host, to decrease or increase the network bandwidth, or routing packets with different mark to different gateway for load balancing etc [22].

8. PERFORMANCE EVALUATIONS

This chapter describes how an experimental lab setup for single traditional firewall and data parallel distributed firewall is implemented for experimental purpose. For implementing these designs we use off-the-shelf components for maintaining lower cost of design. Along with this design, performance measurement tool – Netperf is also described. Finally this chapter ends with experimental results that focuses on how network bandwidth and delay changes according to number of rules in firewall.

8.1 TOOLS USED FOR LAB SETUP AND PERFORMANCE EVALUATION

In this section tools are described, which are used in implementing lab setup and measuring performance of firewall.

8.1.1 Netperf: A Benchmark for Measuring Network Performance

Netperf is a benchmark that can be used to measure various aspects of networking performance. Its primary focus is on bulk data transfer and request/response performance using either TCP or UDP and the Berkeley Sockets interface.

Netperf is designed around the basic client-server model. There are two executables - netperf and netserver. Generally you will only execute the netperf program at client side while the netserver program will be invoked at server side and it keeps running continuously.

When you execute netperf, the first thing that will happen is the establishment of a control connection to the remote system. This connection will be used to pass test configuration information and results to and from the remote system. Regardless of the type of test being run, the control connection will be a TCP connection using BSD sockets.

Once the control connection is up and the configuration information has been passed, a separate connection will be opened for the measurement itself using the APIs and protocols appropriate for the test. The test will be performed, and

the results will be displayed. Netperf places no traffic on the control connection while a test is in progress. Certain TCP options, such as SO_KEEPALIVE, if set as your system's default, may put packets out on the control connection.

Using Netperf we can measure following things:

Bulk Data Transfer Performance:

The most common use of netperf is measuring bulk data transfer performance. This is also referred to as "stream" or "unidirectional stream" performance. Essentially, these tests will measure how fast one system can send data to another and/or how fast that other system can receive it. Again in bulk data transfer you will have two options for TCP stream and UPD stream performance.

Request/Response Performance:

Request/response performance is the second area that can be investigated with netperf. Generally speaking, netperf request/response performance is quoted as “transactions/sec" for a given request and response size. A transaction is defined as the exchange of a single request and a single response. From a transaction rate, one can infer one way and round-trip average latency.

8.1.2 Iproute2

Iproute2 is a collection of utilities for controlling TCP/IP networking and Traffic Control in Linux. It is currently maintained by Stephen Hemminger. The original author, Alexey Kuznetsov, is well known for the QoS implementation in the Linux kernel.

Most network configuration manuals still refer to ifconfig and route as the primary network configuration tools, but ifconfig is known to behave inadequately in modern network environments. They should be deprecated, but most Linux distributions still include them. Most network configuration systems make use of ifconfig and thus provide a limited feature set. The /etc/net project aims to support most modern network technologies, as it doesn't use ifconfig and allows a system administrator to make use of all iproute2 features, including traffic control.

iproute2 has following utilites:

ip address	-	protocol address management
ip interface	-	primary and Secondary Addressing
ip neighbour	-	neighbour/arp table management
ip route	-	routing table management
ip rule	-	routing policy database management
ip tunnel	-	ip tunnelling configuration
ip monitor	-	route state monitoring

8.2 EXPERIMENTAL LAB SETUP - SINGLE FIREWALL DESIGN

Figure 8.1 depicts a single firewall design that consists of three computers, out of this three; one computer is working as firewall system while the rest of two computers are working as server and client respectively, which communicate with each other through firewall system.

Firewall system-having Red hat Linux 9.0 (2.4.18 kernel) as operating system and iptables-1.3.7 serves as basic firewall server. Firewall Sever has two LAN cards for connecting to two different networks. Routing process is enabled in server for routing packets between two different networks. Clients have “Netperf” running for generating traffic and measuring network performance between them.

