ABSTRACT

FLOW-ORIENTED ANOMALY-BASED DETECTION OF DENIAL OF SERVICE ATTACKS WITH FLOW-CONTROL-ASSISTED MITIGATION

by
Sui Song

Flooding-based distributed denial-of-service (DDoS) attacks present a serious and major threat to the targeted enterprises and hosts. Current protection technologies are still largely inadequate in mitigating such attacks, especially if they are large-scale. In this doctoral dissertation, the Computer Network Management and Control System (CNMCS) is proposed and investigated; it consists of the Flow-based Network Intrusion Detection System (FNIDS), the Flow-based Congestion Control (FCC) System, and the Server Bandwidth Management System (SBMS). These components form a composite defense system intended to protect against DDoS flooding attacks. The system as a whole adopts a flow-oriented and anomaly-based approach to the detection of these attacks, as well as a control-theoretic approach to adjust the flow rate of every link to sustain the high priority flow-rates at their desired level. The results showed that the misclassification rates of FNIDS are low, less than 0.1%, for the investigated DDOS attacks, while the fine-grained service differentiation and resource isolation provided within the FCC comprise a novel and powerful built-in protection mechanism that helps mitigate DDoS attacks.

FLOW-ORIENTED ANOMALY-BASED DETECTION OF DENIAL OF SERVICE ATTACKS WITH FLOW-CONTROL-ASSISTED MITIGATION

by
Sui Song

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Department of Electrical and Computer Engineering

January 2006

APPROVAL PAGE

FLOW-ORIENTED ANOMALY-BASED DETECTION OF DENIAL OF SERVICE ATTACKS WITH FLOW-CONTROL-ASSISTED MITIGATION

Sui Song

Dr. Constantine N. Manikopoulos, Dissertation Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Mengchu Zhou, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Roberto Rojas-Cessa, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Jie Hu, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Zhixiong Chen, Committee Member — Date
Associate Professor, Division of Math and CIS, Mercy College

Dr. Robert Statica, Committee Member — Date
Director, Computer Forensic & Cybersecurity Lab, College of Computer Science, NJIT

BIOGRAPHICAL SKETCH

Author: Sui Song

Degree: Doctor of Philosophy

Date: January 2006

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2006

- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2001

- Master of Science in Electrical Engineering,
  Central South University, Hunan, P. R. China, 1990

- Bachelor of Science in Electrical Engineering,
  Huazhong University, Hubei, P. R. China, 1984

Major: Electrical Engineering

Presentations and Publications:

Sui Song and C.Manikopoulos,
"A Control Theoretical Analysis for Flow-based Congestion Control to Mitigate Distributed Denial of Service Attacks,"
In preparation.

Sui Song and C.Manikopoulos,
"IP Spoofing Detection Algorithms for Flow-based Network Intrusion Detection System,"
Accepted by 2006 IEEE Sarnoff Symposium, 27-28 Princeton, NJ, March 2006.

Sui Song and C.Manikopoulos,
"A Novel Flow-based Network Anomaly Intrusion Detection For Distributed Denial of Services (DOS),"
Submitted to International Journal of Information and Computer Security (IJICS) for special issue on: "Nature-Inspired Computation in Cryptology and Computer Security", 2006.

Sui Song and C.Manikopoulos,
"Flow-based Statistical Aggregation Schemes for Network Anomaly Detection," Accepted by 2006 IEEE International Conference On Networking, Sensing and Control, Ft. Lauderdale, Florida, U.S.A. April 23-25, 2006.

Sui Song ,Dekun Zou,and C.Manikopoulos and Yunqing Shi,
"Steganography Detection of Whole-Image-DCT-Based Watermarking in Gray-Scale Images,"
The Third WSEAS Conference on Neural Networks and Applications, July 7th, 2003.

C.Manikopoulos, Yunqing Shi, Sui Song and Dekun Zou,
"Detection of Block DCT-based Steganography in Gray Scale Images,"
The 2002 IEEE Workshop on Multimedia Signal Processing for the topic Data Hiding and Security, 2002: 355-358;May 24th, 2002.

To my beloved wife, Dianhong Luo and my funny son, Fuhua Song

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my advisor, Dr. Constantine N. Manikopoulos, for giving me the opportunity to work with him and leading me into this exciting field. I am greatly indebted to my advisor for the invaluable guidance and encouragement, which he has provided me through my study. He is always open to new ideas and I really appreciate the freedom he gave me while working on my research. This research could not have been possible without his brilliant ideas. I owe much for his unending help to do the research and for his financial support during my graduate study.

I would like to express my sincere appreciation to Dr. Mengchu Zhou who provided me the precious opportunities to start my Ph.D program and introduced me to my advisor. Special thanks to Dr. Roberto Rojas-Cessa, Dr. Jie Hu, Dr. Zhixiong Chen and Dr. Robert Statica for serving on my committee, reviewing this dissertation and providing valuable suggestions.

All the fellow members of my project team and all my friends in and out of NJIT have been great sources of ideas and fun. I thank them for making my Ph.D study productive and enjoyable.

I am forever indebted to the love and trust of my family. My parents and my wife have always been a source of inspiration, support, advice and happiness without which I would not have been able to go this far. I also thank my son, Fuhua Song, for giving me smiles and the will of life.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES (Continued)

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

1.1 Problem Statement and Motivation

Distributed Denial of Service (DDOS) attacks overwhelm a targeted host or network with an immense volume of malicious traffics from distributed or spoofed sources. In Feb. 2000, a number of the world's largest –e-commerce site was brought offline for days by DDOS attacks, even though they had high security prevention services.

According to the CIAC (Computer Incident Advisory Capability), the first DDoS attacks occurred in the summer of 1999 [1]. In February 2000, one of the first major DDoS attacks was waged against Yahoo.com. This attack kept Yahoo off the Internet for about 2 hours and cost Yahoo a significant loss in advertising revenue [2]. Another recent DDoS attack occurred on October 20, 2002 against the 13 root servers that provide the Domain Name System (DNS) service to Internet users around the world. They translate logical addresses such as www.yahoo.edu into a corresponding physical IP address, so that users can connect to websites through more easily remembered names rather than numbers. If all 13 servers were to go down, there would be disastrous problems accessing the World Wide Web. Although the attack only lasted for an hour and the effects were hardly noticeable to the average Internet user, it caused 7 of the 13 root servers to shut down, demonstrating the vulnerability of the Internet to DDoS attacks [3]. If unchecked, more powerful DDoS attacks could potentially cripple or disable essential Internet services in minutes.

There are various ways of launching a DDOS attacks. Flooding-based Distributed DoS attack, or simply DDoS attack, is another form of DoS attack. They simply exploit the huge resource asymmetry between the Internet and the victim in that a sufficient number of compromised hosts are amassed to send useless packets toward a victim around the same time. The magnitude of the combined traffic is significant enough to jam, or even crash, the victim (system resource exhaustion), or its Internet connection (bandwidth exhaustion), or both, therefore effectively taking the victim off the Internet.

Today, researchers are still struggling to devise an effective solution to the DDoS problems. Although many commercial and research defenses have appeared, none of them provide complete protection from the threat. Rather, they detect a small range of attacks that either use malformed packets or create severe disturbances in the network; and they handle those attacks by non-selectively dropping a portion of the traffic destined for the victim. Clearly this strategy relieves the victim from the high-volume attack, but also inflicts damage to legitimate traffic that is erroneously dropped.

There are two main features of DDoS attacks that severely challenge the design of successful defenses:

(1) IP source address spoofing. Attackers frequently use source address spoofing during the attack — they fake information in the IP source address field in attack packet headers. One benefit attackers receive from IP spoofing is that it is extremely difficult to trace the agent machines. The other advantage that IP spoofing offers to the attackers is the ability to perform reflector attacks. To mitigate Spoofing DDoS attacks, much of the current research focuses on anti-spoofing such as ingress filtering [4], route-based packet filtering [5], and various IP traceback protocols [6], [7]. Their effectiveness often

depends on a universal deployment on the Internet. With a partial deployment, source-address spoofing remains feasible. Even if traceback could be successfully performed in the face of IP spoofing, it is difficult to say what actions could be taken against hundreds or thousands of agent machines. Such a large number prevents any but crude automated responses aimed at stopping attack flows close to the sources. Victim cannot filter out spoofed IP packets, wastes resources.

(2) Similarity of attack to legitimate traffic. Many network based attacks can trigger numerous false positives because of normal traffic looking very close to malicious traffic. Therefore, any type of traffic can be used to perform a successful denial-of-service attack. Some traffic types require a higher attack volume for success than others, and attack packets of different types and contents target different resources. However, if the goal is simply to cripple the victim's operation, sending sufficiently large volumes of any traffic and clogging the victim's network can meet it. Attackers tend to generate legitimate-like packets to perform the attack, obscuring the malicious flow within legitimate traffic. Since malicious packets do not stand out from legitimate ones, it is impossible to sieve legitimate from attack traffic based purely on examination of individual packets. A defense system must keep a volume of statistical data in order to extract transaction semantics from packet flows and thus differentiate some legitimate traffic (e.g. belonging to lengthy well-behaved transactions) from the attack traffic. To perform traffic separation, we introduce the concept "flow" to statistic traffics. An IP flow is a unidirectional series of IP packets of a given protocol, traveling between a source and destination, within a certain period of time. Based on "flow" concept, we developed a flow-based aggregation technique to handles high amounts of similar packet

data and keep many statistics on the dynamics of those structures to detect high-volume or anomalous communications.

These two features create contradictory requirements for DDoS defense. In order to perform accurate traffic separation, the defense system requires a lot of resources for record keeping. Therefore, it can only handle small to moderate traffic volumes. On the other hand, the need to control a large portion of the attack traffic requires placement at points that relay a high traffic volume. Those two requirements can hardly be satisfied at a single deployment point. A majority of DDoS defense systems sacrifice the first goal — traffic separation — to achieve the second goal — control of a large portion of the attack traffic. Those systems are located at or near the victim site, which enables them to detect and control the majority of DDoS attacks, but also places the defense system on the path of high-volume traffic, which impairs its selectiveness.

In summary, Existing technologies are simply not up to the task of protecting against today's DDoS attacks. Passive detection technologies working with static filtering solutions won't work against today's complex, sophisticated attacks; they simply don't offer the dynamic detection and mitigation capabilities required to identify and instantly stop attack traffic to protect mission-critical operations. What's required today is a new type of solution that not only detects the most sophisticated DDoS attacks, but also delivers the ability to block increasingly complex and hard-to-detect attack traffic without impacting legitimate business transactions. Such an approach demands more granular inspection and analysis of attack traffic than today's solutions can provide.

This dissertation studies a Computer Network Management and Control System (CNMCS), which consists of Flow-based Network Intrusion Detection System (FNIDS), Flow Congestion Control (FCC) System, and Firewall/Filtering Modulate and Server Bandwidth Management System. The purpose of the FDPS is to construct a multi-layered defense system to protect the private network from DDoS flooding.

CNMCS has been evaluated by the DARPA'98 data or Conex Testbed data. The CNMCS has demonstrably achieved remarkable result, which will strongly support flow control; the false positive, false negative and misclassification rates found are low, less than 0.1%, for DDOS attacks.

1.2 The Proposed Approach

In this dissertation, a multi-layered defense infrastructure has been proposed to detect attacks, control network traffic flow and protect the server system from DDoS attacks. This multi-layered defense infrastructure integrates Flow-based Network Intrusion detection system (FNIDS), Flow-based Congestion Controller, Firewall and DynaTraX™ Physical link switch together; the figure 1.1 shows this architecture.

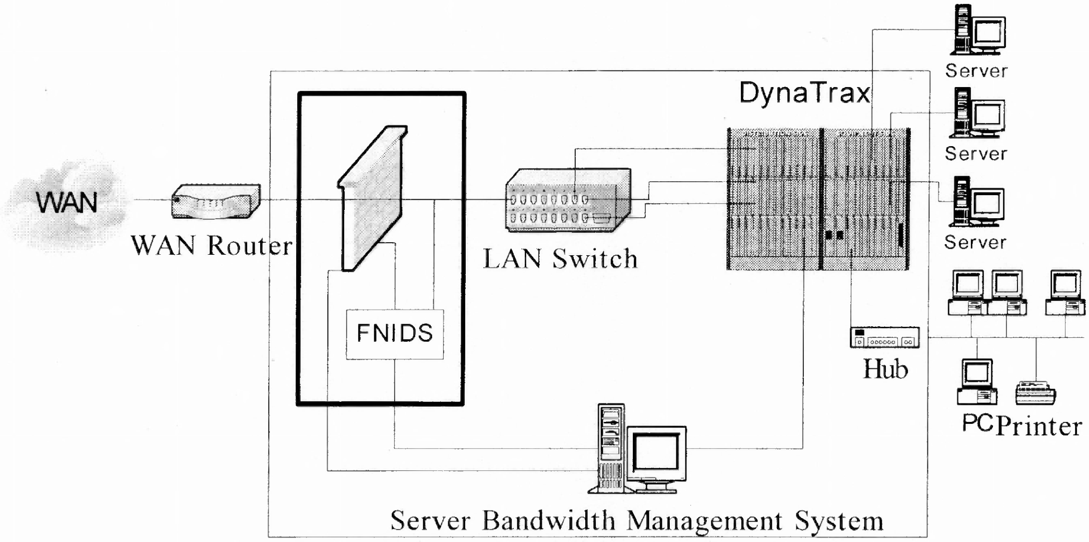


Figure 1.1 The management and control of computer network architecture

The Flow-based Network Intrusion Detection System (FNIDS) is placed behind the firewall to monitor the network status of the whole local area network. A DynaTraX(TM) Physical Link Switch is placed between the LAN switch and the end users to set up physical connections. A Server bandwidth Management System (SBMS) is used to receive the alert information (such as: maliciousness of flow, flow rate etc.) from the FNIDS to decide whether to inform the firewall to blocks malicious flows or drop malicious packets, or to command the DynaTraX™ system to disconnect the overload links to servers.

All of these parts construct a multi-layered defense system described as following:

The first layer of prevention approach is an intelligent firewall, which consists of two distinguishable phases: Flow-based Network Intrusion Detection System (FNIDS) and Firewall/Packet Filtering. The detection part (FNIDS) is responsible for identifying DDoS attacks or attack flow. The firewall is used to drop malicious packets or block malicious clients. After receiving information on packets or flows, the filtering part is responsible for classifying those packets/flows and then dropping them (rate-limiting is another possible action). It is very important to first point out that effective attack detection does not always translate into effective packet/flow filtering. Because of the distributed nature of the attack (DDoS), the detection phase can only use the victim's identities, such as IP address and port number, as the signatures of the attack flows. As a result, packet filtering usually drops attack packets/flows as well as normal packets/flows because both match the signatures (or flow key). As a result, packet filtering does not always help restore the victim's service. Additionally, once faced by large flooding attacks, such as: spoofing attacks, amplifier attacks or reflector attacks etc., the flow number of a flow-based analysis method rapidly increases, which could exhaust a lot of memories and CPU resources and disable the flow-based analysis method. To address these problems, an Adaptive Flow Aggregation Approach (AFAA) is presented, which consists of flow-based aggregation module, Flow Population Density/Distributions Detection Mechanism (FAP2D-DM) and Fuzzy Supervisory Controller. Flow-based aggregation module is responsible for grouping packets based on flow keys, which consist of source IP/Port, destination IP/port and protocol etc. A Flow Population Density/Distributions Detection Mechanism (FAP2D-DM) is proposed to detect how packets aggregate at different levels based on Multi-stage Packet Aggregation Architecture (MPAA), such as: how addresses are aggregating at a given prefix length. Fuzzy Supervisory Controller is to activate the most adaptive flow aggregation scheme for packet aggregates. Therefore, APAA provides a more flexible fine-grained flow aggregation approach and a more effective defense measurement against flooding attacks.

The Second layer of prevention approach is to use feedback control principle to realize network flow control to prevent incoming traffic from exceeding a given threshold, while allowing as much incoming, legitimate traffic as possible and dropping as much malicious traffic as possible. Because current detection of the attack is unreliable and may have high false-positives; rate limiting is a better-suited response than complete filtering. Filtering out all the traffic to the victim would greatly damage misclassified flows, whereas rate limiting still allows some packets to reach the destination and thus keeps connection alive. Allowing some attack packets through is acceptable, since the attack's overall impact depends on the volume of the attack packets. Moreover, if the flow-rate of low-priority is reduced, the high-priority flow will get more chances to access the server they share, which eventually reduce the congestion and improve the throughput of the high-priority flow. This architecture consists of a Fine-grained Quality-of-Service (FQoS) regulator and PID controller. The whole system adopts a control-theoretic approach to adjust the flow rate of every link so as to maintain the high priority flow-rates at their desired level, thus guaranteeing QoS to high-priority flow. The flow-based network intrusion detection is used to classify each flow in the network into different priority classes and give different treatment to the flow-rates belonging to different classes. The architecture is shown to be highly flexible service differentiation and robust against different types of flooding attacks, and traditional network traffic control can be implemented using one common framework. The fine-grained service differentiation and resource isolation provided inside the Flow-based Congestion Control (FCC) is a powerful built-in protection mechanism to mitigate DDoS attacks, reducing the vulnerability of Internet to DDoS attacks.

The last layer is a Server Bandwidth Management System (SBMS), which integrated DynaTraX™, a high-speed digital matrix cross-connect switch, with Flow-based IDS together to thwart the increasing threat posed by cyber terrorists. The DynaTraX™ has the ability to create a critical and meaningful solution to stop hackers from intruding into networks, thereby thwarting cyber terrorists. Especially once FNIDS detects DDOS attacks and some links are overload, the Server Management System will use the fuzzy inference engine to make decision if DynaTraX™ electronically disconnects links and reconnects them to a simulated network port, called "honey pot", within 60-90 nanoseconds that allows you to hold and trace them.

1.3 Key Contributions

This thesis presents a novel multi-layered defense infrastructure, which includes Flow-based Network Intrusion Detection (FNIDS), Flow Congestion Control (FCC) System, Firewall, and Server Bandwidth Management System (SBMS). This thesis makes several key contributions:

(1) It presents a flow-based statistic method in Network Intrusion detection. Efficiently and accurately modeling network behavior is essential for defending attacks. The whole system is based on "flow" concept, we developed a flow-based aggregation technique that dramatically reduces the amount of monitoring data and handles high amounts of statistics and packet data. FNIDS sets up flow-based statistical feature vectors and reports to two parallel detectors: Network Behavior Analyzer and Neural Network Classifier. Network Behavior Analyzer analyzes and visualizes network behavior change. Neural Classifier uses Back-Propagation networks to classify score metric of each flow. Existing Flow-based Network Intrusion Detection Systems (FNIDS) mainly analyze and detect bandwidth type Denial of services attack. By applying up to 22 parameters for each flow, our FNIDS can detect both bandwidth type DOS and protocol type DOS. Moreover, flow here could be any set of packets sharing certain common property as "flow key". FNIDS configures flow flexibly to provide security from network level to application level (IP, TCP, UDP, HTTP, FTP...), and different aggregation schemes, such as server -based, client-based flow.

(2) It presents an Adaptive Flow Aggregation Approach (AFAA). This approach consists of flow-based aggregation module, Flow Population Density/Distributions Detection Mechanism and Fuzzy Supervisory Controller. Flow-based aggregation module is

responsible for grouping packets based on flow keys, which consist of source IP/Port, destination IP/port and protocol etc. A Flow Population Density/Distributions Detection Mechanism (FAP2D-DM) is proposed to detect how packets aggregate at different levels based on Multi-stage Packet Aggregation Architecture (MPAA), such as: how addresses are aggregating at a given prefix length. Expansion and contraction of flow aggregation enable adaptive flow aggregation approach (APAA) to exploit the hierarchical structure of the 5-tuples. Deeper stage or level of the Multi-stage Packet Aggregation Architecture (MPAA) has more fine-grained flow aggregations. The adaptive fine-grained flow aggregations mechanism enables the flow-based defense systems to defense against spoofed flooding attack or amplification attacks. Fuzzy Supervisory Controller is to activate the most adaptive flow aggregation scheme for packet aggregates. Therefore, APAA provides a more flexible fine-grained flow aggregation approach and a more effective defense measurement against flooding attacks.

(3) It presents a control theoretical analysis for flow-based congestion control to mitigate DoS/DDoS attacks. In this thesis, a Flow-rate Congestion Control (FCC) architecture was presented that uses the Flow-based Network Intrusion Detection System (FNIDS) to classify the traffic flows into different priority classes and give different treatment to the flow-rates belonging to different classes, and FCC adopted a control-theoretic approach adaptively to control the low-priority flows so as to maintain the high priority flow-rates at their desired level, thus guaranteeing QoS to high-priority flow. At the same time, It adaptively maximizes low priority flows while maintaining high priority flows at a desired level so that full utilization of network medium can be achieved through adaptive rate control. In this thesis, dynamic network flow model was established, which was

integrated with Fine-grained Quality-of-Service (FQoS) regulator and a PID controller to form a Flow-based Congestion Control (FCC) System. The architecture was shown to be highly flexible and robust against different types of attack patterns, and traditional network traffic control can be implemented using one common framework.

1.4 Roadmap of the Dissertation

This dissertation is organized in the following manner. Chapter 2 presents the philosophy and design of the flow-based Intrusion detection system. It discusses our motivation for the current design and also presents in great detail key components of the system; Chapter 3 provides a highly detailed overview of Adaptive Flow Aggregation Approach (AFAA) for defense system; Chapter 4 presents a control theoretical analysis for flow-based congestion control to mitigate DDoS attacks; Chapter 5 present the server bandwidth management system; We conclude the thesis in Chapter 6.

CHAPTER 2

A FLOW-BASED NETWORK INTRUSION DETECTION FOR DENIAL OF SERVICES

In this chapter, a novel Flow-based Network Intrusion Detection System (FNIDS) is presented. An IP flow is a unidirectional series of IP packets of a given protocol, traveling between a source and destination, within a certain period of time. Based on "flow" concept, a flow-based packet aggregation architecture is developed, which dramatically reduces the amount of monitoring data and handles high amounts of statistics and packet data. FNIDS sets up flow-based statistical feature vectors and reports to two parallel detectors: Network Behavior Analyzer (NBA) and Neural Network Classifier (NNC). Network behavior analyzer analyzes and visualizes network behavior change. Neural network classifier uses back-propagation networks to classify score metric of each flow. Existing flow-based network intrusion detection systems mainly analyze and detect bandwidth type Denial of services attack. By applying up to 22 parameters for each flow, our FNIDS can detect both bandwidth type DOS and protocol type DOS. Moreover, flow here could be any set of packets sharing certain common property as "flow key". FNIDS configures flow flexibly to provide security from network level to application level (IP, TCP, UDP, HTTP, FTP...), and different aggregation schemes, such as server -based, client-based flow. This novel IDS has been evaluated by using DARPA 98 data and CONEX test-bed data. Results show the success in terms of different aggregation schemes for both datasets.

2.1 Introduction

2.1.1 Background

Distributed Denial-of-Service (DDoS) attack presents a very serious threat to the stability of the Internet. In a typical DDoS attack, a large number of compromised hosts are amassed to send useless packets to jam a victim, or its Internet connection, or both. In the last two years, it is discovered that DDoS attack methods and tools are becoming more sophisticated, effective, and also more difficult to trace to the real attackers. Identifying, diagnosing and treating anomalies in a timely fashion are a fundamental part of day-to-day network operations. However, modeling the traffic at the packet level has proven to be very difficult since traffic on a high-speed link is the result of a high level of aggregation of numerous flows. Recently, a new trend has emerged for modeling high-speed Internet traffic at the flow level. Flow aggregation techniques are used to aggregate flows (packets) into one flow with a larger granularity of classification (e.g., from port number to IP address). Aggregated flows have a larger number of packets and longer flow duration that dramatically reduces the amount of monitoring data and handles high amounts of statistics and packet data. Therefore, Internet traffic flow profiling has become a useful technique in the passive measurement and analysis field. The prerequisites for flow-based measurements are now available within the network infrastructure – particularly, in popular Cisco network devices. The integration of this feature has enabled the 'flow' concept to become a valuable method.

Despite a large literature on traffic characterization, traffic anomalies remain poorly understood. There are a number of reasons for this. First, identifying anomalies requires a sophisticated monitoring infrastructure. Unfortunately, most ISPs only collect simple traffic measures, *e.g.,* average traffic volumes (using SNMP). More adventurous ISPs do collect flow counts on edge links, but processing the collected data is a demanding task. A second reason for the lack of understanding of traffic anomalies is that ISPs do not have tools for processing measurements that are fast enough to detect anomalies in real time. Thus, ISPs are typically aware of major events (worms or flooding DoS attacks) after the fact, but are generally not able to detect them while they are in progress.

2.1.2 Motivation

In this chapter, a novel flow-based anomaly network intrusion detection system (FINDS) is presented. An IP *flow* is a unidirectional series of IP packets of a given protocol, traveling between a source and destination, within a certain period of time. FNIDS sets up flow-based statistical feature vectors and reports to two parallel detectors: Network Behavior Analyzer (NBA) and Neural Network Classifier (NNC). Network Behavior Analyzer analyzes and visualizes network behavior change. Neural network classifier uses back-propagation networks to classify score metric of each flow. Existing flow-based network intrusion detection systems mainly analyze and detect bandwidth type Denial of Services (DoS) attack. By applying up to 22 parameters for each flow, our FNIDS can detect both bandwidth depletion DOS and resource depletion DOS. Moreover, flow here could be any set of packets sharing certain common property as "*flow key*". FNIDS configures flow flexibly to provide security from network level to application level (IP, TCP, UDP, HTTP, FTP...), and different aggregation schemes, such as server -based, client-based flow.

The rest of the Chapter is organized as follows: section 2.2 introduces the basic concept of flow and flow aggregation schemes; Section 2.3 describes the system architecture: feature generator, Network Behavior analyzer and neural network classifier. Section 2.4 described the detail of flow-based detector: Network Behavior Analyzer (NBA) and Neural Network Classifier (NNC). Section 2.5 introduces the CONEX Test bed and the attack schemes we simulated. Some experimental results are also reported in that section. Section 2.6 draws some conclusions and outlines future work.

2.2 Flow Management Module

2.2.1 What is Flow?

Strictly speaking, flow is a genetic concept. An IP flow is a set of packets, that are observed in the network within some time period, and that share some common property known as its key, which can be a TCP connection or a UDP stream described by source and destination IP addresses, source and destination port numbers, or the protocol number etc. If we collect and statistic packets from network based on pre-defined flow identifier (or key), there are countless aggregation schemes. The flexible data collection and analysis implementation based on flow make the NIDS have the advantage of greatly reducing the amount of data collected and the features formed by this aggregation schemes can provide detailed network performance information.

For flow-based monitoring, a flow is identified by source-destination addresses, source-destination port numbers and protocol. Thus, the combination of following five fields is used for flow Key:

- Source IP address
- Destination IP address
- Source port number
- Destination port number
- Layer 3 protocol type

2.2.2 Formation of Flow Statistics

The object of flow aggregation is to categorize packets by applying the header fields of a packet. The information relevant for classifying a packet is contained inside the packet in N distinct header fields, denoted H[1],H[2],….,H[N]. For simplify, the fields in our system typically used to classify IP packets are the destination IP address, source IP address, destination port number, source port number, protocol number and protocol flags. The flow key (FK) is denoted by FK=[Destination IP, Source IP, Destination Port, Source IP, Protocol]. Using these fields for classifying IP packets, a flow aggregation scheme specifies a flow key, for example, FK = (192.168.10.120, *, 23, *, TCP), matching traffic addressed to subnet 192.168.10.1 using TCP protocol and destination port 23, which is used for incoming Telnet A firewall may disallow Telnet into its network using a filter to block the flow with this flow key.

The flow key (FK) is an array of N values in program, where H [i] is a specification on i-th header field. The value H [i] specifies what i-th header field of a packet must contain in order for the packet to match the flow key. These specifications often have the following forms: exact match, for example "source address must equal

192.168.10.16"; prefix match, like “destination address must match prefix 192.68.10. * "; Or range match, e.g. “destination port must be in the range 0 to 1023". If specifying Server’s port (destination IP is home net) and protocol, FNIDS can monitor the specific services, such as: using TCP protocol and server’s port 23, which is used for incoming Telnet. Therefore, FNIDS configures flow flexibly to provide security from network level to application level (IP, TCP, UDP, HTTP, FTP, Telnet...).

In conclusion, a packet is said to match a flow Key (FK) if each field of the packet matches the corresponding field of FK. For instance, let FK = (192.168.10.* , *, *, 23,TCP) be a flow key, then, a packet with header(192.168.10.120, 10.10.10.1, TCP, 23, 1025) matches FK, and is therefore aggregated. The packet (192.168.10.120, 10.10.10.1, 21, 1024, TCP), on the other hand, doesn't match.

2.2.3 Flow Key Lifetimes

FNIDS operates by creating Flow-Keys that contain the information for all active flows. The Flow Key is built by processing the first packet of a flow through the standard switching path. A Flow record is maintained within the flow buffers for all active flows. Each flow record in the flow buffers contains key fields that can be later used for exporting data to a collection device. Identifying packets with similar flow characteristics and counting or tracking the packets and bytes per flow create each flow record. Probes collect the flow details or buffer information periodically.

The key to flow buffer is highly intelligent flow buffer management, especially for densely populated flow features in a time window. The flow management module contains a highly sophisticated set of algorithms for efficiently determining if a packet is part of an existing flow or should generate a new flow key. The algorithms are also capable of dynamically updating per-flow accounting measurements rules for expiring Flow keys include:

- Flows that have been idle for a specified time are expired and removed from the buffer.
- Long-lived flows are expired and removed from the buffer.

2.2.4 Flow Aggregation Schemes

Flow aggregation schemes are a series of flow-aggregated methods in that it segregates the network traffic into a series of non-overlapping consecutive time windows and aggregate all features based on their flow Keys. Flow keys are a set of values that determine how a flow is identified. The flow key determines the formation of flow statistic.

Typically in our system the flow keys are a fixed 5 fields of packet header, which are defined as <Source IP Address, Destination IP Address, Source Port, Destination Port, Protocol>. The Flow aggregation schemes have the capability to define Flow Mask including Field Mask and Prefix Mask. They are a predefined set of flow key values.

2.2.4.1 Field Aggregation Schemes

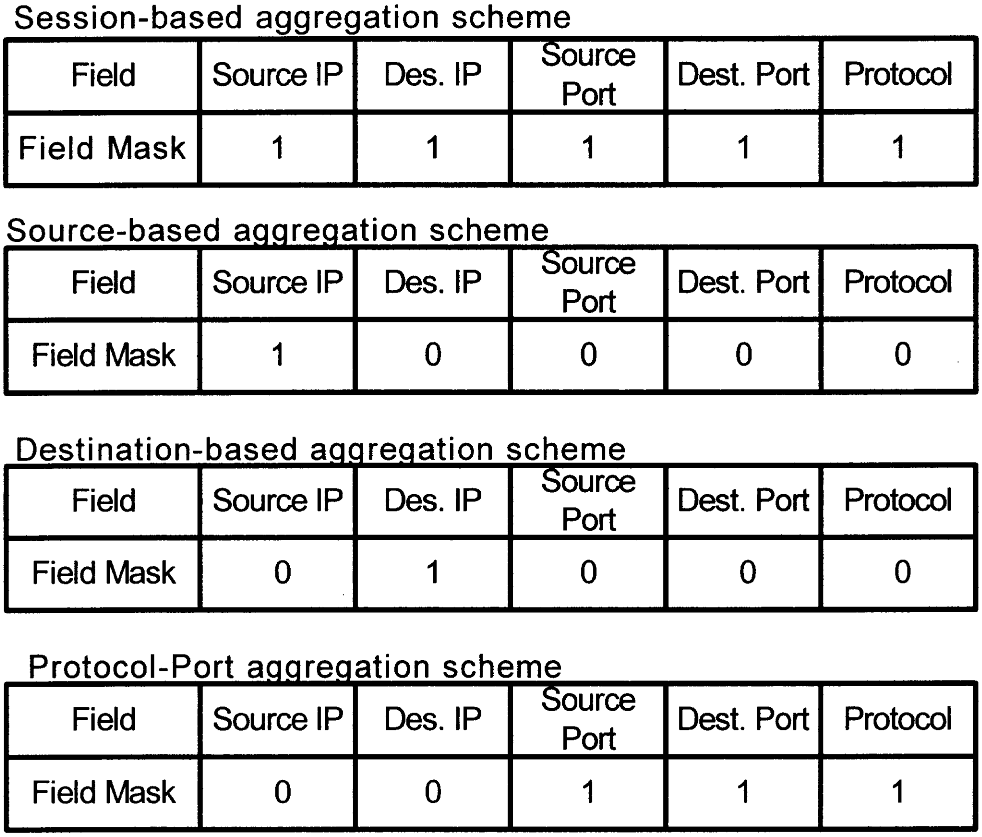
Field aggregation schemes segregates aggregate all features based on their flow Key's fields. In order to form various field aggregation schemes, Field Mask (FM) is defined in this system to mask the filed of flow key, such as Source IP or Port etc. The field mask format is defined in the following table 2.1.

Table 2.1 The Field Mask Format

Field	Source IP	Dest. IP	Source Port	Dest. Port	Protocol
Field Mask	1/0	1/0	1/0	1/0	1/0

Where: '1' means valid and '0' means disable.

To coordinate flow aggregation with different field mask, Field aggregation schemes determine the fields from which you want to collect data. Four typical flow aggregation schemes are showed in Figure 2.1, which also shows which fields are valid for the different aggregation schemes and which fields are parts of the keys. Key fields define a unique flow.



Session-based aggregation scheme

Field	Source IP	Des. IP	Source Port	Dest. Port	Protocol
Field Mask	1	1	1	1	1

Source-based aggregation scheme

Field	Source IP	Des. IP	Source Port	Dest. Port	Protocol
Field Mask	1	0	0	0	0

Destination-based aggregation scheme

Field	Source IP	Des. IP	Source Port	Dest. Port	Protocol
Field Mask	0	1	0	0	0

Protocol-Port aggregation scheme

Field	Source IP	Des. IP	Source Port	Dest. Port	Protocol
Field Mask	0	0	1	1	1

Figure 2.1 The examples of typical flow field aggregation

In figure 2.1, the session-based aggregation scheme has the features aggregated based on the 5 fields of {source IP, destination IP, and source port, and destination port, protocol}. The destination-based aggregation scheme uses a simple scenario way to aggregate flows according to their Destination IP. This algorithm can reduce the number of keeping the flow information in system buffer. The source-based aggregation scheme is similar to Destination-Based aggregation scheme. The source-based aggregation scheme has the flow-based features aggregated by their source IP.

In order to specify some certain field aggregations, we can use some rules in this system. For example: In order to specially monitor and analyze network traffics to protected servers, we can use client-based aggregation scheme as follows:

```
Input:
  S: Sequence of monitored network traffic
  SIP:Homenet IPs
Output: Flow-based aggregation scheme
Begin For each s in S do
        Find IP of s from SIP
          If IP is not found then
            source-based traffic aggregates
          Else
            destination-based traffic aggregates
          Endif
        EndFor
End
```

Figure 2.2 Client-base aggregation scheme

Similarly, we can use the same way to form server-based NIDS in order to monitor traffics to servers.

2.2.4.2 Prefix Aggregation Scheme

We use CIDR notation for prefixes and aggregates. Given an IP address 'a' and prefix length 'p', with $0\leq p\leq 32$, "a/p" refers to the p-bit prefix of 'a' or, equivalently, the aggregate containing all addresses sharing that prefix. An aggregate with prefix length p is called a p-aggregate, or, sometimes, a "/p". A p-aggregate contains 2^{32-p} addresses, so aggregates with short prefix lengths contain more addresses; the single 0-aggregate contains all addresses and a 32-aggregate is equivalent to a single address.

In Prefix Aggregation scheme, prefix lengths are separately defined as five values, which are 0,8,16,24 and 36. The'0' represent all bits are valid and the '32' means all bits are masked. Thus there are only five prefix mask forms defined in Prefix Aggregation Scheme, which are corresponding to the five prefix mask levels. Shown below are examples of used Prefix Aggregation Scheme notation of IPv4 addresses.

0-aggregate: 192.168.10.2/0 ⇒192.168.10.2
8-aggregate: 192.168.10.2/8 ⇒192.168.10.*
16-aggregate: 192.168.10.2/16⇒192.168.*.*
24-aggregate: 192.168.10.2/24⇒192.*.*.*
32-aggregate: 192.168.10.2/32⇒*.*.**

Therefore, there are the following two kinds of prefix aggregation schemes:

- Destination-prefix aggregation
- Source-prefix aggregation

2.2.4.3 Flow Aggregation Schemes

Flow aggregation scheme is to joint field aggregation scheme and prefix aggregation scheme together and group data flows with the same field, source prefix, destination prefix, source prefix mask, and destination prefix.

For example, FK = (192.168.10.*, * , 23, * , TCP), matching traffic addressed to source IP prefix 192.168.10.*.and using TCP protocol and destination port 23, which is used for incoming Telnet A firewall may disallow Telnet into its network using a filter to block the flow with this flow key.

For example: In Unix system, we use "iptables" to drop all packets which match the FK = (192.168.10. *, *, 23, *, TCP) as follows:

```
iptables  - A FORWARD –i eth0 –o eth1   -s 192.168.10.* -sport 23 –p TCP –j drop
```

This command line means drop all packets matching the FK from network card eth0 to network card eth1.

2.2.4.4 Packet Count Distribution of Flow Aggregations

Previous traffic studies [11][21][22] have demonstrated the self-similarity of network traffic, which show noticeable bursts at a wide range of time scales, the lengths of bursts in network traffic and the sizes of files in some systems.

Aggregation population distribution in address domain [21] provides a more fine-grained measurement of how packets are aggregated at a prefix length. Besides prefix aggregation schemes can be used to aggregate traffic at a given prefix length, we presented field aggregation schemes for traffic aggregation .In this section, we further demonstrate that Field Aggregation Population Distribution also has self-similarity in some cases, which provides a more fine-grained measurement of how packets are aggregated at a given field mask.

