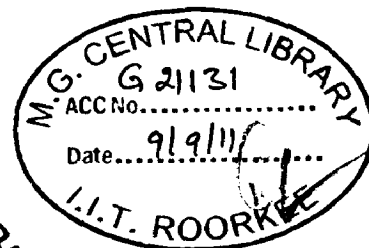
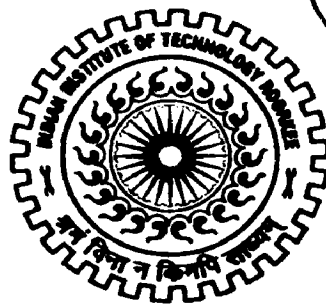


**AN EFFICIENT FUZZY CONTROLLER BASED TECHNIQUE
FOR
NETWORK TRAFFIC CLASSIFICATION TO IMPROVE QoS**

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*
of
MASTER OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

By
AJAY CHAUDHARY



**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
JUNE, 2011**

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being presented in the dissertation entitled "AN EFFICIENT FUZZY CONTROLLER BASED TECHNIQUE FOR NETWORK TRAFFIC CLASSIFICATION TO IMPROVE QoS" towards the partial fulfillment of the requirement for the award of the degree of **Master of Technology in Computer Science and Engineering** submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee, Uttarakhand (India) is an authentic record of my own work carried out during the period from July 2010 to June 2011, under the joint guidance of **Prof. Manoj Misra, Professor** and **Dr. Anjali Sardana, Assistant Professor**, Department of Electronics and Computer Engineering, Indian Institute of Technology, Roorkee.

The matter presented in this dissertation has not been submitted by me for the award of any other degree of this or any other Institute.

Date: 28 June (1)

Place: Roorkee



(AJAY CHAUDHARY)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 28.06.2011

Place: Roorkee


(Prof. Manoj Misra)

Professor

Department of Electronics and Computer Engineering

IIT Roorkee.

Date: 11/7/11

Place: Roorkee


(Dr. Anjali Sardana)

Assistant Professor

Department of Electronics and Computer Engineering

IIT Roorkee.

ACKNOWLEDGEMENTS

I wish to express my heartfelt gratitude for the encouragement, broad knowledge and the expert guidance that **Prof. Manoj Misra** and **Dr. Anjali Sardana** have given me throughout my work on this thesis at the Indian Institute of Technology Roorkee, Roorkee, India. I consider myself very fortunate for having been associated with them. It is their vision and insight that inspired me to undertake research in this interesting field of information security. They have been very generous in providing me the necessary resources to carry out my research. Prof. Manoj Misra and Dr. Anjali Sardana insistence on preparing and giving the best possible presentation, writing the best possible paper, and being the best possible student, definitely changed my way of thinking and enriched my growth as a researcher. I have benefited immensely from his inspiration and advice. I fall short of words to express their excellent and able guidance.

The co-operation and help extended by the Head and faculty members, Department of Electronics and Computer Engineering, Indian Institute of Technology, Roorkee is gratefully acknowledged. I specially thank my research committee members for their patience during discussions and evaluations.

The department of Electronics and Computer Engineering provided an excellent environment for my research in information security. I spent many enjoyable hours working with department members and fellow students. Without their friendly environment and freedom, many of my ideas would not have come to fruition. A special mention of Mr. Shiv Aggarwal, Mr. Prafulla Gupta, Mr. Noaman Siddiqui, Lt. Col. S.S. Chauhan, Mr. Bharat Gupta, Mr. Hemshakar Sahu, Mr. Vinit Kumar and Mr. Kehar Singh. They took the arduous step of proof reading the drafts of the thesis and helping me at various point of my work. Sincere efforts of Mr. Raj Narayan Khati for providing assistance with the use of laboratory facilities are acknowledged.

On a personal note, I owe everything to the Almighty and my parents. I feel deep sense of gratitude for my mother and father for their unending love, inseparable support and prayers during this research experience. Without the sacrifices from my parents, I could never have the

opportunity to undertake this study. They always encouraged me in all my endeavors and felt proud of every achievement of mine. I would like to thank my wife **Mrs. Meenakshi Jakhar** for her sacrifice, motivation, and encouragement during my stay at IIT, Roorkee. I would like to thank my 2 year old son **Master Naitik** for his love, affection and for being constant source of inspiration. I would like to thank my sister nephews, and niece for their constant care and concern that made this journey so congenial and easy. I would like to extend my sincere thanks to my mother in law, late father in law and in laws family members for their affection and wishes.

I feel grateful for the support I have received from my parent institute Engineering College Bikaner, Bikaner, India for sponsoring my stay at Indian Institute of Technology, Roorkee and IITR Heritage Fund for sponsoring my conference participations.

I thank God for everything I am blessed with.

AJAY CHAUDHARY

ABSTRACT

The network traffic management is a key issue today as many new emerging applications are flooding the network with their packets. Several time sensitive and low bandwidth applications run along with other time insensitive and bandwidth intensive applications. As a result, most of the time, channel capacity is exhausted by the bandwidth intensive applications like P2P file sharing, Bittorent etc. Hence, there is limited space left for critical applications. As a result, the accurate identification of network applications through proper observation of associated packets is vital to the areas of network management and surveillance. The overall implication of the accurate traffic management provides space for critical application and also it helps in better management of available network resources like channel capacity etc.

The typical approach of traffic classification is based on 'well known' TCP or UDP port numbers. This approach is not feasible in current scenario in which applications such as tends to use random port numbers. Other approaches like flow based are able to resolve the issue to some extent but they are also limited to certain protocols or applications. The Payload signature based matching algorithm is able to identify the traffic accurately but again it suffers as the processing time is really high.

The work in this dissertation is focused towards exploring the key issues and design factors to be considered for network traffic classification and bandwidth management. An approach is proposed for the network traffic classification and bandwidth consumption by each protocol. It optimizes the usage of the available network capacity and day to day traffic management. The usage of the bandwidth in the networks is managed by taking accurate decisions for various types of applications.

The test results show that by using proposed priority based management approach we are able to classify about 90% of traffic accurately. Also the proposed priority based bandwidth management approach is really fast as compare to non priority bandwidth management. The approach is also able to maintain the QoS requirement of the real time connections by marking the time insensitive but bandwidth intensive traffic. These packets can be dropped for proper traffic management.

Table of Contents

Candidate's Declaration & Certificate.....	i
Acknowledgements	ii
Abstract.....	iv
Table of Contents	v
List of Figures	vii
List of Tables.....	ix
1. Introduction and Problem Statement	1
1.1 Introduction.....	1
1.2 Motivation	4
1.3 Statement of the Problem	6
1.4 Organization of the Report	7
2. Background and Literature Review	9
2.1 Quality of service	9
2.1.1 Quality of service overview	9
2.1.2 Role of Quality of Service.....	10
2.1.3 The Benefits of QoS.....	12
2.1.4 QoS Parameters	13
2.2 Packet Inspection Mechanism and Issues	14
2.2.1 Stateful/Shallow Packet Inspection	14
2.2.2 Medium Depth Packet Inspection.....	14
2.2.3 Deep Packet Inspection (DPI).....	14
2.2.4 Challenges for DPI.....	15
2.2.5 Design Issues for DPI	16
2.2.6 Existing Technologies for DPI Implementation.....	17
2.3 Network traffic classification.....	21
2.3.1 Role of Network traffic classification	22
2.3.2 Types of Network traffic classification.....	23
2.4 Fuzzy Controller and Fuzzy Logic.....	23
2.5 Research Gaps.....	26

3. Proposed Framework for Classification Process.....	27
3.1 System framework	27
3.2 Priority Based Payload Signature Matching (PBPSM).....	28
3.3 Preprocessing of the results	31
3.4 The Fuzzy Controller.....	32
3.4.1 Fuzzy Logic	32
3.4.2 Fuzzy Controller Design	33
4. Details Design and Implementation	37
4.1 Priority Based Bandwidth Management.....	37
4.1.1 The Priority Based Bandwidth Management Algorithm	40
4.2 Preprocessing and Tuning of Initial Results.....	44
4.3 Implementation of Fuzzy Controller	45
4.3.1 The Fuzzy Controller Parameters	45
4.3.2 Design of fuzzy Logic Controller	47
5. Experimental Results and Discussions.....	51
5.1 Packet Inspection Mechanisms	51
5.1.1 Bittorrent Protocol Signatures	51
5.1.2 Packet Detection Module	52
5.1.3 Validation of Payload Signature Based Detection Module	53
5.2 The Priority Based Bandwidth Management Algorithm (PB2MA)	55
5.3 Fuzzy Controller	58
5.3.1 Bandwidth Consumption by various protocols	58
5.4 Threshold Marking	63
5.5 Overview of the Results	65
6. Conclusions and Future Work.....	67
6.12 Conclusions.....	67
6.2 Future Works	68
REFERENCES.....	69
LIST OF PUBLICATIONS	73

LIST OF FIGURES

Figure No.	Page No.
Figure 2.1 Protocol stack and Encapsulation	11
Figure 2.2 Type of Packet Inspection	15
Figure 2.3 Typical DPI implementation	18
Figure 2.4 Aho Corasick model	19
Figure 2.5 Characteristic Function of a Crisp Set A.....	24
Figure 2.6 Characteristic Function of a Fuzzy Set A	25
Figure 3.1 Level 0 System design	28
Figure 3.2 Flow chart of PBPSM	29
Figure 3.3 Fuzzy Inference Model	32
Figure 4.1 System design for Priority based packet Payload Matching	37
Figure 4.2 Internal structures of the packets.....	40
Figure 4.3 Priority Based Bandwidth Management Algorithm (PB2MA)	41
Figure 4.4 Order of list searched in PB2MA	43
Figure 4.5 Algorithm for Fuzzy Controller.....	46
Figure 4.5 Fuzzy Rules and inference	49
Figure 5.1 Netsniff-ng tools	54
Figure 5.2 Shell script for counting number of packets.....	55
Figure 5.3 Running time of priority versus non priority based BWM algorithm ...	57
Figure 5.4 Calculation of high bandwidth consumption by BWM algorithm	60

Figure 5.5	Calculation of low bandwidth consumption by BWM algorithm	60
Figure 5.6	Bandwidth consumption by various applications	62
Figure 5.7	Bandwidth consumption by applications after marking threshold	64