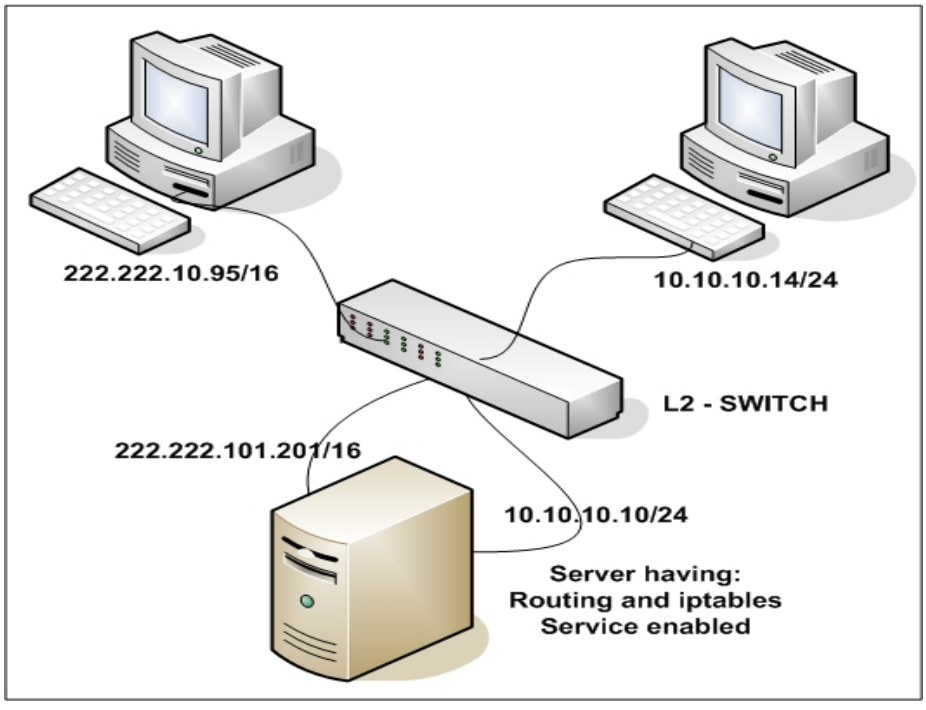

Figure 8.1 LAB setup for single firewall

8.2.1 Firewall Configuration for Single Firewall Design

The Firewall node consists of any computer with two network interface cards for communicating with two different networks. To consider affordability and off-the-shelf components, this design implements the firewall node with computers running the Linux operating system having iptables inbuilt in kernel.

As described in Section 7.1, iptables has set of rules which are placed in different chains and each packet coming into the firewall, is passed through default chains for checking against rules in those chains, with one rule checking at a time in sequential order.

In this design, all rules are placed in FORWARD chain of iptables, as packet arriving in firewall and if that packet is not intended for the firewall system then it processes within rules in that FORWARD chain.

8.3 EXPERIMENTAL LAB SETUP – DISTRIBUTED FIREWALL DESIGN

Figure 4.2 depicts a distributed firewall design that consists of an array of firewall nodes and a packet distributor. In this distributed firewall design, five computers are used. One node used as packet distributor, two nodes as firewall systems and the other two are working as client and server respectively communicating with each other. Netperf is used for measuring network performance.

The distributed firewall implemented here is data parallel distributed firewall, where all the packets arriving into packet distributor are distributed in round robin fashion amongst all firewall nodes in distributed design. As shown in Figure 8.2, packets are distributed between two firewall nodes.

In iproute2, ip rule and ip route utilities are used to configure different routes for differently marked packets and this marking of packets is done with the help of iptables MARK match extension.

8.3.1 Packet Distributor Configuration

Configuration of packet distributor is a tricky job and the main task in designing distributed firewall. This is the place where all packets are identified and distributed among multiple firewalls. This configuration requires the routing service of packet distributor to be enabled for forwarding the packets between two networks. For distributing packets, Author uses nth match extension in iptables, which matches every nth packet coming in the packet distributor. As each and every packet is identified that has to be marked by MARK target extension in iptables before routing process takes place. These marked packets are further processed by iproute2 routing process. Iproute2 can identify marked values of each packet and routes them according to the mark values.

8.3.2 Firewalls Configuration for Distributed Firewall Design

As in data parallel firewall, all firewall nodes in design implement the same rule list, firewall configuration is same as single firewall design. The only difference in the firewall configuration is that, in the functional firewall design, each firewall implements different rule lists in itself, according to the functionality it is providing.