The packet count of an aggregate is the number of active packet of flows contained in that aggregate. Two or more aggregation schemes have similar population distribution, meaning they can express the same or similar characteristic of network traffic. It is very important. Especially under DDoS flooding attack, FNIDS using session-based aggregation scheme would form a lot of flow records, which take an amount of memories and CPU processing time. If using grass-grained aggregation scheme, which has the similar characteristic with session-based aggregation, such as Client-based aggregation scheme, the requirement of memories and CPU process time will drop greatly, and at the same time, it can detect attacks with the similar misclassification rate.

Log-log complementary CDF graphs form a well-known test for heavy-tailed or power-law tail, distribution. In order to demonstrate that flow aggregation population distribution has self-similarity, we obtained 24-hour long trace from New Jersey Bergen community college, and sampling ratio is 1/24.

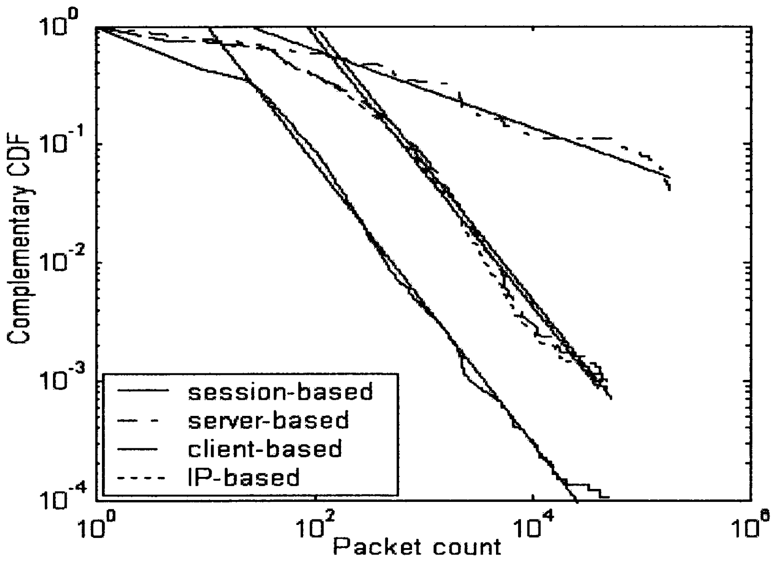


Figure 2.3 Log-log complementary CDF of packet counts

Figure 2.3 presents a log-log complementary CDF of the packet counts of flows. All four distributions appear to have power law tails. Here, α is approximately 0.9179 for session-based, 0.3226 for server-based, 0.9869 for client-based and 0.9711 for IP-base aggregation. These values were calculated by polynomial curve fitting of MATLAB.

2.3 Flow-based Network Intrusion Detection System Architecture

Flow-based Network Intrusion Detection System (FNIDS) is a layered architecture, which include two main parts: feature generator and flow-based detector, shown in Figure 2.4.

The feature generator is similar to Cisco's NetFlow. But FNIDS can provide rich features (more than 22) to be selected. These features represent network states. Thus, they can be used in anomaly analysis or in signature analysis. And the traffic rate of flows can be used for traffic control.

The flow-based detector has two kinds of detection methods: anomaly and signature. In this paper, we will introduce anomaly method. In anomaly method, we will present two methods to detect DDoS: Neural Network Classifier and Network Behavior Analyzer. Network behavior analyzer analyses and visualizes network behavior change. Neural network classifier uses back-propagation networks to classify score metric of each flow. Existing Flow-based Network Intrusion Detection Systems (FNIDS) mainly analyze and detect bandwidth depletion Denial of services attack. By applying up to 22 parameters for each flow, our FNIDS can detect both bandwidth depletion DOS and resource depletion DOS. Moreover, flow here could be any set of packets sharing certain common property as “flow key”. FNIDS configures flow flexibly to provide security from network level to application level (IP, TCP, UDP, HTTP, FTP...), and different aggregation schemes, such as server -based, client-based flow.

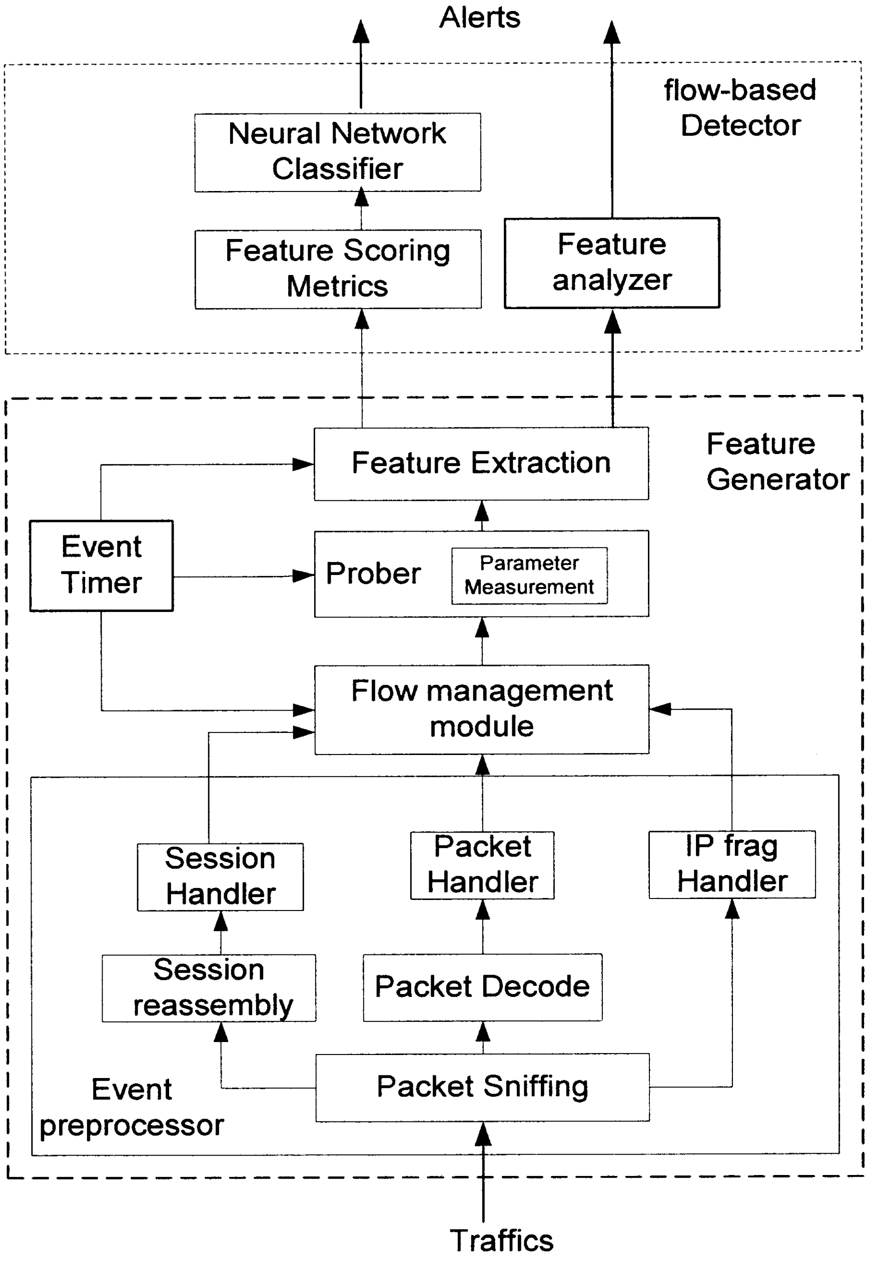


Figure 2.4 The flow-based network intrusion detection system architecture

Event Preprocessor: Collects the network traffic of a host or a network, Event handlers generate reports to Flow management module;

Event Time: Periodically calls Feature Extraction Module to converts the statistic information of flows into the format required by the statistical model.

Flow Management Module: efficiently determines if a packet is part of an existing flow or should generate a new flow key; According to different flow key, this module aggregates flows together based on their flow keys, and dynamically updates per-flow accounting measurements;

Probe: Collects the network traffic, abstracts the traffic into a set of statistical parameters to reflect the network status.

Feature Extraction: Periodically extracts the features, which describe the behaviors of flows.

Feature Scoring Metric: Calculates the probability scores of these features by comparing the features with the reference model generated by past normal and attack users. The probability scores are measurements indicating how likely for a feature to take the observed value.

Neural Network classifier: Classifies the score vectors to prioritize flows with maliciousness. The higher maliciousness of a flow means the flow has the higher possibility of attacker.

Feature Analyzer: Identify the attack by detecting abrupt network behavior changes.

2.4 Flow-based Detector

As described in section 2.3, FNIDS includes two classifiers to detect anomaly attacks: Network Behavior Analyzer (NBA) and Neural Network Classifier (NNC). NBA is a simple and fast response to behavior change of total network traffic, which uses similar principle as in [13] [16] [17]. NBA helps administrator to monitor traffic state of whole network. Moreover, by providing an efficient and effective feature set, our NBA can analyze non-flooding attacks as well as flooding attacks. NNC is an advanced anomaly intrusion detection system, which scores each flow according to reference model, uses neural network to classify each flow. All alerts sent by NND contain IP addresses, which firewalls can use to block malicious flows.

In this section, we first discuss feature set, which is using by both NBA and NND, and then discuss these two detectors in detail.

2.4.1 Statistical Features for Flow-based NIDS

Detecting intrusions involves the multi-variants classifications based on the monitored features. Generally, intrusion detection systems are designed to use as many features as the designers could think they might be useful. For example, JAM [18], an intrusion detection system prototyped by Columbia University, monitors 41 different parameters for each session. In our system, FNIDS provided 31 features, which are relevant in DoS attack detection [15]. The table 2.2 lists 22 selected features.

Table 2.2 The Features in Set$_{22}$

Index	Name	Index	Name
1	in.ip-pkt-rate	12	in.tcp-con-new-aborted
2	in.ip-byt-rate	13	in.tcp-con-half-opened-ratio
3	in.ip-frag-rate	14	in.tcp-con-duration
4	in.ip-defrag-error-rate	15	in.tcp-con-diff-src
5	in.ip-csum-error-rate	16	in.tcp-con-diff-dst
6	in.tcp-pkt-len	17	in.tcp-con-anomalous-entropy
7	in.tcp-pkt-rate	18	in.icmp-pkt-rate
8	in.tcp-syn-pkt-rate	19	in.icmp-byt-rate
9	in.tcp-rst-pkt-rate	20	in.icmp-diff-src
10	in.tcp-con-new-opened	21	in.icmp-diff-dst
11	in.tcp-con-new-closed	22	io.icmp-anomalous-echo-reply

2.4.2 Network Behavior Analyzer (NBA)

FNIDS produces a variety of graphs providing a different view of network behaviors. For example, flow-per-second graphs, packet-per-second and byte-per-second bandwidth utilization graphs. Also, configuration of FNIDS can aggregate counters into larger granularities over longer times. For instance, samples are combined into 30 seconds, 3 minute samples, three hour samples, and finally into 24 hour samples, facilitating both short-term and long-term analysis. These graphs will help administrators fast to understand the whole network states.

Moreover, flow-based aggregation schemes deal with not only the variation of the number of flows, but also with the changes in the distribution of flows. An analysis of the flow-based features could be effective for revealing flood types of attacks. For example:

when a flow is defined as the triple of source address, destination address and destination port, the flood-based attacks spread flows over the destination IP addresses (or ports) in random or dictionary mode style attacks.

In our statistical approach, the abnormality in feature vectors is determined by detecting abrupt changes in their statistics. Our approach is based on the following:

Sudden changes in packet counts, especially when based on flow aggregation scheme, are usually indications of a flood as well.

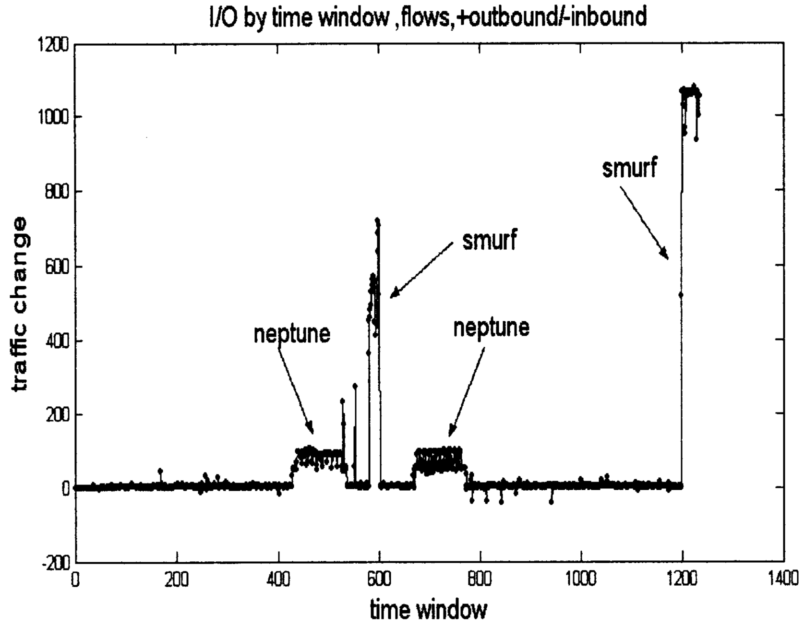


Figure 2.5 Traffic change due to ICMP and SYN flooding using time window

Figure 2.5 is an example graph based on flow counts representing a flood of traffic. Its time-window was set at 30 seconds. The experience showed that a discrepancy between the number of incoming and outgoing flows or packets is a possible indicator of abusive traffic. Specially, in our experience with Flow-based NIDS, we have learned that a discrepancy between the number of inbound and outbound flows or traffics is an indication of DOS flood attacks, such as: Neptune or Smurf.

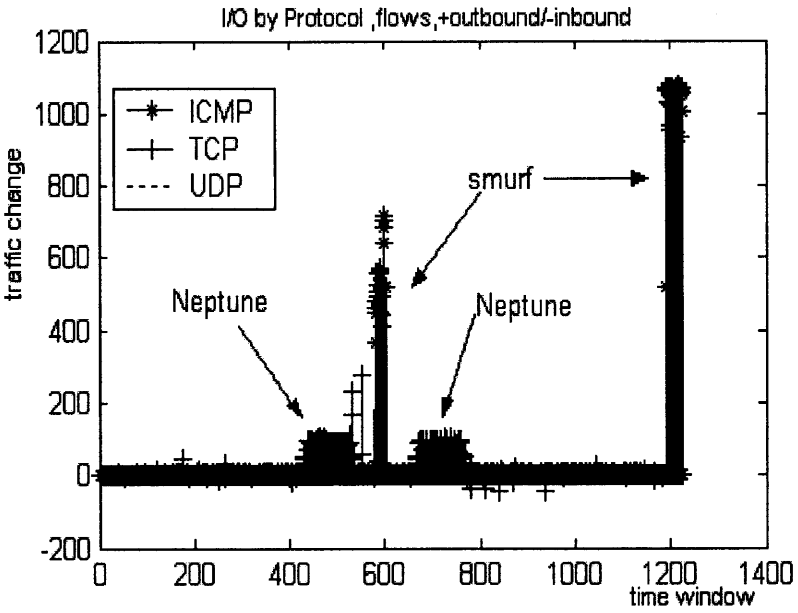


Figure 2.6 Traffic change due to ICMP and SYN flooding using protocol aggregation

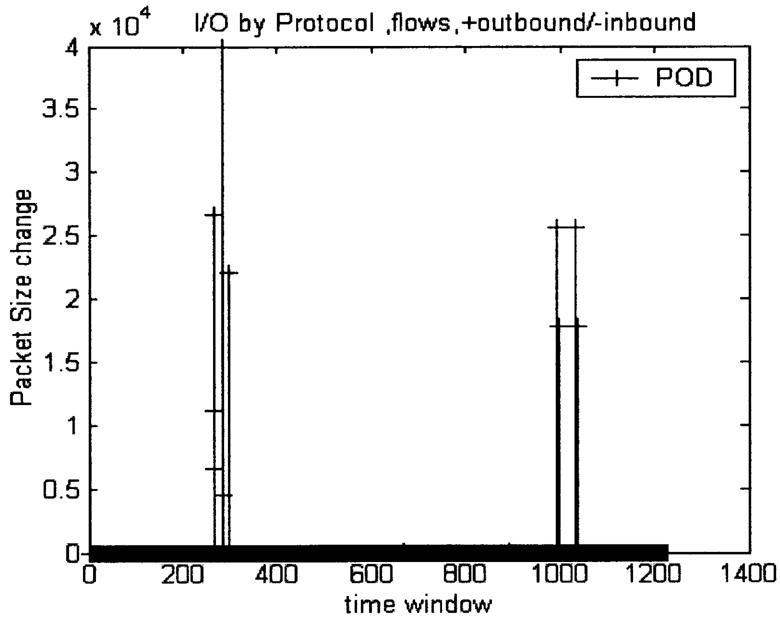


Figure 2.7 Packet size change due to POD attacks using protocol aggregation

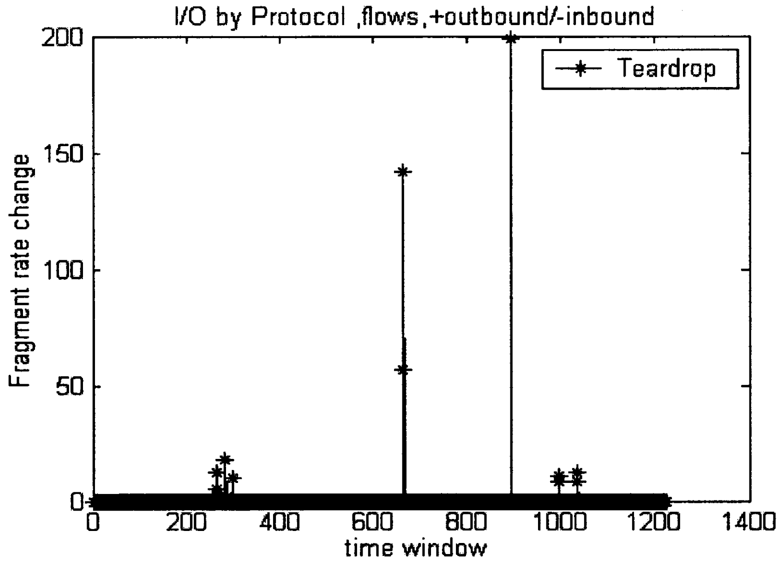


Figure 2.8 Fragment rate change due to teardrop attacks using protocol aggregation

Figure 2.6 is an example based on IP traffic rate representing two kinds of floods. The x-axis of the figure is time window, in this experiment, it was set as 30seconds/time window; the y-axis of the figure is traffic change, which shows the discrepancy between the number of inbound and outbound flows or traffics. The figure 2.6 clearly shows when the flooding attacks (Smurf and Neptune) happened. Figure 2.7 is an example graph based on flow counts representing Pod attacks. Figure 2.8 is an example graph based on IP fragment rate representing 'Teardrop' Attacks.

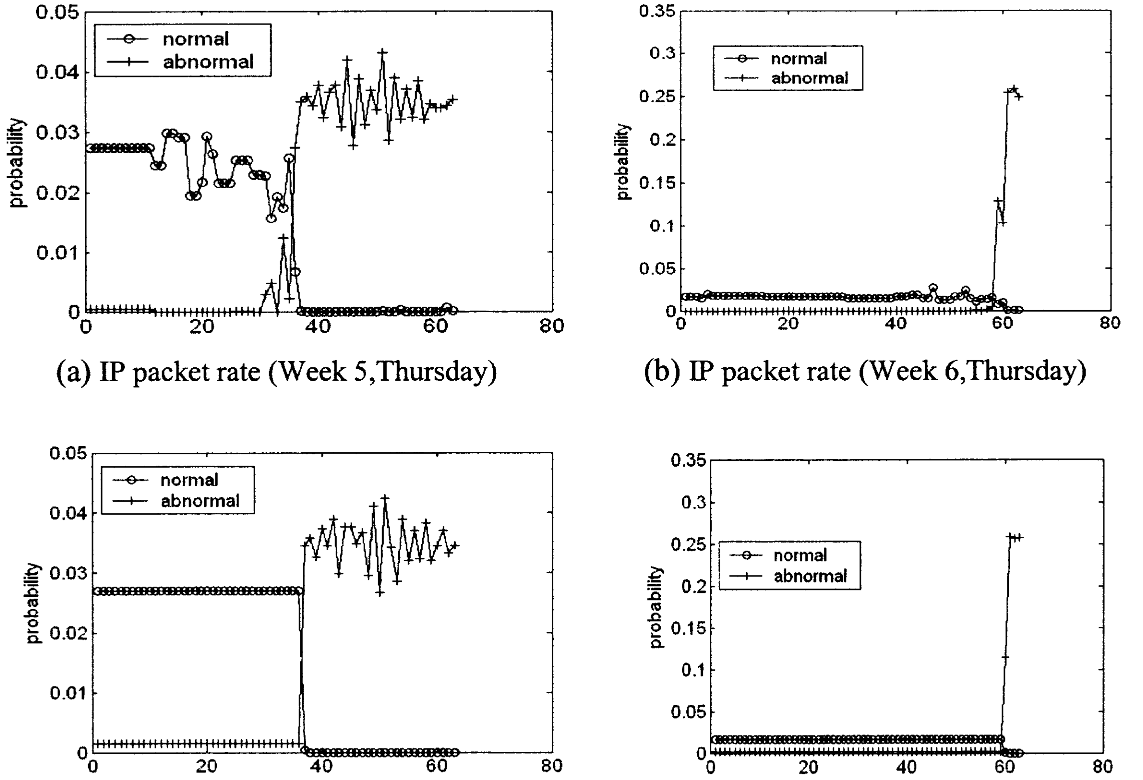
In summary, Network Behavior Analyzer (NBA) identifies and classifies DDOS attacks, viruses and worms in real-time. Changes in network behavior indicate anomalies that were clearly demonstrated in FNIDS data. Although network behavior analyzer can provide network state abnormal changes and detect attacks accordingly, it couldn't show provide details of attacks such as source IP, destination IP. That's why we need a more advanced detector as followed.

2.5 Neural Network Classifier

Neural network classifier extracts flow information from network sniffer, and attempts to detect the presence of anomalies in each flow. Statistics have been used in anomaly intrusion detection systems [16] [17], however most of these systems simply measure the means and the variances of some variables and detect whether certain thresholds are exceeded. SRI's NIDES [18][16] developed a more sophisticated statistical algorithm by using χ^2-like test to measure the similarity between short-term and long-term profiles. Here, neural network classifier maintains reference model of flow statistics features (22 features are using as in section 2.4.1) during training stage, calculates similarity metrics for scoring distance of current flow from reference model, uses back-propagation neural network for classifying each flow as normal or abnormal.

2.5.1 Reference Models

The extracted flow features are then scored based on the information of the reference models. The reference models are in fact probability density functions (PDF) of the feature values. Figure 2.9 showed the measured normal-type and attack-type PDF samples of different features, which were based on Darpa 98 data of the 5th and 6th Thursday, including DoS/DDoS, like 'POD', 'Teardrop',' Neptune' and 'Smurf'. Since the actual legitimate flow distribution during attack is unknown, the approach needs to establish a reference profile of normal users and the reference model of attackers from the past traffic. Just only off-line reference model is not enough. We ensured that the reference model would be updated actively in each time window. While attack packets arrive continuously and the true flow profile changes as new packets arrive, the reference profile is updated and flow is scored at the same time.



(a) IP packet rate (Week 5,Thursday)

(b) IP packet rate (Week 6,Thursday)

(c) ICMP packet rate (Week 5, Thursday)

(d) ICMP packet rate (Week 6, Thursday)

Figure 2.9 The PDF of various packet rates based on Darpa 98 data set

Evidently, as seen from the graphs, there is some overlap between the normal-type PDFs and the abnormal-type PDFs. This means that single parameter threshold classification is error prone. However, it is most significant that, in general, the normal-type and the abnormal-type PDFs are very different from each other at some features. This means that statistical methods that capitalize on these differences can be very effective, especially, if many or all of the features were utilized in unison in arriving at the classification decision. This is exactly how the technique used here and the resulting tool, termed Flow-based Network IDS Intrusion Detection System (FNIDS), achieves its high rate of success.

2.5.2 Score Metrics

There are many statistic metrics used in NIDS. Our current statistical model uses Single Number Statistics (SNS). Let x be the feature value, $p_n(x)$ be the probability of x in the normal reference model, $p_a(x)$ be the probability of x in the attack reference model, $p_{n,max}$ be the maximum probability of the normal reference model, and $p_{a,max}$ be the maximum probability of the attack reference model. The distance of the SNS is defined as the following metrics:

Single Number Statistics version 1(SNS1):

$$S(x) = 2\frac{P_n(x)}{P_{n,\max}} - 1 \tag{2.1}$$

Single Number Statistics version 2(SNS2):

$$S(x) = \frac{P_n(x)}{P_{n,\max}} - \frac{P_a(x)}{P_{a,\max}} \tag{2.2}$$

Single Number Statistics version 3(SNS3):

$$S(x) = 1 - 2\frac{P_a(x)}{P_{a,\max}} \tag{2.3}$$

Where: S is between [-1,1]. S =1 means 'normal'; and S= -1 means 'abnormal'

From the above equations, it can be seen that metric 1 is an anomaly detection approach by comparing feature values with normal reference model. The metric 3 is a normal detection approach by comparing feature value with abnormal reference model.

The metric 2 is a hybrid signature-anomaly detection approach by utilizing both known normal and known attack knowledge. Apparently, the Metric 1 will be more robust to detect new attacks; especially it can identify the low-rate attack without retraining neural network, because the reference model of this way is only based on normal packets.

There are evidently, two kinds of feature vectors here, the normal N-type, labeled as +1, and the malicious M-type, labeled as –1. These labeled N and M vectors can be used for training (2/3 of the total number) and validating (1/3 of the total number) the classifier. In this work, a neural network classifier was employed.

Table 2.3 The Performance of Difference Metrics on CONEX Testbed Data using Session-based Aggregation Scheme

	SNS1	SNS2	SNS3
Number of samples	100701	100701	100701
Number of normal samples	99429	99429	99429
Number of attack samples	1272	1272	1272
Number of False Positives	12	100	363
Number of False Negatives	0	0	0
Misclassification Rate	0.000119165	0.000993039	0.00360473
False Positive Rate	0.000120689	0.00100574	0.00365085
False Negative Rate	0	0	0

In our experiments, we compared these metrics. Form Table 2.3, the results showed that all these metrics can detect Dos. But the Table 2.3 shows that SNS2 and SNS3 have high false positive. Since SNS1 has a good performance, and it is only based on normal behaviors, it will be used in our following discussion.

2.5.3 Neural Network Architecture

Neural network classifiers have been widely considered as an efficient approach to classify challenging patterns. However, here, statistical preprocessing has generated easily discernible and distinguishable normal and abnormal patterns. A 2-layered neural network was built with 22 inputs and one output unit.

The back-propagation neural network classifier [23] (see Figure 2.10) was used to train feature-scoring vectors by modifying the weights of inputs. Each network utilized sigmoid neurons and used on all 22 predictor variables. Again, this was done 1000 times for each random split of the data set .The Back-Propagation Neural Networks was used to evaluate each flow with maliciousness.

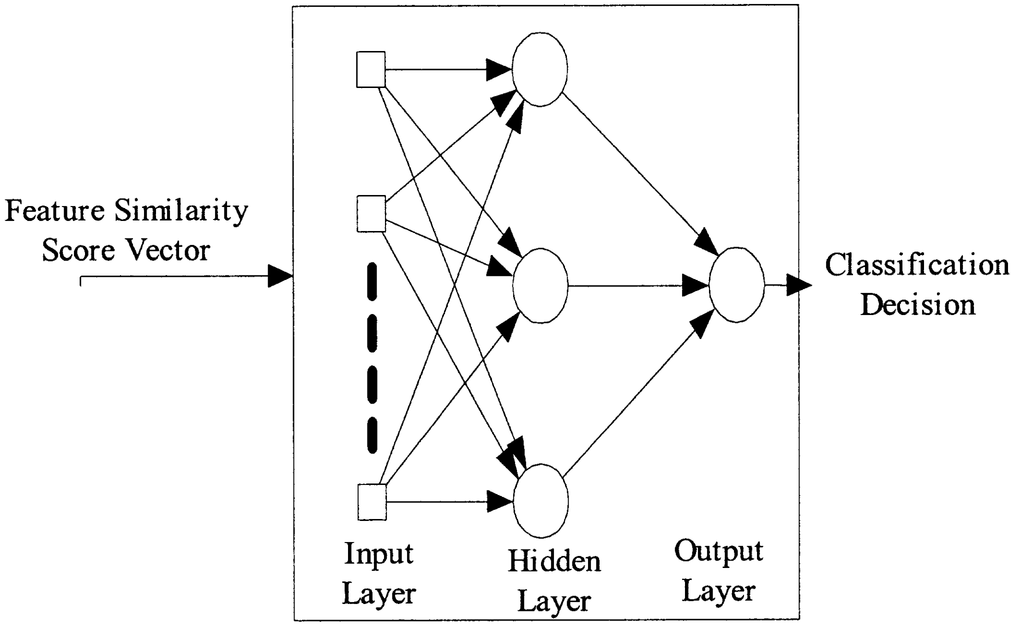


Figure 2.10 Back-propagation classifier

2.6 Evaluation and Experimental Results

Flow-based Network Intrusion Detection System was evaluated by using both 1998 DARPA intrusion detection evaluation data [19] and to CONEX Testbed data. The

DARPA’98 data contains both training data and test data. The training data consists of 7 weeks of labeled network-based attacks inserted in the normal background data. The test data contained 2 weeks of network-based attacks and normal background data. The data contains two main categories of DoS/DDoS attacks: bandwidth depletion, such as: smurf, and resource depletion attacks, such as: Neptune, Ping of Death and Teardrop. A bandwidth depletion attack is designed to flood the victim network with unwanted traffic that prevents legitimate traffic from reaching the (primary) victim system. A resource depletion attack is an attack that is designed to tie up the resources of a victim system. This type of attack targets a server or process on the victim system making it unable to process legitimate requests for service.

DARPA’98 evaluation represents a significant contribution to the field of intrusion detection, there are many unresolved issues associated with its design and execution. In his critique, McHugh [20] questioned a number of results of DARPA’98 evaluation, starting from usage of synthetic simulated data for the background (normal data) and using attacks implemented via scripts and programs collected from a variety of sources. In addition, it is known that the background data contains none of the background noise (packet storms, strange fragments, etc.) that characterizes real data. DARPA’98 data are used here as benchmark of comparison performance with other Intrusion Detection System. Moreover, in order to assess the performance of our anomaly detection algorithms in a live network, we also implement FNIDS in CONEX test-bed. The CONEX TESTBED network is a test-bed network setup in the CONEX lab of NJIT as a platform to launch network-based attacks and real background traffic (HTTP, FTP, SMTP...), for evaluation of intrusion detection prototypes.

The performance of any network intrusion detection system must account for both the detection ability and the false positive rate. Here, we used receiver operating characteristic (ROC) curves to compare intrusion detection ability to false positive. The ROC curve allows us to assess the trade-off between detection ability and false alarm rate in order to properly tune the system for acceptable tolerances. For each curve, the point at the upper left corner represents the optimal detection with high detection rate and low false alarm rate. The x-axis of the figure is the false alarm rate, which is the rate of the typical traffic events being classified as faults or anomalies; the y-axis of the figure is the detection rate, which is calculated as the ratio between the numbers of correctly detected faults/anomalies to their total number.

2.6.1 Anomaly Detection Results on DARPA'98 Data

In order to perform our evaluation of the systems, we applied the above typical aggregation schemes to the data set constructed from DARPA'98 data [19]. According to DoS attack types (Bandwidth and resource depletion attacks) and spoofing situation, the system was evaluated by the following four cases: case 1 is to detect resource depletion attacks; case 2 is to detect bandwidth attacks without spoofing flooding attacks; case 3 is to detect both resource depletion attacks and bandwidth attacks without spoofing flooding attacks; case 4 is to detect both resource depletion attacks and bandwidth attacks with spoofing flooding attacks.

Since the amount of available DARPA'98 data is huge (e.g. some days have several millions of flow records), FNIDS sampled sequences of normal flow records in order to create the normal data set that had the same distribution as the original data set of normal flow. The first 5 weeks of training data was used for training Neural Network Detector. When using session-based aggregation scheme, there were a lot of normal flow records (347747 flow records), 100000 normal flow records were randomly sampled for the training phase. In other flow-based aggregation schemes, all flow records were used for the training phase. The flow records associated with all the attacks from the last two weeks of data were used for testing in order to determine detection rate.

CASE 1: Resource Depletion Attacks

Our purpose here is detecting resource depletion attacks. In Darpa 98's data, there are pod, teardrop and back, which can be used for this experiment.

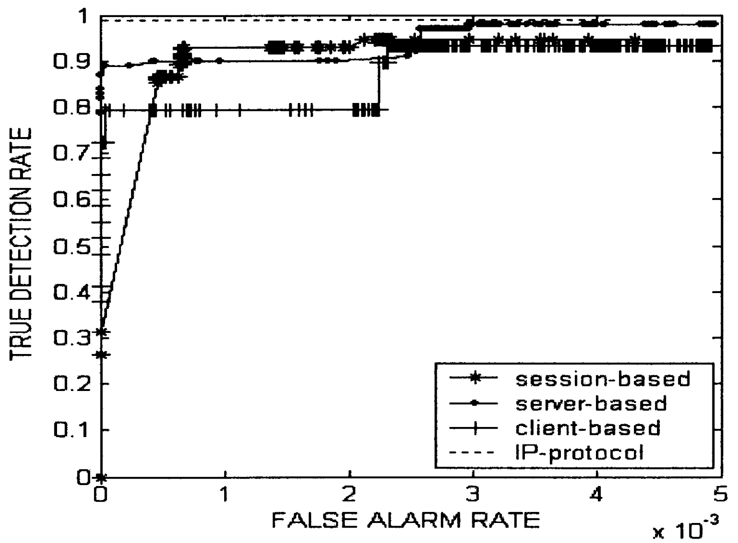


Figure 2.11 ROC curves for resource depletion attacks

Figure 2.11 displays four ROC curves –the first one for session-based NIDS, the second one for server-based NIDS, the third one for client-based NIDS and the fourth one is for IP-based NIDS. From the figure 6, we can observe that all of the ROC curves have very high detection rate. The detection rate is above 80% when false alarm rate is about 0.2%.

CASE 2: Bandwidth Attacks

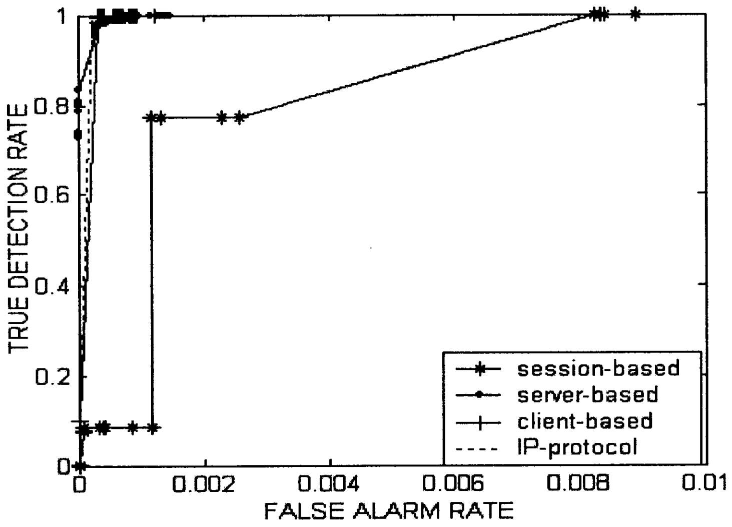


Figure 2.12 ROC curves for bandwidth attacks

Figure 2.12 displays four ROC curves –the first one for session-based NIDS, the second one for server-based NIDS, the third one for client-based NIDS and the fourth one is for IP-based NIDS, which were based on the first five weeks' data for training and used the last week's data for testing. For each curve, the point at the upper left corner represents the optimal detection with high detection rate and low false alarm rate. The x-axis of the figure is the false alarm rate, which is the rate of the typical traffic events being classified as faults or anomalies; the y-axis of the figure is the detection rate, which is calculated as the ratio between the numbers of correctly detected faults/anomalies to their total number. From the figure 2.12, we can observe that all of the ROC curves have very high detection rate. The detection rate is above from 78% when false alarm rate is about 0.2%. This observation also proves our conclusion in the above subsection once again.

CASE 3: Resource Depletion + Bandwidth Attacks

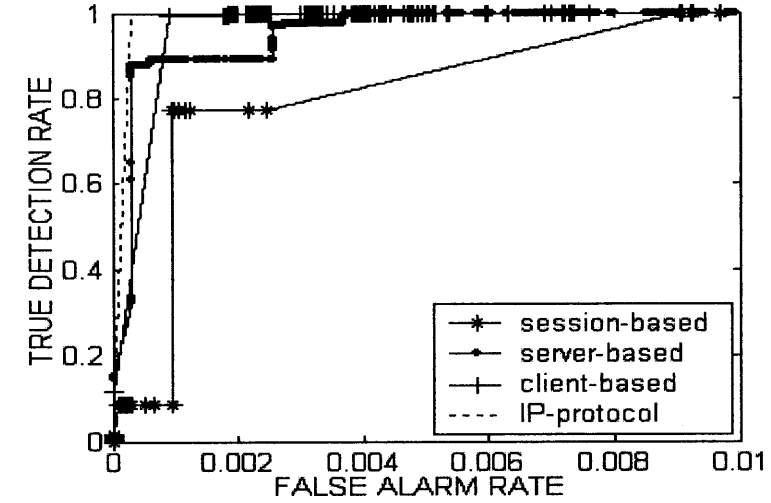


Figure 2.13 ROC curves for resource depletion and bandwidth attacks

From the figure 2.13, Attacks contain both Resource Depletion attacks and Bandwidth attacks, the IDS still have very high detection rate. The detection rate is above 88% when false alarm rate is about 0.2%.