LIST OF TABLES

Table No.	Page No.
<hr/>	
Table 4.1 Threshold values of ART, NPP and PPR	47
Table 5.1 Signature pattern of Bittorrent protocol used for matching.....	51
Table 5.2 Number of packets identified by port based v/s proposed technique ..	52
Table 5.3 Number of protocol detected in trace file.....	56
Table 5.4 Bandwidth consumption by various protocols	61
Table 5.5 Bandwidth consumption by protocols after marking threshold.....	63

Chapter 1

Introduction and Problem Statement

1.1 Introduction

Network traffic in a network consists of data and control information. This information is encapsulated in packets. Different packets may contain data generated by different applications. An application is a program or software which runs at server end or at end host and generates data. This data is communicated over the network. For doing so it uses a protocol so that different intermediate devices can communicate accordingly. Currently there are over two hundred protocols in use. Each protocol is either assigned a unique port number by IANA or they use some random port number to communicate. A network traffic classification is broadly categorization of packets. The amount of traffic on the Internet has increased, both in terms of amount of traffic, and in variety of applications. The introduction of real time applications like voice, video and other applications has changed the way the Internet is used. This has triggered the need for a change in traffic handling methods on the Internet [1]. Many new applications like peer-to-peer (P2P), Voice over IP (VoIP), consume lots of bandwidth. It is really hard to manage such application in any network, as P2P networks are based on the idea of decentralization, sharing of resources across large number of users. Due to rapid increase in total download amount, speed, availability, scalability, P2P traffic costs almost 60% network bandwidth, and saturates ISP networks by Downstream: 50-65% and Upstream: 70-80% [2]. Hence it is really hard to provide QoS to applications unless we categories and classifies them properly.

Detailed knowledge of the network traffic usage is essential for network operation and administrations. The study of network traffic helps in network planning, capacity provisioning, fine tuning of charging schemes and it is also a key aspect of security monitoring. The knowledge of traffic is also helpful in providing QoS to the end users. The trend analysis, network-based QoS mapping, application based access control, lawful interception and intrusion detection, diagnostic monitoring, service differentiation, application-specific traffic engineering are also dependent on the internet traffic profile.

The classification empowers network administrators to create policies to restrict and reduce the amount of undesirable traffic and ensure business critical traffic is prioritized. Also the good understanding of the traffic analysis and trends of different applications can provide important inputs to the network equipment design. The proper understanding of Internet traffic profile at any interface is key issue for several reasons, including network traffic management and prioritization of critical traffic over other less priority traffic like peer to peer traffic. The internet traffic cannot be managed properly unless it is measured and classified properly.

Recently, traffic classification has become a difficult task. In some applications such as P2P are built with features specifically intended to bypass common traffic classification techniques. Hence the detailed internet traffic profiling of several bandwidth intensive and time insensitive protocols like P2P, is especially main concern for Internet Service Providers (ISPs) due to several reasons as explained by Madhukar et. al. [3].

1. Many of current Internet access technologies have asymmetric upstream and downstream bandwidth, to exploit existing access technologies while limiting operating costs for the ISPs. The underlying assumption is that Internet users download much more than they upload, as they do with the Web. However, in Peer to Peer applications, users may upload as much as they download. If a large proportion of the Internet traffic is Peer to Peer, then the underlying assumption of traffic asymmetry may be invalid [3].
2. To cope up with bandwidth demands created due to P2P, increasing the network capacity is expensive solution, and only effective on a short-term basis. Peer to Peer application traffic may soon expand to occupy the additional increased capacity as well, making network congestion inevitable and permanent hindrance to the quality of services provided by the ISPs [3].
3. Many Peer to Peer applications are bandwidth-intensive as they capable of consume almost all the channel available, as a result it leads to Network congestion. Excessive network congestion could lead to dissatisfied customers and possible customer churn [3].

The Universities and campus network operators have similar requirement to limit P2P traffic to avoid congestion and reduce cost which is charged by upstream ISPs.

The aim of network traffic classification is to find out what type of applications is run by the end users, and the share of a particular application in the total traffic mix at a particular network interface. Classification is the key issue for network now a day to provide QoS to end users. New applications, for example P2P-based VoIP, have not only changed characteristic of traffic and behavior of network, but also proposed more different requirements on QoS provisioning ability of network infrastructure. To understand the behavior of network and applications, firstly we should investigate the traffic characteristic of different applications. A deep understanding of traffics of different applications is significant to protocol research [4, 5], anomaly detection [6], network operation and application deployment [7]. There are several attempts to categories the traffic at different level such as at network level, transport level or combination of both and to a some extent even at application level.

The classification itself is key issue for the network administration as most of bandwidth is consumed by time insensitive applications like P2P and hence there is limited bandwidth available for other application. Also several application needs priority based services like real time or time critical application, hence ISP need to assign them free channel so that they can be served within time limit. Application bandwidth utilization may present a correct image of Protocols and application and the Administrator had a power to use it wisely. The bandwidth utilization may also leads to traffic shaping in the network which is also very important. In order to optimally use the available bandwidth and providing service to the end user's for time critical task.

So far the basic methods of traffic classification are Port based, Flow Based, and some statistical method deploying both, application layer payload or deep packet inspection. Some other researcher try to find alternative method for exact identification of network traffic as future of internet now mostly depends on way we shape out network usage and traffic management. The optimal usage of available resources is desirable now. As increasing number of equipment hardly help to curb the demand of added bandwidth as

no matter if we are able to increase the capacity of channel bandwidth intensive and time insensitive P2P download application is ready to utilize the added bandwidth.

1.2 Motivation

A more reliable traffic classification method is always in demand with high network bandwidth i.e. at 10GbE/OC192 (Approaching 40GbE/OC768) line speed, the new application protocols, and frequent practice of traffic misleading techniques.

The classification can be based on the TCP/IP level i.e. based on port number or may be based on flow. The most common identification technique is based on the inspection of 'well known' port numbers. Port-based methods are simple because many well-known applications have specific port numbers (for instance, HTTP traffic uses port 80 and FTP port 21). But it suffers because many applications are no longer use predictable fixed port numbers. Although some applications like HTTP, FTP use ports registered with the Internet Assigned Numbers Authority (IANA) but many applications only utilize available 'well known' default ports that do not guarantee an unambiguous identification [1]. Even these applications can end up using nonstandard ports because [1]:-

- a. Non-privileged users often have to use ports above 1024.
- b. Users especially P2P users may be deliberately trying to hide their existence or bypass port-based filters.
- c. Multiple servers are sharing a single IP address.
- d. Furthermore some applications like passive FTP use dynamic ports unknowable in advance.

A more reliable flow based technique involves stateful reconstruction of session and application information from packet header. Flow based classification is done using 'classic 5-tuple lookup' i.e., they scan five tuples [8] which are source transport layer address (typically TCP or UDP), destination transport layer address (typically TCP or UDP), source IP address, destination IP address and Service type (e.g. FTP, HTTP, SMTP, POP3). Although this avoids reliance on fixed port numbers, it imposes significant complexity and processing load on the identification device, which must be

kept up-to-date with extensive knowledge of application semantics, and must be powerful enough to perform concurrent analysis of a potentially large number of flows.

The application level traffic classification is done using deep packet inspection (DPI) or payload content inspection (PCI). The payload content inspection based classification process is more complicated where the complete content of packet is inspected against the signature. Here the signature of protocol is stored in the form of regular expression or fixed patterns. This pattern then is examined against the packet payload to determine whether a particular protocol is present or not. The protocol identification is done in order to find out whether the data is generated by a simple HTTP application or the packet contains a data of any specific protocol like P2P protocol such as Ares (Ares Galaxy, Warez P2P), Bittorrent (used by application like BitTorrent.Net, G3 Torrent, mlMac, MLdonkey, QTorrent, Shareaza, Torrent, etc.), Direct Connect (used by application like BCDC++, DC++, NeoModus Direct Connect, etc.), Fasttrack (used by application like Grokster, iMesh, Kazaa, Morpheus, etc.), eDonkey (used by application like eMule, Overnet, etc.), Gnutella (used by application like BearShare, iMesh, Gnotella, Gnucleus, GTK-gnutella, LimeWire, Mactella, Shareaza, etc.), MANOLITO/MP2PN (used by application like Blubster, Piolet, RocketItNet) or OpenNAP. P2P traffic present in network is one of the most challenging traffic types to classify and categorize due to its nature to use random port and frequent practice of traffic misleading techniques.

The application payload based technique suffers as it need to perform signature matching in real time and against packet that flow at line speed i.e. 10GbE/OC192. Furthermore they face the problem of low packet throughput, high memory requirement, latency, low accuracy, very high false positive and negative alarm at line speed of 10GbE/OC192 link. Several authors tried to improve existing algorithms to fit into DPI environment but at 10GbE/OC192 (Approaching 40GbE/OC768) line speed we still need a really very fast searching algorithm or there is need to explore the existing hybrid technique like fuzzy logic, artificial neural network, expert system, supervised and semi supervised classification algorithm.

Fuzzy logic controller based application layer packet identification system helps in accurate measure of bandwidth utilization as it is able to processes noisy data i.e. data

with unclear boundary and also once trained, it is really fast in processing the data in real time. Also, fuzzy logic rules help in smoothing the abrupt separation of normality and abnormality bandwidth utilization and hence give the picture of correct deterministic bandwidth measure by a particular protocol.

1.3 Statement of the Problem

“The aim is to design and implement an efficient fuzzy controller based technique for network traffic classification to improve QoS”

Problem Description: The bandwidth in the networks is always limited and should be used wisely in order to keep stack for critical and high priority applications. Data flow in the networks can be classified broadly into two types of connection – real time and non real time. Real time data flow requires data to be sent within the deadline while for non-real time data flow there is no such deadlines. The aim is to maintain QoS of the real time connections while ensuring proper throughput for non-real time connections keeping enough stack of bandwidth for the high priority and time critical applications. The traffic classification is done on payload data of each packet i.e. each packet is checked against hundreds of already available signatures. This payload matching process is based on the exact string matching and may serve as a bottle neck for the overall network. So it needs to be fast and efficient with very low fast positive alarm rate. To do so a priority based classification method is proposed the technique to classify HTTP signature before other protocol. This helps to reduce the ambiguity presents in the signature. The ambiguity occurs as many new protocols use HTTP as their underlying protocol and hence these protocols are wrongly identified as HTTP. Use of fuzzy controller helps to generate the bandwidth utilization of all protocols during surveillance time on an interface. This bandwidth utilization helps to shape the network traffic in future.