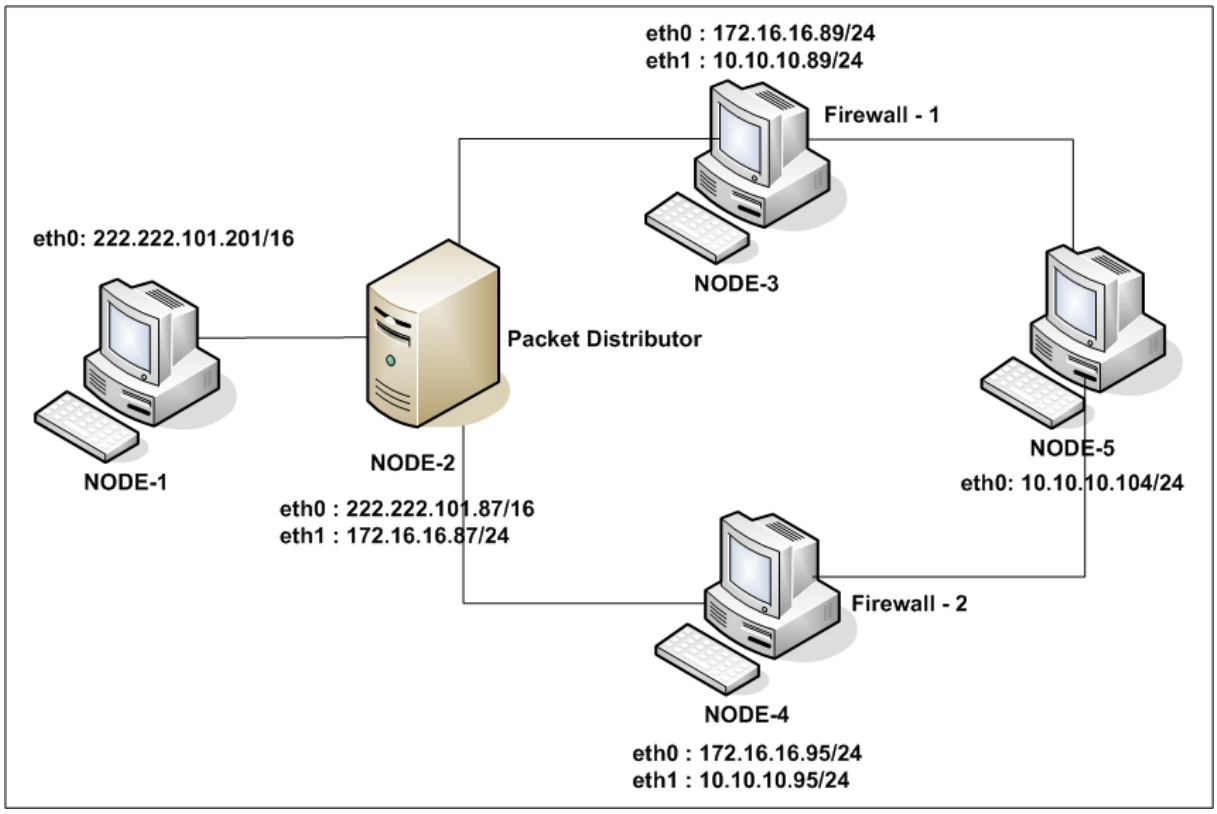

Figure 8.2 Distributed Firewall Design

8.4 EXPERIMENTAL RESULTS

For experimental result, author considered delay and bandwidth as the network performance parameters and measured the effect of number of rules on these network parameters. Results are shown for single (traditional) firewall and data parallel distributed firewall.