CASE 4: Spoofed Flooding Attacks

A session-based aggregation scheme defense system tries to keep a volume of statistical data in order to extract transaction semantics from packet flows and thus differentiate some legitimate traffic from the attack traffic. However, Attackers frequently use source address and port spoofing during the attack — they fake information in the IP source address or port field in attack packet headers. Attackers tend to generate legitimate-like packets to perform the attack, obscuring the malicious flow within legitimate traffic.

From the figure 2.14, Attacks contain spoofing bandwidth attacks, the performance of IDS apparently drop. If the detection rate is kept at above 80%, then the false alarm rate is about 1.6%.

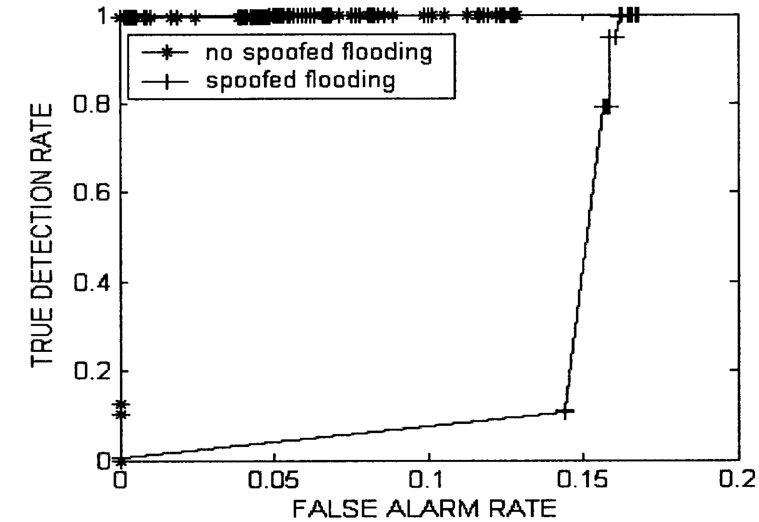


Figure 2.14 ROC curves for spoofed flooding attacks

From Case 1 to case 4, we can see that session-based aggregation schemes performed very well when it was used to detect resource depletion attacks; all aggregation schemes have good performance when detecting non-spoofing flooding attacks. However, the performance of session-based aggregation scheme will drop dramatically when detecting spoofing flooding attacks. The reason is that DDoS attacks have two main features that severely challenge the design of defenses: one is IP source address spoofing; another is similarity of attack to legitimate traffic. These two features create contradictory requirements for DDoS defense. In order to perform accurate traffic separation, the defense system requires a lot of resources for record keeping. Therefore, it can only handle small to moderate traffic volumes. On the other hand, the need to control a large portion of the attack traffic requires placement at points that relay a high traffic volume. Those two requirements can hardly be satisfied at the same time using a fixed aggregation scheme. In the future, we will intend to apply fuzzy logic method adaptively to select flow aggregation scheme to increase the detection rate. Depending on the spoofing level, the fuzzy logic module will automatically activate the most appropriate flow aggregation scheme for FNIDS.

2.6.2 Anomaly Detection Results on CONEX Testbed Data

Due to various limitations of DARPA'98 intrusion detection evaluation data discussed above [20], we have repeated our experiments on live network traffic at the CONEX lab of NJIT. When reporting results on real network data, we report the detection rate, false alarm rate and other evaluation metrics reported for CONEX Test-bed intrusion data. The CONEX TESTBED network, see Figure 2-15, is a test-bed network setup in the CONEX lab of NJIT as a platform to emulate network-based attacks in order to test the performance of intrusion detection prototypes. The network includes three subnets: victim subnet, background subnet and the attack subnet. A more detailed description about the network topology, the attack emulations and the related tools can be found in [8]. We used the data collected on 11/28/2004 for training; and used the data collected on both 11/13/2004 and 11/27/2004 for testing. There are not spoofing flooding attacks in the data.

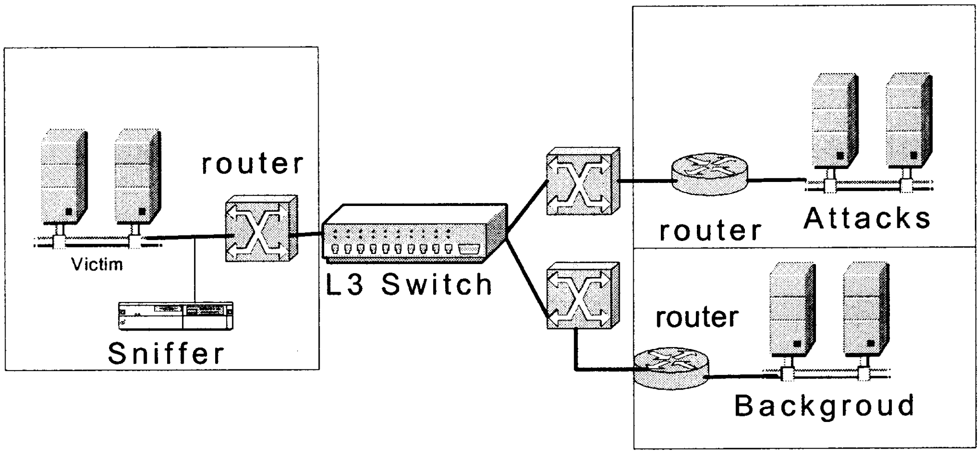


Figure 2.15 Topology of the CONEX testbed network.

The table 2.4 presents the false/true positive ratio of four aggregation schemes that detect various attacks. From this table, it can be seen that session-based aggregation schemes performed very well in detecting DDOS attacks with false positive ratio rates 0%. This indicates that session-based aggregation scheme has the best performance when there are no spoofing flooding attacks.

Table 2.4 False/True Positive Ratio based on CONEX Testbed

Attack Name	Session-based	IP-based	Client-based	Server-based
	False/True Positive Rate	False/True Positive Rate	False/True Positive Rate	False/True Positive Rate
bloop	0	0.037	0	0.037
fawx	0	0.36	0	0.36
fraggle	0	0	0	0
smurf	0	0.027	0.0106	0.027
teardrop	0	0	0	0

Table 2.5 The Performances of Four Aggregation Schemes

	Session-Base	IP-Base	Client-Base	Server-Base
Number of samples	84943	25262	8397	22555
Number of normal samples	83898	24228	7786	22089
Number of attack samples	1045	1034	611	466
Misclassification Rate	0.00401	0.00134	0.00107	0.0047
False Positive Rate	0.00398	0.00103	0.000771	0.0048
False Negative Rate	0.0096	0.0088	0.0049	0.1148

From table 2.5, it can be seen that the misclassification rate is 0.401% for session-based aggregation, 0.102% for IP-based aggregation, 0.107% for client-based aggregation, 0.47% for server-based aggregation. The performances in table 2-5 proved that the more detailed flow-key is, the performance is better when flooding attacks are not spoofed.

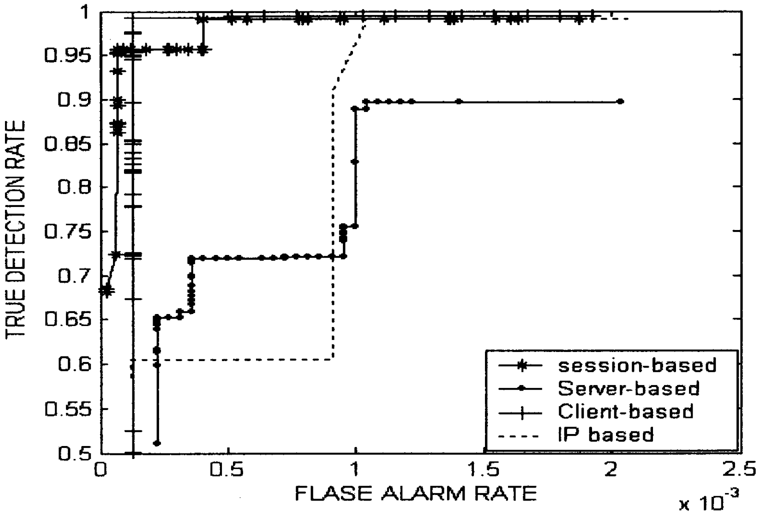


Figure 2.16 ROC curves for CONEX testbed data

Figure 2.16 shows the ROC (Receiver Operating Characteristic) curves of some selected flow-based aggregation schemes. we can observe that all of the ROC curves have very high detection rate. The detection rate is above 94% when false alarm rate is about 0.2%.

2.7 Related Work

NetFlow [10], first implemented in Cisco routers, is the most widely used flow measurement solution today. Exported NetFlow data can be used for a variety of purposes, including in- and out-bound Internet traffic analyses, network management, combating Denial of Service (DoS) attacks, and data mining etc. Based on NetFlow, there are a variety of tools for flow-based measurement arisen. Cflowd is a flow analysis tool created by CAIDA [9]. It is currently used for analyzing Cisco's NetFlow [10] enabled switching method. This analysis package permits data collection and analysis by ISPs and network engineers in support of capacity planning, trends analysis, and characterization of workloads in a network service provider environment. FlowScan [8] is one such freely available tool. It analyzes and reports on NetFlow format data collected using Cflowd tool. In [8], flow profiling was introduced, which analyzes and reports on flow data exported by Internet Protocol routers. The information presented by FlowScan assists in understanding the nature of the traffic that your network is carrying. It can be useful in the identification and investigation of anomalies such as poor performance and attacks on hosts. Kohler [21] investigates the structure of addresses contained in IP traffic and used the changing ratios (i.e., the rate of decrease) between the flow numbers of neighboring specific bit-prefix aggregate flows for detecting peculiarities.

Flow level measurements are also widely used in security detection or to provide insight into the traffic crossing a network. Many Flow-based NIDS have the ability to detect anomaly attacks. The main difference between Flow-based NIDS and traditional IDS is that recording does not contain the high level information, like payload, which make NetFlow process packets in high-speed network like hardware way. Although NetFlow cannot make the in-depth analysis to the data packet, but already had the enough information to discover the suspicious flows. On the other side, aggregation has the advantage of greatly reducing the amount of data collected. From a security standpoint, the aggregated data provides primarily baseline and gross change information. Abrupt or unexpected changes, like traffic behavior, could be used in NIDS to identify the anomaly actions. Bro [16] is a Unix-based Network Intrusion Detection System (IDS). Bro monitors network traffic and detects intrusion attempts by comparing network traffic against rules describing events that are deemed troublesome. Marina [17] described a statistical anomaly detection based on abrupt traffic change and correlated information from SNMP-MIB variables. Androulidakis [13] presented an anomaly detection solution that relies on network flow data exported from CISCO NetFlow-enabled devices. The proposed detection algorithm in [13] monitors flow data from all interfaces of border routing equipment and calculates specific metrics that are compared against adaptive thresholds that characterize the "normal" network utilization. Almost all of them are to detect bandwidth depletion attacks or some certain attacks [14]. There are some other flow-based NIDSs, which made the in-depth analysis, for example: Murai [12] proposed Session Based NIDS which used signature method to monitor the response against the detected attack in order to cuts down the operational costs of NID.

Actually, these flow-based detection technologies are still unable to withstand large-scale distributed attacks. One reason is because filtering the malicious attack traffic requires identifying the (potentially thousands of) attackers, which is complicated, especially if the source addresses are spoofed. A lot of flow will overload router or IDS.

2.8 Summary

A novel flow-based anomaly network intrusion detection system (FNIDS) for detecting network intrusion is proposed in this paper. FNIDS sets up flow-based statistical feature vectors and reports to two parallel detectors: Network Behavior Analyzer and Neural Network Classifier. Network Behavior Analyzer analyzes and visualizes network behavior change. Neural network classifier uses Back-Propagation networks to classify score metric of each flow. FNIDS configures flow flexibly to provide security from network level to application level (IP, TCP, UDP, HTTP, FTP...), and different aggregation schemes, such as server -based, client-based flow. Experimental results evaluated by DARPA 98 and CONEX test-bed data indicate that FNIDS detects both bandwidth and resource depletion DOS attacks successfully for different flow aggregation algorithms. For DARPA data, detection rates are above 80% for all aggregation schemes when false alarm is 0.2% and there are no spoofing flooding attacks.

For CONEX data, the performance is even better .Its misclassification rate is 0.401% for session-based aggregation, 0.102% for IP-based aggregation, 0.107% for client-based aggregation, 0.47% for server-based aggregation.

However, the flow-based analysis still has certain vulnerabilities in terms of processing speed and memory requirement when facing large spoofed sources during DOS attack. In such an attack, the attacker directs a huge amount of malicious spoofed flow, which could exhaust FNIDS memories and CPU resources, disable FNIDS for tracking and detecting all flows. In the future, our work will concentrate on developing an overall framework, which automatically build flow aggregation schemes, recognizes and blocks spoofed flows as soon as possible, maximizes the effectiveness and efficiency of FNIDS.

In the next chapter, Multi-stage Packet Aggregation Architecture (MPAA) is introduced to provide an adaptive flow aggregation approach to defense spoofed flooding or amplification attacks.

CHAPTER 3

ADAPTIVE FLOW AGGREGATES APPROACH FOR FLOW-BASED DEFENSE SYSTEM

Abstract— Flooding-based Distributed Denial-of-Service (DDoS) attack presents a very serious threat to the stability of the Internet. It attempts to disrupt an online service by generating a lot of useless packets to clog links or cause defense system overloaded to crash.

In this chapter, an Adaptive Flow Aggregation Approach (AFAA) is presented, which consists of Flow Aggregation Schemes, Multi-stage Packet Aggregation Architecture, Flow Aggregate Population Density/Distributions Detection Mechanism and Fuzzy Supervisory Controller. Flow aggregation Schemes are responsible for grouping packets based on flow keys, which consist of source IP/Port, destination IP/port and protocol etc. A Flow Aggregate Population Density/Distributions Detection is proposed to detect how packets aggregate at different levels based on Multi-stage Packet Aggregation Architecture (MPAA), such as: how addresses are aggregating at a given prefix length. Fuzzy Supervisory Controller is to activate the most adaptive flow aggregation scheme for packet aggregates. Therefore, APAA provides a more flexible fine-grained flow aggregation approach and a more effective defense measurement against flooding attacks. Finally, the performance of applying AFAA in FNIDS against flooding DDOS attacks is evaluated by using DARPA 98 data. Results show that the adaptive flow-based NIDS made a significant improvement due to the application of the embedded AFAA.

3.1 Introduction

3.1.1 Background

Flooding-based distributed denial-of-service (DDoS) attack presents a very serious threat to the stability of the Internet. In a typical DDoS attack, a large number of compromised hosts are amassed to send useless packets to jam a victim, or its Internet connection, or defense system including IDS, firewall and traffic control system etc.

Spoofing the source IP address/port of packets on the Internet is one of the major tools used by hackers to mount Denial of Service (DoS) attacks. In such attacks the attackers forge the source IP/port of packets that are used in the attack. Instead of carrying the source IP of the machine the packet came from, it contains an arbitrary IP address, which is selected either randomly or intentionally. Once faced by large flooding attacks, such as spoofing attacks, amplification attacks or reflection attacks etc., the flow number of a flow-based analysis method rapidly increases, which could exhaust a lot of memories and CPU resources and disable the flow-based analysis method.

There are very few and not very effective mechanisms that network operators may use today to detect and filter out spoofed packets. The most prominent of them is the ingress and egress filtering. In ingress filtering an ISP prohibits receiving from its stub-connected networks, packets whose source address does not belong to the corresponding stub network address space [34]. In egress filtering a router or a firewall, which is the gateway of a stub network, filters out any packet leaving the network whose source addresses do not belong to the network address space. Thus, both mechanisms ensure that the traffic leaving out of a stub network may only spoof addresses that belong to the same stub network. The former does it at the ISP router while the latter at the stub edge equipment. However, all filters proposed in the literature so far fall short to detect IP address spoofing from the domain in which the attacker resides. For example, in Figure 3.1, if A1 uses the IP addresses of domain D2 as its source IP, the filters R7 will stop such forged packets to reach the victim V1, but if A1 uses some unused IP addresses of domain D1, the filters R7 will not be able to stop such forged packets to reach the victim V1. For another example: if A1 uses hosts in Domain D2 to reflect a lot of packets to the victim V1, The filter R3 will not be able to stop such packets. Ingress filtering can drastically reduce the DoS attack by IP spoofing if all domains use it. If there are some unchecked points, it is possible to launch DoS attacks from those points. Unlike ingress filters, egress filters reside at the exit points of network domain and checks whether the source addresses of existing packets belong to this domain. Actually, it is hard to deploy ingress/egress filters in all Internet domains.

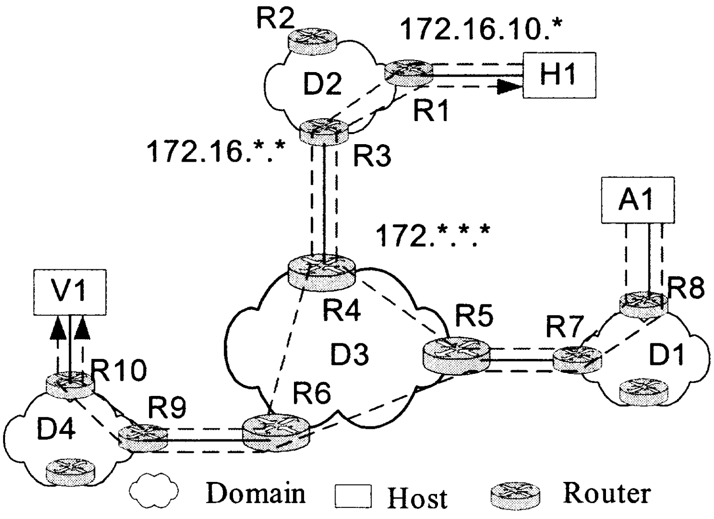


Figure 3.1 Different scenarios for DDoS

3.1.2 Related work

We are not aware of similar previous work on characteristics of Multi-level Packet Classification Architecture to defense flooding attacks. More broadly, much effort has gone into modeling the structures of IP address structure in the Internet for the self-similarity of packet aggregation [22].

MULTOPS [53] proposed a tree of nodes, which contained packet rate statistics for subnet prefixes at different aggregation levels, to monitors certain traffic characteristics to detect (and eliminate) bandwidth attacks.

The properties of client addresses aggregated according to BGP routing prefixes have previously investigated in [78]. The results indicate that client cluster size has a heavy-tailed distribution. Source IP address Monitoring (SIM) was studied in [52] to detect the Highly Distributed Denial of Service (HDDoS). This detection scheme uses an intrinsic feature of HDDoS attacks, namely the huge number of new IP addresses in the attack traffic to the victim, i.e., and the presence of a large number of spoofed IP addresses to identify a distributed denial of service attack by detecting an abnormal increase in the new IP addresses. However, this approach is too gross-grained. Recently, researchers have started to investigate IP address prefix based aggregate properties for aggregate congestion control [39]. A novel flow-based congestion control will be presented in chapter 5.

To mitigate spoofing DDoS attacks, much of the current research focuses on anti-spoofing such as ingress filtering [4], route-based packet filtering [5], and various IP traceback protocols [6], [7]. Their effectiveness often depends on a universal deployment on the Internet.

3.1.3 Motivation

Our work was initially motivated by addressing the poor performance of the statistic flow-based NIDS under high-stress spoofed flooding. Although the static flow aggregation schemes are proved to detect attacks with good performance [10][17][26][27], the performance of the system with a static flow aggregation scheme will drop rapidly under "stress"(i.e., large volume of flooding packets with spoofed IPs)[25][37]. Two main features of DDoS attacks severely challenge the design of defenses: one is a lot of IP source addresses; another is similarity of attack to legitimate traffic. These two features create contradictory requirements for DDoS defense. In order to perform accurate traffic separation, the defense system requires a lot of resources (such as: buffers) for record keeping. Therefore, it can only handle small to moderate traffic volumes. On the other hand, the need to control a large portion of the attack traffic requires placement at points that relay a high traffic volume. Those two requirements can hardly be satisfied at the same time using a fixed aggregation scheme. Once faced by large flooding attacks, such as spoofing attacks, amplification attacks or reflection attacks etc., the flow number of a flow-based analysis method rapidly increases, which could exhaust a lot of memories and CPU resources and disable the flow-based analysis method.

In Figure 3.2, the performances of session-based NIDS are presented in two cases: a lot of spoofed flooding DDoS and non-flooding DDoS, which were based on Darpa 98 data of 5 weeks for training and two weeks for testing. The figure 3.2 shows that performance of FNIDS drops dramatically once it is attacked by a lot of flooding packets including ICMP flooding and Neptune flooding.

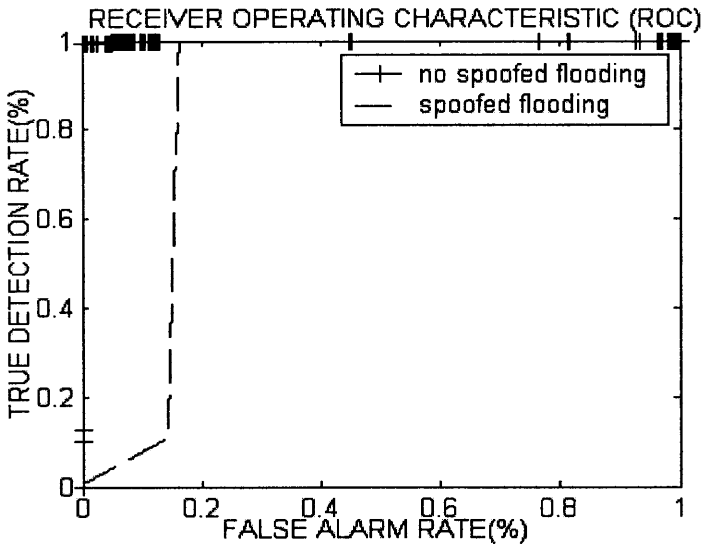


Figure 3.2 ROC of session-based NIDS under: spoofed/no spoofed flooding attacks

In this chapter, an Adaptive Flow Aggregation Approach (AFAA) is presented, which consists of flow-based aggregation module, Aggregate Population Distributions Analyzer and Fuzzy Supervisory Controller. Flow-based aggregation module is responsible for grouping packets based on flow keys, which consist of source IP/Port, destination IP/port and protocol etc. Aggregate Population Distributions Analyzer is to find how packets aggregate at different levels based on Multi-stage Flow Aggregation Architecture (MFAA), such as: how addresses are aggregating at a given prefix length. . Therefore, AFAA provides a more flexible fine-grained flow aggregation approach and a more effective defense measurement against flooding attacks by detecting the abnormal changes of flow population density and distribution. Fuzzy Supervisory Controller is to activate the most adaptive flow aggregation scheme for packet aggregates

Finally, the performance of applying AFAA in FNIDS against flooding DDOS attacks is evaluated by using DARPA 98 data. Results showed that adaptive flow-based NIDS made a significant improvement due to the application of the embedded AFAA.

3.2 Adaptive Flow Aggregates Approach Design

In chapter 2, "flow" concept was introduced. Theatrically, flows with all fields (source IP, destination IP, source port, destination port, and protocol) can completely express a stream of packets. But attackers commonly use randomly generated spoofed source address and port numbers to launch attacks to a victim. Therefore, more information of flow records, using high dimensional flow key, like session-based aggregation scheme, is easy to overload buffer or take longer time processing once the system is attacked by flooding.

In this section, Multi-stage Flow Aggregation Architecture (MFAA) and Flow Aggregation Population Density/Distribution Detection Mechanism are proposed to detect and defense spoofed flooding attacks.

3.2.1 Multi-stage Flow Aggregation Architecture (MFAA)

As the key component of the APDA, the proposed MFAA uses 5-tuples (Source IP, Source Port, destination IP, destination Port, Protocol) in the IP header to classify packets. Certainly, transport-layer information can also be extracted to further divide a protocol-based aggregates into a UDP aggregate, a TCP aggregate, and an ICMP aggregate and, then, to distinguish IP Flags in the TCP aggregate etc. For simplification, MPAA in this chapter will be based on 5-tuples.

In section 2.2.2, two kinds of flow aggregation schemes were proposed: field aggregation schemes and IP prefix aggregation schemes. Based on these flow aggregation schemes, a Multi-staged Flow Aggregation Architecture (MFAA) is shown in Figure 3.3. MFAA contains two multi-level structures: 4-stage Field Aggregation Architecture and Source IP Aggregation Level Structure shown in Figure 3.3.

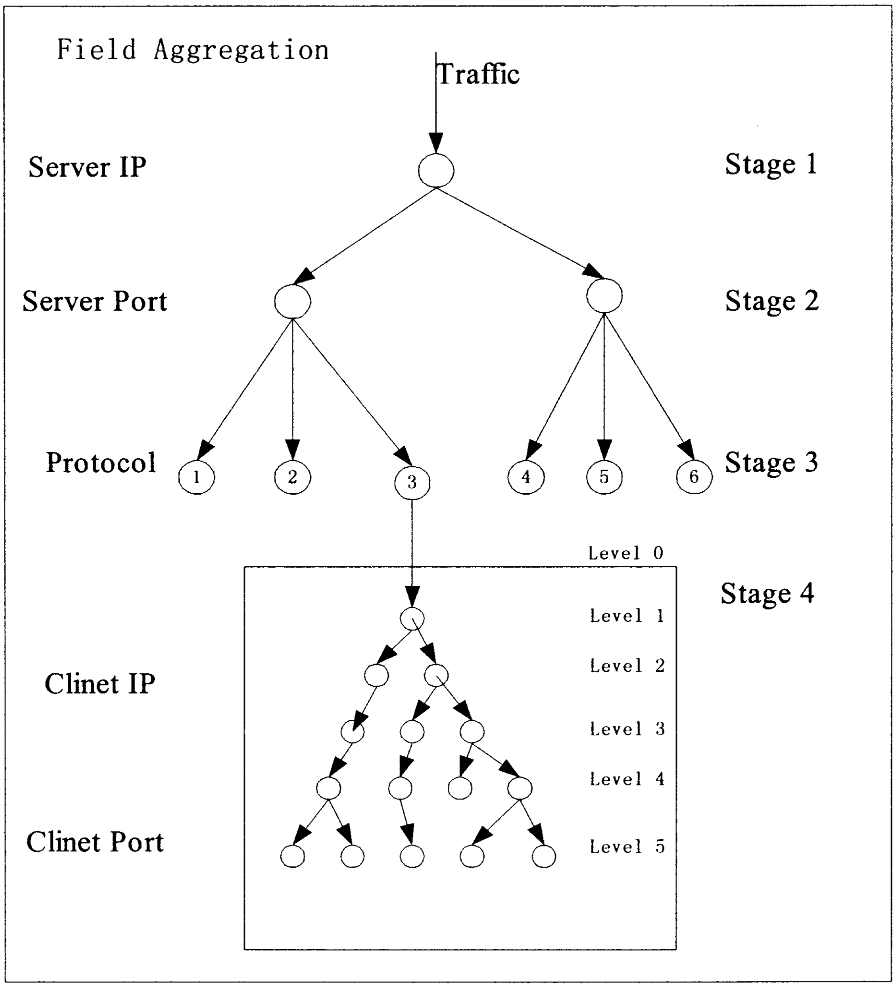


Figure 3.3 Multi-stage flow aggregates architecture

In Figure 3.3, it is seen that Source IP Aggregation Level Structure is a sub-tree structure of stage 4 detailed in Figure 3.4. Nodes represent flow keys and lines represent flows, which means how many flows are based on each flow key. They are used to realize an adaptive flow-based aggregation approach to detect and defense spoofed flooding attacks.

Upon the multi-stage flow aggregation architecture, expansion and contraction of flow aggregation enable adaptive flow aggregation approach (APAA) to exploit the hierarchical structure of the 5-tuples. Deeper stage or level of the architecture has more fine-grained flow aggregations. APAA detects the attack on a high stage in the structure (where fields are few) and expands toward the most possible fields including source IP addresses. According to this described architecture, using different-stage flow aggregation algorithm can effectively to defense flooding attacks and realize various fine-grained traffic aggregates. For example: Suppose that there is a spoofed flooding attack with forged source ports, if 4-stage packet classification architecture is chosen to aggregate packets, which means that the flow key is composed of destination IP/port, protocol and client IP, then the flow key doesn't contain port field. So the spoofed ports of flooding attacks do not influence the flow number.

Not only can AFAA be used to defense spoofed flooding, but also it can defense other flooding attacks from some subnets, such as reflection flooding attack or amplification flooding attack. For example: in Figure 3.1, if hosts in the domain D2 are used to reflect a lot of packets with the same subnet prefix (172.16.*.*). The simply way to defense this flooding attack is to adopt a 2-level IP prefix of 4-stage field aggregation to group packets. The flow key doesn't contain last two octets of client IP and ports, so the reflected flooding does not affect the flow number.

AFAA detects the attack on a low stage in the structure (where fields are more) and contract toward the fewer fields, in order to real-timely monitor general network state. Some stages or levels are not included in flow keys to realize flow aggregation. For example, In Figure 2.6, we only used protocol-based aggregation scheme to aggregate packet and monitor traffic change. This Figure shows that there are two kinds of flooding attacks happening: one is Smurf and another is Neptune. Smurf is ICMP flooding and the Neptune is SYN flooding. Apparently, protocol-based aggregation scheme can defense flooding attacks, but is not the most effective defense method. Because NIDS would inform a *filtering engine* to block all ICMP packets or SYN packets once there is flooding happening. However, this way also drops normal packets. That is why a more fine-grained traffic aggregation model is needed.

There are many methods can be used to detect network flooding. In section 2.42, we described a Network Behavior Analyzer (NBA), which can efficiently detect various flooding attacks by observing abrupt changes of statistics features, detailed in section 2.42.

Upon the Multi-stage Flow Aggregates Architecture, Flow Population Density /Distribution Detection Mechanism (FP2D-DM) is proposed here to realize detection and defense of spoofed flooding attacks.

3.2.2 Flow Population Density /Distribution Detection Mechanism (FP2D-DM)

The purpose of FP2D-Dm is to find which stage/level spoofed flooding is at and which flow contains flooding. Based on the multi-stage flow aggregates architecture, we propose two detection mechanisms: Flow population Density Detection Mechanism (FPD-DM-1) and Flow Population Distribution Detection Mechanism (FPD-DM-2). It is analogous to population survey that FPD-DM-1 is to survey how many flows belong to a node (flow key) and how nodes (flow keys) are distributed.

In order to survey flow population, we adopt two detection directions in MFAA tree: horizontal direction and vertical direction. Horizontal direction is to calculate relative density among contiguous nodes and Vertical direction is to calculate how nodes distribute in different stages or levels. FPD-DM-1 uses the horizontal direction to detect flow population density and FPD-DM-2 uses the vertical direction to detect how many leaves a parent node has, which is useful in expansion and contraction of flow aggregation to defense spoofed flooding.

3.2.2.1 Flow Population Density Detection Mechanism (FPD-DM-1)

To detect which flow contains flooding is simply the measurement of how densely packets are packed in each flow. Based on the multi-stage packet aggregates architecture, FPD-DM works on the same stages to find flow aggregation population density at each stage, which will tell which flow contains the most quantity of packets.

Given a flooding attack, let $n_{i,j}$ be the leaf counts (or sub-flows) of the ith node(or flow) at stage j and N_j be the total leaf counts at the stage j , then the Flow Population Density (FPD) of this node i at this stage j is denoted by:

$$FPD_{i,j} = \frac{n_{i,j}}{N_j} \qquad (3.1)$$

Where: i ≤ the maximum node (or flow) number at the stage j; and j is from 1 to 5.

Formula (3.1) expresses a flow population relative density at the jth stage. The *flow population* is the number of active leaves (sub-flows) contained in the node (flow). If there are flooding happening and the FPD of a flow is over threshold, then this flow contains flooding attacks. Except Flow Population Density (FPD), Packet Population Density (PPD) is used to score the packet density in a flow (or node), which gives the possibility of flooding that a flow contains.

Given a flooding attack, let $r_{i,j}$ be a packet rate of the ith flow at stage j and R_j be the total packet rate at the stage j .The Flow Population Density (FPD) of this flow i at this stage j is denoted by:

$$FPD_{i,j} = \frac{r_{i,j}}{R_j} \qquad (3.2)$$

Where: i ≤ the maximum flow number at the stage j; and j is from 1 to 5.

Actually, formula (3.2) expresses a flow population relative density at the jth stage. The *population* of a flow is the number of active packets contained in that flow. If there are flooding happening and the FPD of a flow is over threshold, then this flow contains flooding attacks. For example, Figure 2.6 used a protocol-based aggregation scheme. Thus it shows a more fine-grained packet aggregation than one in figure 2.5. All flows in figure 2.6 are differentiated by protocol, like TCP, UDP and ICMP. In Figure 2.6, The FPD of the "TCP" flow during 400~480 is over 0.8, but the FPD of "UDP" or

"ICMP" flow is below 0.1. Apparently, the attacker used TCP protocol to launch attack. This example shows that FPD-DM can find which flow contains flooding at the same stage.

Although FPD-DM-1 is successful to find flooding flow from the flows at the same stage, it is still not enough to defense spoofed flooding. For example: in Example 3.2, when the server-based (1-stage) detection method detects an abruptly abnormal traffic increase, it only controls a *filtering engine* to block all packets to this server while normal client can not access this server too. If there are some ways to identify domain (D2) that flooding attacks come from, the *filtering engine* only drops the packets from this domain (D2) while keeping the server alive. Expansion and contraction of flow key based on the Multi-stage Packet Aggregates Architecture (MPAA) provide an approach to realize more fine-grained detection mechanism while keeping maximum valid flow keys for packet aggregation.

3.2.2.2 Flow Population Distribution Detection Mechanism (FPD-DM-2)

Although FP-DDM-1 is successful to find flooding flows from the flows at the same stage, it is still not enough to defense spoofed flooding, since many attacks directly disable a victim by sending a lot of spoofed flooding and causing it to keep a lot of flow records in a short time.

In this section, we will present FPD-DM-2, which detects flow population distribution along vertical direction in Multi-stage Flow Aggregates Architecture (MFAA). For example: if there is flooding attack happening, the FPD-DM-2 compares the flow numbers at the stage 3 and stage 4; If the flow number of stage 4 is much more than the number of stage 3; apparently, the source IP/Ports are spoofed. Thus choosing 3-stage aggregation will defense the port-spoofed flooding attacks.

In section 2.2.4, we described two kinds of flow aggregation schemes: field aggregation schemes and prefix aggregation schemes, which separately work in flow aggregates architecture, we further propose two kinds of flow population distribution detections: Field Aggregation Population Distribution Detection (FA-PDD) and Source IP Aggregation Population Distribution Detection (SIPA-PDD). FA-PDD mainly detects the flow aggregation distribution based on filed aggregation schemes. Because Source IP/Port is often utilized by attack to launch a lot of packets, SIPA-PDD is proposed to take more fine-grained flow aggregation distribution analysis based on source IP/Port, which is a sub-tree of a note at stage 4. Because FA-PDD and SIPA-PDD have the similar tree structures, both FA-PDD and SIPA-PDD have the same form of flow population distribution detection algorithms.

(1) Field Aggregation Population Distribution Detection Algorithms (FA-PDD)

In order to detect the flow aggregation distribution between two stages or levels in the Multi-stage Flow Aggregation Architecture, suppose there is a flooding attack happening, which can be detected by the above method.

Let there are N_i flows at the ith stage (or level) and there are N_{i+1} flows at the i+1th stage (or level). If we aggregate flows separately based on the stage (or level) i and stage (or level) i+1, then *Single-stage (or level) Flow Aggregation Likelihood Ratio (SFALR)* from stage (or level) i to i+1 can be defined as followed.

$$SFALR_{i,i+1} = \frac{N_{i+1}}{N_i} \qquad (3.3)$$

Where: 1≤ i≤ 4 for FA-PDD or 0≤ i≤ 4 for SIPA-PDD.

In multi-stage structure, there are N_i flows at stage (or level) i and N_j flows at stage (or level) j. then the *Multi-stage (or level) Field Aggregation Likelihood Ratio (MFALR)* from i to j (i<j) is the multiplication of all $SFALR_{i,i+1}$ from level i to j.

Here is:

$$\begin{aligned} MFALR_{i,j} &= SFALR_{i,i+1} \times SFALR_{i+1,i+2} \times \cdots \times SFALR_{i-2,j-1} \times SFALR_{i-1,j} \\ &= \frac{N_{i+1}}{N_i} \times \frac{N_{i+2}}{N_{i+1}} \times \cdots \times \frac{N_{j-1}}{N_{j-2}} \times \frac{N_j}{N_{j-1}} \\ &= \frac{N_j}{N_i} \qquad (3.4) \end{aligned}$$

Where: 1≤ i≤ j≤ 4 for FA-PDD or 0≤ i≤ j≤ 4 for SIPA-PDD.

These formulas (3.3) and (3.4) represent a relative ratio of flow number between two stages (or levels), which represent the flow aggregation degrees between two stages (or levels). Especially at stage 3 and stage 4 of FA-PDD, once there are spoofed flooding attacks, the flow aggregation likelihood ratio becomes great. Thus 3 stage flow aggregation schemes can be activated to group packets for defending against spoofed flooding.

(2) Source IP Aggregation Level Structures for SIPA-PDD

In order to exploit the hierarchical structure of source IP address/port space, more fine-grained flow aggregations based on source IP is proposed.

In the section 3.1, we introduced three approaches to prevent Spoofed DDoS: Ingress Filtering, Router-based and victim-based (Server-based). However, all these filters proposed in the literature so far fall short to detect IP address spoofing from the domain in which the attacker resides. In other words, attackers often use the devices in some subnets to reflect or amplify a lot of packets to a victim or spoof a limited set of IP addresses—with a common prefix. And also there are very few and not very effective mechanisms that network operators may use today to detect and filter out spoofed packets. Thus, network administrator has to uses two ways to protect server from a lot of spoofed flooding: one is to stop all packets to victim and another way is to limit total traffic volume. The former can stop DDoS while normal clients also cannot access the server which attacker wants. The later only protect servers from flooding, but it cannot block malicious clients.