Thus the above problem has been divided into following sub problems:

- (i) To record the flow of the connection set in a priority list based on header information of TCP/IP layer to decide priority.

- (ii) To match the packet's application with existing signatures and update the priority list.
- (iii) To implement fuzzy logic controller to decide the bandwidth consumption.
- (iv) To log the protocol into database along with bandwidth utilized by the protocol.
- (v) To mark the selective packets in order to achieve desire threshold bandwidth.

1.4 Organization of the Report

This dissertation report comprises of six chapters including this chapter that introduces the topic and states the problem. The rest of the report is organized as follows.

Chapter 2 gives the background study of network traffic classification, Deep Packet Inspection and Fuzzy set and Fuzzy logic based controller used in other applications.

Chapter 3 describes the slot reservation concept and the proposed solution.

Chapter 4 gives the implementation details in terms of Algorithm: priority based bandwidth management and fuzzy controller, logical flow graph of complete implementation, datasets, signature used, design of fuzzy controller, test bed description, assumptions and log description to log the intermediately flow and results.

Chapter 5 discusses and analyzes the results obtained from the implementation.

Chapter 6 concludes the dissertation work and gives suggestions for future work.

Chapter 2

Background and Literature Review

In this chapter we discuss the technical details of network traffic classification in order to get a brief idea about network traffic classification methodologies. Then we give a brief idea about packet inspection methods and finally we discuss about hybrid technologies like artificial intelligence, neural network, Fuzzy logic etc and a overview of fuzzy based method implied in classification process.

2. 1 Quality of Service

2.1.1 Quality of Service Overview

Quality of service (QoS means providing consistent, predictable data delivery without any delay and within the time provided. There are n numbers of way to characterize Quality of Service (QoS) for a particular application. In other words Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow. It is basically satisfying customer application minimum requirements.

QoS does mean availability of enough bandwidth or creating bandwidth space. It isn't possible for the network create additional bandwidth, and if somehow we chose the expensive method of adding even tera bytes of bandwidth still the bandwidth intensive application utilizes that in no time. QoS only manages bandwidth in an optimal way so that it can be used more effectively to meet the range of application requirements. The main aim of quality of service is to provide some level of control and predictability to the administrator beyond the current best effort service. The implementation of QoS should be done in such a way that best-effort traffic is not starved.

Any QoS assurances are only as good as the weakest chain between sender and receiver as its all depends on the communicating network and as long as network stands the QoS is stands. Hence any solution to support real-time traffic should take into consideration

the overall QoS architecture that spans the entire network. The required bit rate, delay, jitter, packet dropping probability and/or bit error rate may be guaranteed. Quality of service guarantees are important if the network capacity is insufficient, especially for real-time streaming multimedia applications such as voice over IP, online games and IP-TV, since these often require fixed bit rate and are delay sensitive, and in networks where the capacity is a limited resource, for example in cellular data communication. In other words, just increasing the bandwidth at the backbone is not sufficient to support applications that are running on a network with low bandwidth and high congestion[9].

2.1.2 Role of Quality of Service

The network architecture of the Internet itself is very simple and based on the concept that packets (datagrams) with source and destination addresses can traverse a network of (IP) routers independently, without the help of their sender or receiver. The overall protocol stack itself makes this task much easier and the sender/receiver application or program lies on the top of this stack. The robustness of Internet is a result of this simplified model, which is keeping it still running despite an enormous growth in its size and traffic over the past several years.

2.1.2.1 The Classical End-to-End Principle

A central design principle of the internet for the last 25 years has been the “end to end” argument[10]. This basically meant that the data link, the network and the transport layers of the protocol stack should only care about sending data packets from the sender to the receiver, without worrying about their content, their security, Quality of service, or even the fact that they reached their destination. These issues should rather be addressed higher up the protocol stack, at the session or application layers.

Application data (emails, music files, voice streams etc.) is encapsulated into TCP packets, which are in turn encapsulated into IP packets, which are in turn encapsulated into Ethernet frames, which are then sent over the wires or radio. Because of this encapsulation, the lower protocol layers do not have to care about what is in the packets or frames they transport. Because of encapsulation, the network that does not care about the content of the packets it moves is a network that cannot easily be used for censorship

or communications surveillance. Maximum freedom for the endpoints means maximum freedom for the users.

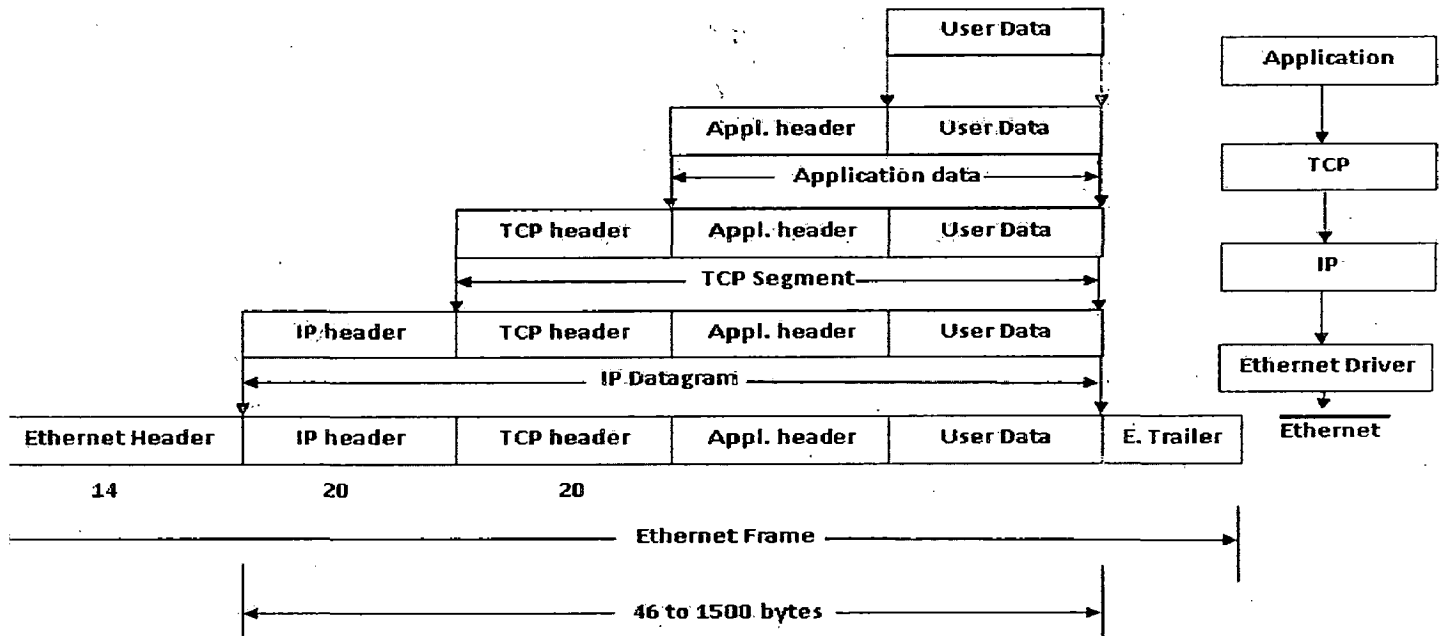


Figure 2.1. Protocol Stack and Encapsulation

The internet has so far been a loose network of interconnected data networks that share few central characteristics[11].

- [1]. *Technical Simplicity*: Because of the layered approach, they are only connected through the TCP/IP protocol suite and a shared address space. Therefore, they are highly open to new transportation methods (WiMax, UMTS etc.) as well as new applications (e.g. Twitter, Bittorrent, or XMPP/Jabber).
- [2]. *Political Freedom*: Because the higher-layer payloads are encapsulated for the lower layers, the users have end-to-end communication channels at the application layer, which are normally not interfered with in transport.
- [3]. *Economic Openness*: Because of the openness for new applications, they do not discriminate traffic according to its source, therefore treating all innovations at the application layer equally and giving them a fair chance to succeed at the market.

There is a price to pay for this simplicity, however. The reason IP is simple is because it does not provide many services. IP provides addressing, and that enables the

independence of each datagram. IP can fragment datagrams (in routers) and reassemble them (at the receiver), and that allows traversal of different network media. But IP does not provide reliable data delivery. Routers are allowed to drop IP datagrams without notice to sender or receiver. IP relies on upper-level transports (e.g. TCP) to keep track of datagrams, and retransmit as necessary. These “reliability” mechanisms can only assure data delivery; neither IP nor its high-level protocols can ensure timely delivery or provide any guarantees about data throughput. IP provides what is called a best effort service. It can make guarantees about when data will arrive, or how much it can deliver.

This limitation has not been a problem for traditional Internet applications like web, email, file transfer, and the like. But the new breed of applications, including audio and video streaming, demand high data throughput capacity (bandwidth) and have low-latency requirements when used in two-way communications (i.e. conferencing and telephony). Public and private IP Networks are also being used increasingly for delivery of mission critical information that cannot tolerate unpredictable losses.

Hence this simplicity IP protocol stack now demands some efforts to provide QoS by network itself so the traffic classification is now a key issue in order to assure QoS to these task and end users.

2.1.3 The Benefits of QoS

As business is increasingly conducted over the web from booking a railway ticket or airline ticketing even movie ticketing are now booked online. Other than these sensitive bank information sends over network for fund transfer etc a failure in transaction due to network problem may leads to dissatisfaction to the end user, it becomes more important that IT managers ensure that these networks deliver appropriate levels of quality. QoS technologies provide tools for IT managers to deliver mission critical business over the public network.

With the passage of time, applications are getting more demanding. Mission-critical applications deployed over IP networks increasingly require quality, reliability, and timeliness assurances. In particular, applications that use voice, video streams, or multimedia must be carefully managed within an IP network to preserve their integrity.