8.4.1 Throughput as Function of Number of Rules

Table 8.1 Throughput as Function of Number of Rules

No of Rules	Ex -1	Ex -2	Ex -3	Ex - 4	Average
0	40.28	40.20	37.78	41.21	39.87
1000	38.00	40.00	40.07	40.18	39.56
2000	21.44	22.84	38.82	28.35	27.86
3000	26.03	25.86	25.95	25.90	25.94
4000	19.77	19.82	19.74	19.79	19.78
5000	16.15	16.17	16.13	16.17	16.16
6000	13.54	13.57	13.60	13.59	13.58
7000	11.73	11.74	11.73	11.73	11.73
8000	10.38	10.38	10.38	10.38	10.38
9000	9.32	9.33	9.33	9.32	9.33
10000	8.42	8.42	8.42	8.42	8.42

** Throughput in 10^6 bits/sec

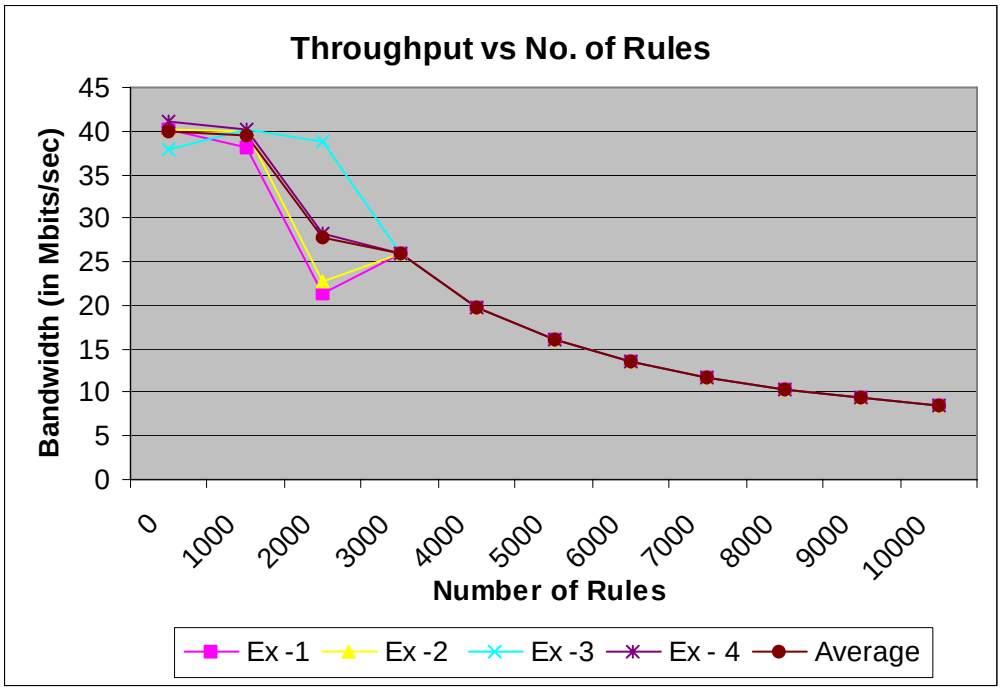

Figure 8.3 Throughput as Function of Number of Rules

We can see from the graph (Figure 8.3) that as the number of rules increases in firewall policy lists, throughput (bandwidth) decreases. Initially, when there are no rules in the firewall policy table, the bandwidth of approximately 40 Mbps is observed. By increasing the number of rules in the firewall, this bandwidth goes down up to 16 Mbps for 5000 rules. In this period, the performance decrease of 60% is observed. Moreover, the performance of the firewall decreases more rapidly than that of between 5000 to 10000 rules. At 5000 rules, the bandwidth of 16 Mbps is observed and this decreases up to 8 Mbps at 10000 rules that indicates 47% performance decrease.

8.4.2 Request/Response Performance as Function of Number of Rules

Table 8.2 Request/Response Performance as Function of Number of Rules

No of Rules	Ex -1	Ex -2	Ex -3	Ex - 4	Average
0	2151.08	2164.40	2176.11	2176.11	2166.93
1000	1755.52	1804.95	1781.87	1796.81	1784.79
2000	1254.24	1251.77	1262.24	1252.57	1255.21
3000	971.37	992.93	1001.46	1002.38	992.04
4000	853.05	840.74	821.10	856.85	842.94
5000	746.68	747.74	723.29	745.20	740.73
6000	652.22	656.21	657.41	656.24	655.52
7000	579.19	578.26	591.38	590.98	584.95
8000	535.04	470.01	453.58	536.68	498.83
9000	486.83	484.43	484.05	493.52	487.21
10000	443.53	385.91	445.50	452.90	431.96

**Transaction Rate per Second

Request/response performance is the second area that can be investigated for performance of firewalls. Generally speaking, request/response performance is quoted as “transactions/second" for a given request and response size (here it is 1 byte). A transaction is defined as the exchange of a single request and a single response.