Expansion and contraction enable SIPA-PDD to exploit the hierarchical structure of the IP address space and the fact that a bandwidth attack is usually directed at (or coming from) a limited set of IP addresses—with a common prefix—only. SIPA-PDD detects the attack on a high level in the IP structure (where prefixes are short) and expands toward the largest possible common prefix of the victim's IP address, potentially establishing single IP address that are under flooding attack.

There are two steps in the SIPA-PDD to detect and defense flooding attacks. First, the SIP-ADD sorts the incoming IP flows according to source IP addresses, and identifies whether there is an IP flow with an unusually large number of packets. If there is, the SIPA-PDD activates the *filtering engine* to block this abnormal IP flow. This step is very effective for defending against some naive DoS attacks launched from a single or small number of sources. The second step is the core technology of our SIPA-PDD scheme, which is designed to defend against sophisticated DDoS attacks and is described in detail in the following.

Source IP Aggregation Level Structures

In section 2.2, we presented various flow aggregation schemes based on the five fields, which include two kinds of IP/port: destination IP/Port and Source IP/Port. Principally, all of them can be spoofed. But in the real world, attack is usually directed at (or coming from) a limited set of IP addresses—with a common prefix or attack spoofs the source IPs with a common prefix.

As described in section 2.2.4.2, an aggregate with prefix length p is called a p-aggregate with 0≤ p≤ 32. "a/p" refers to the p-bit prefix of IP address "a" or, equivalently, the aggregate containing all addresses sharing that prefix. The'0' represent all bits are valid and the '32' means all bits are masked. IP address is usually expressed as four octets, each representing eight bits. The purpose of Spoofing is to forge some or all of the four octets. Thus, the prefix lengths are defined separately by five values, which are 0,8,16,24 and 36. Dietrich and David [35] described the spoofing levels of IP address as network boundary, partial, first octet and Full. For example: spoofing the last octet (192.168.2. *) is typical network boundary level; Spoofing the last two octet (192.168. *. *) is partial level. If all octets are spoofed, it is call full level of spoofing.

Similarly, the Source IP Aggregation Levels are divided into six levels (from 0 to 5), which include a source port. Higher level has more fine-grained aggregation. Level 5 is the finest-grained aggregation, which is often used to aggregate packets without spoofed flooding or reflection flooding or amplification flooding. Figure 3.4 shows the Source IP Aggregation Level of IP+Port. As a victim defense system, both spoofed packets and packets from some subnets have the same effect that flow number will increase rapidly when attacked. Therefore, spoofed flooding is used as a template flooding.

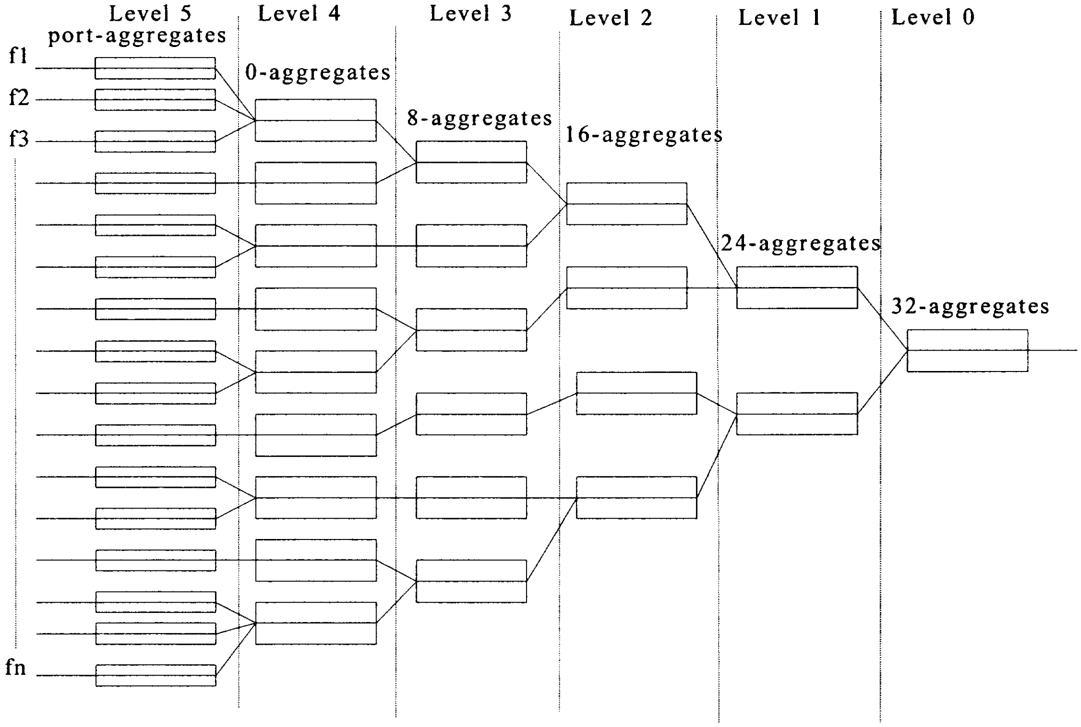


Figure 3.4 The source IP aggregation level structure

The Source IP Aggregation Levels correspond to spoofing levels of IP address [35] if the IP addresses are spoofed. Therefore, this architecture is sometime called as *IP spoofing level architecture.*

Source IP Aggregate Likelihood Ratio (SIALR)

Formulas (3.3) and (3.4) can be used in the Source IP Aggregation Level Structure to calculate Source IP Aggregate Likelihood Ratio.

For example: when aggregating flows from spoofing level 2 to 3 in Figure 3.4, N2=4 and N1=2. Therefore, the SIALR between level 1 and level 2 is 2.

For example: In Figure 3.4, N_4=9 and N_0=1, the Likelihood ratio from level 0 to 4 is 9.

Logarithm function is used here to express *10-time aggregation degree* or simply called *aggregation degree* for flow Aggregation Likelihood Ratio.

The formula (3.3) can be denoted by

$$SFAD_{i,i+1} = \log(SFALR_{i,i+1}) \ ; \qquad (3.5)$$

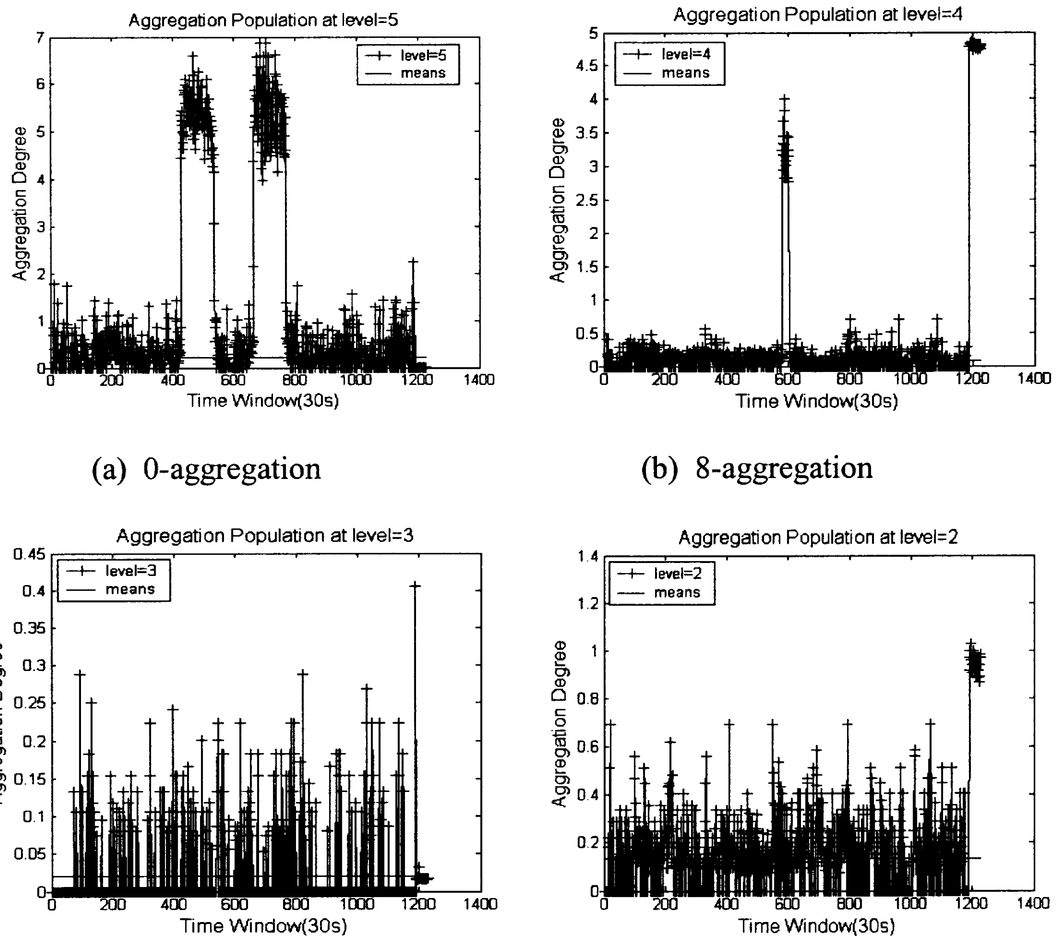
The formula (3.4) can be denoted by

$$MFAD_{i,j} = \log(MFALR_{i,j}) = \sum_{k=i}^{j-1} \mathrm{SFAD}_{k,k+1} \qquad (3.6)$$

Where: $0 \leq i \leq j \leq 4$.

In fact, there are no other differences between (3.3) and (3.5) or between (3.4) and (3.6) except that the formulas (3.5) and (3.6) express the distribution situation by aggregation degree. In the left of chapter, we used (3.5) and (3.6) in all experiments.

Six examples of aggregation degrees corresponding to different aggregation population levels or stages are shown in Figures 3.5(a) ~(f). In figure 3.5(a), flows are aggregated at level 5; this figure shows abrupt changes of *aggregation degree* happening during two periods (about 400~500 and 700~800 time windows). These changes were caused by a lot of “Neptune” attacks with spoofed source ports. In Figure 3.5(b), flows were aggregated at level 4; the abrupt changes of *aggregation degrees* were caused by "smurf" attacks. Apparently, Figure 3.5(a) and figure (b) show when traffics contain attacks and how to choose flow aggregate level to defense flooding. For example: based on the result of figure 3.5(a), there were "smurf" happening between 400 and 500. Apparently, the source ports were spoofed during this period, so the system could ignore the spoofed ports and used 4 tuples (destination IP/port and Source IP,protocol) as flow keys to aggregate packets. Figure 3.5(f) shows the total *aggregation degree*.

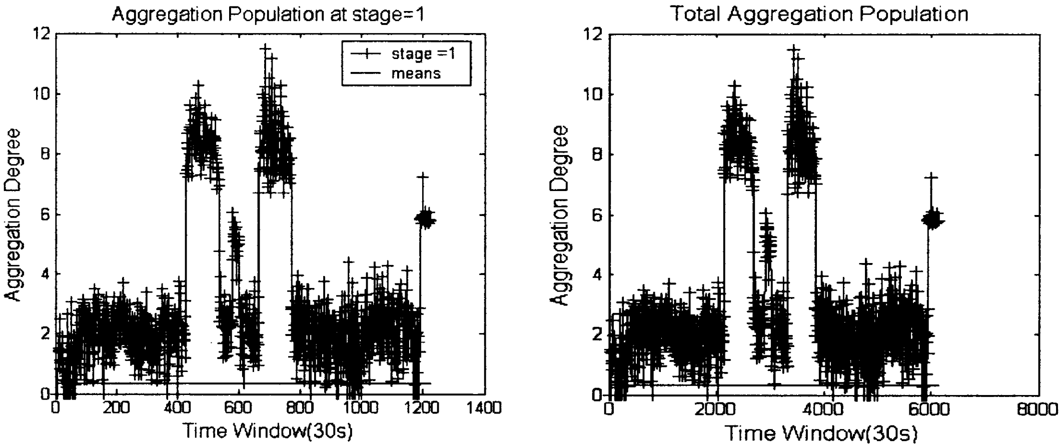


(a) 0-aggregation

(b) 8-aggregation

(c) 16-aggregation

(d) 24-aggregation



(e) Server-based aggregation (f) Total aggregation degree

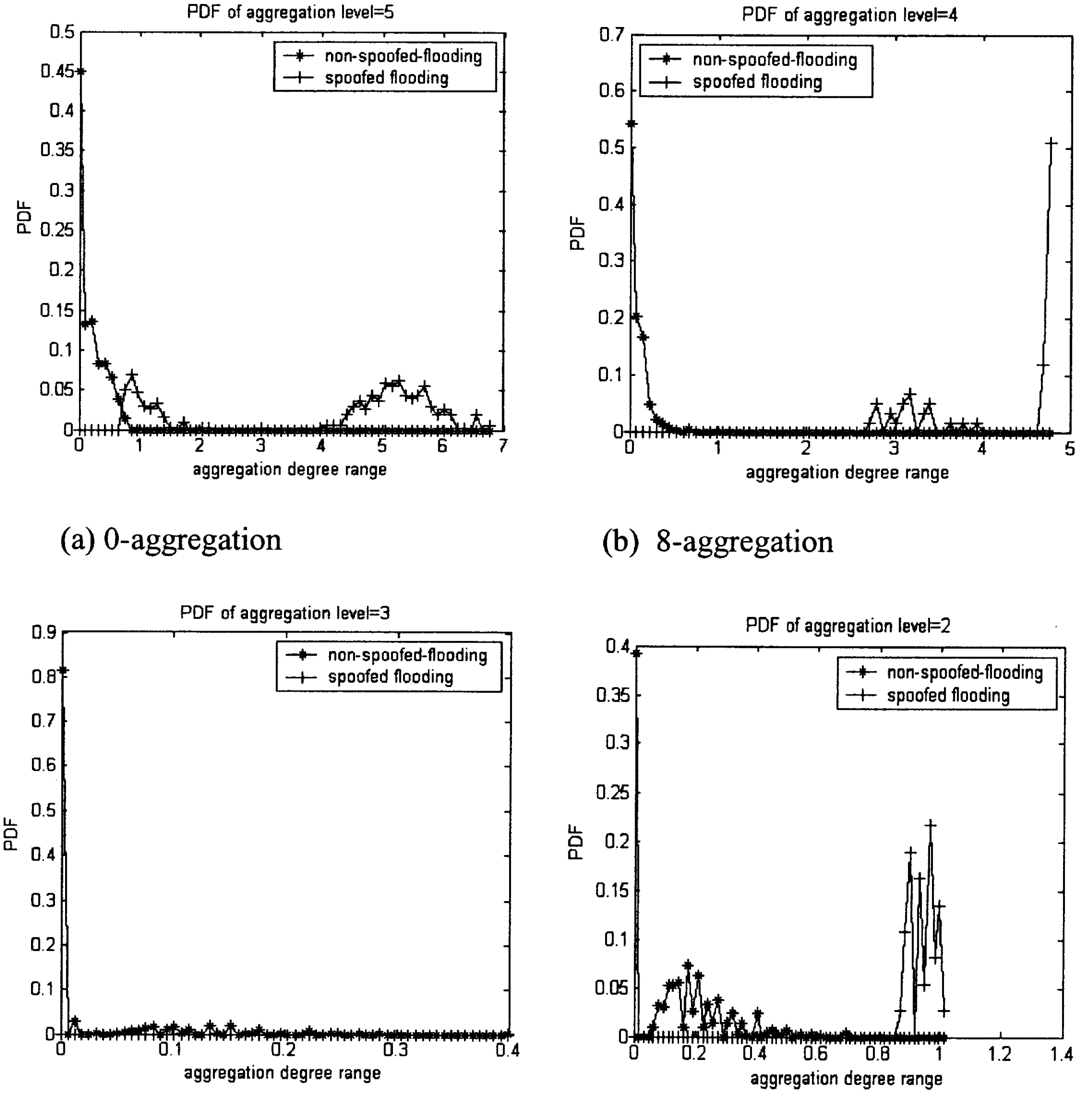
Figure 3.5 Six aggregation levels in client side under spoofed flooding attacks

In order to compare the spoofed flooding degree and non-spoofing degrees, it is necessary to set up a reference model of non-spoofed flooding degrees.

(3) Statistical Modeling for Aggregation Degree

The calculated *aggregation degrees* are scored based on the information of the reference models. The reference models are in fact probability density functions (PDF) of the values of *aggregation degrees*. Figure 3.6 shows the measured non-spoofed-flooding-type and spoofed-flooding-type PDF samples of *aggregation degrees* at different levels, which were based on Darpa 98 data of the 6th Thursday, including spoofed 'Neptune' and 'Smurf'. Since the actual legitimate *aggregation degree* distribution is unknown, the approach needs to establish a reference profile of non-spoofed traffic. The reference model would be updated actively in each time window. While packets arrive continuously and the normal-type profile changes as new packets arrive, the reference profile is updated.

Six examples of PDFs of *aggregation degrees* between non-spoofed-flooding-type and spoofed-flooding-type flows are shown in Figure 3.6 (a)~(f). In each figure, both kinds of PDFs were plotted for easy comparison.

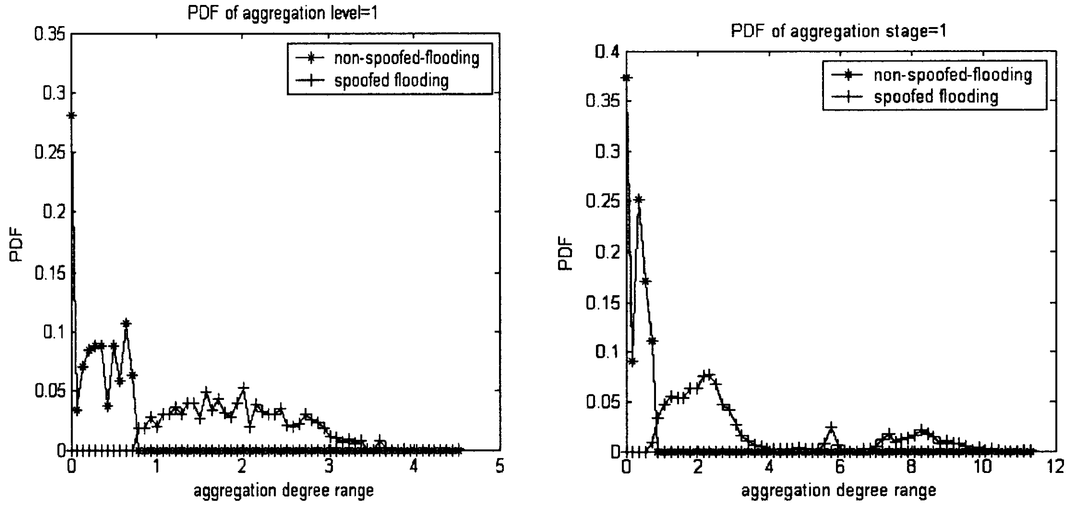


(a) 0-aggregation

(b) 8-aggregation

(c) 16-aggregation

(d) 24-aggregation



(e) 32-aggregation

(f) Server-based Aggregation

Figure 3.6 PDFs of aggregation degree between non-spoofed-flooding-type and flooding-type at different aggregation levels.

There are differences of aggregation degree's PDFs between non-spoofed flooding and spoofed flooding in Figure 3.6(a), (b), (e) and (f). Figure 3.6(f) shows the difference between both type PDFs. These figures further show that there is self-similarity among different grained aggregations. Evidently, as seen from the graphs, there is some overlap between the non-spoofed-flooding-type PDFs and the spoofed-flooding-type PDFs. This means that single aggregation level threshold classification is error prone. However, it is most significant that, in general, the non-spoofed-flooding-type and the spoofed-flooding-type PDFs are very different from each other at some aggregation levels. This means that statistical methods that capitalize on these differences can be very effective. Such as: IP-protocol aggregation scheme is better for case (a) and (b); server-based aggregation scheme is the best for case (f). According to different cases, Fuzzy supervisor controller (described later) can make decision which scheme will be activated dynamically in real system, which will be described later.

(4) Score metrics

There are many statistic metrics to score a single value. Our current statistical model uses Single Number Statistics (SNS).

Let x be the feature value, $p_n(x)$ be the probability of x in the non-spoofed-flooding reference model, $p_{n,max}$ be the maximum probability of the non-spoofing reference model. The distance of the SNS is defined as the following metrics:

$$S\ (x) = \frac{P_n(x)}{P_{n,\max}} \qquad (3.7)$$

Where: S is between [0,1]. S =1 means 'non-spoofed-flooding'; and S= 0 means 'spoofed-flooding'. From the above equations, it can be seen that metric (3.7) is a spoofing degree change detection approach by comparing spoofing degree with the reference model.

In order adaptively to select flow aggregation scheme to increase the detection rate, we used Fuzzy Supervisory Switch to identify the spoofing level according to the difference of spoofing degree.

3.3 Fuzzy Supervisory Controller (FSC)

As described above, Multi-stage Flow Aggregates Architecture (MFAA) provides a multi-layered flow aggregation structure for flow-based defense system and Flow Population Density/Distribution Detection Mechanism (FP2D-DM) provides a way to detect flooding distribution and provide a aggregation degrees for a defense system to choose the adaptive flow aggregation schemes to defense spoofed flooding attacks or other bandwidth flooding, like reflection flooding attacks.

In this section, we use fuzzy control theory and Multi-level Spoofing Levels Structure to realize an automatic selection of flow aggregation scheme.

3.3.1 The Relationship between Aggregation Levels and Flow Aggregation Schemes

In section 3.2, Multi-stage Flow Aggregates Architecture (MFAA) was proposed, which constructed a multi-layered flow aggregation structure. Furthermore, The MFAA contains 4-stage Field Aggregation Architecture and Source IP Aggregation Level Structure. In 4-stage Field Aggregation Architecture, only Client IP/Port is unreliable due to spoofing attack. Other fields, like Server IP/Port and Protocol, must be true. Source IP Aggregation Level Structure is a more fine-grained flow aggregation based on the 4th stage in 4-stage Field Aggregation Architecture, detailed in 3.2.2.2(2). Therefore, it is necessary to establish a relationship between Source IP Aggregation Levels and Flow Aggregation Schemes in order to realize adaptive expansion and contraction of source IP/Port or provide more fine-grained flow aggregation schemes..

As discussed in the above section 3.2.2.2(2), there are 5 aggregation levels in Source IP/Port. In order to emphasize the spoofed flooding, the "spoofing level" is used for "aggregation level". Thus, 6 flow aggregation schemes were separately used in 5 spoofing levels and 1 normal state, shown in figure 3.7.

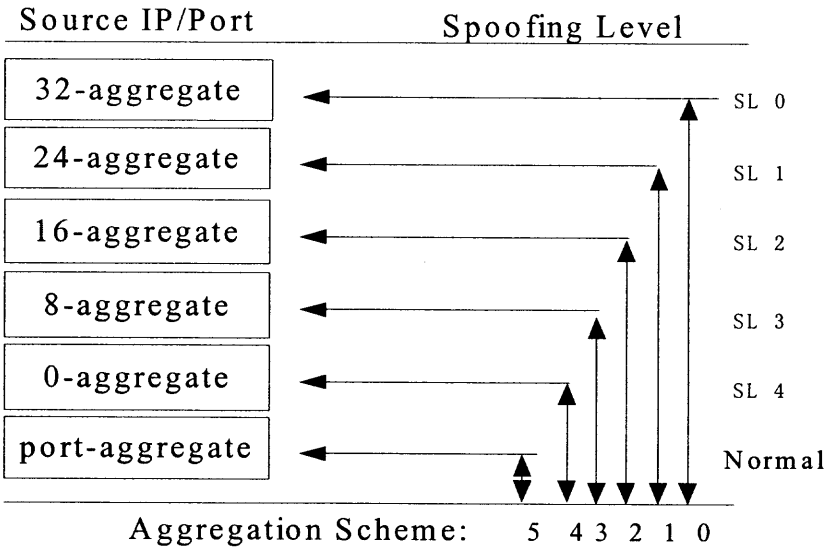


Figure 3.7 The relationship between spoofing levels and flow aggregation schemes

Based on the spoofing levels, the system separately uses one of 6 aggregation schemes to group packets.

Each spoofing degree change between spoofing degree and reference model in this system is considered as an input of fuzzy supervisory controller. The FSC is considered as a multi-input Single-output (MISO) control system.

3.3.2 Fuzzy Supervisory Controller (FSC) Design

Fuzzy control theory has been successfully used in various fields, especially in make decision based on multi factors. The Fuzzy Supervisory Controller (FSC) developed here contains five Multi-Input Single-Output engines (MISO). Depending on Spoofing levels, the fuzzy Supervisory controller will automatically activate the most appropriate flow aggregation scheme. To get this through, the controller uses Fuzzy Inference Engine (FIE) and Knowledge Base (KB).

(1) Fuzzification / Defuzzification

Fuzzification and Defuzzification involve mapping the fuzzy variables of interest to "crisp" numbers. Fuzzification translates spoofing degree changes into linguistic values. In order to activate the most appropriate flow aggregation scheme, the FSC uses five kinds of variables as the control inputs and one parameter for the output. These variables represent the criteria or objectives to be optimized in this system, which are as follows:

Spoofing Level 0: detect the aggregation degree change of 32-aggregation
Spoofing Level 1: detect the aggregation degree change of 24-aggregation
Spoofing Level 2: detect the aggregation degree change of 16-aggregation
Spoofing Level 3: detect the aggregation degree change of 8-aggregation
Spoofing Level 4: detect the aggregation degree change of 0-aggregation

Six Aggregation Schemes (AS 0~5) of supervision and scheduling have been defined as follows:

AS0: 3-stage flow aggregation.
AS1: 24-aggregates.
AS2: 16-aggregates.
AS3: 8-aggregates.
AS4: 0-aggregates.
AS5: port-aggregates, or full-4-stage aggregates.

The output of the fuzzy controller is used to activate the aggregation scheme with highest priority value.

Figure 3.8 shows the input member functions. Figure 3.9 shows the output member function. All input/output values are divided into three ranges corresponding to HIGH, MEDIUM and LOW.

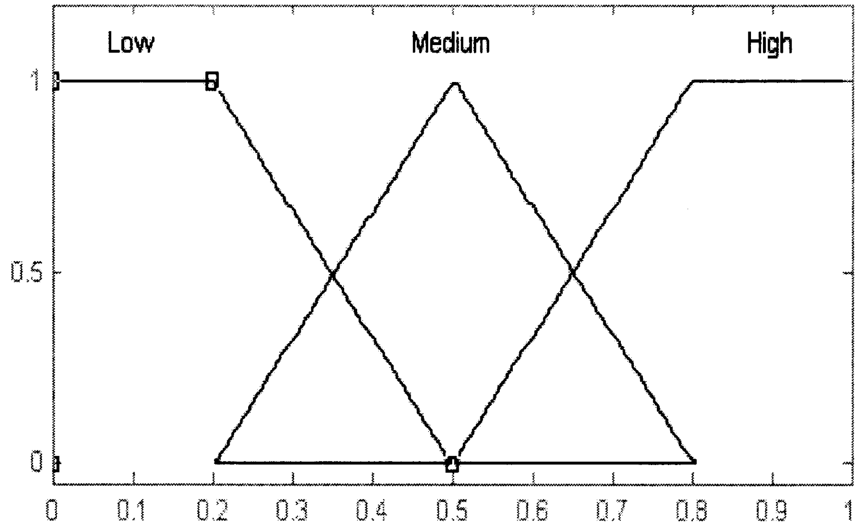


Figure 3.8 The input membership function

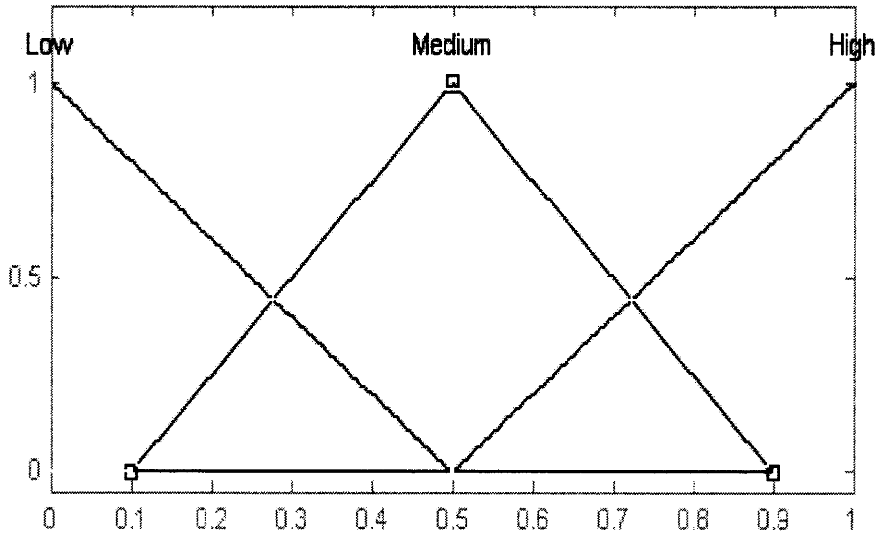


Figure 3.9 The output membership function

The input variables are fuzzified and run through the fuzzy rule base, which is discussed below. In a crisp system, the intersection of two sets contains the element that is common to both sets. This is equivalent to the common logical AND operation. In conventional fuzzy logic, the AND operator is supported by taking the minimum of the truth membership grades.

There are many methods that can be used to defuzzy the outputs. One of the most commonly used methods for defuzzification is the Centroid method also known as Center of Area (COA) method. The model well coincides with the human images of fuzzy inference rules in cases when the system has many input and many outputs. In the present simulator, the Centroid method has been used, in which the centroid of the consequent fuzzy set is found and it corresponding priority value on the x-axis is the final crisp output.

(2) Inference

The fuzzification process finds the degree of membership of each input to their corresponding fuzzy sets. In the main working part of fuzzy logic, there are sets of rules provided by an expert, which relate various fuzzy sets to a desired fuzzy output. Fuzzy sets in this case are, high, medium and low. Fuzzy output sets are high-priority, medium-priority and low-priority. Priority indicates the suitability of a particular flow aggregation scheme to a particular request. Depending on the fuzzy inputs and the rule bases, the output fuzzy set, `priority' is computed using an inference scheme. Several inference schemes are available like Mamdani etc. For the present simulator, the Mamdani scheme has been adopted (through MATLAB). Figure 3.10 shows the pictorial representation of the inference scheme for several rules and its combined fuzzy output.

(3) Rule Development

Our rule development strategy is to select different flow aggregation schemes to make the FSC active the most adaptive flow aggregation scheme. The output of the FSC is based on the current input. The main idea is that if and only if the system is under a lot of spoofed flooding attacks, the FSC will select different flow mask to activate different flow aggregation scheme for the applications (such as: Flow-based NIDS, flow control or firewall) in next time window. Otherwise, the applications will use default aggregation scheme (in our system, session-based aggregation scheme was set as default scheme).

There are many ways to realize inference rules. Here we presented a set of rules as an example. All of these inference rules are described below table 3.1.

Table 3.1 FSC inference rules

IF	(SL4,SL3,SL2,SL1,SL0)=	THEN	(AS5,AS4,AS3,AS2,AS1,AS0)=
Rule 1	(L,L,L,L,L)		(H,L,L,L,L,L)
Rule 2	(H/M,L,L,L,L)		(L,H,L,L,L,L)
Rule 3	(L/M,H/M,L,L,L)		(L,L,H,L,L,L)
Rule 4	(L/M,L/M,H/M,L,L)		(L,L,L,H,L,L)
Rule 5	(L/M,L/M,L/M,H/M,L)		(L,L,L,L,H,L)
Rule 6	(L/M,L/M,L/M,L/M,H/M)		(L,L,L,L,L,H)
Rule 7	(H,M,L,L,L)		(L,M,H,L,L,L)
Rule 8	(N,H,M,L,L)		(L,M,M,H,L,L)
Rule 9	(N,N,H,M,L)		(L,M,M,M,H,L)
Rule 10	(N,N,N,H,M)		(L,M,M,M,M,H)

Keywords like IF THEN represent operations on the fuzzy inputs, which gives the output of that particular rule. For example, (SL0, SL1, SL2, SL3, SL4) is a minimum operator, which selects the minimum of the five values that it operates on. At the end of the inference a resultant fuzzy set is obtained, which needs to be defuzzified to get a meaningful crisp output.

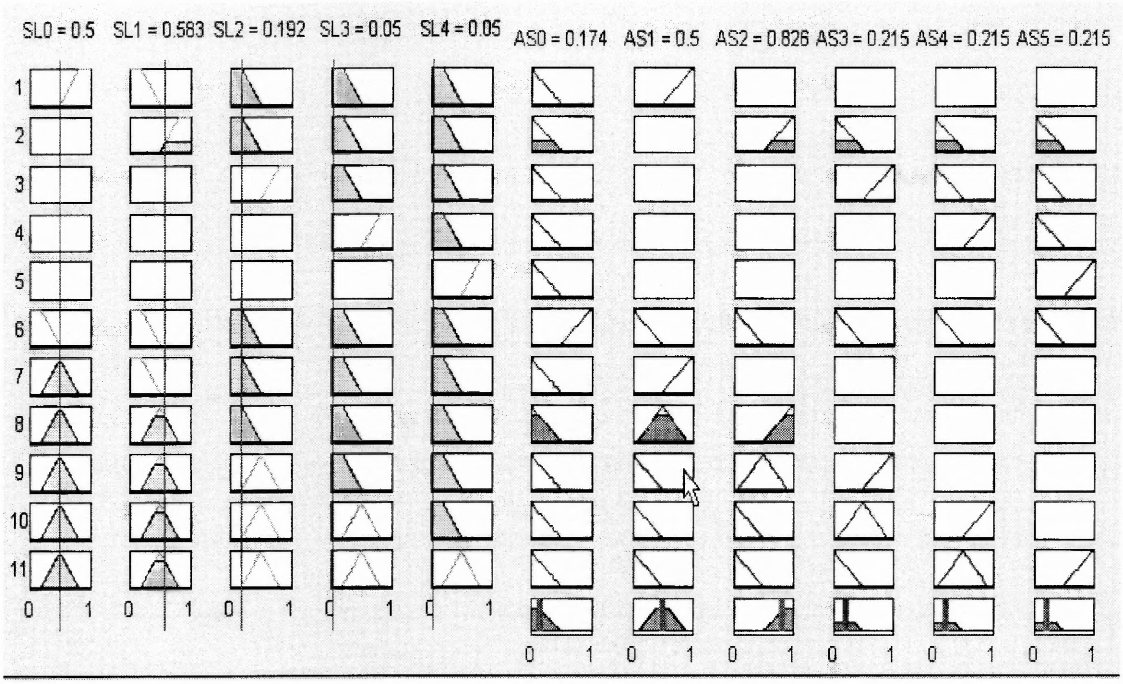


Figure 3.10 The pictorial representation of inference schemes

In figure 3.10, when the SL0=0.5 and SL1=0.583, the AS2 has the highest priority, which is 0.826.The FSC will activate the 16-aggregation scheme to group packets.

3.4 Adaptive Flow-based Aggregation Architecture Implementation

Adaptive flow-based aggregation architecture (AFAA) includes two main parts: Flow-based Aggregation Engine (FAE) and Fuzzy Supervisory Controller (FSC). The sample architecture of AFAA is shown in the figure 3.11.

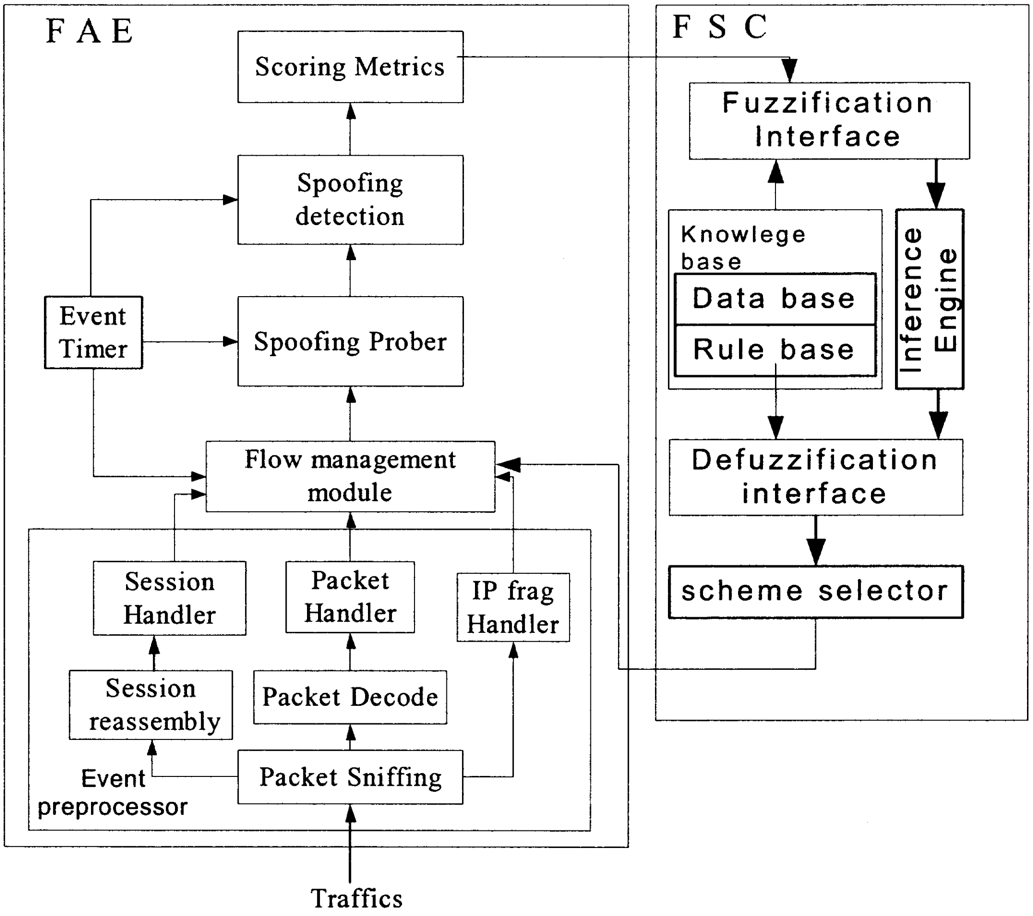


Figure 3.11 The adaptive flow aggregation architecture

Flow-based Aggregation Engine sniffs packets from network, decodes packets, groups all packets based on flow-based aggregation schemes, sets up flow-based statistical feature vectors. Moreover, flow here could be any set of packets sharing certain common property as “flow key”. The aggregation engine configures flow flexibly to provide security from network level to application level (IP, TCP, UDP, HTTP, FTP...), and different aggregation schemes, such as server -based, client-based flow. The fuzzy Supervisory controller will automatically activate the most appropriate flow aggregation scheme depending on spoofing level analysis.