2.1.4 QoS Parameters

Following are a list of parameters that give a measure of Quality of Service[12 1998 #388]:

- [1]. **Delay** is the elapsed time between a node sending a message and another node receiving that message. It is a measure of the amount of data held in transit in the network. The greater the delay between sender and receiver, the more insensitive the feedback loop becomes and therefore the end-to-end protocols become more insensitive to short term dynamic changes in network load. For interactive voice and video applications, the introduction of delay causes the system to appear unresponsive, e.g. in an Internet telephony application there is sometimes a delay of upto several seconds before the receiver can hear what the sender is saying. This greatly hinders interactive communication between the two parties.
- [2]. **Jitter** is the variation in end-to-end transit delay. It is an aberration that occurs when video or voice is transmitted over a network, and packets do not arrive at its destination in consecutive order or on a timely basis, i.e. they vary in latency. High levels of jitter in applications are unacceptable in situations where the application is real-time based, such as an audio or video signal. In such cases, jitter causes the signal to be distorted which is particularly damaging to multimedia traffic. For example, the playback of audio or video data may have a jittery or shaky quality.
- [3]. **Bandwidth** is a measure of data transmission capacity. It is the maximal data transfer rate that can be sustained between two end points. By increasing bandwidth we can transfer more data. Network bandwidth can be visualized as a pipe that transfers data. The larger the pipe, the more data can be sent through it. By increasing bandwidth, we can always achieve QoS. This is a brute force solution and not applicable because bandwidth is not cheap. Hence the issue here is to obtain certain level of QoS by using the minimum bandwidth required.
- [4]. **Reliability** is a property of the transmission medium and can be thought of as the average error rate of the medium. An unreliable or error-prone network is a result

of faulty channels that not only drop packets in transit but also alter their order. Unreliability causes induced distortion in the original signal at the receiver's end.

2.2 Packet Inspection Mechanism and Issues

2.2.1 Stateful/Shallow Packet Inspection

It is basically, a process in which the headers are parsed, and the results are compared to a rule set defined by the system administrator. It has access to Layers 3 and 4 of the OSI stack (sometimes Layer 2, as well). SPI firewalls perform the 'classic 5-tuple lookup' that is, they scan five tuples [8] which are source transport layer address (typically TCP or UDP), destination transport layer address (typically TCP or UDP), source IP address, destination IP address and Service type (e.g. FTP, HTTP, SMTP, POP3) These rule sets are commonly based upon above fields or a combination of the two and defines the type of traffic is subsequently allowed or denied.

2.2.2 Medium Depth Packet Inspection

Medium Depth Packet Inspection is done by AP(Application proxies) or gateway. An AP is providing intermediary services to the hosts/users that reside on different networks/locations while maintaining complete details of the TCP connection state and sequencing. In practice, a client host (running, for example, mail service) first negotiates a service request with the AP, which acts as a surrogate or medium for the host that provides the service (the web server). Hence there are two connections are required for each session - one between the client and the gateway or proxy server, and one between the gateway or proxy server and the server. there is no direct link between hosts. Additionally, proxy server also provide limited amount of packet filtering based upon rudimentary application-level data parsing.

2.2.3 Deep Packet Inspection (DPI)

Deep Packet Inspection (DPI) is a computer network surveillance technique that uses device and technologies that inspect and take action based on the contents of the packet i.e. the complete payload of packet rather than just the packet header which includes data up to layer 7 of OSI model(see figure 2.2).

The common packet inspection analyzes only the content below the layer 4 of the IP packet, including the source address, destination address, source port, destination port and the protocol type. It identifies application types in the network through the port number.

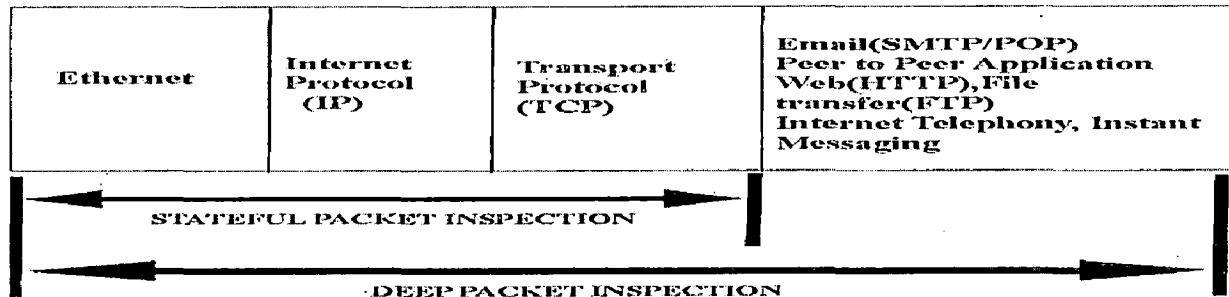


Figure 2.2. Type of Packet Inspection

2.2.4 Challenges for DPI

There are several challenges to build an efficient DPI system. The most common are [13]

2.2.4.1 The false alarms

An Intrusion Detection System (IDS) might generate more than thousands of alerts per day when deployed in any infrastructure. The number of generated alarms that need to be reviewed can escalate rapidly, making the task very difficult to manage. Moreover, due to this, a significant problem is faced by current IDS technology is the high level of false alarms. Snort [14] generates as much as 69% of false alarms.

2.2.4.2 The search algorithm complexity

The complexity of an algorithm is important aspect for implementing a signature based systems like deep packet inspection. As most of the time system is busy performing string matching. This is the main concern, as string matching time accounts total share of 40% to 70% [15] of the Snort [14] running time i.e. most of time the IDS busy with string matching activity so efficient algorithm is desirable.

2.2.4.3 Rapid growth in intruder signatures

With advancement in the number and types of attacks, there is a need to define the corresponding intruder signature. Hence, the pace of increasing intruder signature is very high. So the DPI system must be scalable. Snort [14] contains more than 17209 rule sets(as on 25 august 2010) in version 2.8.6.

2.2.4.4 Overlapping/similarity of signature

The most of the false alarms generated [14]-[16] by snort are basically due to http signature overlapping as there are 1096 rules in http rule set [13].

2.2.4.5 Location of signature or pattern unknown

Compared to other IDS/IPS systems the deep packet inspection is more difficult to implement as it examines the whole packet and the pattern of threat/application data is not localized to a particular location. So, the complete payload needs to be inspected against signatures.

2.2.4.6 Data may be Encrypted or coded

The data which is encrypted cannot be simply inspected by DPI so, DPI component must be deployed behind some decryption unit.

2.2.4.7 Speed of Line inspection

The current communication systems works at a speed of 10GbE/OC192 and approaching to 40GbE/OC768.Hence, the deep packet inspection system need to work at such high speeds otherwise, it might be a bottleneck in the system.

2.2.5 Design Issues for DPI

The DPI systems need to fulfill certain design principles in order to keep in pace with current attack and communication channel speed. These are

2.2.5.1 Availability of signature

The signature must be dynamically updated and must be able to define and defend current threats.

2.2.5.2 Performance and Scalability of system

The hardware based systems, especially ASIC are really fast and can do matching at line speed but the main problem with them is that they are not reconfigurable. In order to update them we need to design chip again and re setup factory to fabricate. Software systems are scalable in nature but need really very fast processor to do operations at line speed of 10GbE/OC192.

2.2.5.3 Space & Power consumption

The space is key issue in design of DPI, as rapid change in signature might lead to memory scare situation. So the system must be memory efficient. Power consumption also plays a key role when systems are deployed at remote location. Hardware like FPGA is reconfigurable but they consume as much as 40 times power compare to ASIC.

2.2.5.4 Quality of Service issue

DPI needs to address some of the Quality of Service issues like bandwidth management. So it is a pure over head on system to maintain efficient flow of traffic as ISP (internet Service Provider) uses DPI for providing privileged services to their customer. Hence DPI system must provide the support for the same.

2.2.6 Existing Technologies for DPI Implementation

2.2.7.1 String Matching Algorithm

To implement DPI on hardware systems or on software systems we need to develop a process which includes techniques like exact string matching/searching algorithm, regular expressions and a DFA of consisting signatures. Figure 2.3 shows detailed diagram about relation among them.

Signature consists of sequence of alphabets (string) or regular expressions. Regular expression may represents repetition of characters, special characters, pattern with length constraints and wildcards. Now a day PCRE (Perl Compactable Regular Expression) is widely used to define the signatures of virus, worms, attacks etc.

Every application has its own signature pattern which may be written in PCRE. Finite state machine are used to represent these signature in either DFA (Deterministic Finite Automaton) or NFA (Nondeterministic Finite Automaton) or lezzy DFA, which is somewhere in between both. Finally signatures matched against packet data (including

payload) using pattern matching algorithms. Aho-Corasick, Boyer Moore and Wu-Manber are widely used pattern matching algorithms in most of the IDS/IPS and deep packet systems. But there is need of significant changes in implementation techniques of

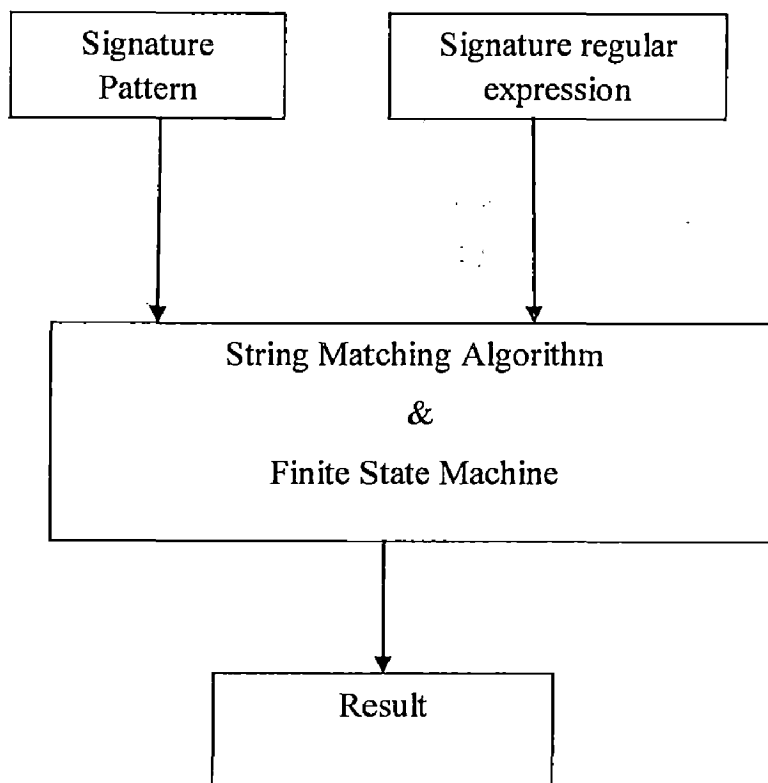


Figure 2.3. Typical DPI Implementation

these algorithms to work at such a high speed.