Same scenario, as of bandwidth can be noticed here in Figure 8.4. Here request/response performance without firewall is 2166 transactions/sec, and it decreases up to 746 transactions/sec that is 65% of reduction in performance. While between 5000 rules to 10000 rules reduction of only 40% is observed.

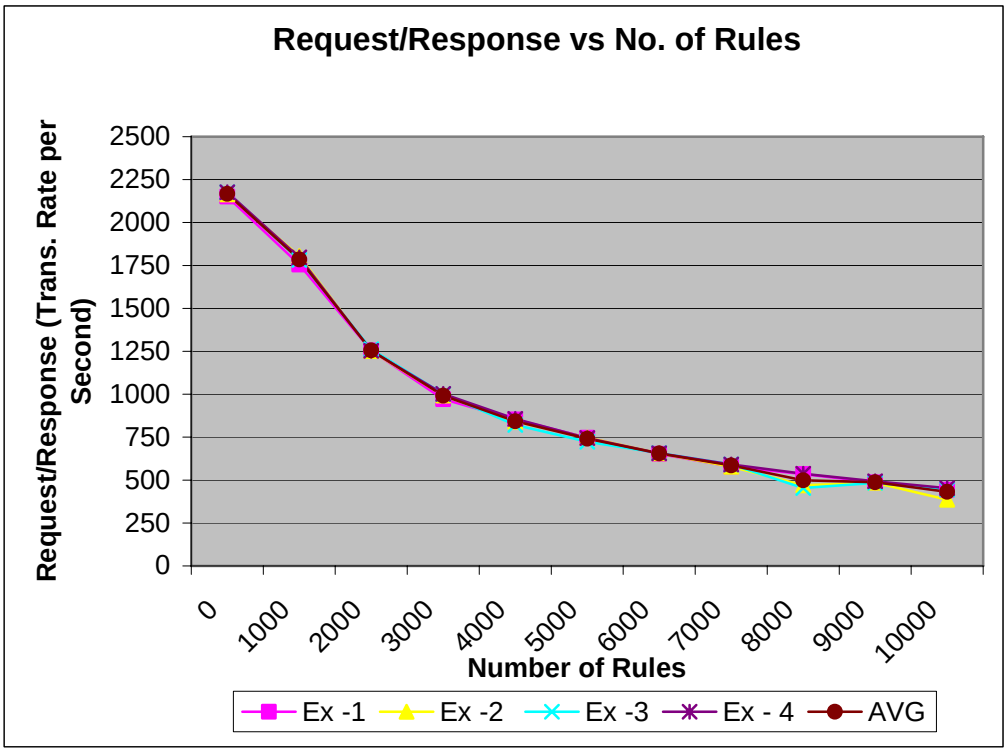

Figure 8.4 Request/Response Performances as Function of Number of Rules

8.4.3 Delay as Function of Number of Rules – Single Firewall

Table 8.3 Delay as Function of Number of Rules – Single Firewall

No of Rules	RTT for 56 bytes packet	RTT for 500 bytes packet	RTT for 1000 bytes packet	RTT for 1500 bytes packet
0	0.546	1.599	2.671	3.787
1000	0.648	1.668	2.794	3.911
2000	0.914	1.905	3.032	4.164
3000	1.101	2.109	3.234	4.367
4000	1.297	2.278	3.403	4.612
5000	1.483	2.469	3.683	5.025
6000	1.654	2.641	3.852	5.377
7000	1.805	2.842	4.004	5.680
8000	1.973	3.007	4.121	6.000
9000	2.148	3.158	4.294	6.364
10000	2.323	3.340	4.457	6.705