Event Preprocessor: Collects the network traffic of a host or a network, Event handlers generate reports to Flow management module;

Event Time: Periodically calls Feature Extraction Module to converts the statistic information of flows into the format required by the statistical model.

Flow Management Module: efficiently determines if a packet is part of an existing flow or should generate a new flow key; According to different flow key, this module aggregates flows together based on their flow keys, and dynamically updates per-flow accounting measurements;

Spoofing Probe: Statistic flow numbers based on various aggregation schemes;

Spoofing detection: Periodically extracts the statistic flow number and calculate Aggregate Likelihood Ratio (ALR) or Spoofing Degree..

Scoring Metric: Calculates the probability scores of these spoofing degree by comparing the values with the reference model generated by past normal and flooding attack users. The probability scores are measurements indicating how likely for a spoofing degree to take the observed value.

Adaptive flow-based aggregation module can be plug in the flow-based Network Intrusion detection or lonely used for firewall to block spoofing DDoS.

3.5 Applications and Evaluations

In this section, the application of Adaptive Flow Aggregation Architecture (AFAA) in Flow-based Network Intrusion Detection (FNIDS) was presented, which is called adaptive flow-based NIDS. The evaluations of the application show that source IP aggregation level structure and flow population density/distribution detection mechanism improve the performance of NIDS under spoofed flooding packets.

In these experiments, 1998 DARPA Intrusion Detection Evaluation Data [19] was used, which contained both training data and test data. The training data consisted of 7 weeks of labeled network-based attacks inserted in the normal background data. The test data contained 2 weeks of network-based attacks and normal background data. The data contained spoofed bandwidth depletion attack, such as: smurf, SYN flood, and other low stress attacks, such as: ping of death and teardrop. These flooding attacks had spoofed source IP addresses. Since the amount of available DARPA'98 data was huge (e.g. some days have several millions of flow records), we sampled sequences of normal flow records in order to create the normal data set that had the same distribution as the original data set of normal flow. The normal data set was used for training a neural network. The flow records were collected from the first 5 weeks of training data (1491010 flow records), where 618677 data records were sampled that corresponded to normal connections, and used for the training phase. For testing purposes, the flow records associated with all the attacks from the first 5 weeks of data were used in order to determine detection rate. Also a random sample of 200000 flow records was considered that correspond to normal data in order to determine the false alarm rate. It is important to note that the sample used for testing purposes had the same distribution as the original set of normal connections. After the features are constructed and normalized, anomaly detection schemes were tested separately for the DoS attacks.

The performance of any intrusion detection system must account for both the detection ability and the false positive rate. Here, we used receiver operating characteristic (ROC) curves to compare intrusion detection ability to false positive. The ROC curve allows us to assess the trade-off between detection ability and false alarm rate in order to properly tune the system for acceptable tolerances. For each curve, the point at the upper left corner in a graph represents the optimal detection with high detection rate and low false alarm rate. The x-axis of the figure is the false alarm rate, which is the rate of the typical traffic events being classified as faults or anomalies; the y-axis of the figure is the detection rate, which is calculated as the ratio between the numbers of correctly detected faults/anomalies to their total number. Different aggregation schemes produce different ROC curves.

In order to compare the performance of adaptive flow-based NIDS with other aggregation scheme, we choose the session-based aggregation scheme as comparison. The system was evaluated on the following five cases: case 1 is only to detect attacks with spoofing level 5; case 2 is only to detect attacks with spoofing level 4; case 3 is only to detect all spoofed flooding attacks; case 4 is to detect all of both spoofed flooding and non-spoofed flooding attacks.

Case 1: detect attacks with spoofing level 5

Our purpose here is to show that the performance of FNIDS under flooding attacks with spoofed ports has been improved greatly after using spoofing detection algorithms. In Darpa 98's data, a Neptune attack sent 20 SYN packets to every port from 1 to 1024 of the Solaris server once every ten minutes and sending twenty SYN packets to a port on a Solaris 2.6 system will cause that port to drop incoming requests for approximately ten minutes. Each time it sent a lot of SYN packets to block multi ports of a server at the same time. For example, In Darpa data 98 on 6/18/1998, Neptune attack launched about 2840 SYN packets each time windows (30 seconds) to 140 ports at a server and In the next time window, Neptune attack will launched another 280 SYN packets with new spoofed source ports. Thus, the session-based NIDS must keep a lot of flow records (at least 2840) in each time window, but adaptive flow-based NIDS only keeps 140 flow records because the forged information has been removed so that the flow number does not be affected by spoofed ports.

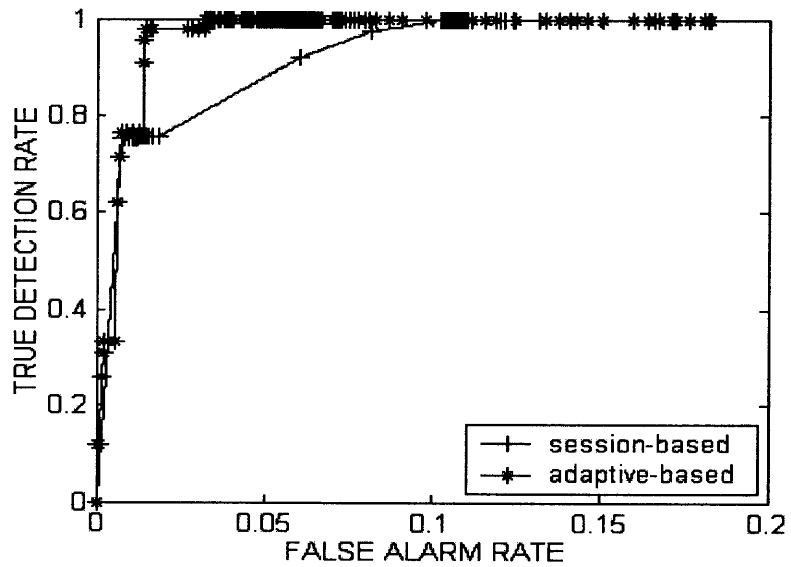


Figure 3.12 ROC curves under attacks with spoofing level 5

Figure 3.12 displays two ROC curves –the first one for session-based NIDS and the second one for adaptive flow-based NIDS, which were based on the first five weeks' data for training and used the last week's data for testing. From the figure 3.12, we can observe that under a lot of Neptune flooding attacks, the session-based NIDS apparently drop (about 78% detection rat when false alarm rate is 0.02.), but the adaptive flow-based NIDS still has very high detection rate (above 99%) when the false alarm is between 0.02 and 0.1.

Case 2: detect attacks with spoofing level 4

In this experiment, we will use ICMP flooding with spoofed level to evaluate the performance of FNIDS under flooding attacks with spoofed level 4 and to show that its performance has been improved after using spoofing detection algorithms.

In Darpa 98's data, a smurf attack sends ICMP "echo request" packets to the broadcast address (xxx.xxx.xxx.255) of many subnets with the source address spoofed to be that of the intended victim. Any machines that are listening on these subnets will respond by sending ICMP "echo reply" packets to the victim. In adaptive NIDs, IP spoofing detection agent can detect which subnet ICMP "echo reply" packets are from and scenarios them based on subnet address as well as server IP/port and protocol. For example: in Darpa 98 data on 6/17/1998, a smurf attack sent ICMP "echo request" packets to 8 subnets. More than 1040 machines send "echo reply" to the victim in each time window, and in next window. In Session-based NIDS, all flow records (over 1040) would be kept in machine storage, but in Adaptive NIDS, only 8 flow records will be kept in each time window, which greatly reduce the memory requirement.

The figure 3.13 shows that the performance of the adaptive flow-based NIDS is much better than the performance of session-based NIDS. The detection rate of adaptive flow-based NIDS is closed to 1 when false alarm rate is about 0.01.

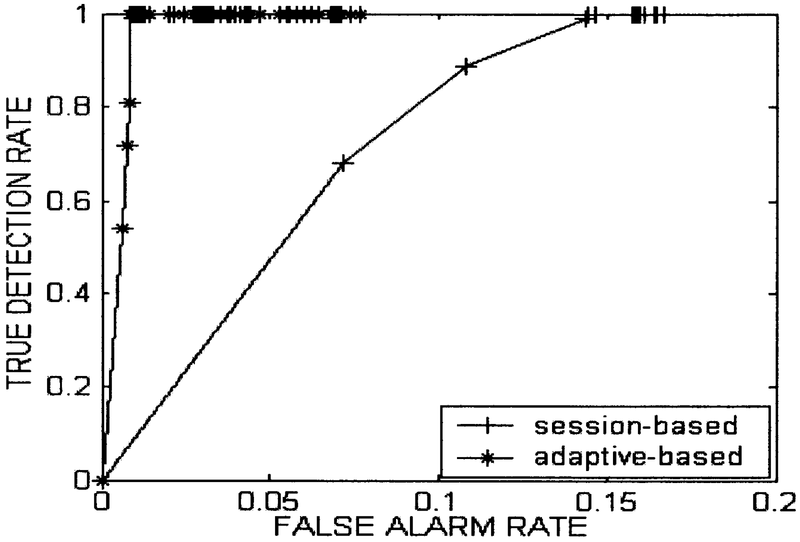


Figure 3.13 ROC curves under attacks with spoofing level 4

Case 3: Detect all spoofed flooding attacks

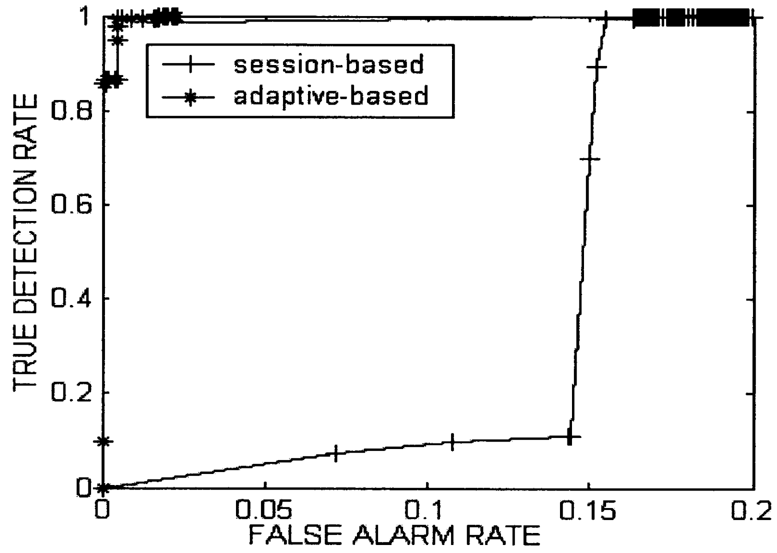


Figure 3.14 ROC curves under spoofed flooding attacks

This experiment was based on case 1 and case 2 to show that the performance of FNIDS under two kinds of spoofed flooding attacks (spoofing level 5 and spoofing level

4) has been improved after using spoofing detection algorithms. As described in case 1 and case 2, in Darpa 98 data, the "smurf" attack belongs to spoofing level 4 and the "Neptune" attack belongs to spoofing level 5. Figure 3.14 shows the performance of both session-based NIDS and adaptive flow-based NIDS.

From the figure 3.14, we can observe that the performance of FNIDS with SLDS is much better than the performance of session-based NIDS. The detection rate of adaptive flow-based NIDS is closed to 1.

Case 4: Overall detection of both spoofing attacks and non-spoofing attacks

In Darpa 98's data, except the spoofing attacks, like "Smurf" and "Neptune", there were other DDoSs, like pod, teardrop and back etc. In this experiment, we tried to show an overall performance comparison between adaptive flow-based NIDS and session-based NIDS.

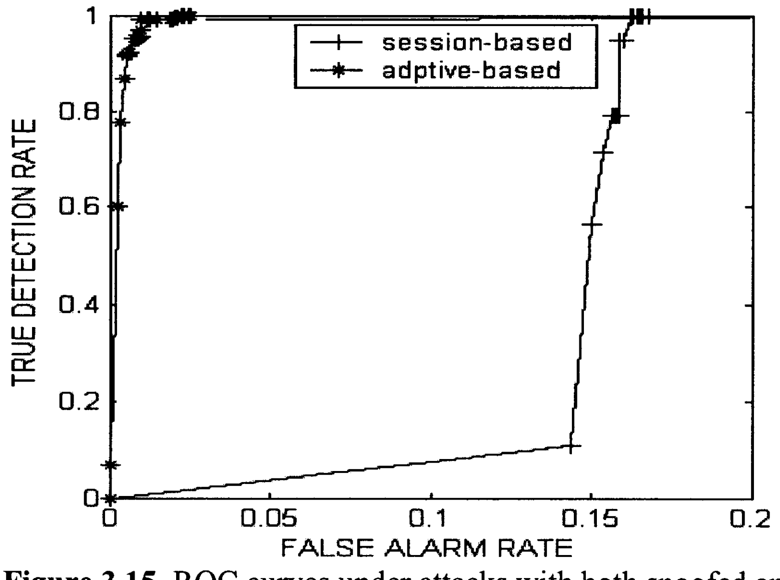


Figure 3.15 ROC curves under attacks with both spoofed and non-spoofed flooding

The figure 3.15 shows that the performance of FNIDS with SLDS is much better than the performance of session-based NIDS. The detection rate of adaptive flow-based NIDS is closed to 1.

Table 3.2 The Classification Results of Typical Flow Scheme on DARPA 98 Data

Number\ scheme	Session-Based scheme	Adaptive scheme
Samples Number	1423182	383010
Normal samples	449610	378630
Attack samples	973572	4380
Misc. Rate	0.18	0.00886
False Positive Rate	0.21	0.0089
False Negative Rate	0.26	0.00829

Table 3.2 shows that there were 973572 different attack flow records in the session-based Aggregation scheme, but there were 4380 attack flow records in adaptive flow-bas NIDS, which greatly reduced the memory requirements and processing time caused by spoofed flooding attacks. Apparently, session-based aggregation scheme took more then 222 times (973572/4380 ≈222.27) buffer units than the adaptive aggregation scheme needed for spoofed flooding attacks.

3.6 Summary

Adaptive Flow Aggregation Approach for Flow-based Defense System was presented in this paper, which only address the poor performance of the statistic flow-based NIDS under high-stress spoofed flooding, but also provide a fine-grained traffic classification model to recognizes and blocks spoofed flows for other flow-based defense system, like firewall or traffic control.. Moreover, we presented fuzzy logic method adaptively to select flow aggregation scheme to increase the detection rate depending on the spoofing

CHAPTER 4

A CONTROL THEORETICAL ANALYSIS FOR FLOW-BASED CONGESTION CONTROL TO MITIGATE DOS ATTACKS

Abstract— flooding-based distributed denial-of-service (DoS) attack presents a very serious threat to the stability of the Internet. However, current detection of the attack is unreliable and may have high false-positives. Rate-limiting is a better-suited response than complete filtering. Filtering out all the traffic to the victim would greatly damage misclassified flows, whereas rate-limiting still allows some packets to reach the destination and thus keeps connection alive. Allowing some attack packets through is acceptable, since the attack's overall impact depends on the volume of the attack packets. Moreover, if the flow-rate of low-priority is reduced, the high-priority flow will get more chances to access the server they share, which eventually reduce the congestion and improve the throughput of the high-priority flow. Guaranteeing Quality of Service (QoS) has been proposed using two different models-the integrated services (Intserv) and the differentiated services (Diffserv) model. Intserv uses the per-flow approach to provide guarantees to individual stream, but its scalability is double since per flow state information has to be kept in every router on the path from sender to receiver(s). DiffServ provides aggregate assurances for a group of application. But the lack of fine-grained service differentiation and resource isolation by current DiffServ routers exposes their vulnerability to Distributed Denial of Service (DDoS) attacks [36], causing a serious threat to the availability of Internet services. Based on the concept of flow-based aggregation, we present a flow-based congestion control architecture that provides fine-grained service differentiation and resource isolation among different classes of traffic

aggregates. The Flow-based Congestion Control (FCC) architecture consists of a Fine-grained Quality-of-Service (FQoS) regulator and PID controller. The whole system adopts a control-theoretic approach to adjust the flow rate of every link so as to maintain the high priority flow-rates at their desired level, thus guaranteeing QoS to high-priority flow. To realize Fine-grained Quality-of-Service (FQoS) regulator, we proposed a Multi-Level Packet Classification (MLPC), dynamical traffic models and flow rate-limit scheme. The flow-based network intrusion detection in chapter 2 is used to classify each flow in the network into different priority classes and give different treatment to the flow-rates belonging to different classes. The architecture is shown to be highly flexible service differentiation and robust against different types of flooding attacks, and traditional network traffic control can be implemented using one common framework. The fine-grained service differentiation and resource isolation provided inside the FCC is a powerful built-in protection mechanism to mitigate DDoS attacks, reducing the vulnerability of Internet to DDoS attacks. This system has been evaluated by using CONEX test-bed data. Results show the success that the system mitigates Distributed Denial of Service Attacks

4.1 Introduction

Internet-enabled business, or e-business, has mushroomed into a significant part of the US economy, yet the further advancement of e-business is plagued by various Quality-of-Service (QoS) and security problems. One of the worst is the Distributed Denial-of-Service (DDoS) attack, which aggregates junk data traffic from up to thousands of computers into a formidable volume and floods and effectively blocks a certain victim website. Bandwidth depletion can be caused by packet flooding, or by redirect amplification of attack packets. In a flooding attack, agents-zombies send large volumes of traffic to the victim to consume the victim system's bandwidth, whereas in amplification attack the broadcast router redirects and amplifies the attack to the victim. However, current detection of the attack is unreliable and may have false-positives; rate limiting is a better-suited response than complete filtering. Filtering out all the traffic to the victim would greatly damage misclassified flows, whereas rate limiting still allows some packets to reach the destination and thus keeps connection alive. Allowing some attack packets through is acceptable, since the attack's overall impact depends on the volume of the attack packets.

From the view of traffic flow (instead of packets), congestion control regulates the behaviors of flows in the Internet. Many rate-limiting algorithms have been proposed [41][49][50]. Aggregate-based Congestion Control (ACC) [39] is to minimize the immediate damage done by high-bandwidth aggregates. It includes a detecting mechanism, aggregate controlling mechanism, and a cooperative pushback mechanism in which a router can ask upstream routers to control an aggregate. Sally Floyd[42] argues that router mechanisms are needed to identify and restrict the flows that are using a disproportionate share of the bandwidth in times if congestion. Ratual Manhajan [40] proposed a mechanism name RED with preferential Dropping (RED-PD), which keeps partial flow state for the high-bandwidth flows. It uses the packet drop history at the router to detect high-bandwidth flows in times of congestion and preferentially drops

packets from these flows. By restricting high-bandwidth flows, it improves the performance of low-bandwidth flows.

As a response to the attacks, traffic control restricts the allowed sending rate to the victim of attack. Since the detection of the attack is unreliable and may false-positives, rate-limiting is a better-suited response than complete filtering. Filtering out all the traffic to the victim would greatly damage misclassified flows, whereas rate-limiting still allows some packets to reach the destination and thus keeps connection alive. Allowing some attack packets through is acceptable, since the attack's overall impact depends on the volume of the attack packets. Thus, we propose a Flow-rate Congestion Control (FCC) architecture that will use the flow-based network intrusion detection [45] to classify the traffic flows in the network into different priority classes and give different treatment to the flow-rates belonging to different classes, and adopted a control-theoretic approach to adaptively control the low-priority flows so as to maintain the high priority flow-rates at their desired level, thus guaranteeing QoS to high-priority flow. To provide this desired service guarantee to high priority flows, we set up network flow model and network flow rate-limit scheme. This system has been evaluated by using CONEX test-bed data. Results show the success that the system mitigates Distributed Denial of Service Attacks.

The rest of the paper is organized as follows: section 4.2 introduces the related work; section 4.3 discusses a Fine-grained QoS regulator design; Section 4.4 describes the Controller design; Section 4.5 explains experiments and results; Section 4.6 draws some conclusions and outlines future work.

4.2 Related work

Today there are many studied [38][43]~[60] on solving the congestion problem of a network caused by DDOS flooding. However, they rarely achieve complete blocking of attack traffic in general conditions. Meanwhile, the DDoS attack is likely to remain as a critical threat for the public servers. To thwart DDoS attacks and provide service differentiation at victim servers, sophisticated resource management schemes at end-servers have been proposed, such as Resource Containers, WebQoS, and Escort [65], [67], [67], [68], [69]. Like the computing resources of an end-server, there is only a limited amount of network resources—such as bandwidth, buffer, and processing power of routers—available to Internet users. The vulnerability of the Internet to DDoS attacks roots in its best-effort model that provides no resource isolation among different IP flows, making it easy for attacking traffic to hog network resources. Having sufficient service differentiation and resource isolation at IP routers is essential not only to provide network Quality of Service (QoS) to end-users, but also to counter DDoS attacks as a powerful built-in protection mechanism inside the Internet.

To support network QoS, the Differentiated Service (DiffServ) infrastructure [42] has been proposed as a promising solution due mainly to its scalability and robustness. Based on the DS field in the IP header, IP flows are classified into different Behavior Aggregates (BAs).Services are provided for aggregates, not for individual flows, and defined by a small set of Per-Hop Behaviors (PHBs), which are the forwarding behaviors applied to different aggregates at IP routers. According to the three different services provided by DiffServ, three types of PHBs are specified: Expedited Forwarding (EF), Assured Forwarding (AF), and Best-Effort (BE). Although EF traffic is strictly policed and conditioned at edge routers, which protects the rest of the network resources from the flooding EF traffic, the violated AF traffic is only remarked without strict policing at network edges. More importantly, no conditioning is applied to BE traffic, which is the main component of the Internet traffic. Compared with the best-effort service model, DiffServ is more resilient against DDoS attacks, but it is still susceptible to DDoS attacks, especially the BE flooding traffic. Because in the current DiffServ architecture, the QoS classification at core routers depends solely upon the DS field in the IP header, yielding only coarse-grained service differentiation and resource isolation. No further service differentiation and resource isolation are provided among different transport-layer protocols within a BA. On the other hand, UDP and TCP are two dominant transport-layer protocols in the current Internet, but their services and traffic behaviors are quite different. Furthermore, UDP and ICMP flooding attacks have been widely used for stealing network bandwidth and disabling a victim server. It is necessary to provide resource isolation among TCP, UDP, and ICMP traffic and the resource consumption of UDP and ICMP traffic should be bounded. Besides meeting the requirement of the bi-directional service differentiation to TCP sessions, which the current DiffServ fails to achieve [70], [71], there are some reasons for differentiating TCP control segments—SYNs, FINs, ACKs, and RSTs—from data segments, especially in the best-effort service model. Usually, TCP control segments have much smaller packet size than data segments, so they consume much less network bandwidth than data segments. The loss of a TCP control segment, especially SYNs, incurs more serious performance degradation than the loss of a data segment. DDoS attack tools usually utilize TCP control segments for generation of DoS attacks, such as TCP SYN and ACK flooding attacks. In other words, the coarse-grained service differentiation and the lack of resource isolation on metadata packets not only degrade the assured service of TCP sessions, but also expose the vulnerability of Internet to DDoS attacks [72].

Recent literatures have shown that the control theory can be successfully applied to rate control in ACM[50][51], and high-speed network infrastructure[40][45] respectively. They demonstrated the effectiveness of feedback control mechanisms in achieving predictable system performance without precise knowledge of worst-case load patterns. Binary feedback control [42] is widely used in network traffic control for its simplicity and efficiency. It relies upon a one-bit indicator in each probing packet to determine the congestion status of a network resource. The key design issue here is the threshold value(s) to set and unset the indicator bit. Mitigating DDoS Attacks using a PID Controller [44] uses a filtering method based on Proportional-Integral-Derivative (PID) control theory to predict traffic flow change in each border router. Since it is based on numerous assumptions and only limited to particular types that it becomes impractical in real network.

In summary, all the schemes up to now can punish a small number of large unresponsive traffic flows to maintain a certain level of fairness. However, no scheme works well during the time of DDoS attack, which is characterized by a large number of small traffic.To some degree, both filtering-based and congestion control techniques will alleviate the anguished symptom of DDoS attacks, but they may inevitably block some legitimate traffic, the more effective methods still further research.

4.3 Fine-grained QoS Regulator Design

In the previous section, we described that the diffserv architecture can counter DDoS attacks in some degrees, but it is still susceptible to DDoS attacks, especially the BE flooding traffic. Because in the current DiffServ architecture, the QoS classification at core routers depends solely upon the DS field in the IP header, yielding only coarse-grained service differentiation and resource isolation.

To provide transport-layer fine-grained service differentiation and resource isolation, we will present a novel fine-grained QoS regulator (FQoS), which extracts transport-layer information to further divide a behavior aggregates into a UDP aggregate, a TCP aggregate, and an ICMP aggregate and, then, to distinguish IP Flags in the TCP aggregate or source IP/port or QoS etc. The granularity of the regulator deploys Multi-Level Packet Classification detailed in section 4.3.3, which is based on flow-based aggregate schemes in chapter 2, and uses Adaptive Flow Aggregation Architecture detailed in chapter 3 to defense spoofed flooding attacks. The fine-grained QoS regulator drops packets from traffic streams at transport-layer. This filtering approach is based on flow rate limits, where the rate limits dynamically varies according to the output of controller (see section 4.4), and the output (maliciousness) of flow-based network intrusion detection system to identify the malicious nature of traffic streams. Setting the limit rates for the packets from the suspected malicious flows and dropping packets reaching in the next window can adaptively limit low priority (high possible malicious) flows while maintaining high priority (normal) flows at a desired level so that full utilization of network bandwidth can be achieved through adaptive flow control.

4.3.1 Bandwidth and Regulation

QoS regulator is used to balance or protect server's resource, like network bandwidth, protocol state memory buffer, kernel interrupt processing, memory and CPU cycles. Some of these resources can be protected by rate-based schemes, such as: traffic shaping or rate control. Traffic control is a means to achieve the required QoS goals, including bandwidth management.

There are many mechanisms that can be used for traffic control including input firewall, ingress router, incoming link or at server. In this chapter, a combination of rate-control and control-theoretical method is used, which has a better control over bandwidth resource. Figure 4.1 shows a typical topology of Flow-based Congestion Controller (FCC). The advantages of this approach are that it does not require any modification on the server farm being protected. By separating the QOS regulator from the end servers, the regulator also relieves the end servers of expensive interrupt processing of attack packets [58]. Moreover, with the availability of network processors and other ASICs specially designed for classifying and handling network traffic [61], the FCC can be easily made to operate at line-speeds.

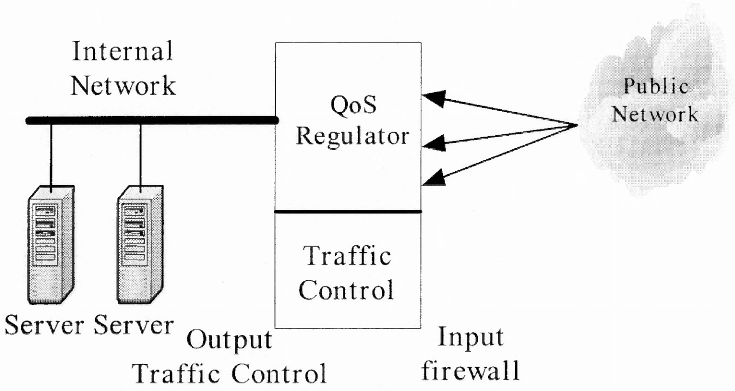


Figure 4.1 The flow-based congestion controller (FCC) architecture

This approach deploys flow-based aggregate schemes detailed in chapter 2 and Multi-Level Packet classification in the section 4.3.3 at the QoS regulator to support fine-grained service differentiation and resource isolation, and establishes a flexible fine-grained QoS regulator to reduce the impact of attacks on the end servers. In order for the QoS regulator to function as desired, it needs to have a quantitative idea of the bandwidth resources available in the end servers and the level of bandwidth consumption by different packets, flows or traffic classes. Therefore, there is a need to set up network rate-limit models. At the same time, it is necessary to *map* or reflect incoming packets to flow aggregates, and controlling flows at transport-layer. This leads us to the creation of *traffic classes*, where each traffic class is supposed to correspond to a certain kind of flow.

The whole system adopts a control-theoretic approach to adjust the flow rate of every link so as to maintain the high priority flow-rates at their desired level, thus guaranteeing QoS to high-priority flow. By regulating bandwidth consumption, attacks (like Distributed Denial of Service) can be effectively limited. Figure 4.1 shows how output traffic control is implemented on the QoS regulator.

4.3.2 Flow Aggregation Algorithms

The goal of a flooding DDoS attack is to send a large amount of traffic to exhaust a target bandwidth so that legitimate users cannot access the server. The offending traffic can be characterized as an aggregate of packets. A traffic aggregate is defined by matching the fields in the packet headers (at IP, UDP, TCP, and/or application layers) against a set of values. For example, the traffic aggregate for a SYN attack consists of all SYN packets to a destination address/port pair, and the aggregate for a Smurf attack consists of all ICMP echo-reply packets from a subnet to a destination address, blocking the SYN packets to the destination can stop the SYN attacks or blocking ICMP packets to the destination also can stop ICMP attacks.

The object of flow aggregation is to categorize packets by applying the header fields of a packet. For simplify, the fields in our system typically used to classify IP packets are the destination IP address, source IP address, destination port number, source port number, protocol number and protocol flags. The flow key (FK) is denoted by FK=[Destination IP, Source IP, Destination Port, Source IP, Protocol]. Using these fields for classifying IP packets, a flow aggregation scheme specifies a flow key, for example, FK = (192.168.10.120, *, 23, *, TCP), matching traffic addressed to subnet 192.168.10.1 using TCP protocol and destination port 23, which is used for incoming Telnet A firewall may disallow Telnet into its network using a filter to block the flow with this flow key. The flow key (FK) is an array of N values in program, where H [i] is a specification on ith header field. The value H [i] specifies what i-th header field of a packet must contain in order for the packet to match the flow key. In Linux 'iptable', these flow keys form a chain array seen in figure 4.3. These specifications often have the following forms: exact match, for example "source address must equal 192.168.10.16"; prefix match, like "destination address must match prefix 192.68.10. * "; Or range match, e.g. "destination port must be in the range 0 to 1023". If specifying Server's port (destination IP is home net) and protocol, flow aggregation mechanism can monitor the specific services, such as: using TCP protocol and server's port 23, which is used for incoming Telnet. Therefore, flow aggregation algorithms provide a flexible service differentiation and resource isolation at layer-4. For instance, let FK = (192.168.10.* , *, *, 23,TCP) be a flow key, then, a packet with header(192.168.10.120, 10.10.10.1, TCP, 23, 1025) matches FK, and is therefore aggregated. The packet (192.168.10.120, 10.10.10.1, 21, 1024, TCP), on the other hand, doesn't match.

From the view of protecting bandwidth from flooding attacks and reducing the number of flow keys, the flow keys of the Fine-grained QoS regulator are based on the destination address as well as on the port and protocol at the transport or higher layer. Therefore, the forwarding database of a router keeps some filters to be applied on flow keys. Since flow key is based on server side, the number of these filters is not too large to influence the performance of QoS regulator.

4.3.3 Multi-Level Packet Classification

Internet routers that operate as firewalls, or provide a variety of service classes, perform different operations on different flows. In order to classify a packet, a router consults a table (or classifier) using one or more fields from the packet header to search for the corresponding flow. The classifier is a list of rules that identify each flow and the actions to be performed on each flow. With the increasing demands on router performance, there is a need for algorithms that can classify packets quickly with minimal storage requirements and allow new flows to be frequently added and deleted. In this system, we adapted the multi-level packet classification architecture to manage these rules.

For service differentiation, a Fine-grained QoS regulator is proposed here, which employs a multi-stage packet classification mechanism. Similar to Figure 3.3, Packet classification can be modeled at multi different hierarchical levels, as shown in Figure 4.2. The deeper level of packet classification has the finer grain of flow aggregation. There are two kinds of information from IP header for packet classification: one is reliable and another unreliable. The unreliable information includes source IP/Port that is often forged by attacker, or some subnets used for amplification attack or reflection attacks. The other information is reliable information. As the key component of the multi-level packet classification architecture, the proposed QoS regulator at routers uses both kinds of information for packet classification. In order to remove forged information to reduce flow number caused by spoofed flooding attacks, adaptive flow aggregation architecture in chapter 3 can be used in QoS regulator. Except that IP-layer reliable information like destination IP/port aggregation can be used for level 1~2 packet aggregates as shown in figure 4.2, transport-layer information also can be extracted to further divide a behavior aggregates into a UDP aggregate, a TCP aggregate, and an ICMP aggregate for level-3 packet aggregation and, then, to distinguish IP Flags in the

TCP aggregate for level-4 packet aggregation etc. Routing and QoS lookups are integrated into a single framework to fulfill layer-4 packet forwarding and, therefore, the forwarding database of a router consists of many filters to be applied on multiple header fields.

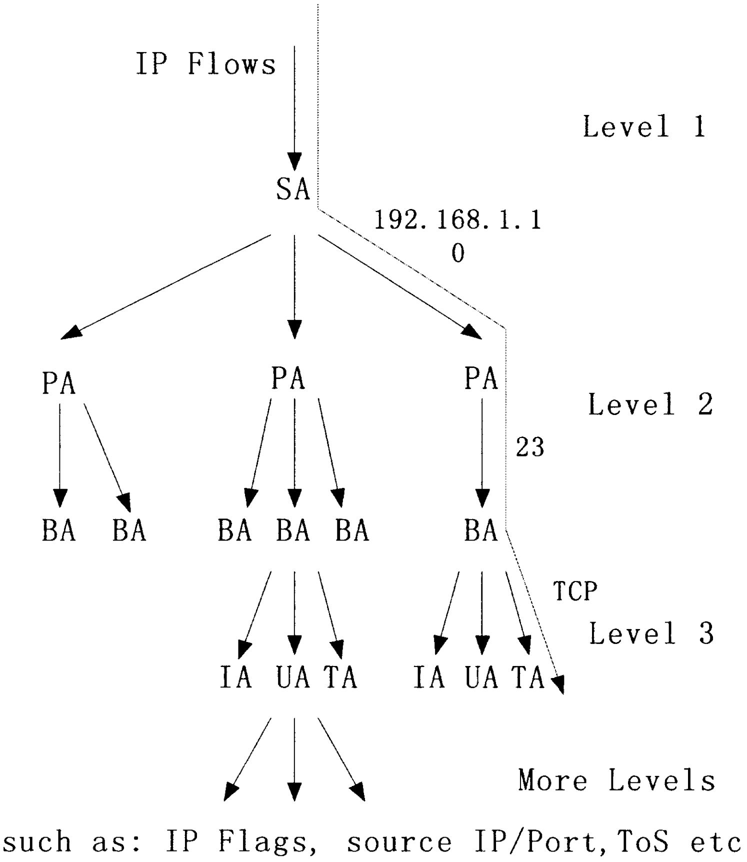


Figure 4.2 The multi-level packet classification

In figure 4.2, SA means server aggregates; PA means port aggregates; BA means Behavior Aggregates; TA, UA and IA are TCP, UDS and ICMP aggregates separately. At the first two levels, it is straightforward to set the filtering rules. By checking destination (server) IP address/port and protocol type fields in the IP header against the filtering rules, the QoS classification is simple and does not cause any ambiguity. Moreover, a general TCP control segment identification (Flags) can be used for the identification of each individual TCP control segment, such as: SYN, FIN, ACK, and RST. The only difference at the port-based identification is to check different bits in the 6-bit flag field.

4.3.4 Traffic Rate-limit Algorithms

As described above, an IP flow is a set of packets, which are observed in the network within some time period, and share some common property known as its key. It can be a TCP connection or a UDP stream described by source and destination IP addresses, source and destination port numbers, or the protocol number etc. For the fine-grained QoS regulator, flow key is based on the destination address as well as on the port and protocol at the transport or higher layer. Therefore, the forwarding database of a router consists of a large number of filters to be applied on flow keys.

Based on the Multi-level Packet Classification and flow aggregation schemes, all packets can be classified as different flows. Limiting traffic rate of each flow can solve the network congestion problem. Therefore, It is necessary to establish a flow-based traffic model for QoS regulator.

4.3.4.1 Traffic Model

Let T denote the discrete time unit, and assume that fluid approximation for flow rates holds. Consider a network that consists of a set £= {1, 2...L} of unidirectional links of capacity c_k, $k \in L$. Each link is connected to a server. Each flow must go through one of these links to reach its corresponding sever. We will assume that the data streams are lossless across links. The set of all flows is denoted by S, a subset of s. The incoming flow rate F_k on the kth link uses a set of flows $F_k := \{f_k^{(s)} \mid s \in S\}$. The rate F_k satisfies $m_k \leq F_k \leq M_k$, where $m_k \geq 0$ and $M_k \leq C_k$ are the minimum and maximum transmission rates, respectively.

Hence, we have:

$$F_k(nT) = \sum_{s \in S} f_k{}^{(s)}(nT) + m_k \tag{4.1}$$

The following constrains on the vector $F_k := \{f_k^{(s)} \mid s \in S\}$ of the incoming flow rate:

$$\begin{aligned} f_k^{(s)} &\geq 0, & \forall\, \text{s} \in \text{S}, \\ F_k &\leq C_k, & \forall\, k \in L. \end{aligned} \tag{4.2}$$

Note that: for a given vector $C := \{C_k \mid k \in L\}$ of capacities, no F_k of rates might exist if $m_k > C_k$ for any k. Therefore, we further assume that the vector C of capacities satisfies

$$C_k > m_k, \quad \forall k \in L$$

A pair of vectors (F_k, C) satisfying these constraints is said to be feasible.