2.2.7.1 Exact string Searching Algorithms

String matching algorithms can be divided into exact and approximate string matching algorithms. DPI implementation basically depends only on exact string matching algorithm to reduce false alarms.

One of the most oldest exact string matching algorithm is Aho-Corasick algorithm [17]. It is able to find all occurrences of a pattern in a text. It constructs a finite state machine using patterns or keywords. Basic idea behind this algorithm is quite simple. It works as follows: Starting from root node which is the initial state each transaction adds a state to FSM and the result is successful if final accept state or end of pattern is achieved. If in

between there is a mismatch, then the failure pointer transfers the state to root or other similar state. Figure 2.4 illustrates the states of Aho Corasick[17] FSM for set of keywords {hers, his, she, he }.The Aho-Corasick algorithm is still in use due to its linear time complexity and simplicity in implementing. Along with Aho-Corasick other algorithms like Boyer-Moore [18] and Wu-Manber [19] algorithms are used in Intrusion detection/prevention systems like Snort [14] [16] BRO [20], Linux L7-filter [21] with/without modification. The implementation in IDS/IPS is slightly less sophisticated

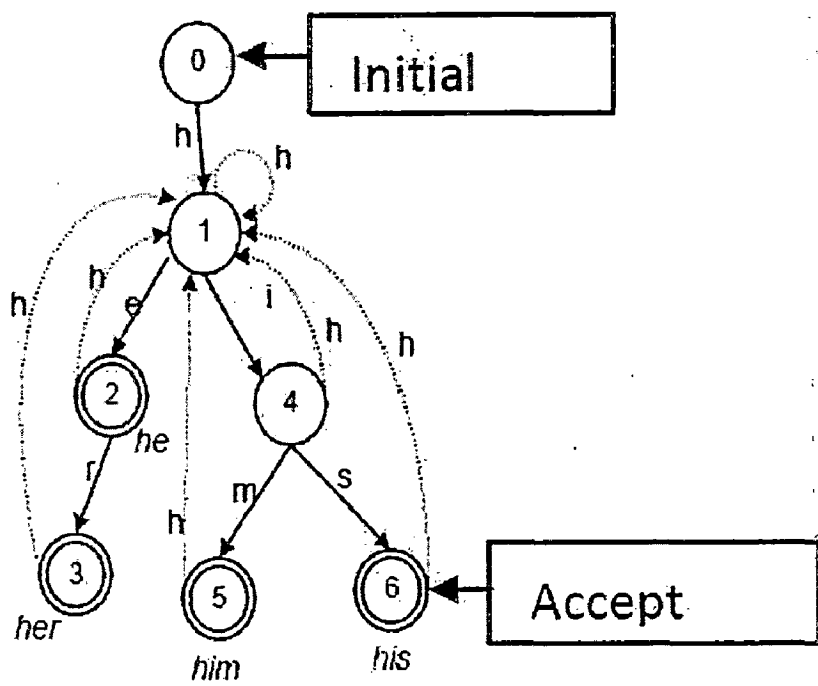


Figure 2.4. Aho Corasick Model

than the original papers describing these algorithms.

The Aho-Corasick algorithm is one of the best algorithm for string matching but it requires large cache for storage. Tuck et al. [22] implement Aho-Corasick algorithm with bitmap node compression and path compression to gain compact storage and worst-case performance. They show that the use of such compression techniques they are able to gain almost 50 times in database size reductions on current rule sets. Hence, it can be easily implemented with FPGA, ASIC, and Network Processor as the space required by Aho-Corasick is reduced by remarkable ratio of 50%. There are several variants of Boyer-Moore algorithm especially designed for deep packet inspection such as Set-wise Boyer-

Moore-Horspool (SBMH) algorithm [23] which was proposed by Fisk and Varghese based on Boyer-Moore-Horspool algorithm [24] which matches the rule set simultaneously.

The set of patterns can be compared easily to any position in the text quickly by storing the reversed patterns. Their experiments show that on medium-sized pattern sets, SBHM is faster as compared to both Aho-Corasick and Boyer-Moore algorithms. But the maximum number of shifts is bounded by the LSP (length of the shortest pattern) in the pattern set. Rafiq et. al. suggest an algorithm [25], with some improvisation over Boyer-Moore algorithm. To perform better in long pattern, long text, and large alphabet set. They suggest two way checking and speed searching phase. As a result the complexity decreases by a considerable value as worst, average and best case complexities are $O(nm)$, $O(n)$ and $O(n/m)$ respectively. FNP Algorithm (Fast String Matching) [26] proposed by Tai et al. works well when size of rule sets is less i.e. $LSP < 4(35\% \text{ of snort [14] rule sets})$. In such conditions FNP gives better performance as compared to Aho-Corasick, Boyer-Moore (SBHM) & Wu-Manber algorithm. They have shown that the FNP algorithm is very efficient for small LSP regardless of search set size, further same group of author designed FNP2 [26], which uses the characteristic of signature rule sets and the hardware facility of Network Processor to maximize performance. They implemented FNP2 on Vitesse IQ2000 Network Processor platform to evaluate the relation between performance and the number of memory accesses for processing multi-pattern matching. Their experimental results reveal that FNP2 is far better than the other algorithms {AC, SBMH, E2xB and MWM}, when the LSP is small. Also this design is able to process L7 payload efficiently when implemented on network processor. The complete list of string matching algorithm is listed in our paper [27].

2.2.7.2 Regular Expression

The pure exact string matching is no longer valid in deep packet inspection as new signatures contain

- i. Wildcards : length restrictions ‘?’ & ‘+’ notations), dot-star notations (e.g. pattern for Internet radio protocol is “membername.*session.*player” contains two ‘.*’ notation.
- ii. Character ranges : pattern for ftp protocol is “^220[\x09_\x0d=~]*ftp”, contains class inside brackets that includes all the printing characters.
- iii. Counting constraints :Bounded repetition of simple characters, sub-patterns, character set and wildcards.
- iv. Perl compatible regular expressions:

The PCRE library is a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl 5.PCRE has developed an extensive and in some ways unique feature set which includes Consistent escaping rules, Extended character classes, Minimal matching, Unicode character properties, Multiline matching, Newline/linebreak options, Backslash-R options, Other beginning of pattern options Back references, Subroutines, Atomic grouping , Look-ahead and look-behind assertions , Escape sequences for zero-width assertions , Recursive patterns, Generic callouts us with these all features it is possible to write every possible combination of the signature.

2.3 Network Traffic Classification

The network traffic classification is detail overview of the bandwidth consumption/utilizations in the network by various applications i.e. it is a snapshot of traffic generated by various application protocol at any interface in network.

2.3.1 Role of Network Traffic Classification

It is the complete overview of computer network traffic which helps in

- a) *optimize or guarantee performance*: Some applications like real time application need minimum QoS in order to run properly hence the traffic shaping need to be

done which use the traffic classification for some kinds of packets by delaying other kinds of packets that meet certain criteria.

- b) *Improve latency* : Latency itself a big issue in network so traffic must be properly managed to improve it. If a link becomes saturated to the point where there is a significant level of contention (either upstream or downstream) latency can rise substantially. As a result, traffic shaping can be used to prevent this from occurring and keep latency in check [28].
- c) *Increase usable bandwidth*: Today mostly bandwidth is consumed by time insensitive high bandwidth intensive application like P2P. But if we delay or stop traffic generated by these application during peak time in order to increase usable bandwidth for other application.
- d) *Bandwidth throttling*: It is a means to control the volume of traffic being sent into a network over a specified period by a specific user. It is a reactive measure employed in communication networks to regulate network traffic and minimize bandwidth congestion for applying bandwidth throttling the network administrator must have image of the network traffic at that incidence. The network administrator may employ bandwidth throttling to help limit network congestion and run it smoothly at least during peak hours which bandwidth is needed for critical applications. The Internet Service Provider (ISP) may use bandwidth throttling to reduce a user's usage of bandwidth that is supplied to the local network as they had clear image of user's bandwidth consumption profile over a period of time. This can be used to actively limit a user's upload and download rates on programs such as BitTorrent protocols and other file sharing applications, as well as even out the usage of the total bandwidth supplied across all users on the network.

2.3.2 Types of Network Traffic Classification

The network traffic broadly classifies follows:

- 1) *Port Based classification*: The port based classification is done by using the well known port number of application which are assigned by The Internet Assigned Numbers Authority port-numbers[29].
- 2) *Flow Based Classification*: It is based on the flow of the network traffic a flow is the active connection between to communicating machine[30-32].
- 3) *Using Soft computing with other approach*: Many researchers [1, 31-33] tries to use soft computing technique for network traffic classification.
- 4) *Application signature based approach*: In this approach the well known signature of application are matched against the payload of packet [34].

2.4 Fuzzy Controller and Fuzzy Logic

In 1965 Lotfi A. Zadeh[35-37] introduced the concept of fuzzy logic. He was professor for computer science at the University of California in Berkeley. Fuzzy Logic (FL) is a generally a multi-valued logic, which allows intermediate values to be defined between conventional evaluations like true/false, yes/no, high/low, etc. The philosophy of rather tall or very fast is formulated mathematically using the concept of fuzzy logic. It is processed by computers so as to give a human like way of thoughts in the programming of computers. Fuzzy systems are the substitute to conventional concept of set membership and logic which has its origins in ancient Greek philosophy.

The accuracy of mathematics has its success due to the efforts of Aristotle and the philosophers who came after him. They efforts to formulate a brief theory of logic, and later mathematics, hence the “Laws of Thought” were given. One of them, the “Law of the Excluded Middle,” states that every proposition must either be True or False. The first version of this law was proposed by Parminedes around 400 B.C. At that time there were strong and immediate objections. Heraclitus proposed that things could be simultaneously True and not True. Plato laid the foundation of what becomes the fuzzy logic which indicates that there is a third region (beyond True and False).