**All results are in milliseconds

Graph in Figure 8.5 shows how delay varies according to number of rules in firewall. In this experiment, performance is measured in terms of the delay for different size of packets and varying number of rules in firewall. Experiment is performed for packets having sizes of 56 bytes, 500 bytes, 1000 bytes and 1500

bytes. From the graph, it can be seen that as number of rules increase, delay also increases constantly for the packets of 56 bytes, 500 bytes and 1000 bytes. On the other hand, for large packet sizes like 1500 bytes, we have some constant changes in delay up to 4000 rules, as rules increase from 4000 rules, delay increases more rapidly. Initially, at no rules, delay of 0.546 ms for 56 bytes packet is observed and at 10000 rules, delay of 2.323ms is observed, which is 77% reduction in performance of delay.

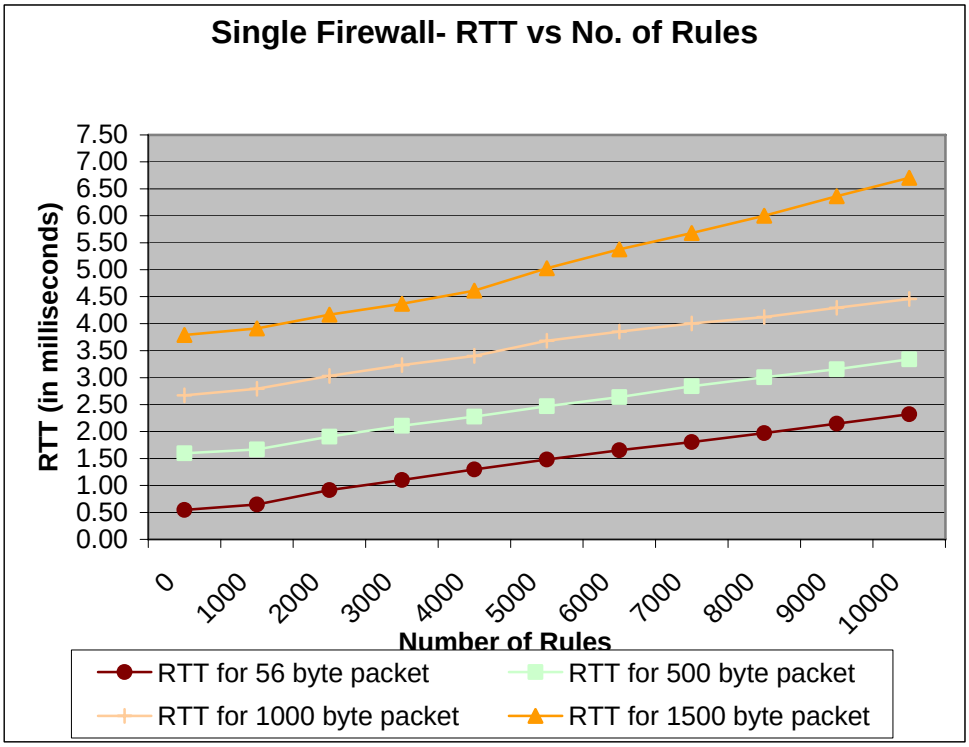

Figure 8.5 Delay as Function of Number of Rules – Single Firewall

8.4.4 Delay as Function of Number of Rules – Distributed Firewall

In this experiment, two-firewall nodes are used in distributed firewall design and one packet distributor. Figure 8.6 shows that by incrementing number of rules in distributed firewall, delay is increased constantly for 56 bytes, 500 bytes and 1000 bytes, while for 1500 bytes packets delay is increased more rapidly after 4000 rules in firewall.

Table 8.4 Delay as Function of Number of Rules – Distributed Firewall

No of Rules	RTT for 56 bytes packet	RTT for 500 bytes packet	RTT for 1000 bytes packet	RTT for 1500 bytes packet
0	0.555	1.613	2.784	3.829
1000	0.673	1.762	2.785	3.981
2000	1.039	1.987	3.023	4.249
3000	1.220	2.263	3.325	6.243
4000	1.968	2.358	3.434	4.706
5000	1.558	2.619	3.679	4.873
6000	1.731	2.737	3.917	5.156
7000	1.929	2.846	3.909	5.492
8000	2.101	3.096	4.251	5.742
9000	2.221	3.181	4.259	6.130
10000	2.510	3.349	4.558	6.498