Let F (nT) denote total network flow rate going through the network backbone during the time window [(n-1)T, nT]. Then,

$$F(nT) = \sum_{k=1}^{L} (F_k(nT)) \tag{4.3}$$

In order to protect all of links or control network traffic, we assume that all traffic must go through a network backbone. N_T denotes the capacity of the network backbone. Then, the maximum network traffic rate is denoted by:

$$F(nT) = \min\left\{N_T, \sum_{k=1}^{L} c_k\right\} \tag{4.4}$$

The goal of implementing any network traffic rate adjustment scheme is to show that the schemes can provide a mechanism, which will allow for the provisioning of the network traffic rate requirements in a dynamic traffic environment.

Because the QoS regulation algorithm is based on flow-based aggregation, the output (maliciousness) of flow-based Network Intrusion Detection can be used for the QoS regulator adaptively to limit low priority (high possible malicious) streams while maintaining high priority (normal) streams at a desired level so that full utilization of network bandwidth can be achieved, and prevent servers from overloading or being flooded.

4.3.4.2 Flow Rate-limit Schemes

In order to limit the traffic rate of a flow in a link, a flow rate-limit is introduced, which is described as following:

Suppose $\lambda_k^{(s)}(nT)$ is the incoming byte (or packet) number and $\nu_k^{(s)}(nT)$ is the outgoing byte number during the period [(n-1)T,nT] in the sth flow ,then the admitted portion of bytes(or packet) in a flow can be defined by:

$$\alpha_k^{(s)}(nt) = \frac{\nu_k^{(s)}}{\lambda_k^{(s)}} \tag{4.5}$$

Where: k∈£ and $0 \le \beta \le 1$.

Let the sth incoming flow rate (byte rate or packet rate) be denoted by:

$$\psi_k^{(s)}(nT) = \frac{\lambda_k^{(s)}(nT)}{T} \tag{4.6}$$

And let the sth outgoing flow rate (byte rate or packet rate) be denoted by :

$$f_k^{(s)}(nT) = \frac{v_k^{(s)}(nT)}{T} \tag{4.7}$$

Then we have:

$$\alpha_k^{(s)}(nt) = \frac{f_k^{(s)}(nT)}{\psi_k^{(s)}(nT)} \tag{4.8}$$

Where: α is said to be a byte counting rate ratio or packet counting rate ratio. When α =0 or 1,all bytes or packets are dropped or accepted at the sample time nT. Because both byte counting and packet counting rate belong to flow level, we call α as flow rate-limit or flow weight. For example: there is a flow, which contains 1000 packets and each packet has the fixed packet size (53bytes) at a sample time. If the flow rate-limit $\alpha^{(s)}$=0.7,then there are 700 packets (37100bytes) allowed to go through the QoS regulator and others will be dropped at this sample time.

Substitute (4.8) into (4.1); the flow rate in the kth link is expressed by:

$$\begin{aligned} F_k(nT) &= \frac{1}{T} \sum_{s \in S}^{k} \left(\alpha_k^{(s)}(nT) \times \lambda_k^{(s)}(nT) \right) + m_k \\ &= \sum_{s \in S}^{k} \left(\alpha_k^{(s)}(nT) \times \psi_k^{(s)}(nT) \right) + m_k \end{aligned} \tag{4.9}$$

And the flow rate is constrained by:

$$F_k \leq c_k, \text{ k=1,2,.....L} \qquad (4.10)$$

Where: c_k is the capacity of the link k.

In flow rate-limit scheme, the system timely monitors the traffic rate in the link and feedbacks the information to the regulator and predict every flow limit rate or flow weight to support a good rate regulator as shown in figure 4-3.

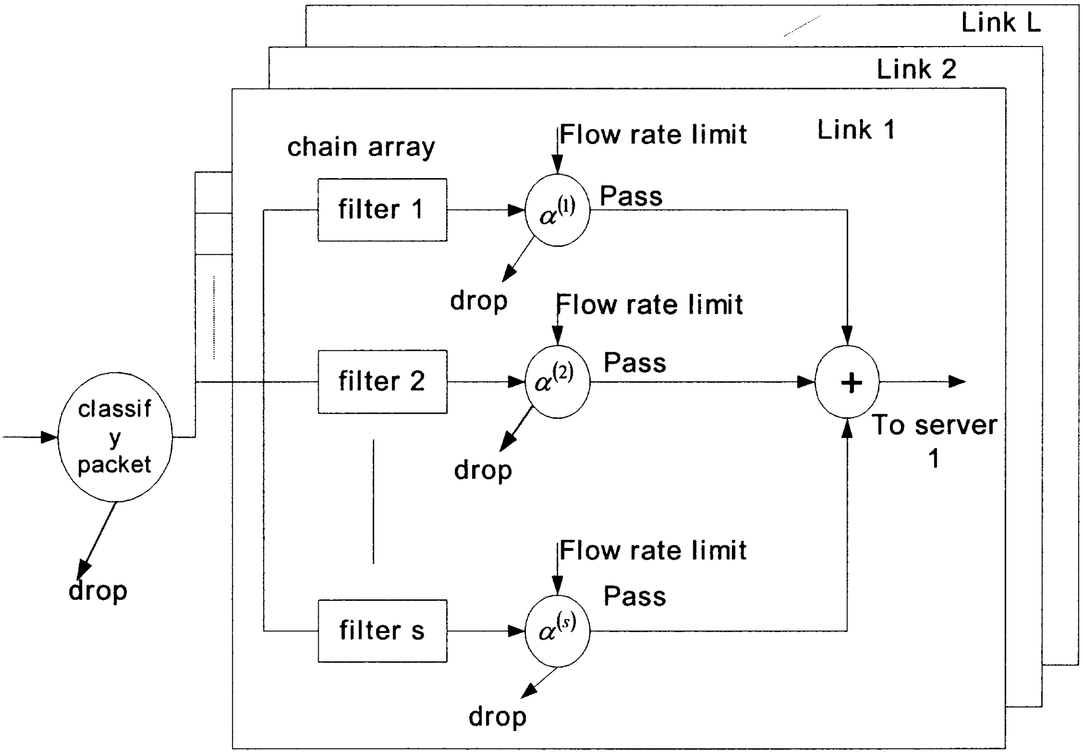


Figure 4.3 Flow rate regulators for servers

Figure 4.3 shows flow rate regulators for servers. There are L servers, thus L links is shown in this figure. In each Link, there are s filter ,which correspond to a series of flow keys. After entering flow regulator, each packet is classified based on flow keys. The packets with unknown flow key would be discarded. Each kind of packets will be transferred to the relevant filters. All filters form chain arrays. In hardware design, chain arrays are made of ASIC chips (Application-Specific Integrated Circuit, a chip designed for a particular application), which work parallel to deal with packets that greatly improve process ability of the QoS regulator. In software design, the chain arrays can be implemented by 'iptable' or 'tc' instruction in Linux system. Therefore, these chains could be statistic (predefined) or dynamic (provided an updated by a flow management module). In our implement, we used software to realize dynamical chain array.

4.3.4.3 Maliciousness-based Rate-limit Control Algorithms (MRCA)

In the above section, the traffic regulator models were discussed. Although these models can be used to control network traffic including flooding DDoS, it cannot maximally provide bandwidth resource for normal clients. In order maximally to block malicious flows while maintaining normal flows at a desired level, Flow-based Network Intrusion detection was introduced into FCC to realize MRCA.

Based on formula (4.9) and figure 4.3, we introduce a set of maliciousness factor, denoted by $m_k := \{m_k^{(s)} \mid s \in S\}$, which expresses the maliciousness degrees of every flow in the link k. The more malicious flow is, the lower priority the flow has to pass through QoS regulator. If the flow-rate of low-priority is reduced, the high-priority flow will get more chances to access the medium they share, which eventually reduce the congestion and improve the throughput of the high-priority flow. Thus maliciousness-based rate-limit control is an effective means to provide service differentiation to different class of flow. Thus, the general form of the maliciousness-based rate-limit function can be defined by:

$$\eta_k^{(s)}(nT) = Q(m_k^{(s)}(nT), p_k^{(s)}(nT)) \quad ;s \in S \qquad (4.11)$$

Where: Q is the feedback flow control function, which uses discrete form. "nT" is sample time, which corresponds to nth time window. "p_k" is a output of controller, which correspond to rate-limits.

Although formula (4.11) provides a general traffic control methods, it requires too many controllers to provide rate limits. In order to simplify the control method, one controller is used in our system, which is:

$$p_k^{(s)}(nT) = \omega_k^{(s)} \times \mu_k(nT) \;;s \in S \qquad (4.12)$$

From (4.11), we have:

$$\eta_k^{(s)}(nT) = Q\left(m_k^{(s)}(nT), \omega_k^{(s)}, \mu_k(nT)\right) \;;s \in S \qquad (4.13)$$

Where: ω is a constant weight vector for each flow. The default value is 1. μ is PID controller's output. In addition to reducing the number of controller, there are several other benefits by using (4.13) as follows:

- Using one single-loop control system can realize link bandwidth control;
- Different weight can be used to emphasize different service;

Many functions could be used to model the formula (4.13). Such as: the incomplete gamma function, Hyperbolic tangent function, Inverse tangent function etc. In this system, the maliciousness-based flow rate-limit function is denoted by:

$$\eta_k^{(s)}(nT) = \omega_k^{(s)} \times \left(1 - \frac{2 \times \mathrm{atan}\left(\mu_k(nT) \times (2 - m_k^{(s)}(nT)\right)\right)}{\pi}\right) \qquad (4.14)$$

Or $$\eta_k^{(s)}(nT) = \omega_k^{(s)} \times \left(1 - \tanh\left(\mu_k(nT) \times (1 - m_k^{(s)}(nT)\right)\right)\right) \qquad (4.14')$$

Where: k = 1,2...L and $s \in S$. S is the set of all flows in this link. $\eta_k(nT) := \{\eta_k^{(s)}(nT) \mid s \in S\}$ is a vector of rate-limits in the link k. The condition $m_k^{(s)}$ is the maliciousness of the s^th^ flows in the link k at the n^th^ time window, which is provided by flow-based network intrusion detection and it satisfies $|m_k^{(s)}| \le 1$.

If supposing that each flow has the same weight (ω =1), the vector expression of (4.14) and (4.14') can be written by:

$$\eta_k^{(s)}(nT) = 1 - \frac{2 \times \text{atan}\left(\mu_k(nT) \times (2 - m_k^{(s)}(nT)\right)}{\pi} \qquad (4.15)$$

Or

$$\eta_k^{(s)}(nT) = 1 - \tanh\left(\mu_k(nT) \times (1 - m_k^{(s)}(nT))\right) \qquad (4.15')$$

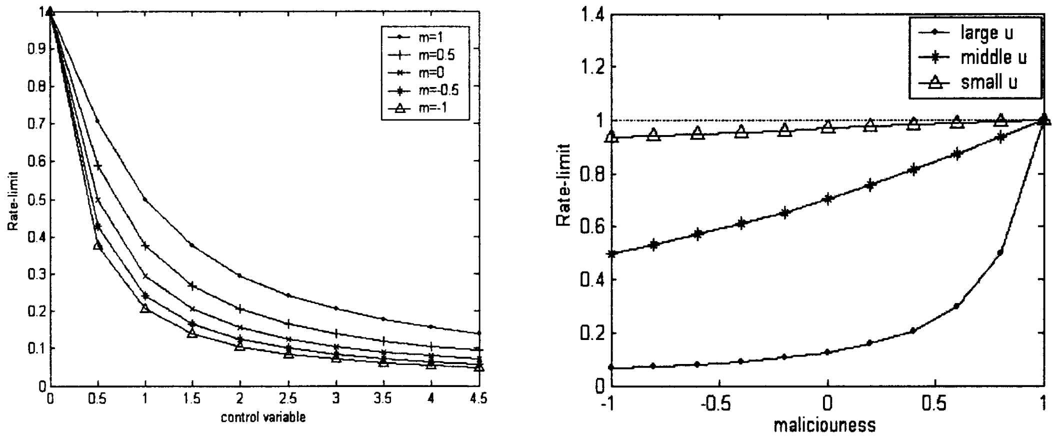
Special case: when m_k = I means that there are no malicious packet or FNIDS is failed, the (4.15) is written by

$$\eta_k^{(s)}(nT) = 1 - \frac{2 \times \text{atan}\left(\mu_k(nT)\right)}{\pi} \qquad (4.16)$$

The formula (4.16) is used for traditional traffic control.

Flows that have more maliciousness are quickly restricted to very low rates whereas this restriction is more gradual for better-behaving flows. The figure 4.4 shows the relationship between the rate-limit and control variables and the relationship between the rate-limit and maliciousness.

From the Figure 4.4, it is can be seen that: (1) the rate-limit decreases as control variable increase, which means that when network traffic of line is over the expected rate, the controller will give out a positive value (control variable) and rate-limit will decrease as control variable increases in order to limit the traffic going through this link; (2) the rate-limit decreases as maliciousness decreases, which proved that the formula (4.14) or (4.15) has the function of maximal dropping malicious packets.



(a) by control variable; (b) by maliciousness

Figure 4.4 The performance of rate-limit η

4.4 Controller Design

We have developed network traffic models for rate-based QoS regulator in section 4.3. In figure 4.1, we can see that the system consists of QoS regulator and traffic controller. In this section, we will further discuss control theoretical approaches for traffic controller.

There are many methods for controller. In our system, we choose PID control as the controller for the following reasons. (1) In term of control theory, the network traffic control system is a dynamic system; it is known that PID control is a widely applicable control technique in dynamic systems. (2) Compared with other control techniques, an important feature of PID control is that it does not require a precise analytical model of the system being controlled.

4.4.1 PID Controller Design

A conventional PID regulator is the most frequently control element in the industrial world. It is estimated that, at least, the 90% of the controllers employed in the industry are PIDs or its variations.

A conventional PID controller is described by equation (4.17). The control signal incorporates three actions on the error signal: proportional (process gain), integral (transient response) and derivative (stationary error). Each of them is modified by a constant, which are respectively called proportional gain *Kp*, integral time *Ti*, and derivative time *Td*. They are also named proportional,

In figure 4.13, the feedback controller (PID) is designed to generate an output µ that causes some corrective effort to be applied to a process so as to drive a measurable process variable Y towards a desired value R known as the set point. The controller uses an actuator to affect the process and a sensor to measure the results. Virtually all feedback controllers determine their output by observing the error e between the set-point (R) and a measurement of the process variable (Y). Errors occur when a disturbance or a load on the process changes the process variable. The controller's mission is to eliminate the error automatically.

4.4.1.1 PID Algorithms

Consider the ideal Proportional-Integral-Derivative (PID) controller written in the continuous time domain form:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} + u_0 \qquad (4.17)$$

Where: K_p, K_i and K_d are weighting constants; u_0 is an offset; $\alpha(t)$ is the controller output.

$$e(t) = R - c(t) \tag{4.18}$$

Where: R is the target incoming traffic (or set point). C(t) is the observed coming traffic. The variable e(t) represents the tracking error, the difference between the target incoming traffic R and the observed incoming traffic C(t). This error signal e(t) is sent to the PID controller, and the controller computes both the derivative and the integral of this error signal. The signal u(t) just past the controller is now equal to the proportional gain (Kp) times the magnitude of the error plus the integral gain (Ki) times the integral of the error plus the derivative gain (Kd) times the derivative of the error.

The above equation (4.17) is a continuous representation of the controller and it must be converted to a discrete representation and the final form of the equation is. To discrete the controller, we need to approximate the integral and the derivative terms to forms suitable for computation by a computer. From a purely numerical point of view, we can use:

$$\frac{de(t)}{dt} \approx \frac{e(nT) - e(nT-1)}{T} \quad \text{and} \quad \int_0^t e(t)dt \approx T\sum_{i=0}^{nT} e(i)$$

Therefore, the discrete PID algorithm is denoted by :

$$u(nT) = K_p e(nT) + K_i T \sum_{i=0}^{nT} e(i) + K_d \frac{e(nT) - e(nT-1)}{T} + \mathrm{u}_0 \tag{4.19}$$

This is now in the form of a difference equation, suitable for coding in an appropriate programming language.

From (4.19), the final form of PID controller is denoted by:

$$u(nT) = u(nT - 1) + K_p[e(nT) - e(nT - 1)] + K_iTe(nT) + \frac{K_d}{T}[e(nT) - 2e(nT - 1) + e(nT - 2)] \tag{4.20}$$

If K_d =0, the PI controller is denoted by:

$$u(nT) = u(nT - 1) + K_p[e(nT) - e(nT - 1)] + K_iTe(nT) \tag{4.21}$$

If K_i =0, the PD controller is denoted by:

$$u(nT) = u(nT - 1) + K_p[e(nT) - e(nT - 1)] + \frac{K_d}{T}[e(nT) - 2e(nT - 1) + e(nT - 2)] \tag{4.22}$$

Where: T is the sampling period or Time Window.

4.4.1.2 PID Tuning and Stability

An important task of building a stable and high performance PID-based rate controller is to tune the control parameters (i.e., K_p, K_i and K_d). One of the traditional ways to design a PID controller was to use Ziegler and Nichols method, which gave two techniques for selecting the PID parameters. The advantage of this method is that no knowledge of the system model is needed; all the information required to choose the parameters is obtained from simple experiments on the open-loop system. However, this method is usually used for linear time-invariant system. In network system, it is very difficult to precisely model real network traffic. Therefore, we used empirical tuning rules to determine the PID parameters.

Proportional

Proportional control is the easiest feedback control to implement, and simple proportional control is probably the most common kind of control loop. A proportional controller is just the error signal multiplied by a constant and fed out to the drive.

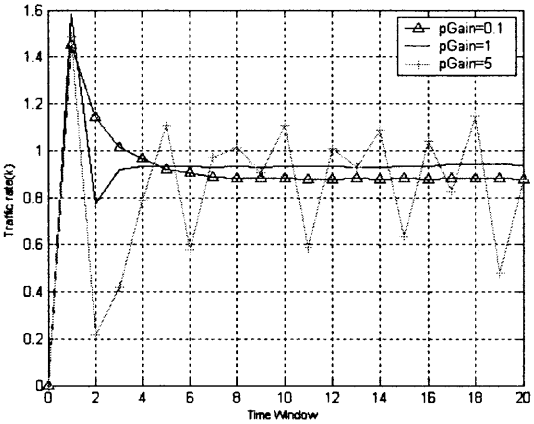
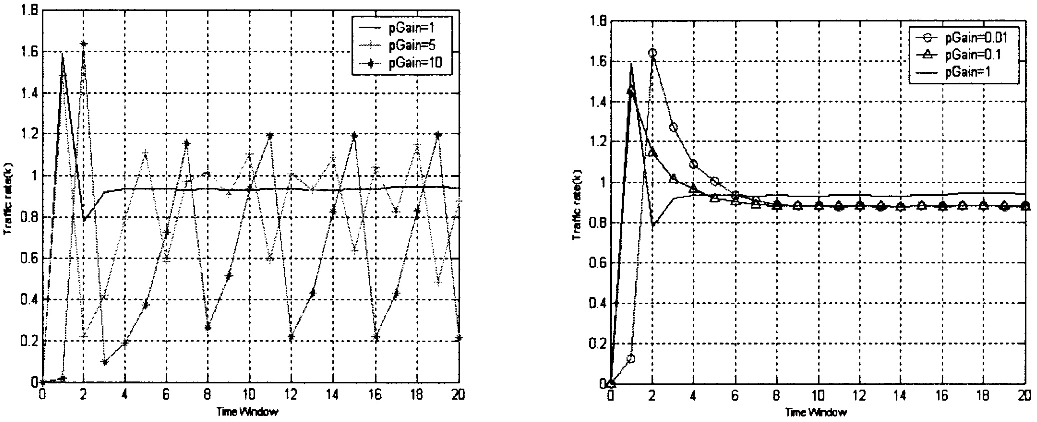


Figure 4.5 P-type controller with different proportional gains



(a) Big Proportional Gains.

(b) Small Proportional Gains;

Figure 4.6 P-type controller with big / small P proportional gain

Figure 4.5 shows what happens when you add proportional feedback to the traffic control system. For small gains (k_p = 0.01, 0.1) the traffic rate goes to the correct target, but it does so quite slowly, shown in figure 4.6(b). Increasing the gain (k_p = 1) speeds up the response to a point. Beyond that point (k_p = 5, 10, 20) the traffic rate starts out faster, but it overshoots the target, shown in figure 4.6(a). In the end, the system doesn't settle out any quicker than it would have with lower gain, but there is more overshoot. If the controller keeps increasing the gain, it would eventually reach a point where the system just oscillated around the target value.

The traffic controller starts to overshoot with high gains because of the delay in the traffic rate response. If looking back at figure 4.5 and formula (4.26), it can be seen that the traffic rate doesn't start ramping up immediately. This delay (at least one time window), plus high feedback gain, is what causes the overshoot seen in Figure 4.5 and Figure 4.6(a). Figure 4.6 shows the response of the precision actuator with proportional feedback only. Proportional control alone obviously doesn't help this system. No matter how low the gain is, the system will oscillate. As the gain is increased, the frequency of the output will increase but the system just won't settle.

The experiment shows that a proportional controller alone can be useful for some things, but it doesn't always help. Network system that has too much delay can't be stabilized as the gains increase with proportional control. As the gains decrease, these stable error increases. Therefore, traffic rate cannot be brought to the desired set point. To solve these control problems we need to add integral or differential control or both.

Integral

Integral control is used to add long-term precision to a control loop. It is almost always used in conjunction with proportional control.

Integral control by itself usually decreases stability, or destroys it altogether. Figure 4.7 shows the traffic controller with pure integral control ($P_k = 0$). The system doesn't settle as the gain is increased. Like the traffic controller with proportional control, the traffic controller with integral control alone will oscillate with smaller and smaller swings until the rate approaches the expected value. The network traffic control system with integral control alone has a bigger stable error as the gain decreases. This system takes longer time to settle out than the same network traffic system with proportional control (see figure 4.5), but notice that when it does settle out, it settles out to the target value-even with the disturbance added in.

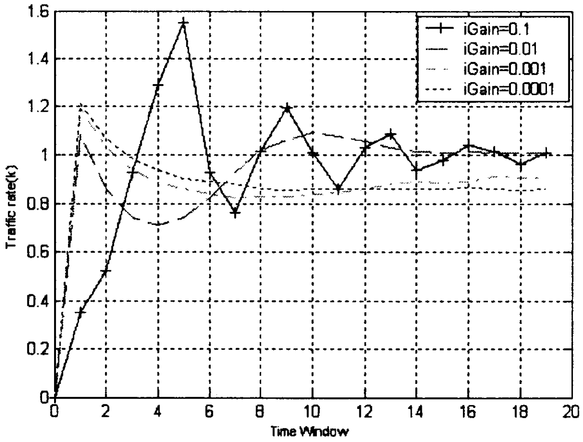


Figure 4.7 I-type controller with different integral gains

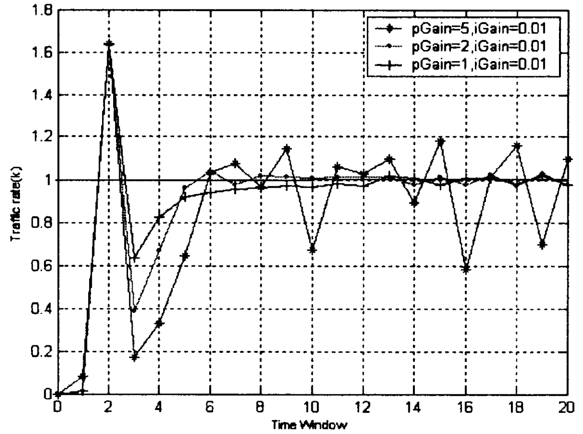


Figure 4.8 PI-type controller with different proportional gains

Figure 4.7 shows that the integrator "remembers" all that has gone on before, which is what allows the controller to cancel out any long-term errors in the output. This same memory also contributes to instability. To stabilize the previous systems, the system needs a little bit of their present value, which comes from a proportional term.

Figure 4.8 shows the network traffic control system with proportional and integral (PI) control. Compare this with Figures 4.5 and 4.7. The network traffic control system takes longer to settle out than the system with pure proportional control, but it will not settle to the wrong spot. The QoS regulator still settles out to the exact target traffic rate, as with pure integral control (see figure 4.7), but with PI control, it settles out two to three times faster.

Differential

As describled above, pure integral control by itself usually decreases. Since proportional control deals with the present behavior of the network traffic control system, and integral control deals with the past behavior of the system, P-type and PI-type controller can't stabilize. If there are some elements that predict the network traffic behavior then this might be used to stabilize the network traffic control system. A differentiator will do the trick.

With differential control, a controller can stabilize the network traffic control system. Figure 4.9 shows the response of the network traffic control system with PID control. We can see the performance improvement to be had by using full PID control with this network traffic system.

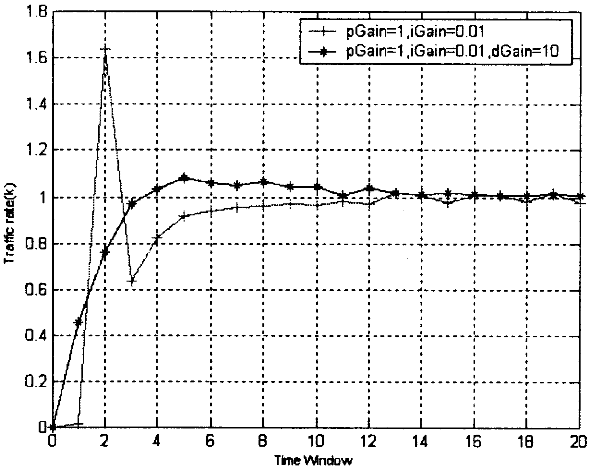


Figure 4.9 PID-type controllers with different gains

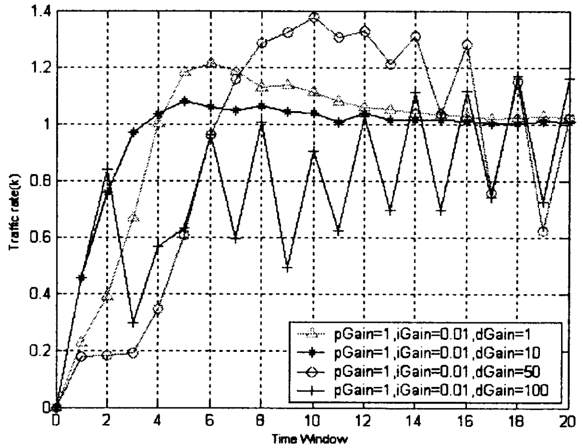


Figure 4.10 PID-type controllers with big derivative gains

Differential control is very powerful, but it is also the most problematic of the control types presented here. The main problem is that differential control suffers from noise problems because noise is usually spread relatively evenly across the frequency spectrum. Proportional control passes noise through unmolested. Integral control averages its input signal, which tends to kill noise. Differential control enhances high frequency signals, so it enhances noise. We can low-pass filter the differential output to reduce the noise, but this can severely affect its usefulness. The theory behind how to do this and how to determine if it will work is beyond the scope of this thesis.

A more scientific and systematic approach is to apply control theory analysis to select the PID control parameters. Such analysis requires an analytical model of the system. Because the system is nonlinear and time variant, it is difficult to precisely model a system such as a real-time system.

The procedure of tuning traditional PID is hard work in network traffic control system because the parameters of PID depends on real network system and the mode of control (PID) cannot be changed once the PID controller is started. Actually, it is not necessary for controller to have a high accuracy. In other ward, the parameters of PID can be in some ranges.

The fuzzy logic controller design is based on the linguistic description of the control strategy that a skilled operator or expert should use in the manual control of the process. Therefore, in the following section, we will design a fuzzy logic according to the design of traditional PID in this section to simulate a PID-like controller.

4.4.2 The Fuzzy PID Controller Design

The fuzzy logic controller design is based on the linguistic description of the control strategy that a skilled operator or expert should use in the manual control of the process. The fuzzy controllers can be further classified into three types: the direct action (DA) type, the gain scheduling (GS) type and a combination of DA and GS types.

The majority of fuzzy PID applications belong to the DA type; here the fuzzy PID controller is placed within the feedback control loop, and computes the PID actions through fuzzy inference. In GS type controllers, fuzzy inference is used to compute the individual PID gains and the inference is either error driven self-tuning [62] or performance-based supervisory tuning [63].

There are several methods [64] available for the implementation of fuzzy PID controllers, one of which utilizes the variables of the error, integral of error and derivative of error. This method leads to too many rules, and its realization in practice is considered difficult to implement and tune. Another realization of a fuzzy PID controller is the parallel combination of fuzzy PI and PD controllers. The rule base is simpler because each one has only two fuzzy input variables, and from the deduction process the relationship between conventional PID parameters and fuzzy ones can be built up, making it possible to apply conventional PID tuning methods.

In this section, we introduce a simple fuzzy PID-like structure for Flow-based Congestion Controller for a reference. The PID-like controller utilizes the fuzzy logic principle and simulates the performance of PID to control QoS regulators. More professional PID methods can be found in [64].

The traffic rate-difference E and traffic rate-difference change CE are selected as the inputs of fuzzy controller and rate-limit η is an output. Therefore, the PID-like controller is a two-input and one-output fuzzy logic system. The system consists of a rule base, membership functions, and an inference procedure, shown in Figure 4.11.

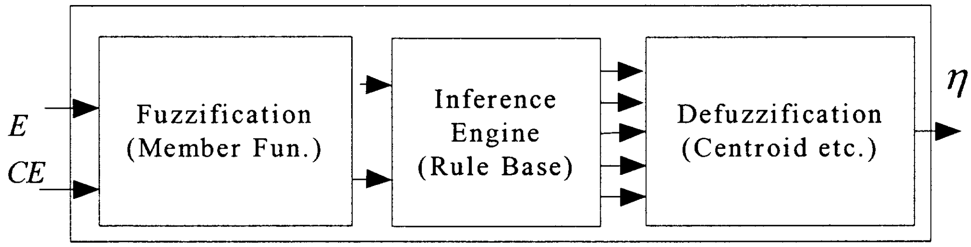


Figure 4.11 The operational structure of fuzzy PID-like controller

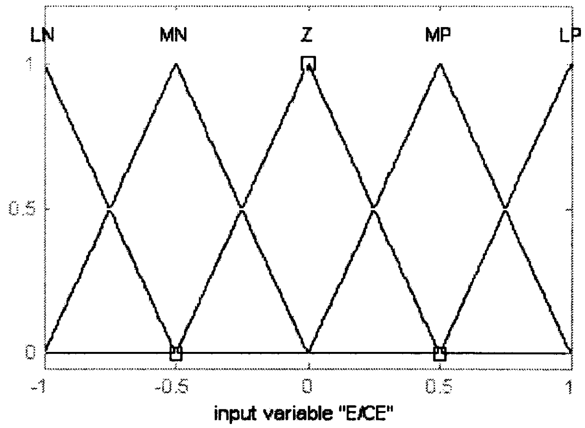
Fuzzification / Defuzzification

Fuzzification and Defuzzification involve mapping the fuzzy variables of interest to "crisp" numbers. Fuzzification translates traffic rates and traffic rate changes into linguistic values.

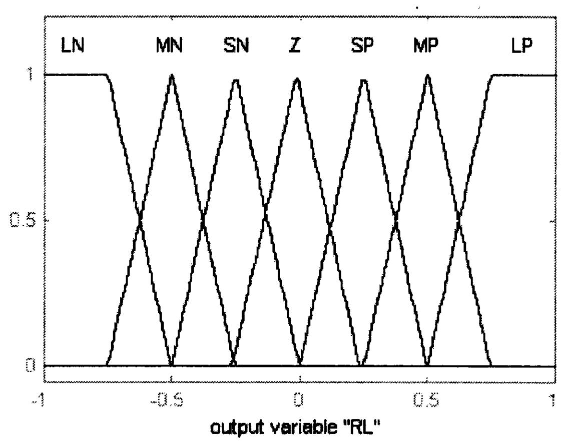
In order to performance like PID controller during traffic control, the PID-like controller uses rate-difference between the target incoming traffic and the observed incoming traffic, and rate-different change as inputs of Fuzzy controller, which represent the change and trends of real network traffic.

Fuzzification translates the traffic rate-difference E and traffic rate-difference change CE into linguistic values. Three triangular membership functions (see figure 4.12.a) are defined for each input while three triangular membership functions (see figure 4.12.b) over the range [-1,1] are defined for the output η.

Once the input variables are fuzzified and run through the fuzzy rule bases, which are discussed below. In a crisp system, the intersection of two sets contains the element that is common to both sets. This is equivalent to the common logical AND operation. In conventional fuzzy logic, the AND operator is supported by taking the minimum of the truth membership grades.



(a) The Input Membership Functions



(b) The Output Membership Functions

Figure 4.12 The input/output member functions of fuzzy PID-like controller

There are many methods that can be used to defuzzy the outputs. One of the most commonly used methods for defuzzification is the Centroid method also known as Center of Area (COA) method. The model well coincides with the human images of fuzzy inference rules in cases when the system has many input and many outputs. In the present simulator, the Centroid method has been used, in which the Centroid of the consequent fuzzy set is found and it corresponding priority value on the x-axis is the final output.

Inference

The fuzzification process finds the degree of membership of each input to their corresponding fuzzy sets. In the main working part of fuzzy logic, there are sets of rules provided by an expert, which relate various fuzzy sets to a desired fuzzy output. Fuzzy sets in this case are, high, medium and low. Fuzzy output sets are high-priority, medium-priority and low-priority. Priority indicates the suitability of a particular flow aggregation scheme to a particular request. Depending on the fuzzy inputs and the rule bases, the output fuzzy set, `priority' is computed using an inference scheme. Several inference schemes are available like Mamdani, etc. For the present simulator, the Mamdani scheme has been adopted (through MATLAB). Figure 4.10 shows the pictorial representation of the inference scheme for several rules and its combined fuzzy output.

Table 4.1 The Inference Rules of PID-like Control

The rule table of fuzzy inference engine (RL)					
CE \ E	LN	MN	Z	MP	LP
LN	LN	LN	LN	LN	MN
MN	LN	MN	SN	Z	SP
Z	MN	SN	Z	SP	MP
MP	SN	Z	SP	SP	LP
LP	MP	LP	LP	LP	LP

Table 4.2 Meanings of the Linguistic variables in PID-like Controller

LN	Large Negative
MN	Medium Negative
SN	Small Negative
Z	Zero
SP	Small Positive
MP	Medium Positive
LP	Large Positive
E	Error
CE	Change in Error
RL	Rate Limit

Rule Development

Our rule development strategy is to select the traffic rate-difference E and traffic rate-difference change CE as the inputs of fuzzy controller and use our experience of using PID above to make the FSC be similar to PID. There are different rules to achieve different objectives, which depend on human experiences. As an example, all the inference rules are shown in the table 4.1.

From table 4.1, there are 25 inference rules. Keywords like IF THEN represent operations on the fuzzy inputs, which gives the output of that particular rule.

For example:

IF E = Z And CE = MP Then RL = SP

At the end of the inference a resultant fuzzy set is obtained, which needs to be defuzzified to get a meaningful crisp output.

4.5 Experiments and Results

4.5.1 Simulation Testbed Model

Although more and more high-speed routers adopt ASICs to fulfill routing and packet forward, Linux-based routers still play important roles in Internet due to their scalability and flexibility. Almost most of traditional traffic controls use the Linux QoS architecture. In this section, we highlight how the ideas presented earlier are implemented in our prototype. This simulation model is used to study the performance of FQoS and rate-limit algorithms. The network traffic includes background traffic caused by normal clients and abnormal traffic caused by attackers. Our simulation model consisted of a traffic generator that generate background traffic; an attack launcher that launches different types of flooding attacks; an Flow Congestion Controller (FCC) that makes admission/rejection decisions on incoming packets or flows; end servers that are used as victims for various kinds of attacks, while at the same time providing some useful services to good clients; and a flow-based network intrusion detection system (FNIDS) that on-line classify each flow.

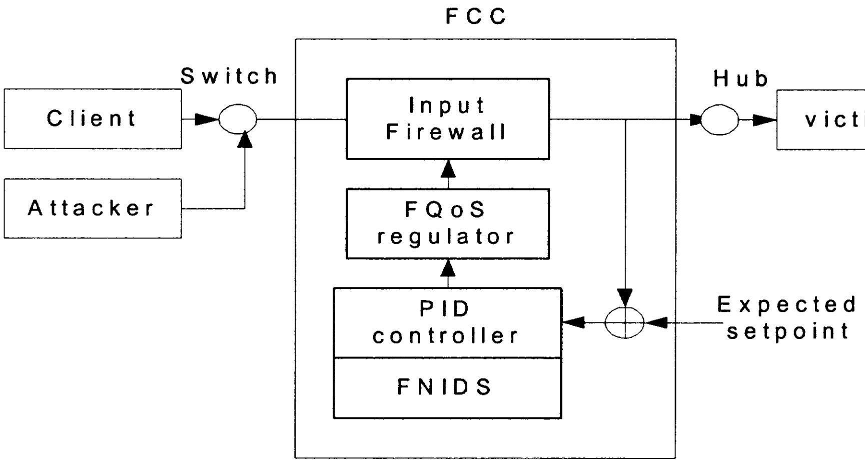


Figure 4.13 The Prototype of FCC simulator model

The following configuration is shown in the figure 4.13. The simulator network was interconnected with 100 Mbps Ethernet cards. A fast machine (Pentium 2.6GHz,1 G Ram, two 100Mbps Ethernet cards, one switch card) was chosen for FCC ,which includes Input firewall, QoS regulator, PID controller and flow-based Network Intrusion Detection System. The fast CPU enables FCC to carry out all incoming packets without getting slowed down by a slow processing speed. Two PCs(Pentium 1GHz , 256MB Ram ,one 100Mbps Ethernet card) were chose as sender (client and attacker) .And one PC(Pentium 1GHz , 256MB Ram ,one 100Mbps Ethernet card) was chose as a victim. To simulate multi-homed machine, we set multi-IP addresses for the network interface card (NICs) of the victim supposing each IP address corresponds to one server. All machines were equipped with the Red Linux 9.0.