The basic notion of fuzzy systems is a fuzzy (sub) set. In classical mathematics there are values which are crisp. For example, the possible values from the set X of all real numbers lies between 0 and 1. From this set X a subset A can be defined. The

characteristic function of A which assigns a number 1 or 0 to each element in X, depending on whether the element is in the subset A or not, is shown in Figure 3.3. The elements that are in the set A are assigned the number 1 and the elements that are not in the set are assigned a number 0.



Figure 2.5: Characteristic Function of a Crisp Set A.

The above concept is sufficient for several areas of applications, but it lacks flexibility for some applications like classification of remotely sensed data analysis where data values correspond to several classes. A fuzzy set allows us to define such a notation. The idea is to use a fuzzy set for making computers more intelligent. In the example, all the elements were coded with 0 or 1. A simple way to extend this concept is to allow more values between 0 and 1. Hence infinitely many alternatives can be allowed between the boundaries 0 and 1, namely the unit interval $I = [0, 1]$.

The explanation of the numbers is now assigned to all elements and is much more difficult. Again the number 1 assigned to an element means that the element is in the set B and 0 means that the element is definitely not in the set B. All other values mean a gradual membership to the set B. This is shown in Figure 2.5. The graphical representation of the magnitude of participation of each input is termed as membership function. It links a weight with each of the inputs that are processed, defines functional overlap between inputs, and ultimately determines an output. The rules use the input membership values as weighting factors to determine their control on the fuzzy output sets of the final output conclusion.

The membership function on the fuzzy set returns a value between 0.0 and 1.0. For example, a set of 0.3 has a membership of 0.5 to the set A (see Figure 2.6). It is significant to show the distinction between fuzzy logic and probability. Both operate over the same numeric range, and have similar values. In both the cases 0.0 represents False

(or non-membership) and 1.0 represents True (or full-membership). But, there is a difference to be made between the two statements: The probabilistic approach yield the statement, “There is a 50% chance that A is low,” while the fuzzy terminology corresponds to “A’s degree of membership within the set of low is 0.50.” The semantic difference is important: the first view supposes that A is or is not low; it is just that we only have a 50% chance of knowing which set it is in. By contrast, fuzzy terminology supposes that A is “more or less” low, or in some other term corresponding to the value of 0.50.

Fuzzy classifiers are one application of fuzzy theory. The expert knowledge is used and is expressed in a very simple way using linguistic variables, which are described by fuzzy sets. Expert knowledge for this variable can be formulated as a rules like IF feature X is low AND feature Y is medium AND feature Z is medium THEN R is r.

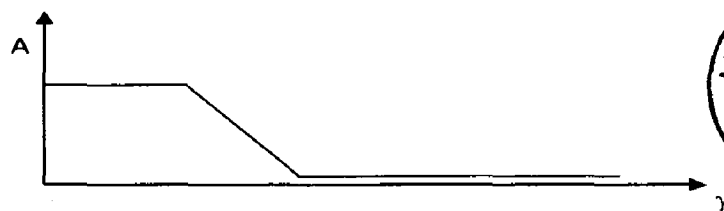


Figure 2.6: Characteristic Function of a Fuzzy Set A

Fuzzy logic, a widely deployed technology for developing sophisticated control systems [35, 36, 38], provides a simple way to get definite precise conclusion and solutions based on unclear, imprecise, ambiguous or missing input information. In general, two major components are needed to develop the fuzzy logic controller: (1) define membership functions for each input/output parameter and (2) design the fuzzy rules. The membership function is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs, define functional overlap between inputs, and determines an output response. The fuzzy logic rules use the input membership values as weighting factors to determine their influence on the output sets

2.5 Research Gaps

The port based approach[29] of traffic classification is not accurate as applications tends to use random ports. The flow based approaches is based on the flow of the network

traffic. A flow is the active connection between two communicating machines [30-32]. But it is limited to certain applications or protocols. In the payload signature based approach the well known signature of application are matched against the payload of packets [34]. It also suffers as the processing time of matching signatures is really high. Several researchers [1, 31-33] try to use soft computing technique for network traffic classification and they are able to classify the certain traffic accurately. But none of these approaches propose the bandwidth consumptions by various protocols. The payload and flow approaches are suffers from their flaws of limited protocols and high processing time. In our approach we use flow based method to indentify the flow of the traffic which helps payload method to decrease the overall processing time by narrowing its search from 100-200 protocol to 10-15 protocols. Also we use fuzzy controller for bandwidth management to mark the bandwidth consumed by each protocol. The marking of protocols helps to improve the channel capacity and overall QoS.

Chapter 3

Proposed Framework for Classification Process

3.1 System Framework

To address some issues discussed in research gaps, we propose a framework of priority based classification for network traffic. Framework comprises of broadly four independent systems:

- (a) Application signature based matching,
- (b) Pre processing and fine tuning of the initial results,
- (c) Fuzzy controller for calculation of bandwidth consumption,
- (d) Marking the required packets.

Applications signature based matching is the primary process in which the packets are classified based on the signature of application present in the payload of packet.

Pre processing consists of filtering out wanted fields from the packets. Once the required fields such as number of protocol packets, total packets, average inter-arrival time, protocol ratio are filtered out from the output of signature based matching, they are logged into mysql for further processing.

Fuzzy controller is designed to calculate the total bandwidth consumed by various processes during the period of surveillance.

Marking of the packets is done in order to drop out packets of an application to lower its bandwidth consumption based on the result returned by fuzzy controller.

Figure 3.1 shows the overall design of the proposed framework. This only shows the main parts of the framework. Internal details of the parts give in the next sections. All parts are autonomous. They are designed in a manner so that they not do need each other

for their functioning.

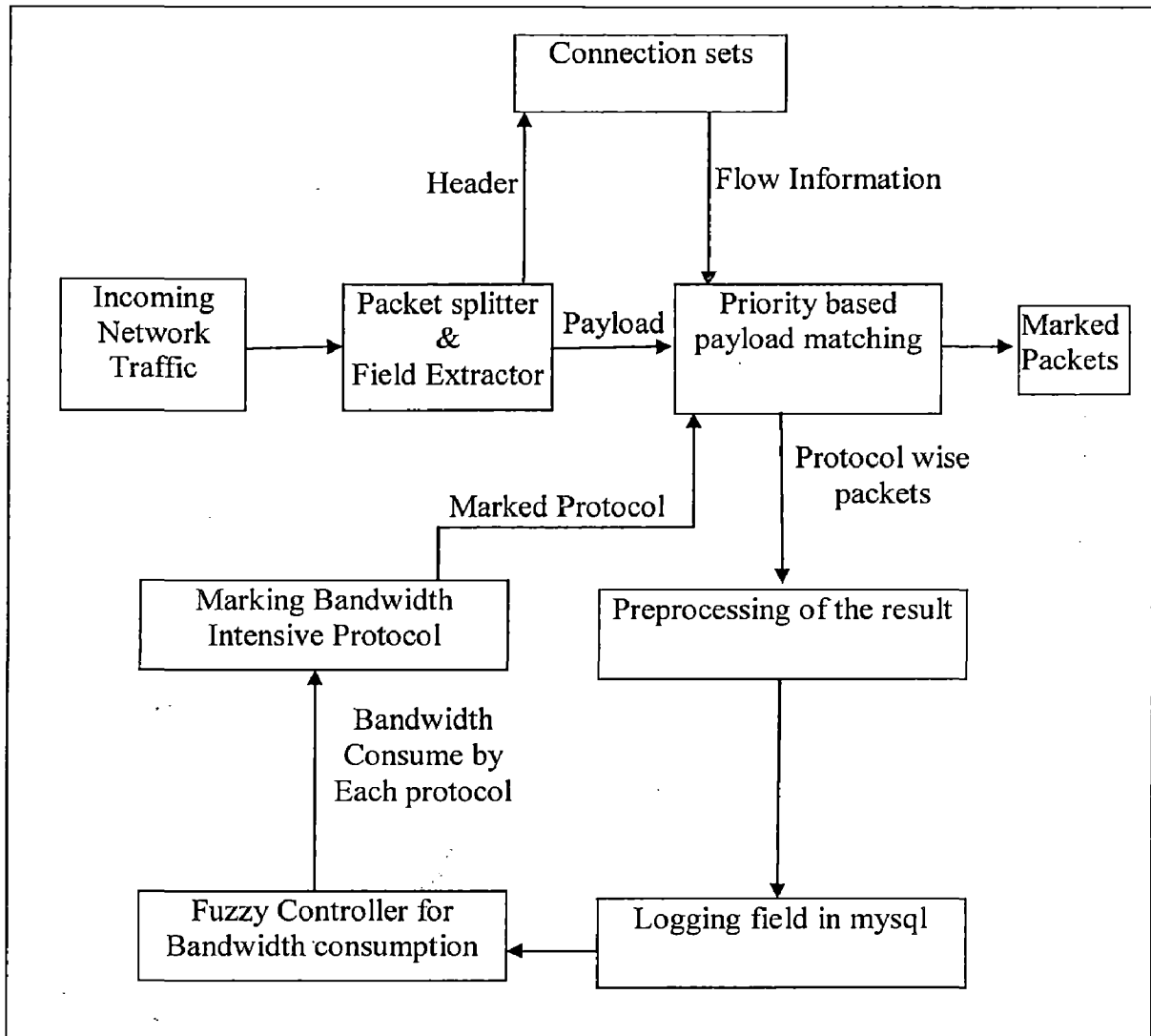


Figure 3.1 Level 0 System Design

3.2 Priority Based Payload Signature Matching (PBPSM)

As already pointed out that the main problem with port based classification is that their limited classification capability. Also it is also more prone to do false classification too. So when a packet is dropped we can't trust port based approach. So we need to design payload based approach in order to classify packets more accurately. The concept of flow based approach is introduced by extracting header information for connection set. As

shown in figure 3.1, the incoming packets are splits into header and payload, and then the header information is used to maintain connection set and the payload is used for signature matching. The main role of the connection set is to maintain the flow of the active connections.

A flow is a five tuple entry of source and destination ip, source and destination port, and the protocol. The flow chart of Priority Based Payload Signature Matching (PBPSM) is shown in figure 3.2.