**All results are in milliseconds

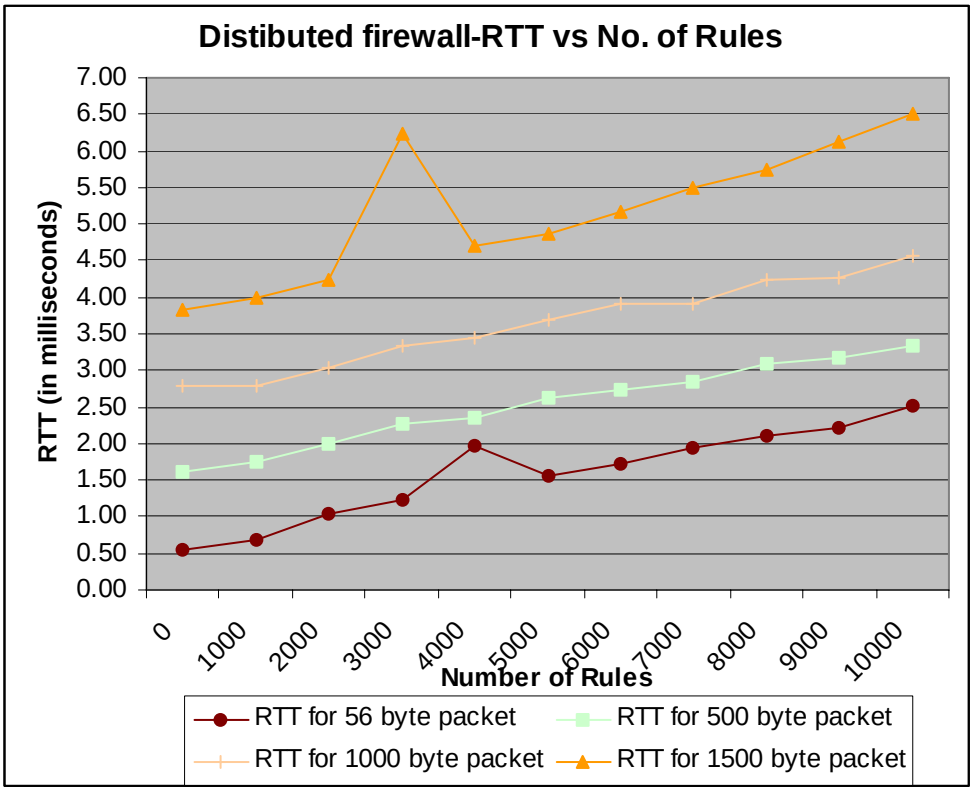

Figure 8.6 Delay as Function of Number of Rules – Distributed Firewall

While comparing the results of single firewall and distributed firewall for delay vs. number of rules, we don't have more difference. As data parallel firewall system duplicates the policy to all the firewall nodes in design, it does not reduce the processing time per packet but just the arrival rate into any firewall node. The performance benefit over traditional firewall models is only evident under high traffic loads. So this design is beneficial to backbones where network bandwidth is of the order of gigabits.

In distributed firewall design, packet distributor is used which introduces additional delay in network path. If any other dedicated embedded device is used that can work as packet distributor, then packet distribution can be done in efficient way and with lower processing delay. There are many such solutions available in market, one of them is Network Taps, which is a hardware solution that accesses dataflow through cables and duplicates that data to the other cables.

9. CONCLUSION AND FUTURE WORK

Firewall is an inevitable element of security. Although it is used to filter the traffic effectively, it proves to be the bottleneck for the network traffic. The problem get worsen when number of rules increases, which is the case of organizations with large and complex networks. Here a simple firewall (with one node) can be thought of with massive resources, but again there are some limitations when it comes to update the resources of single node. Here comes the concept of distributed firewall (with multiple nodes sharing the filtering task of packets). The thesis focused on the study of distributed firewall for performance evaluation in order to improve it. This includes the extensive study on iptables. Policy optimization study is made and three effective methods are described which will have great effect on performance by reducing the number of comparisons to be made for a single packet within rules table. Again, with large and complex network, the management of rules becomes more difficult. We presented firewall policy modeling and set of anomalies to be identified. That becomes the must-to-have toolset for firewall administrator. We could identify that the distributed firewall works more effectively and efficiently even in large and complex networks with thousands of rules implemented by applying aforesaid improvements. In addition, when the number of rules increases up to certain level, the distributed firewall becomes the only feasible option.