FCC Implementation

The FCC is to limit the traffic rate to protect servers. It includes Input firewall, flow-based QoS regulator, PID controller and Flow-based Network Intrusion Detection System (FNIDS). All of these parts can be separated or integrated together. In this simulator, a PID controller and a Fine-grained QoS regulator are plugged into Flow-based Network Intrusion Detection System to construct an integrated FCC, shown in Figure 14.

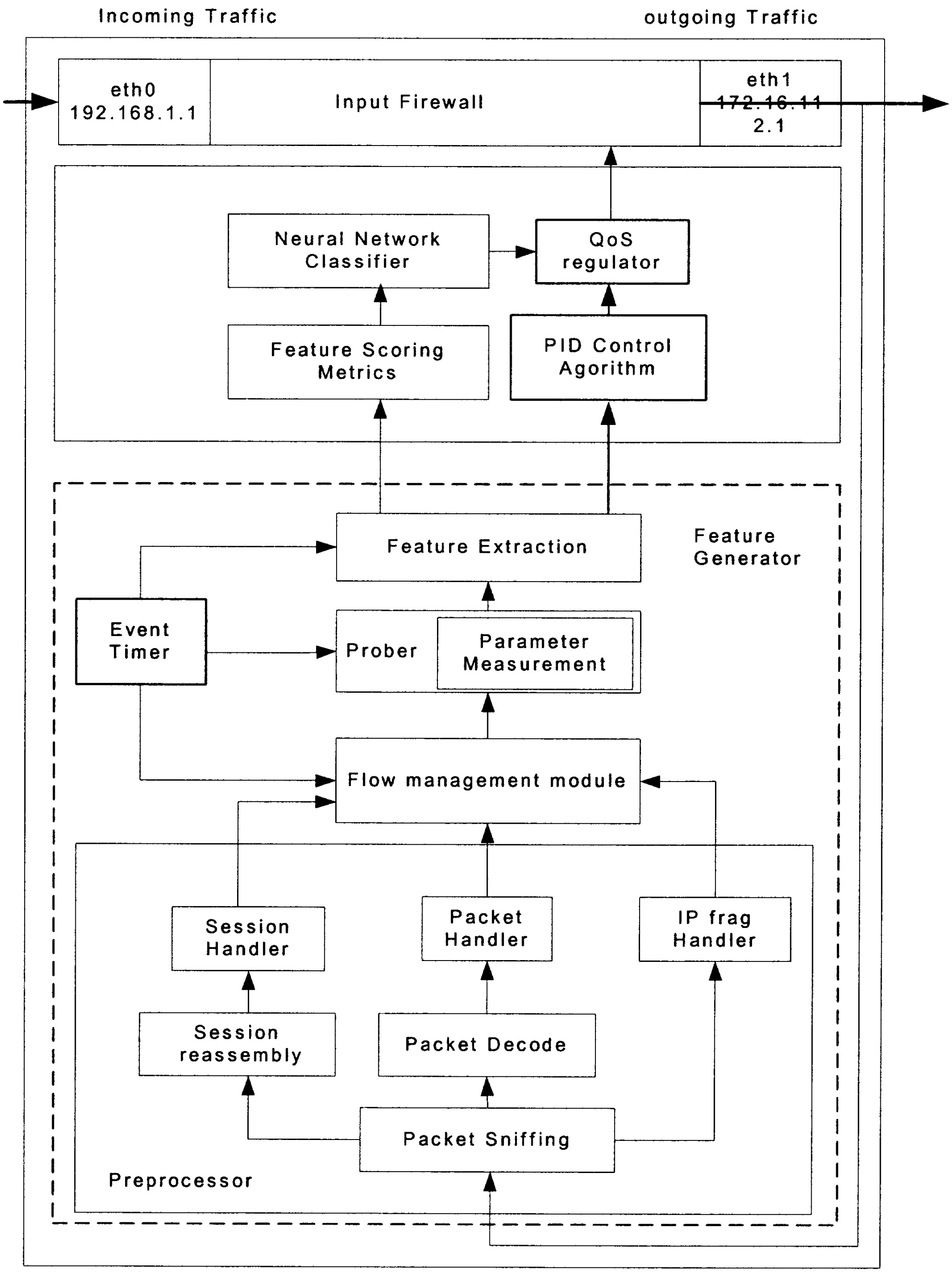


Figure 4.14 Flow congestion control model

4.5.2 Results and Interpretation

Several experiments were conducted on the test-bed to test out FCC and its effectiveness in mitigating DDoS attacks. Various parameters, like allocation of bandwidth under normal user or attackers, service differentiation of FCC, traffic control by using PID controller or fuzzy controller were evaluated in this section.

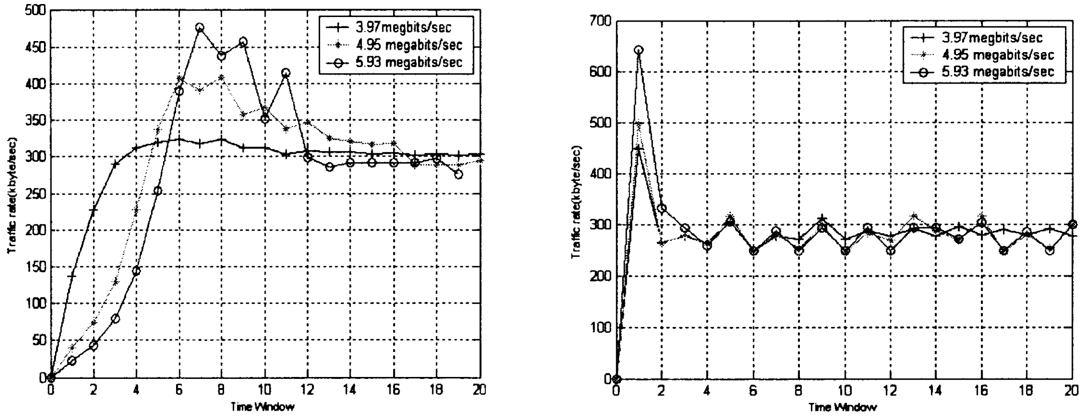
There are two kinds of counting rate: Packet counting rate and Byte counting rate, for network traffic. When flows are mapped to classes, it is important to consider the per-packet processing-cost in order to do a fair allocation of end server resources across flows. Thus it is better to set a packets/sec rate limit than a bytes/sec limit [58].

In this section, we shall discuss the results of our experiment and how may be applied. We separately used traditional PID and Fuzzy PID for flow rate control system. Based on these two controllers, we tested the performance of flow-based control system under different stress flooding attacks. Moreover, we used flooding attacks to detect the effects of FCC.

(a) Stress Testing of PID controller

In this section, stress testing was done for the sensitivity, and thus effectiveness, of PID controller.

The results of stress testing are shown in Figure 4.15. The Figure 4.15 shows that the outgoing flow rates of both the PID and Fuzzy PID approach the expected traffic rate (300kbyte/sec) under various stress incoming traffic. We can notice that the traditional PID took longer time to settles out to the exact target traffic rate than fuzzy PID. This is because traditional PID is more sensitive to real network system because of static parameters, but Fuzzy PID possesses self-adjusted function that make controller adaptive to various real network traffic state. It is also noticed that the final steady performance of tradition PID is consistently better than that of fuzzy PID in this case, which is because Fuzzy PID has a bigger stable error than traditional PID and oscillated around the target.



(a) Traditional PID (b) Fuzzy PID

Figure 4.15 Stress testing of PID controller

In summary, Fuzzy PID has the same performance with the traditional PID and even better. And the Fuzzy PID-like has better dynamical response performance than traditional PID, which makes Fuzzy PID-like in traffic control system rapidly to stop abrupt flooding attack while keeping a good performance..

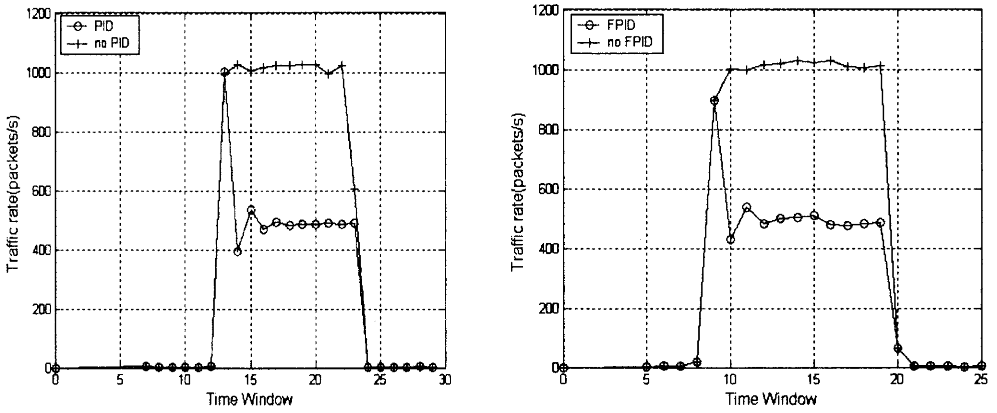
(b) The Control Performance of FCC with/without PID under Flooding Attacks

There are two types of packets best suited to flooding DoS/DDoS attacks: Smurf --ICMP flooding and Fraggle --UDP flooding. In a DDoS Smurf attack, the attacker sends packets to a network amplifier (a system supporting broadcast addressing), with the return address spoofed to the victim's IP address. The attacking packets are typically ICMP ECHO REQUESTs, which are packets (similar to a "ping") that request the receiver to generate an ICMP ECHO REPLY packet. The amplifier sends the ICMP ECHO

REQUEST packets to all of the systems within the broadcast address range, and each of these systems will return an ICMP ECHO REPLY to the target victim's IP address· This type of attack amplifies the original packet tens or hundreds of times. A DDoS Fraggle attack is similar to a Smurf attack in that the attacker sends packets to a network amplifier. Fraggle is different from Smurf in that Fraggle uses UDP ECHO packets instead of ICMP ECHO packets. Because the controlled object of FCC is packet/bytes number, ICMP flooding and UDP flooding attacks has the same characteristic: a lot of packets is launched to a victim.

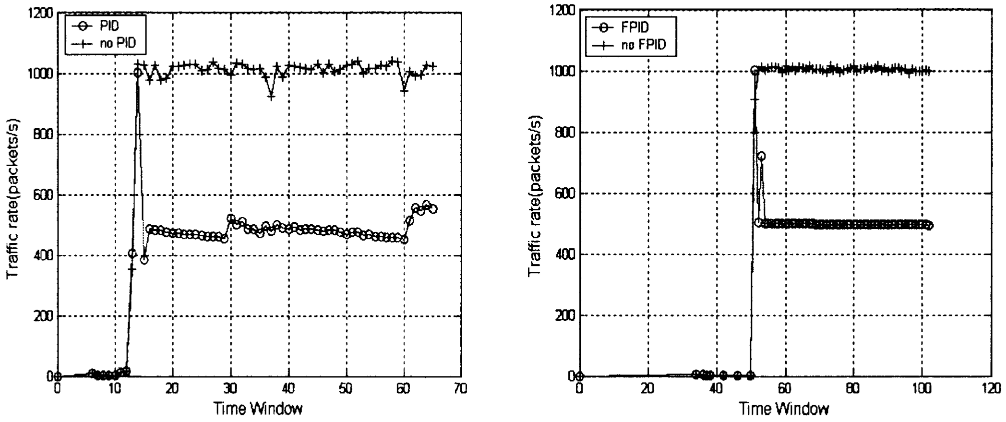
In this experiment, the attacking machine kept sending a constant flood at 1000packets/s to the end server. The packet size of ICMP flooding packets was 1066bytes. And the packet size of UDP flooding was the length of IP head (1066bytes). The rate limit on the QoS regulator was varied and the results plotted in Figure. It is seen that if there were no PID/FPID control, the traffic rate would be over the desirable rate (500 Packets/s) to the server; if there were PID/FPID control, the traffic rate was controlled around the desirable value.

To measure the effectiveness of FCC and the impact that it has on attack, several experiments were run in the identical testbed by using PID control and FPID control. Figure 4.16 presents the results of the impact that PID/FPID has on TCP flooding. The figure 4.16 shows that the packet rates settle out to the desirable traffic rate (500packets/sec) in several time windows.



(a) with/without PID control (b) with/without FPID control

Figure 4.16 The control performance of FCC for Smurf flooding



(a) with/without PID (b) with/without FPID.

Figure 4.17 The control performance of FCC for Fraggle flooding

Likely, Figure 4.17 presents the results of the impact that PID/FPID has on UDP flooding. The flooding traffic rates are rapidly approaching to the target rate.

(c) The allocation of bandwidth under different users

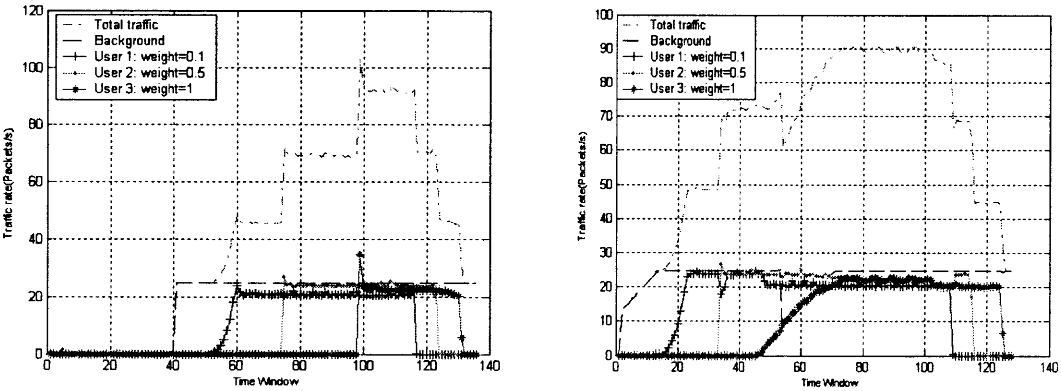
The allocation of bandwidth in PID or Fuzzy PID mechanisms without flooding attacks is discussed here. The dynamic sharing of bandwidth between competing users has to be achieved via flow-based congestion control schemes.

The experiments show two cases of accommodating the allocation of bandwidth. In the first case, the background traffic is low; there is enough bandwidth for new users.

Every new user entering the area will take a certain percentage of the bandwidth. In the second case, the background traffic takes 80% of bandwidth and only 20% of bandwidth for new users. Thus, all users will compete for the bandwidth. A lower priority application is asked to relinquish some of its bandwidth for the new higher priority application. In order to show the influence of weights or priorities on the bandwidth allocation, the fixed rate and highest-priority are assumed for background traffic. The workings of both cases are described with the use of Figure 4.20 and Figure 4.21.

Case 1 : Enough Bandwidth for New Users

In figure 18, user 1, 2 and 3 transfer files by FTP by 20 packets/s(each packet contains 1448 bytes) and a traffic generator launches background traffic. Initially, traffic generator launches traffic to a server by 20 packets/s. User1 starts to sent data to the same server. After some time, a new user 2 enters this same server. Then, the user 3 enters the region. Because the total traffic is not over the target value (100packets/s), the system allocates the bandwidth to each user.



(a) With different weight in PID

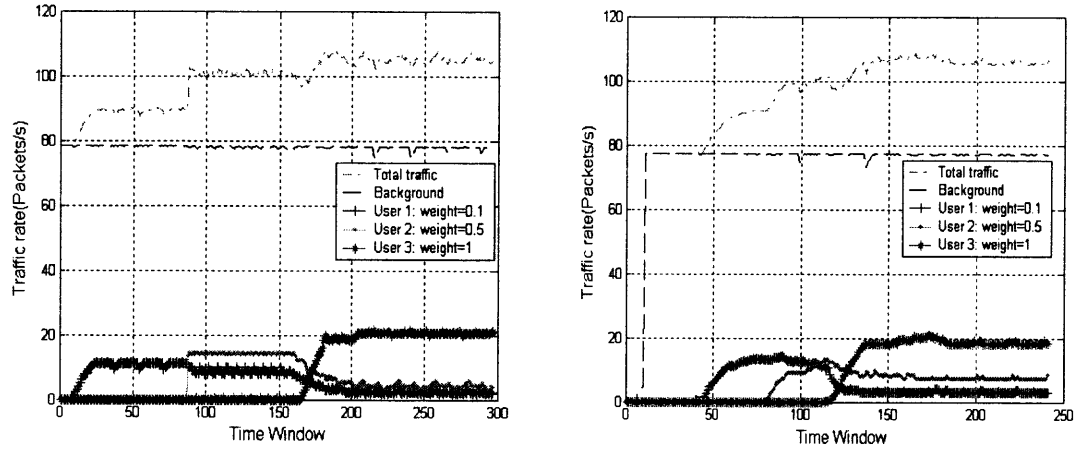
(b) With different weight in FPID

Figure 4.18 Bandwidth allocation for users

Case 2: Limited Bandwidth for New Users with different Priorities (Weights)

In Figure 4.19, User 1, 2 and 3 are users in the system and a traffic generator launches background traffic by 80 packets/s. Each user launches transfer a file by 20 packets/s (each packet contains 1448bytes) through FTP. This figures show that since the total traffic at this time is over the desired value (10Opackets/s), the lower priority userl is asked to relinquish some of its bandwidth for the new higher priority user 2, and when new user 3 with the highest priority enters the region, both of userl and user2 reduce their sending rate to allocate bandwidth for the user3.

This experiment shows that the FCC has the formula (4.14) for setting the weight for different flows, which is that a low priority application is asked to relinquish some of its bandwidth for higher priority users once the bandwidth is not enough.



(a) Using PID

(b) Using FPID

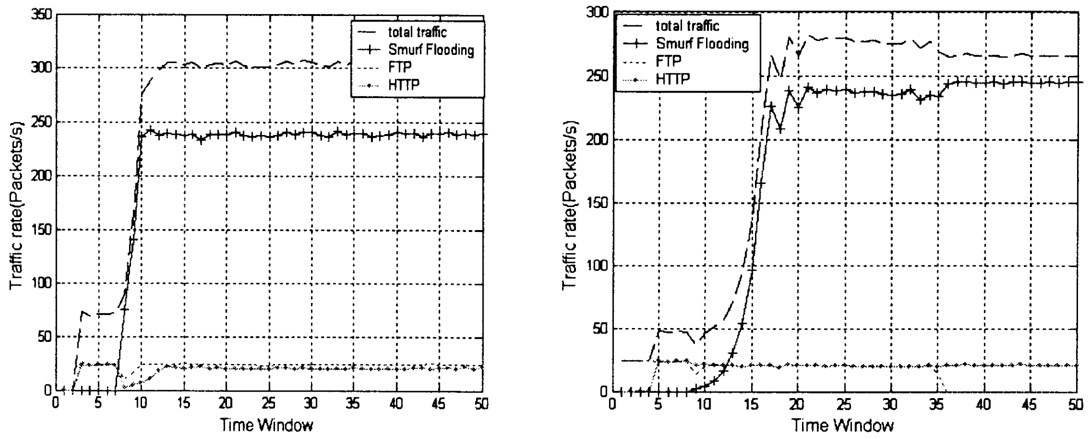
Figure 4.19 Bandwidth allocation for users with different priorities (weights)

(d) The allocation of bandwidth under normal user and attacker

The allocation of bandwidth in PID and Fuzzy PID mechanisms during flooding attacks is discussed here. The dynamic sharing of bandwidth among normal users and attacker has to be achieved via flow-based congestion control scheme. Two cases were performed in these experiments: case 1: attack rate below the target value of server; case 2 attack rate over the target value of server.

The experiments show two cases of accommodating the allocation of bandwidth for normal and malicious user by using formula (4.15). In both cases, a certain percentage of the bandwidth is reserved for the normal user entering the area and the left percentage of the bandwidth for malicious user (attacker). Total traffic rate approaches to the target value (300packets/s). The working of both cases is described with the use of Figure 4.20 and Figure 4.21.

Case 1: Attack Rate below the Target Value



(a) using PID

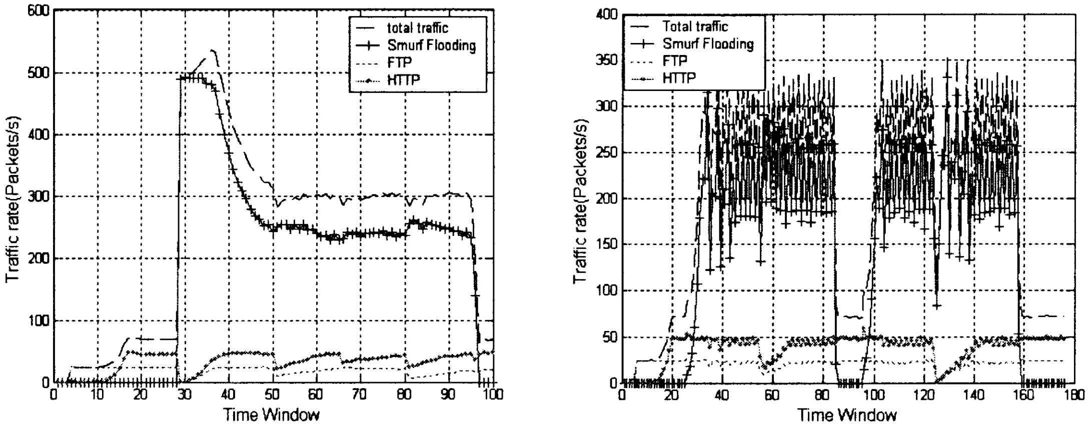
(b) using fuzzy PID

Figure 4.20 Attack rate below the target value

In figure 4.20, the bandwidth is reserved for user 1 and user 2. When the malicious client enters this region, it is allocated by the left bandwidth. Because the total traffic rate is not over the desired traffic rate, the FCC did not limit the bandwidth of malicious client.

Case 2: Attack Rate over the Target Value

In figure 4.21, user 1 and user 2 are running in the system. At time instance, a flooding attacker enters this region, it can only use some of new bandwidth and the bandwidths of user 1 and user 2 almost keep unchanged. FCC allocated the left bandwidth for malicious client while normal FTP user and HTTP user kept unchanged bandwidth.



(a) using PID

(b) using FPID

Figure 4.21 Attack over the target value

These experiments show that a lower priority application will be asked to relinquish some of its bandwidth for the new higher priority application. This further proved that Flow-rate Congestion Control (FCC) architecture could adaptively control the low-priority flows so as to maintain the high priority flow-rates at their desired level.

4.6 Summary

The experiment results show that the mechanism outlined in this paper can effectively detect and handle individual flooding DDoS flows at a QoS router by using PID control and FNIDS.

Based on the concept of flow-based aggregation, we present a flow-based congestion control architecture that provides fine-grained service differentiation and resource isolation among different classes of traffic aggregates. The Flow-based Congestion Control (FCC) architecture consists of a Fine-grained Quality-of-Service (FQoS) regulator, which deploys a Multi-Level Packet Classification (MLPC) to realize packet classification and traffic Rate-limit algorithms to limit traffic rate, and PID controller. The whole system adopts a control-theoretic approach to adjust the flow rate of every link so as to maintain the high priority flow-rates at their desired level, thus guaranteeing QoS to high-priority flow. The architecture is shown to be highly flexible service differentiation and robust against different types of flooding attacks, and traditional network traffic control can be implemented using one common framework. The fine-grained service differentiation and resource isolation provided inside the FCC is a powerful built-in protection mechanism to mitigate DDoS attacks, reducing the vulnerability of Internet to DDoS attacks.

The performance results evaluated by using CONEX test-bed data show the success that the system mitigates Distributed Denial of Service Attacks. The Flow-based QoS regulator guarantees that high-priority flows receive better service, hence, yield better performance in terms of loss rate, effective throughput than low-priority flows.

Furthermore, the simulation demonstrates that a FCC can provide good service differentiation and resource isolation when network bandwidths are under provisioned. It shows the effectiveness of FCC to defense the flooding attacks. Therefore, the flow-based QoS regulator architecture can provide better network QoS and is a simple yet powerful built-in protection mechanism to counter DDoS attacks.

CHAPTER 5

SERVER BANDWIDTH MANAGEMENT SYSTEM (SBMS)

5.1 Introduction

In the above chapters, we discussed FIDS-firewall defense mechanism and network Flow-based Congestion Control (FCC) mechanism. In this chapter, we will mainly present Server Bandwidth Defense System (SBMS), which utilizes the DynaTraX™ switch to create a critical and meaningful solution to stop hackers from intruding into networks, thereby thwarting cyber terrorists and an Intelligent Decision Machine (IDM) to analyze the information from FNIDS to sent alarms or commands to ask DynaTraX(TM) electronically to disconnect and reconnect links. In Chapter 2, we introduced Flow-based NIDS. Here the server-based aggregation scheme was selected for Flow-based Network Intrusion Detection for detecting each flow.

5.1.1 Problem Statement and Motivation

As a last layer of defense system, the SBMS integrated DynaTraX™(see appendix B), Flow-based NIDS and network management station together ,which uses the information from FNIDS to automatically take action to disconnect or reconnect some links to prevent servers from crashing under flooding attacks. The traditional way is to use threshold as a referent value. Once the traffic rate is over threshold, the link is disconnected and if the traffic is below the threshold, then the link is reconnected. This way would cause the two problems:

- The desired upper limit of traffic rate is not deterministic. Although the flow-based control is used to limit the network traffic, the maximum traffic rate is not easy to determine, because if it is high, which is closed to the maximum capacity of link, then sudden incoming flooding attack could crash the server because of the controller response delay and false negative will be high. If the maximum traffic rate is too low, the false positive will be high and the bandwidth cannot be fully utilized.
- High false alarms from network Intrusion detection system. In statistic approached for anomaly detection, the threshold is always used to determine if a packet or flow is normal or abnormal. This fixed threshold always causes high positive or false negative. That is why the alarms directly from IDS are not trusted

In this chapter, we propose an SBMS to reduce the risk of flow control under high threshold and at the same time reduce the false alarms from Flow network Intrusion Detection System in order to improve the liability of our whole defense system. Because the proposed SBMS directly manages the physical links between the protected user networks and the public Internet, as the last resort, any wrong link-control action, whether wrong connection while under attack or wrong disconnection while no attack, could have significant impacts on the QoS and the effectiveness of this SBMS in protecting the customer networks from the network-based attacks. Therefore, Fuzzy logic algorithms have been proposed to reduce the false link-control actions to the lowest possible level.

Fuzzy control theory has been successfully used in various fields, especially in classification of a crisp boundary. Conventional classification approaches always assign a new unidentified object into exactly one category by means of classifier constructed from the training data set. Even though they are suitable for various applications and have proven to be an important tool, they do not reflect the nature of human concepts and thoughts, which tend to be abstract and imprecise. Thus the introduction of fuzzy logic into the realm of classification becomes necessary. Fuzzy Logic is a departure from classical two-valued sets and logic, which uses "soft" linguistic (e.g. large, hot, tall) system variables and a continuous range of truth-values in the interval [0,1], rather than strict binary (True or False) decisions and assignments.

5.1.2 The Basic Fuzzy Logic System Structure

Formally, fuzzy logic is a structured, model-free estimator that approximates a function through linguistic input/output associations.

Fuzzy rule-based systems apply these methods to solve many types of "real-world" problems, especially where a system is difficult to model, is controlled by a human operator or expert, or where ambiguity or vagueness is common. A typical fuzzy system consists of a rule base, membership functions, and an inference procedure (see Figure 5.1).

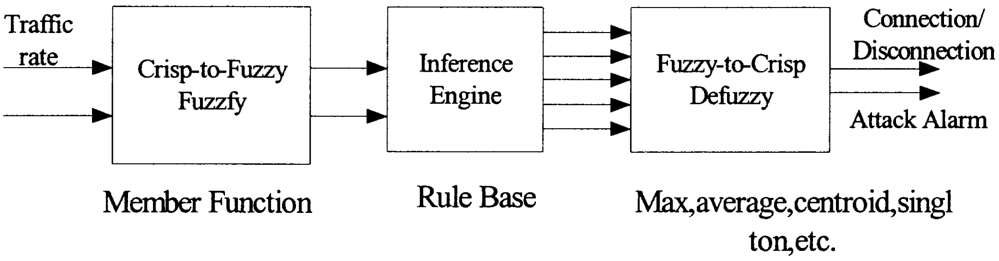


Figure 5.1 The operational structure of SMS

5.2 SBMS Architecture

As described in section 1.2, the SBMS is used as the last-lined protection mechanism in multi-lined defense system. It integrated DynaTraX(TM), a high-speed digital matrix cross-connect switch, with Flow-based IDS together to thwart the increasing threat posed by cyber terrorists. The relationship of SBMS with Flow-based NIDS is shown in figure 5.2.

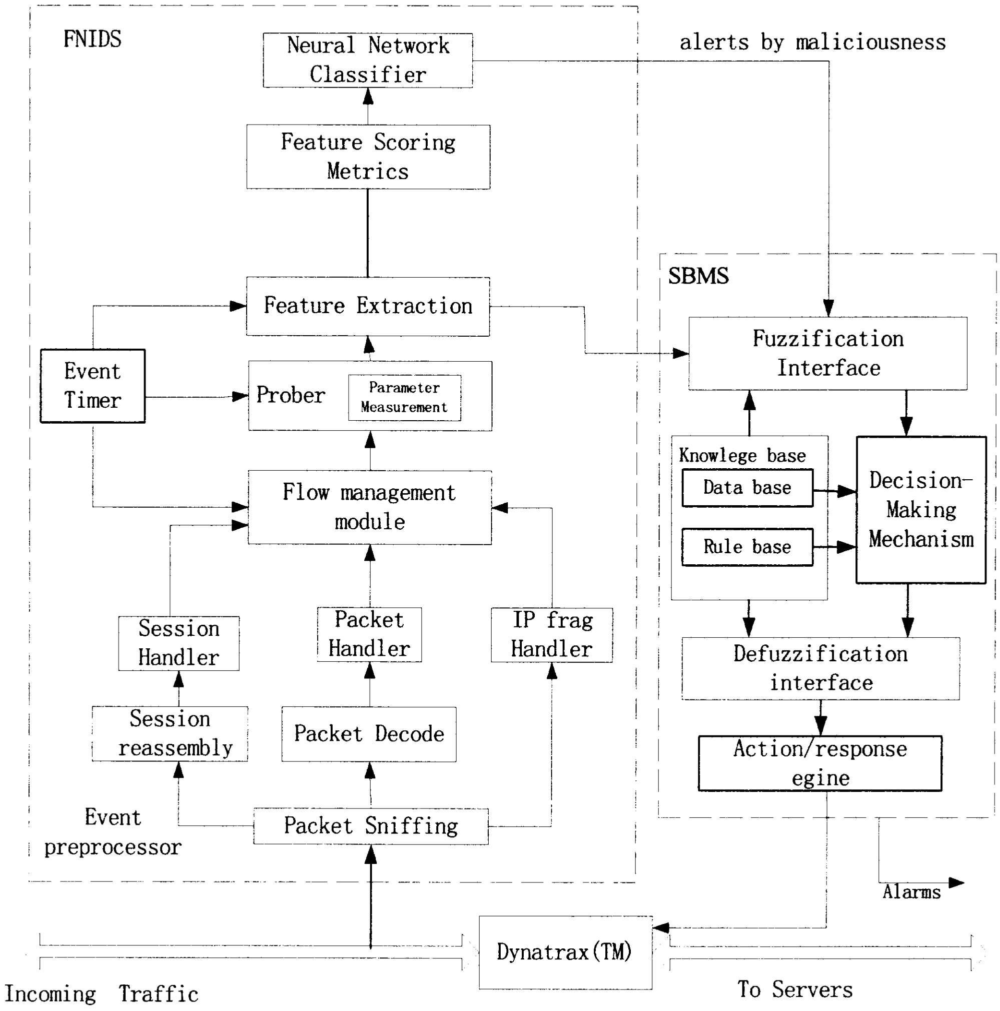


Figure 5.2 The relationship of SBMS with flow-based NIDS

Once there exist attacking action or suspicious packets in the network, FNIDS would send alerts to the SBMS. Then the SBMS analyzes the information from FNIDS to automatically take action to disconnect or reconnect some links to prevent servers from crashing under flooding attacks.

5.2.1 Fuzzy Decision-Making Mechanism

This mechanism is involved in making decisions based on the information received from FNIDS. In particular, it determines the type of attack and recommends actions when attacks are detected. There are different decision support modules, which are specialized in dealing with various anomalous situations. To accomplish this task, the agent uses decision modules, such as Fuzzy Inference Engine (FIE) and Knowledge Base (KB). In order to decide the final response, a bidding system is implemented, where each module generates a bid along with its suggested action; the action with the largest bid is selected. It may be possible to use weight vector to differentiate the importance and role of each module as necessary. Also the bid value may represent the confidence of the decision in taking a particular response. However, the final decision is passed to the Action/Response agent.

5.2.2 Knowledge Base

This base provides a knowledge base of known attacks, which are stored as a set of condition-action rules. The rules represent the expert and common sense knowledge as well as some system level policies. We used a classifier system, which is an adaptive learning system that evolves a set action selection rules to cope with the environment. The condition-action rules are coded as fixed length strings (classifiers) and are evolved using a genetic search. These classifiers are evolved based on the security policy – this rule set forms a security model with which the current system environment needs to be compared. The decision agent activates this module to determine the attack type whenever a deviation occurs.

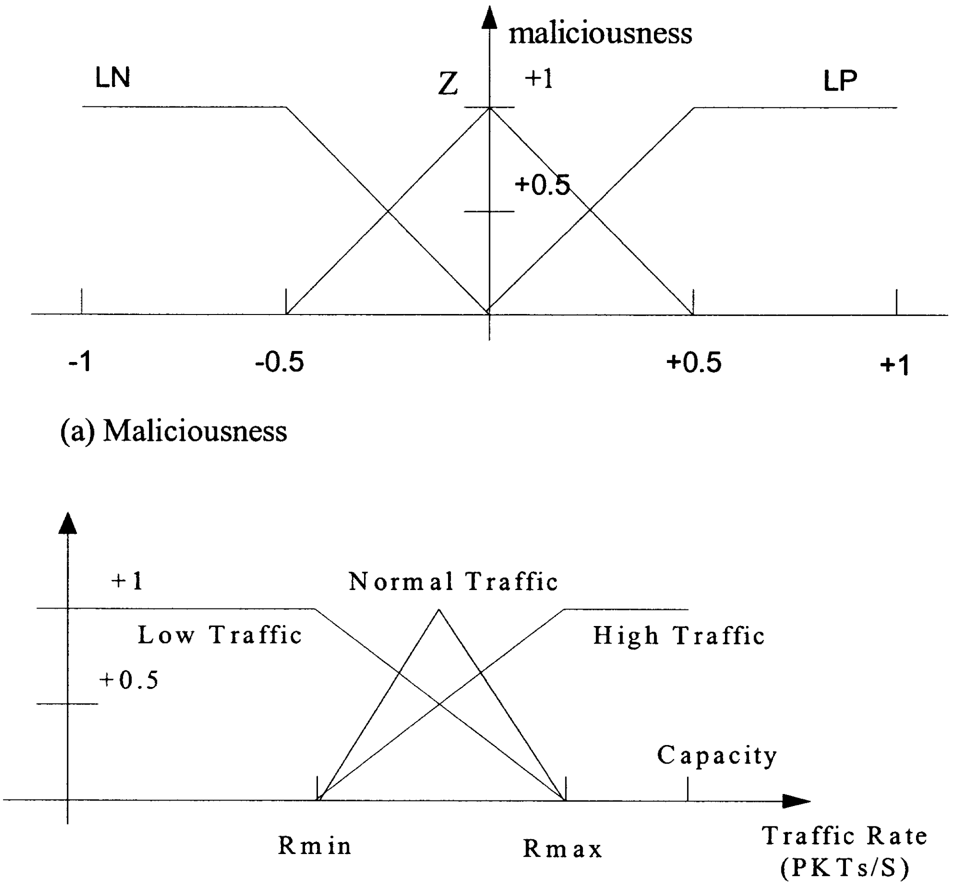
5.2.3 Fuzzy Inference Engine

As the difference between the normal and the abnormal activities are not distinct, but rather fuzzy, this module can reduce the false signal in determining intrusive activities. The purpose of Fuzzy Inference System is to use imprecise and heuristic knowledge to generate appropriate response. The imprecise knowledge is represented by fuzzy logic; this allows representing vague concepts as 'small', 'high', etc. A fuzzy knowledge base and a fuzzy inference system provide the following functionalities of this System. In this system, we mainly manage the bandwidth of server. The traffic rate and maliciousness are the monitored variables. Therefore, the Fuzzy Inference System receives the monitored parameters (Traffic rate) and attack alerts (maliciousness) from FNIDS. The values for these parameters are normalized between 0.0 and 1.0. The fuzzy engine loads fuzzy knowledge before it starts reasoning process. The fuzzy reasoning applies the fuzzy rules over the monitored values and deviation indicators and produces a diagnosis and recommendation, which are then sent to the action agent.

5.2.4 Fuzzification / Defuzzification

The SBMS developed here is a two-input/two-output controller. The two inputs are the maliciousness and traffic rate provided by FNIDS. The two outputs are the reconnection/disconnection (1 or 0) of link and attack alarm.

The Input/Output member functions of SBMS are shown in Figure 5.3 and 5.4. Fuzzication translates the traffic rate r(k) and flow maliciousness m(k) into a linguistic value. Three triangular membership functions (see figure 5.3) are defined for each input while three triangular membership functions (see figure 5.4) over the range [-1,1] are defined for the output.



(a) Maliciousness

(b) Traffic Rate

Figure 5.3 The input member functions of SBMS

In figure 5.3(a), maliciousness is divided into three ranges: large negative (more possible malicious traffic), zero (undetermined range) and large positive (more possible normal traffic). In figure 5.3(b), traffic rate is also divided into three ranges: law-stress traffic, normal traffic and high-stress traffic. In figure 5.4, the output for attack alarm and the output for control DynaTraX™ have the same forms of output member functions.