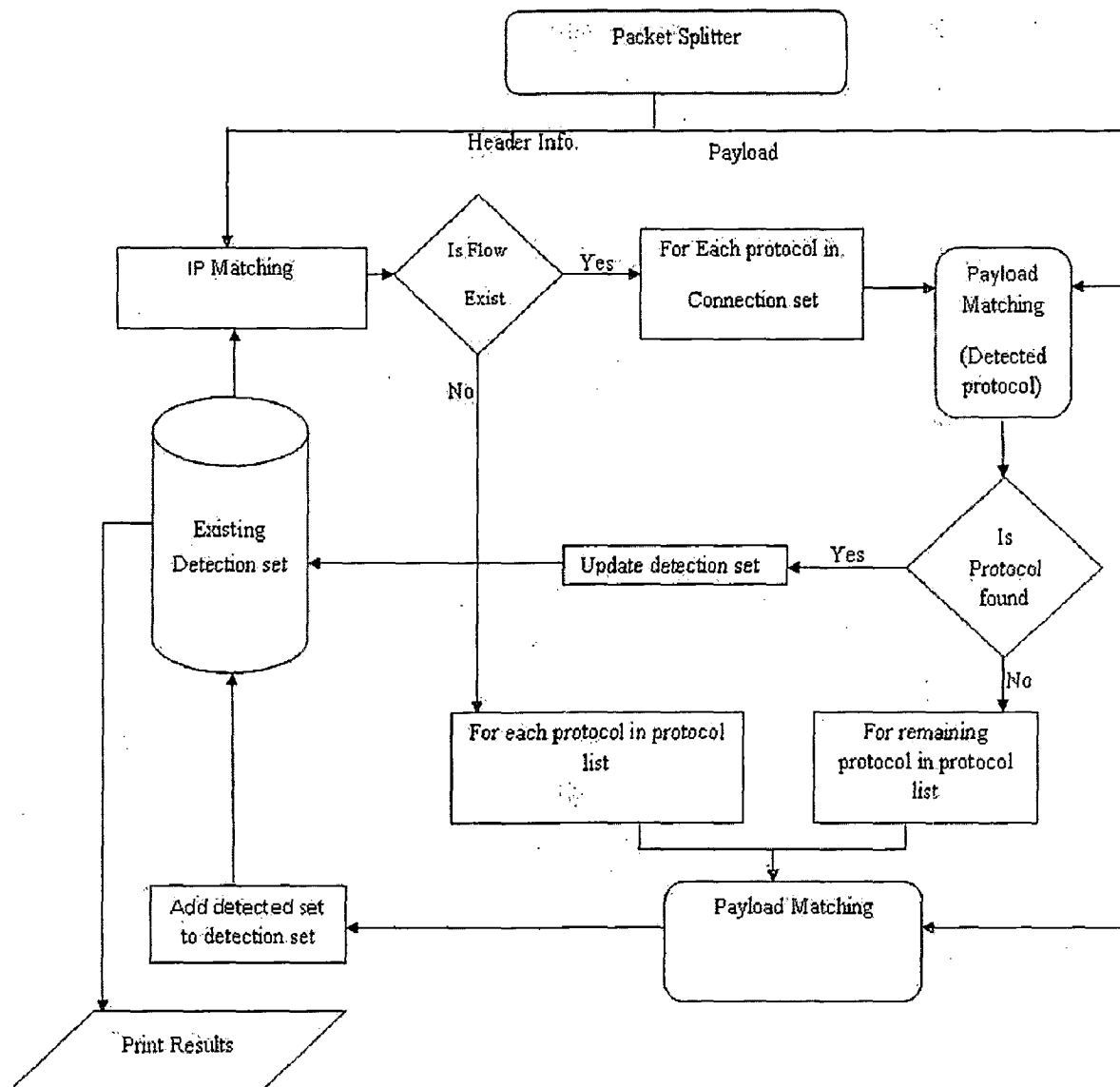


Figure 3.2 Flow Chart of PBPSM

The overall process takes place in following steps:

- 1) Capture incoming packet,

- 2) Splits packets into header and payload,
- 3) Extract header information for connection set,
- 4) Check if the connection set is present in the list, if not add the connection set into list,
- 5) For each connection set check active protocols already detected if so Search Priority list of detected protocol,
- 6) If protocol is not found in connection set then search complete protocol list and add protocol into connection set,
- 7) Log the packet into mysql for further processing.

The step by step process of the priority based payload signature matching algorithm is shown above. The process starts with capturing the network packet at the gateway interface. The capture packets are stored in form of *pcap* files. Then we process each *pcap* files for detection of protocols. In detection process packets are splits into header and payload. The header is processed for flow or connection set of each active communications during the time of surveillance. The connection list is first searched, if the connection set is not present then the newly detected connection set is added to the list. Now for the connection set, extract the protocols already detected, for each protocol detect search priority list and if protocol is found in this list, return the searched result. If protocol is not found in the priority list, search complete protocol list and add detected protocol into the priority list.

The simple concept behind the priority based payload signature matching mechanism is that generally two communicating machine uses at most 10 different application which uses internet. The priority list added an advantage, as it speed up the whole searching process as generally complete list is searched only once for each protocol in active connection set. If same application packets detect again then the result is returned from priority list which contains only 10 protocols, not the complete list of 100 protocols. This improves the running performance of the system.

3.3 Preprocessing of the Results

The results are pre processed and logged into the mysql. The payload based matching returns the complete packet and this packet as whole is not used for the fuzzy controller and hence the preprocessing is needed. The fuzzy controller calculates the bandwidth based on three input parameters: average inter-arrival time, number of protocol packets, total packet or packet ratio. To calculate these three parameters we need following information regarding the packets of the trace file.

- 1) Protocol: It is the protocol of the application which generates the current packets. There are around 100 plus protocols which are in use, it varies from simple HTTP protocol to Bittorrent or ICMP protocol. The protocol field is needed as the bandwidth consumption regarding particular protocol is needed.
- 2) Number of protocol packets: It is the total number of protocol packets present in the trace file. The number of protocol packets is needed as it gives an overview of the protocol participation in the total bandwidth consumption.
- 3) Total packets: It is the packet present in the trace file which includes packets from all protocols packets generated by the user of the internet during the surveillance period.
- 4) Average inter-arrival time: It is the average mean time between any two consecutive packets of the same protocol. The average inter-arrival time is calculated as shown below:

$$\text{average inter-arrival time} = \frac{(\text{Timestamp of last packet} - \text{Timestamp of first packet})}{\text{Total number of the packets}}$$

The value of average inter-arrival time gives the intensity by which application generates traffic. If the intensity is high it tries to flood the network hence it consumes high bandwidth.

After calculating the values of these parameters, they are logged into the log file. The preprocessing phase tunes the results of priority based payload matching mechanism and

they are further used by Fuzzy Controller.

3.4 The Fuzzy Controller

3.4.1 Fuzzy Logic

Fuzzy logic uses qualitative terms and linguistic labels to represent trust as a fuzzy concept, and membership function is used to describe at what degree an entity can be labeled as trustworthy or untrustworthy. Fuzzy logic provides rules for calculation with fuzzy measures of this type. In modeling trust, concepts such as trustworthy, honesty, and accuracy are defined and quantified. Since these linguistic labels are fuzzy, we can apply fuzzy logic to handle the uncertainty and the imprecision in any trust model.

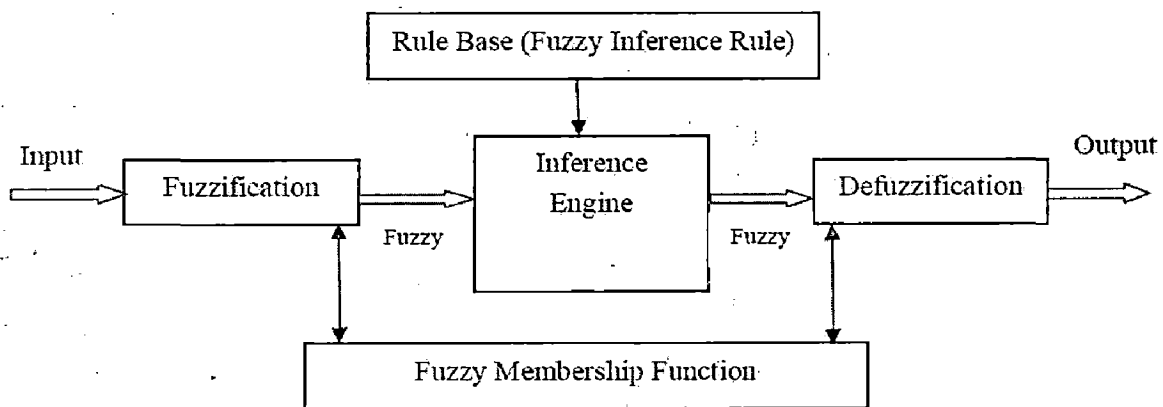


Figure 3.3: Fuzzy Inference Model

Fuzzy inference is the process of creating the mapping from a given input to an output using fuzzy logic. The mapping provides a basis from which decisions can be made, or patterns discriminate. The five parts of the fuzzy inference process are shown in Figure 3.3 [39] and described as follows:

(1) Fuzzification: It is the first step in the fuzzy inference process. During this the inputs are considered and the degree, to which they belong to each of the fuzzy sets via membership functions, is determined.

(2) Fuzzy membership function: It is a curve which defines the mapping of each point in the input space to a membership value (or degree of membership). Its value lies between

0 and 1. There are several membership functions such as Guass, Bell, Z, PI, Trapez, S-Polynome, Dreieck and so on.

(3) Fuzzy Inference Rule Base: It comprises several fuzzy rules which relate the input fuzzy set to output fuzzy set. In fuzzy approximate reasoning, there are mainly two important inference rules. First one is Generalized Modus Ponens (GMP) and other one is Generalized Modus Tollens (GMT).

Generalized Modus Pones (GMP) is defined as follows:

Implication: If X is A Then Y is B.

Premise: X is A. > Conclusion: Y is B.