At this juncture of the current thesis, looking at the horizons of future work, lot of possible research can be thought of. This includes the distributed firewall study and implementation for real time network traffic like multimedia, by considering QoS parameters. Further study can be made on iptables for policy optimization. Moreover, considering more number of nodes in distributed firewall, the extended study can be made for the same. Performance analysis for connection tracking and NAT is another way towards which study can be made. Similarly, future work can be expected in direction of identifying the effect of VPN on firewalls in terms of network traffic statistics. Although we believe that this project proves to be the milestone in the firewall optimization, looking at the wide field of network security, every study, including this ends by opening new horizons for further research.

REFERENCES

[1] Firewall Architecture, “Understanding the purpose of a firewall when connecting to ADSL network services”. *A Nextep Broadband White Paper, 2001*

[2] Steve Suehring and Robert L. Ziegler*, Linux Firewall Third Edition*, ISBN: 81-7758-964-4

[3] Ryan Joseph Farley; “Parallel Firewall Designs For High-Speed Networks”, Wake Forest University; December 2005

[4] John Wack, Ken Cutler, Jamie Pole; “Guidelines on Firewalls and Firewall Policy”; National Institute of Standards and Technology; January 2002

[5] CERT. Cert/cc statistics 1988-2005, http://www.cert.org/stats/cert_stats.html

[6] Greg Bastien, Christian Abera Degu; *CCSP Cisco PIX Firewall Advanced*; ISBN 81-297-0291-6

[7] Check Point FireWall-1™ Architecture and Administration Ver 4; Check Point Software Technologies Ltd.; Sep-1998

[8] Robert Ellis, “Dynamic Firewall Techniques For Residential Use”, Southern Connecticut State University, Connecticut, February 2004

[9] Ehab S. Al-Shaer and Hazem H. Hamed, “Firewall Policy Advisor For Anomaly Discovery And Rule Editing”, IFIP/IEEE Eighth International Symposium on Integrated Network Management, March 24-28, 2003

[10] Errin W. Fulp. “Optimization of network firewall policies using directed acyclical graphs”. In Proceedings of the IEEE Internet Management Conference (IM’05), 2005

[11] Utz Roedig and Jens Schmitt, **“**Performance Modelling and Evaluation of Firewall Architectures for Multimedia Applications”, Springer Berlin / Heidelberg, Volume 3042/2004, April 08, 2004, pp 38-51

[12] A. A. Hassan, "Algorithms for Verifying Firewall and Router Access Lists",Department of Computer Science and Engineering,Slovak University of Technology

[13] Errin W. Fulp. “Firewall policy models using ordered-sets and directed acyclical graphs”. Technical report,Wake Forest University Computer Science Department, 2004.

[14] Errin W. Fulp, "Firewall Architectures for High-Speed Networks", Department of Computer Science, WAKE FOREST University, September 15-17, 2004

[15] Lucian Gheorghe, *Desing and Implimenting Linux Firewalls and QoS using netfilter, iproute2, NAT and L7 filter*, Packt Publishing, ISBN 1-904811-65-5, October 2006

[16] Netperf, http://www.netperf.org/netperf/

[17] Brian Komar, Ronald Beekelaar, and Joern Wettern, *Firewall For Dummies, 2nd Edition*, Wiley Publishing, Inc., ISBN: 0-7645-4048-3

[18] Firewall Technologies, Rober Zalenski, IEEE Potentials, 2002

[19] http://en.wikipedia.org/wiki/netfilter

[20] Advanced Features of netfilter/iptables, Barry O'Donovan, http://linuxgazette.net/108/odonovan.html

[21]http://www.netfilter.org/documentation/HOWTO/netfilter-extensions-HOWTO.html

[22] http://www.faqs.org/docs/iptables/targets.html