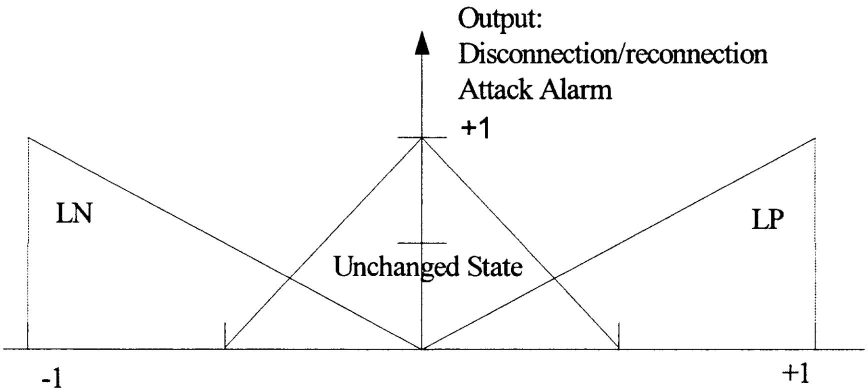


Figure 5.4 The output member functions of SBMS

Once the input variables are fuzzified and run through the fuzzy rule base, which is discussed below, the output of the rules are then aggregated and defuzzified. Aggregation of the results of fuzzy rules takes the logical sum of all the output fuzzy sets. Then, a numerical control signal is generated. A typical formula for this purpose is the so-called centroid method [55].

The rules may use several variables both in the condition and conclusion of the rules. The controller can therefore be applied to multi-input-multi-output (MIMO) problem.

The figures 5.5 show the rule table of the fuzzy inference engine. The meaning of the linguistic variables is explained in Table 5.1.

		Taffic Rate		
		LT	NT	HT
Maliciousness	HN	RC/NA	RC/NA	DC/NA
	NZ	RC/NA	NC/NA	DC/SA
	HA	RC/SA	NC/SA	DC/SA

Figure 5.5 The rule table of fuzzy inference engine

Table 5.1 The Meaning of the Linguistic Variables in SBMS

LT	Low Traffic
NT	Normal Traffic
HT	High Traffic
HN	High Normal
NZ	Near Zero
HA	High Attack
RC	Reconnection
NC	No Change
DC	Disconnection
NA	No Alarm
SA	Send Alarm

5.3 Implementation Description and Evaluations

5.3.1 The Graph Window of SBMS

SBMS was implemented by Visual Basic language working on window system. It has two kinds of information sources: one is from Dynatrax™, which real-time reports the network channel connection states (on/off); another is from FNIDS, which reports the actual network traffic and attack alerts of the monitored network channels. At the same time, SBMS provides two kinds of outputs: one is the command to Dynatrax™ to disconnect or connect the channel between outside and the server; another is to issue alarms to inform administrators. A GUI front end is designed to display all information.

Figure 5.6 shows the main window of the GUI front end.

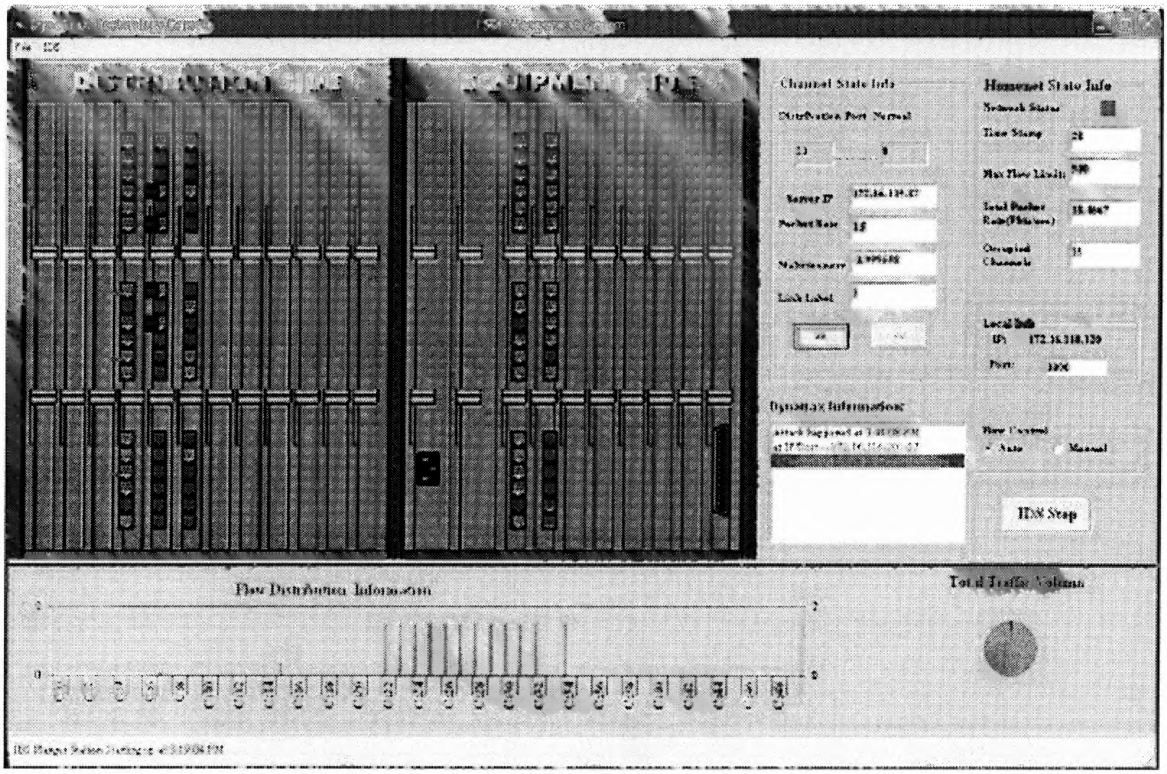


Figure 5.6 The GUI of SBMS

In Figure 5.6, there are three information boards. The first board on left upper is to show connection states and alarm information; the second board on bottom is to show traffic distribution based on each channel; the third board on the right upper is to show the detailed information, such as flow rate, IP address, maliciousness etc., of each channel and total flow information.

5.3.2 The Experiment Results

The SBMS has been tested using two different test sets, the DARPA'98 data set and the CONEX Testbed data set, to test the effectiveness of SBMS and the three above-proposed algorithms.

Similar to the evaluation criteria of Flow-based NIDS, the SBMS is evaluated based on the following measurements:

False Positive Alarms: are incidences that SBMS fails to issue negative alarms;

False Negative Alarms: are incidences that fail to issue positive alarms.

(1) The Results of the DARPA'98 Data Set

The DARPA'98 data set was provided by the MIT Lincoln labs as a research effort to evaluate intrusion detection system. This data set includes seven weeks of training TCPDUMP trace files and two weeks of testing TCPDUMP trace files. Totally 83989 events are collected when Server-based NIDS processes these files.

The misclassifications when using these two algorithms at flooding attack are given in the table below.

Table 5.2 Misclassifications for the DARPA'98 Data Set

	Server-Based	Fuzzy logic
Number of samples	82890	83752
Number of normal samples	82366	82366
Number of attack samples	524	524
Misclassification Rate	0. 0063	1. 1940e-005
False Positive Rate	0. 0064	1. 1940e-005
False Negative Rate	1. 2064e-005	0

From the table 5.2, it can be seen that, misconnections decreases rapidly after using SBMS. Because there are false messages in the DARPA'98 data set, thus, fuzzy logic can intelligently correct the mistakes of server-based NIDS according to network traffic state.

(2) The Results on the CONEX TESTBED Data Set

The CONEX TESTBED data set is collected from an attack-emulation network setup within the CONEX lab of NJIT. Three day worth of TCPDUMP traces are processed by Fuzzy logic. Totally 22555 events are generated during processing.

Table 5.3 Misclassifications for the CONEX TESTBED Data Set

	Server-Base	Fuzzy logic
Number of samples	22555	22555
Number of normal samples	22089	22089
Number of attack samples	466	466
Misclassification Rate	0.0047	0. 000982263
False Positive Rate	0.0048	0. 000873628
False Negative Rate	0.1148	0. 0210526

From the table 5.2, it can be seen that, misconnections decreases rapidly after using SBMS. Because there are false messages in the Conex Testbed data set, thus, fuzzy logic can intelligently correct the mistakes of server-based NIDS according to network traffic state.

5.4 Summary

In this chapter, we introduced a Server-based Bandwidth Defense System (SBMS), which utilized the DynaTraX™ switch to create a critical and meaningful solution to stop hackers from intruding into networks, thereby thwarting cyber terrorists and an Intelligent Decision Machine (IDM) to analyze the information from FNIDS to sent alarms or commands to ask DynaTraX(TM) electronically to disconnect and reconnect links. The system adopts fuzzy logic method to analysis each alert from FNIDS in order to reduce false alarms and at the same time control DynaTraX™ to protect server from flooding attacks.

CHAPTER 6

CONCLUSION

In this dissertation, we introduced "flow" concept into our system. We proposed a novel Network Intrusion Detection System. Because it is based on flow aggregates that dramatically reduces the amount of monitoring data and handles high amounts of statistics and packet data. Moreover, FNIDS includes flow-based statistical feature vectors and two parallel detectors, that make FNIDS not only be used to monitor network behavior change ,but also be used classify every flow. Therefore, FNIDS configures flow flexibly to provide security from network level to application level (IP, TCP, UDP, HTTP, FTP...), and different aggregation schemes, such as server -based, client-based flow.

We presented Adaptive Flow Aggregation Approach to address the problem that a huge amount of malicious spoofed flow exhausts memories and CPU resources. As an application of this approach, an adaptive flow-based network intrusion detection system was introduced in this dissertation. Comparing the flow-based NIDS, the adaptive flow-based NIDS can automatically choose the optimal flow aggregation scheme to detect the DDoS. Therefore, the adaptive flow-based NIDS possesses much better performance of detecting DDoS. Except this application, adaptive flow aggregation approach can be used in firewall to block spoofed attacks while open the port alive to normal clients.

In this dissertation, we presented a Flow-based Congestion Control(FCC) algorithm. The Flow-based Congestion Control (FCC) consists of a Fine-grained Quality-of-Service (FQoS) regulator and PID controller. The whole system adopts a control-theoretic approach to adjust the flow rate of every link so as to maintain the high priority flow-rates at their desired level, thus guaranteeing QoS to high-priority flow. To realize Fine-grained Quality-of-Service (FQoS) regulator, we proposed traffic model based on flow aggregation algorithms and defined flow rate-limit scheme. The flow-based network intrusion detection is used to classify each flow in the network into different priority classes and give different treatment to the flow-rates belonging to different classes. The architecture is shown to be highly flexible service differentiation and robust against different types of flooding attacks, and traditional network traffic control can be implemented using one common framework. The fine-grained service differentiation and resource isolation provided inside the FFCS is a powerful built-in protection mechanism to mitigate DDoS attacks, reducing the vulnerability of Internet to DDoS attacks.

In the last part of this dissertation, we introduced a Switch-based Bandwidth Defense System (SBMS), which utilize the DynaTraX™ switch to create a critical and meaningful solution to stop hackers from intruding into networks, thereby thwarting cyber terrorists and an Intelligent Decision Machine (IDM) to analyze the information from FNIDS to sent alarms or commands to ask DynaTraX™ electronically to disconnect and reconnect links. The system adopts fuzzy logic method to analysis each alert from FNIDS in order to reduce false alarms and at the same time control DynaTraX™ to protect server from flooding attacks.

APPENDIX A

MONITORED STATISTICAL FEATURES

This appendix gives the detailed descriptions on the statistical features monitored by Flow-base Network Intrusion detection.

The FNIDS is capable of monitoring traffic into and out of the protected network. The FNIDS statistical features can be categorized based on the protocols of network traffic.

1. **IP Packet Length** measures the averages and the distributions of the IP packet lengths within a time window. For simplicity, this parameter is symbolized as "ip-pkt-len" afterward.
2. **IP Packet Rate** measures the averages and the distributions of the packet rates of all observed IP packets within a time window. This feature will be symbolized as "ip-pkt-rate" afterward.
3. **IP Byte Rate** measures the averages and the distributions of the byte rates of all observed IP packets within a time window. This feature will be symbolized as "ip-byt-rate" afterward.
4. **IP Fragment Rate** measures the averages and the distributions of the packet rates of all observed IP fragments within a time window. This feature will be symbolized as "ip-frag-rate" afterward.
5. **IP Defragmentation Error Rate** measures the averages and the distributions of the IP defragmentation error rates occurred within a time window. This feature will be symbolized as "ip-defrag-error" afterward.

6. **IP Checksum Error Rate** measures the averages and the distributions of the IP checksum error rates occurred within a time window. This feature will be symbolized as "ip-csum-error" afterward.
7. **TCP Invalid Packet Rate** measures the averages and the distributions of the rates of the TCP packets with invalid combinations of TCP control flags. This feature will be symbolized as "tcp-pkt-invalid".
8. **TCP Packet Length** measures the averages and the distributions of the lengths of IP packets within a time window. This feature will be symbolized as "tcp-pkt-len".
9. **TCP Packet Rate** measures the averages and the distributions of the TCP packet rates within a time window. This feature will be symbolized as "tcp-pkt-rate".
10. **TCP SYN Packet Rate** measures the averages and the distributions of the rates of TCP control packets with SYN flag set within a time window. This feature will be symbolized as "tcp-syn-pkt-rate" afterward.
11. **TCP FIN Packet Rate** measures the averages and the distributions of the rates of TCP control packets with FIN flag set within a time window. This feature will be symbolized as "tcp-fin-pkt-rate" afterward.
12. **TCP RST Packet Rate** measures the averages and the distributions of the rates of TCP control packets with RST flag set within a time window. This feature will be symbolized as "tcp-rst-pkt-rate" afterward.
13. **TCP Connection Open Rate** measures the averages and the distributions of the TCP connection open rates within a time window. This feature will be symbolized as "tcp-con-new-opened" afterward.

14. **TCP Connection Close Rate** measures the averages and the distributions of the TCP connection close rate within a time window. This feature will be symbolized as "tcp-con-new-closed" afterward.
15. **TCP Connection Abort Rate** measures the averages and the distributions of the TCP connection abort rate (connection closed by RESET or TIMEOUT other than normal three-way hand shaking) within a time window. This feature will be symbolized as "tcp-con-new-aborted" afterward.
16. **TCP Connections from Different Source Address** measures the distributions of TCP connections from different source IP addresses within a time window. This feature will be symbolized as "tcp-con-diff-src" afterward.
17. **TCP Connection to Different Destination Address** measures the distributions of TCP connections to different destination IP addresses within a time window. This feature will be symbolized as "tcp-con-diff-dst" afterward.
18. **TCP Connection Anomalous Entropy** measures the averages and the distributions of the anomalous entropies of all TCP connections within a time window. This feature was first proposed in Staniford [37]. The equation to calculate the connection anomalous entropy is given in Equation 7.1. This feature will be symbolized as "tcp-con-anomalous-entropy" afterward.
19. **TCP Connection Half Opened Ratio** measures the averages and the distributions of the ratio between the half-opened TCP connections and all TCP connections within a time window. This feature will be symbolized as "tcp-con-half-opened-ratio" afterward.

20. **TCP Connection Duration** measures the averages and the distributions of the TCP connection durations within a time window. This feature will be symbolized as "tcp-con-duration" afterward.
21. **UDP Packet Length** measures the averages and the distributions of UDP packets within a time window. This feature is symbolized as "udp-pkt-len" afterward.
22. **UDP Packet Rate** measures the averages and the distributions of UDP packets within a time window. This feature will be referred as "udp-pkt-rate" afterward.
23. **UDP Byte Rate** measures the averages and the distributions of UDP packets within a time window. This feature will be referred as "udp-byt-rate" afterward.
24. **UDP Packets from Different Sources** measures the distributions of UDP packets from different IP addresses. This feature will be referred as "udp-diff-src" afterward.
25. **UDP Packet to Different Destinations** measures the distributions of UDP packets destined to different IP addresses. This feature will be referred as "udp-diff-dst" afterward.

26. **ICMP Packet Length** measures the averages and the distributions of ICMP packet lengths within a time window. This feature will be referred as "icmp-pkt-len" afterward.
27. **ICMP Packet Rate** measures the averages and the distributions of ICMP packets within a time window. This feature will be symbolized as "icmp-pkt-rate" afterward.
28. **ICMP Packets from Different Sources** measures the distributions of ICMP packets originated from different IP addresses within a time window. This feature will be symbolized as "icmp-diff-src" afterward.

29. **ICMP Packet to Different Destinations** measures the distributions of ICMP packets destined to different IP addresses within a time window. This feature will be referred as “icmp-diff-dst” afterward.

30. **ICMP Anomalous Echo Replies** measures the averages and the distributions of anomalous ICMP echo replies, which are ICMP echo reply packets without previous echo request packets, within a time window. This feature will be referred as “icmp-anomalous-echo-reply” afterward.

31. **ICMP DUR Packet Rate** measures the averages and the distributions of ICMP DUR (destination-Unreachable) packets within a time window. This feature will be referred as “icmp-dur-pkt-rate” afterward.

APPENDIX B

DYNATRAX™

The DynaTraX system maintains the network configuration, allowing the user to establish data links electronically from a remote location. This eliminates the need to physically move, add and change (MAC) connections. Changes to the network can be made quickly, simply, and safely. DynaTraX combines a comprehensive, flexible cable management scheme and a unified management system that meet the EIA/TIA 606 standard.

1. Features of DynaTraX

- Electronic patch panel capable of cross-connecting up to 108 ports of 4-pair twisted-pair cable to 108 ports of 4-pair (2-pair active) service ports;
- Remote provisioning capability for managing the network configuration;
- Support for Ethernet, Token Ring, 3270, 3X/AS400, TP-PMD (TP-DDI), Fast Ethernet, ATM-25, ATM-52, ATM-155, DS-1(T1), DS-3(T3) or any mix of these networks (Other protocols will be supported in the future. Please check with your local representative regarding your specific requirements);
- SNMP capability facilitating network in-band and out-of-band (serial) management;
- A Test Card capable of testing the condition of the distribution cables; and
- A software package for managing the DynaTraX unit from a PC.

2. Main applications

- Main office: reduce MAC (moves, adds and changes) costs by centralizing the operation and maintaining an up-to-date cable management database;

- Remote management: eliminate technician travel costs and downtime by managing cross-connections in a remote, centralized location;
- Fast network reconfiguration: optimize LAN performance by resegmenting the network to fit the traffic pattern in the organization;
- Improved network reliability: provide back-up LAN services in case of network failure;

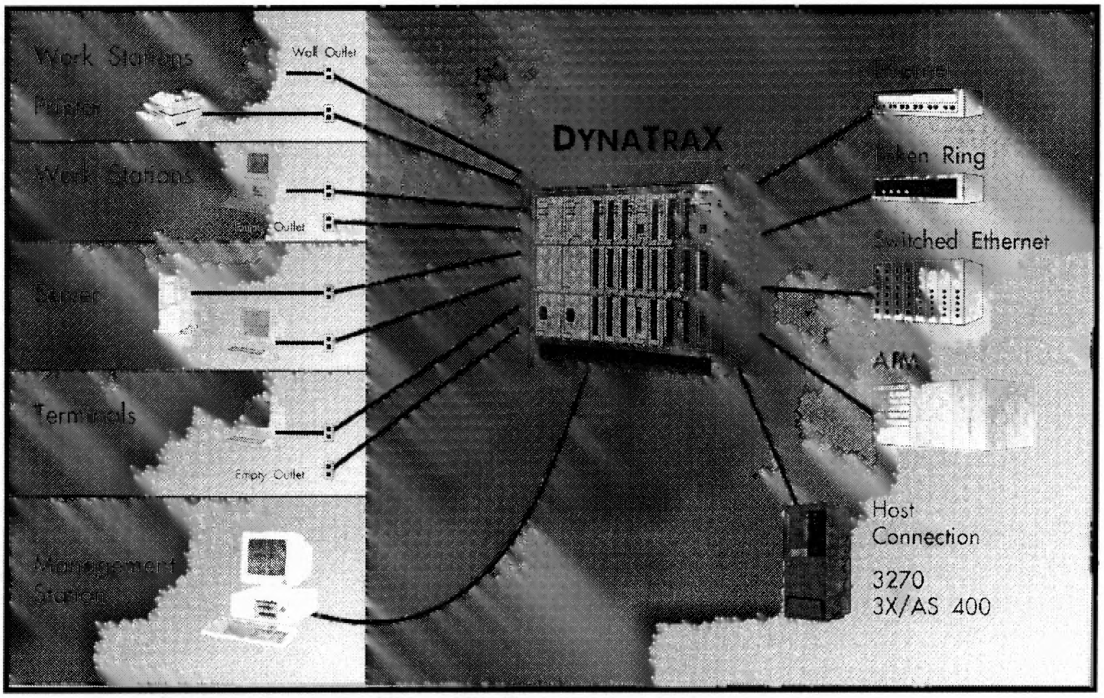


Figure B.1 Generic DynaTraX application

3. Physical Description

Figures B.2 and B.3 show front and rear views, respectively, of the DynaTraX unit,which includes three main cards: Main Controller Cards, Equipment card and Distribution Cards.

The Main Controller (MC) Card initiates all communication and is responsible for the overall control of the system. When the DynaTraX unit is powered up, it queries each card slot to ask for a status or sends a message to a card to perform an action.

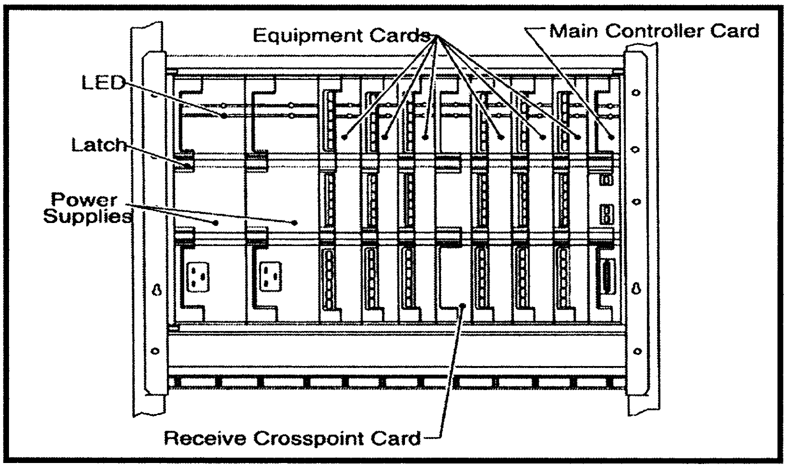


Figure B.2 Front view of the DynaTraX unit

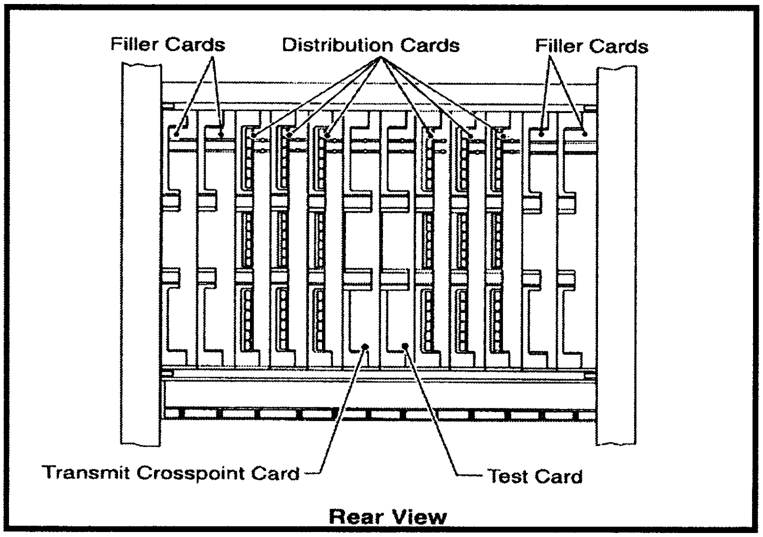


Figure B.3 Rear view of the DynaTraX unit

The Equipment Card (EQUIP) provides universal transmit/receive connections to any of a multitude of data services. It is available in the following three configurations: 1) up to 10 Mb/s (except 4Mb/s TR), 2) up to 16 Mb/s, and 3) up to 100 Mb/s. Upgrading a card from one level to another is done via a firmware download. The cards offer a maximum of 18 8-pin modular ports to which the hubs are connected. Figure B-3 shows an Equipment Card faceplate.

The Distribution Card (DIST) provides universal transmit/receive connection points through 18 standard 8-pin modular connectors. A DynaTraX unit containing 6 of these cards can support up to 108 ports of 4-pair cables (any 2-pairs active) from the terminal stations.

REFERENCES

1. Paul J. Criscuolo, "Distributed Denial of Service," *CIAC-2319*, Department of Energy Computer Incident Advisory Capability, February 14, 2000.

2. Ted Bridis,"Powerful Attack Cripples Internet". *Associated Press* for Fox News 23, October 2002.

3. H.Wang, D. Zhang, and K. G. Shin, "SYN-dog: Sniffing SYN Flooding Sources," *Proceedings of 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, July 2002.

4. P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing," *IETF*, RFC 2267, January 1998.

5. K. Park and H. Lee, "On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets," *Proceeding. of ACM SIGCOMM'2001*, August 2001.

6. D. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," *Proceedings of IEEE INFOCOM'2001*, March 2001.

7. A. C. Snoren, C. Partridge and L. A. Sanchez, "Single-Packet IP Traceback," *IEEE/ACM Transactions on Networking*, December 2002.

8. Dave Plonka, "FlowScan:A Network Traffic Flow Reporting and Visualization Tool," *Proceedings of the 14th Systems Administration Conference (LISA 2000)*, (New Orleans), December 3, 2000.

9. CAIDA, Cooperative Association for Internet Data Analysis, http://www.caida.org/home/.

10. CSICO Company, Cisco IOS NetFlow. http://www.cisco.com/warp/public/732/Tech/nmp/netflow/.

11. Subhabrata Sen and Jia Wang, "Analyzing Peer-To-Peer Traffic Across Large Networks," *IEEE/ACM Transactions on Networking*, VOL. 12, NO.2, April 2004.

12. Masayoshi Mizutani, Shin Shirahata, Masaki Minami, Jun Murai ,"The Design and Implementation of Session Based NIDS," http://www.sfc.wide.ad.jp/~mizutani/article/sb-ids/article.pdf.

13. G. Androulidakis, V. Chatzigiannakis etc., "Network Flow-Based Anomaly Detection of DDoS Attacks," *TERENA Networking Conference* 2004, June 2004.

14. Jana Dunn, "Security Applications for Cisco NetFlow Data," July 23, 2001, http://www.sans.org/rr/whitepapers/commerical/778.php.

15. Z.Zhang , "Statistical Anomaly Denial of Service and Reconnaissance Intrusion Detection," PhD Dissertation, NJIT, 2004.

16. Vern Paxson, http://bro-ids.org/Overview.html.

17. Marina Thottan and Chuanyi Ji, "Anomaly Detection in IP Networks," *IEEE Transactions on Signal Processing,* Vol. 51, No. 8, Aug. 2003

18. W. Lee, S. Stolfo, and K. Mok, "A Data Mining Framework for Building Intrusion Detection Models," *Proceedings of 1999 IEEE Symposium of Security and Privacy*, pp. 120–132, 1999.

19. 1998 Intrusion Detection Evaluation Data Set. http://www.ll.mit.edu/IST/ideval/.

20. Anukool Lakhina, Mark Crovella and Christophe Diot , "Diagnosing Network-Wide Traffic Anomalies", *Proceedings of ACM SIGCOMM ,* pp. 219—230, 2004.

21. J. McHugh, "The 1998 Lincoln Laboratory IDS Evaluation (A Critique)", *Proceedings of the Recent Advances in Intrusion Detection*, 145-161, Toulouse, France, 2000.

22. E. Kohler, J. Li, V. Paxson and S. Shenker, "Observed Structure of Addresses in IP Traffic", *Proceedings of ACM SIGCOMM Internet Measurement Worksho*p, (Marseille, France), pp. 253-266, November, 2002.

23. R. M. Dillon, C. N. Manikopoulos, Neural Net Nolinear Prediction for Speech Data, IEEE Electronics Letters, Vol. 27, Issue 10, pp. 824-826, May 1991.

24. Dave Plonka, "FlowScan:A Network Traffic Flow Reporting and Visualization Tool", *Proceedings of the 14th Systems Administration Conference (LISA 2000)*, (New Orleans), December 3– 8, 2000.

25. Cristian Estan, Ken Keys, "Building a Better NetFlow," Presented at *SIGCOMM'04*, Aug. 30-Sept. 3, 2004.

26. Masayoshi Mizutani, Shin Shirahata, Masaki Minami and Jun Murai , " The Design and Implementation of Session Based NIDS ," http://www.sfc.wide.ad.jp/~mizutani/article/sb-ids/article.pdf.

27. G. Androulidakis and V. Chatzigiannakis, "Network Flow-Based Anomaly Detection of DDoS Attacks," *TERENA Networking Conference 2004*, June 2004.

28. Jana Dunn, "Security Applications for Cisco NetFlow Data," July 23, 2001, http://www.sans.org/rr/whitepapers/commerical/778.php.

29. Nick Duffield, Carsten Lund, Mikkel Thorup, "Properties and prediction of flow statistics from sampled packet streams," *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*, P159 - 171, 2002.

30. W. Lee, S. Stolfo, and K. Mok, "A Data Mining Framework for Building Intrusion Detection Models," *Proceedings of 1999 IEEE Symposium of Security and Privacy*, pp. 120–132, 1999.

31. P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing Agreements Performance Monitoring," *RFC 2827*, May 2000.

32. K. Park and H. Lee, "A Proactive Approach to Distributed DoS attack Prevention using Route-Based Packet Filtering," *Proceedings of ACM SIGCOMM*, San Diego, CA, Aug. 2001.

33. T.Ohnishi,"Fuzzy Reasoning by Ordian Structure Model of Control Rule", *Journal of Japan Society for Fuzzy Theory and System*,Vol 2,No.4,pp 125-132,1990.

34. P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service attacks which Employ IP Source Address Spoofing," Technical Report., IETF, *RFC 2267*, May 2000.

35. Sven Dietrich and David Dittrich, "DDoS for fun and profit", *The 14th USENIX Security Symposium*, Baltimore, MD USA, July 2005.

36. F.A fanasiev, V.Grachev, "Flow-based Analysis of Internet Traffic", *NI/0306037*, v1, 9 June 2003.

37. Nick Du_eld and Carsten, Lund, "Predicting Resource Usage and Estimation Accuracy in an IP Flow Measurement Collection Infrastructure", *Internet Measurement Conference*, October 2003.

38. Sheng-Ya Lin, Yong Xiong etc. "A Ratio-Based Control Algorithm for Defense of DDoS Attacks", Technical Report 2005-1-4, January 2005

39. Ratul Mahajan, Steven M. Bellovin and Sally Floyd, "Controlling High Bandwidth Aggregates in the Network", *Computer Communications Review*, Vol.32, No.3, pp.62-73, July, 2002.

40. Ratul Manajan, Steven M. Bellovin, "Controlling High Bandwidth Flows at the Congested Router", *Proceedings of IEEE ICNP 2001*, Riverside, CA, December 2001.

41. Eddie Kohler, Mark Handley, and Sally Floyd, "Datagram Congestion Control Protocol (DCCP)", *Internet Draft, Internet Engineering Task Force*, February, 2004.

42. Sally Floyd and Kevin Fall, "Promoting the use of End-to-End Congestion Control in the Internet", *IEEE/ACM Transactions on Networking*, Augest,1999

43. Dong Lin and Robert Morris, "Dynamics of Random Early Detection", *ACM SIGCOMM 1997*, 1997.

44. Sally Floyd, "TCP and Explicit Congestion Notification", *ACM Computer Communication Review*, Vol.24, No.5, pp.10-23, October 1994.

45. A. Kolarov, G. Ramamurthy, "A Control-Theoretic Approach to the Design of an Explicit Rate Controller for ABR Service", *IEEE/ACM Transactions on Networking*, Vol.7 No.5 Oct. 1999.

46. Vance J. VanDoren, "Understanding PID Control: Familiar Examples Show How and Why Proportional-Integral-Derivative Controllers Behave the Way They Do," *Control Engineering* , June 1, 2000.

47. Aleksandar Kuzmanovic and Edward W.Knightly, "Low-Rate TCP-Targeted Denial of Service Attacks", *SIGCOMM03*, pp.75-86, August 25-29,2003.

48. David K.Y.Yau, John C. S. Lui, and Feng Liang, "Defending Against Distributed Denial-of-service Attacks with Max-Min Fair Server-centric Router Throttles", *Proceedings of IEEE International Workshop on Quality of Service (IWQoS),* Miami Beach, FL, May, 2002.

49. Yong Xiong, Jyh-Charn Liu1, "On the Modeling and Optimization of Discontinuous Network Congestion Control Systems," *IEEE INFOCOM*, 2004

50. K.B. Kim, Ao Tang, Steven H. Low, "Design of AQM in Supporting TCP based on Well-known AIMD Model," *IEEE Globecom*, 2003.

51. J.S. Sun, G.R. Chen, K.T. Ko, S. Chan, M. Zukerman, "PD-controller: A New Active Queue Management Scheme," *IEEE Globecom*, 2003.

52. Tao Peng , Christopher Leckie , Kotagiri Ramamohanarao, " Detecting Distributed Denial of Service Attacks using Source IP Address Monitoring," Technical Report, November 2002, http://www.cs.mu.oz.au/~tpeng/mudguard/research/detection.pdf.

53. Thomer M. Gil and Massimiliano Poletto, "MULTOPS: a data-structure for bandwidth attack detection, " *Proceedings of the 10th USENIX Security Symposium*, pages 23–38, Washington, DC, July 2001.

54. Yih Huang and J. Mark Pullen, “Countering Denial-of-Service Attacks Using Congestion Triggered Packet Sampling and Filtering,” *Proceedings of Computer Communications and Networks*, 2001.

55. David Mankins, Rajesh Krishnan, Ceilyn Boyd, John Zao, Michael Frentz, “Mitigating Distributed Denial of Service Attacks with Dynamic Resource Pricing”, *Proceedings of 17th Annual conference on Computer Security Applications*, 2001.

56. Yong Xiong, Steve Liu, and Peter Sun, "On the Defense of the Distributed Denial of Service Attacks: An On-Off Feedback Control Approach," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol. 31, No. 4. pp. 282-293, July 2001.

57. David K. Y. Yau, John C. S. Lui, and Feng Liang, “Defending Against Distributed Denial-of-Service Attacks with Max-Min Fair Server-centric Router Throttles,” *Proceedings of IEEE International Workshop on Quality of Service*, pp. 35-42, 2002.

58. Jose Carlos Brustoloni, "Protecting Electronic Commerce From Distributed Denial-of-Service Attacks," *Proceedings of ACM WWW*, May 2002.

59. A. Garg and A. L. Narasimha Reddy, "Mitigation of DOS Attacks through QOS Regulation," *Proceedings of IWQOS Workshop*, May 2002.

60. David Mankins, Rajesh Krishnan, Ceilyn Boyd and John Zao, “Mitigating Distributed Denial of Service Attacks with Dynamic Resource Pricing”, *Proceedings of 17th Annual Conference on Computer Security Applications*, 2001.

61. Kihong Park and Heejo Lee, "On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets," *Proceedings of ACM SIGCOMM*, August 2001.

62. Juniper Systems, “Internet Processor II ASIC: Performance without Compromise,” *White Paper*, Sept. 2000.

63. Z.-Y. Zhao, M. Tomizuka, and S. Isaka, “Fuzzy Gain Scheduling of PID Controllers,” *Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 23, pp. 1392–1398,1993.

64. C. W. de Silva, “Intelligent Control: Fuzzy Logic Applications,” New York: CRC, 1995.

65. George K. I. Mann, Bao-Gang Hu and Raymond G. Gosine, “Analysis of Direct Action Fuzzy PID Controller Structures”, *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 29, No. 3, June 1999.

66. T. Abdelzaher and K.G. Shin, "End-Host Architecture for QoS Adaptive Communication," *Proceedings of IEEE Real-Time Technology and Applications Symposium*, June 1998.

67. G. Banga, P. Druschel, and J. Mogul, "Resource Containers: A New Facility for Resource Management in Server Systems," *Proceedings of Third Symposium. Operating System Design and Implementation*, Feb. 1999.

68. N. Bhatti and R. Friedrich, "Web Server Support for Tiered Services," *IEEE Network*, vol. 13, no. 5, Sept./Oct. 1999.

69. A. Miyoshi and R. Rajkumar, "Protecting Resources with Resource Control Lists," *Proceedings of IEEE Real-Time Technology and Applications* ,Symposium, May 2001.

70. O. Spatscheck and L. Peterson, "Defending against Denial of Service Attacks in Scout," *Proceedings of Third Symposium Operating System Design and Implementation*, February 1999.

71. K. Papagiannaki, P. Thiran, J. Crowcroft, and C. Diot, "Preferential Treatment of Acknowledgment Packets in a Differentiated Services Network," In *Proceedings International Workshop QoS*, June 2001.

72. P. Wang, Y. Yemini, D. Florissi, J. Zinky, and P. Florissi, "Experimental QoS Performances of Multimedia Applications," *Proceedings of IEEE INFOCOM*, March 2000.

73. L. Garber, "Denial-of-Service Attack Rip the Internet," *Computer*, Apr. 2000.

74. J. McQuillan, "Layer 4 Switching," *Data Communication*, Oct. 1997.

75. V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast and Scalable Layer Four Switching," *Proceedings of ACM SIGCOMM,* Sept. 1998.

76. P. Gupta and N. McKeown, "Packet Classification on Multiple Fields," *Proceedings of ACM SIGCOMM*, Sept. 1999.

77. T.V. Lakshman and D. Stiliadis, "High Speed Policy-based Packet Forwarding Using Efficient Multi-Dimensional Range Matching," *Proceedings of ACM SIGCOMM*, September. 1998.

78. Balachander Krishnamurthy and JiaWang,"On network-aware clustering of Web clients," *Proceedings of ACM SIGCOMM 2000 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, August 2000.