(4) Fuzzy Inference Engine: Mamdani [40]-type fuzzy inference method is the frequently used fuzzy methodology. It was along with the first control systems built using fuzzy set theory. It supposes the output membership functions to be fuzzy sets. When the aggregation process is done, there is a fuzzy set corresponding to each output variable that is to be de-fuzzified. Mamdani is basically a Min-Min-Max fuzzy inference method.

(5) Defuzzification: The defuzzification process is a mapping of fuzzy set to a single number. The input to a de-fuzzifier is an aggregated fuzzy set and output is a single crisp value. The most common defuzzification method is the centroid calculation, which returns the center of area under the curve.

3.4.2 Fuzzy Controller Design

The fuzzy controller is used to generate bandwidth consumed by various protocols present in the trace file. The fuzzy controller is based on fuzzy logic suggested by L. A. Zadeh [35-37]. The fuzzy controller takes three input parameters and it returns the bandwidth consumption by various protocols in current trace file. The fuzzy controller works on fuzzy values and it generates the output values using the Mamdani Inference. The fuzzy rules are used to define intensity of each parameter. The intensity level can be any value out of following:

- 1) Low: The low term defines the impact that value of parameter is minimum. For

example, if the intensity level of protocol packet is low, then it uses minimum bandwidth and there is still space for other application.

- 2) Medium: The medium term defines the impact that value of parameter is moderate and hence it consumes almost equal or more resource than it is supposed to use. For example, the medium value for protocol packet means that it almost consumes 30-50 % of channel capacity.
- 3) High: The high term defines the impact that value of parameter is alarming. The application tries to exhaust the resource available and hence there is either no or few space for other application to expand. The high value of bandwidth consume means the application tries to exhausts the channel capacity with its high bandwidth consumption.

The Fuzzy Controller works as follow:

- 1) Fuzzification of input parameters: The process of translating input values to fuzzy truth values is called Fuzzification. It fuzzifies the input parameters so that they can be used in fuzzy inference process. In fuzzification, we define the range for the three parameters as low, medium, and high intensity level. Then the input supplied to these parameters is fuzzified based on the range of its participation. In Fuzzification the membership values is decided from range of 0.0-0.5-1.0 (here 0 equal to crisp 0 and 1 equal to crisp 1).
- 2) Rule Evaluation: This step incorporates the if-then clauses which were programmed into the controller. For our classification we have used the rule of the form:

if protocol packet is high and average inter-arrival time is low and total packet is low then bandwidth consumption is high.

This rule checks the fuzzy value of protocol packet and if it falls in high intensity range, it checks the fuzzy value of the average inter-arrival time and if it lies in the range of low intensity level, it checks the total packets and if it is low, then it returns the bandwidth consumption as high. This is due to the fact that if protocol

packet is more and the inter-arrival time between two packets is less and the total packets present in network is also less, so it contains most of the protocol packets which are generated in a quick succession.

3) Defuzzification: This transformation of a fuzzy set to a crisp value is called defuzzification. This is the final process of producing a quantifiable result in fuzzy logic. The values obtained after the Rule Evaluation process are handled here using given fuzzy sets and corresponding membership degrees. It is not a unique operation as different approaches are possible. The most important ones for control are described in the following [41, 42].

i) *Max membership principle*: Also known as the height method, this scheme is limited to peaked output functions. This method is given by the algebraic expression

$$\mu_c(z^*) \geq \mu_c(z) \text{ for all } z \in Z$$

where Z^* is the defuzzified value.

ii) *Centroid method or center of gravity*: This procedure (also called center of area) is the most prevalent and physically appealing of all the defuzzification methods [43] it is given by the algebraic expression

$$z^* = \frac{\int \mu_c(z) \cdot z dz}{\int \mu_c(z) dz}$$

where \int denotes an algebraic integration.

iii) *Weighted average method*: The weighted average method is the most frequently used in fuzzy applications since it is one of the more computationally efficient methods. Unfortunately it is usually restricted to symmetrical output membership functions. its algebraic expression.

$$z^* = \frac{\sum \mu_{C_i}(\bar{z}) \cdot \bar{z}}{\sum \mu_{C_i}(\bar{z})}$$

where \sum denotes the algebraic sum.

iv) *Center of sums*: This is faster than many defuzzification methods that are presently in use, and the method is not restricted to symmetric membership functions. It is represented by following equation is given by the following equation:

$$z^* = \frac{\int z \sum_{k=1}^n \mu_{C_k}(z) dz}{\int \sum_{k=1}^n \mu_{C_k}(z) dz}$$

where the symbol \bar{z} is the distance to the centroid of each of the respective membership functions.

v) *Center of largest area*: If the output fuzzy set has at least two convex subregions, then the center of gravity.

$$z^* = \frac{\int \mu_{C_m}(z) \cdot z dz}{\int \mu_{C_m}(z) dz}$$

where μ_{C_m} is the convex subregion that has the largest area making up C_k .

vi) *Mean max membership*: This method (also called middle-of-maxima) is closely related to the first method, except that the locations of the maximum membership can be nonunique (i.e., the maximum membership can be a plateau rather than a single point). This method is given by the expression:

$$z^* = \frac{a+b}{2}$$

Chapter 4

Detailed Design and Implementation

4.1 Priority Based Bandwidth Management

Design of this system includes three major modules: packet splitter, connection set, and payload matching. Packet splitter is used to extract header and payload information,

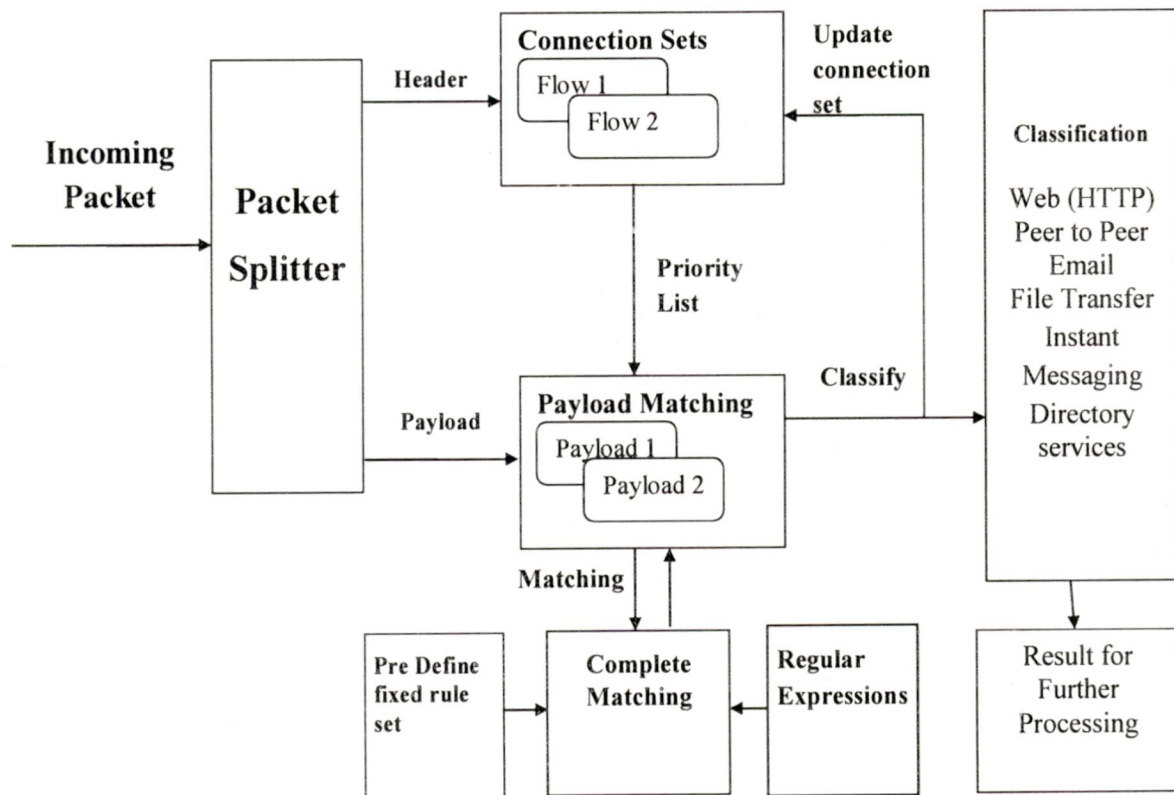


Figure 4.1 System Design for Priority Based Packet Payload Matching

Connection set lists the active connections with used protocols, and Payload based signature matching contains following phases:

- i. *Field Extractor*: The nature of all the network traffics stored in the log files are detected based on header and payload information. Packet is divided into Header and payload. The tools required are TCPDump, Snort, and Wireshark etc.

- ii. *Connection set/ Flow*: The header information like the source IP address (srcIP), destination IP address (dstIP), source port (srcPrt), destination port (dstPrt), and protocol fields (ToP) are matched at this point and then they are searched in connection set. If the protocol is not found within the connection set then it added to connection list for next time. Then the current connections set along with already detected protocols are passed to the payload matching phase.
- iii. *Payload Analysis*: This phase analyses the traffic based on packet content i.e. it is examined against predefine signatures, extract the information and if signature is matched then traffic is classified accordingly. The payload analysis takes place only after it gets result from the connection set phase. The connection set phase provides all active protocol, if the connection exists in the connection list then the payload phase first searches the priority list and if protocol is found there then it returns back else it searches the exhaustive list and once it found the desired protocol it update the connection list by adding the newly detected protocol and then returns the result. In the matching process the payload of packets are matched against standard signature[44] of protocols.
- iv. *Complete Matching*: Once regular expression and some fixed fields returns the result of its matching portion then the complete matching classifies the packet. For example, Bittorrent protocol contains '0X13' followed by keyword "bittorrent protocol" just in the starting of user payload, is matched by using pre-define fixed rule. This message start new connection set or flow between any two machines.
- v. *Regular Expression*: In this the signature are stored as regular PCRE expressions. The regular expressions are used where there are repetitions of string in the signature and the position of the signature is not fixed i.e. signature can exist at beginning, end, or middle of the payload data. Signature can be present in two parts, in that case we need to skip few characters to match the packet. The internal layered structure of a packet in Wireshark[45] is shown in the figure 4.2. The values are in HEX format and payload is the last portion of the packet. To search any string in payload the headers are skipped and then only searching is done.

- vi. *Fixed Signature*: Some signature is stored as fixed string like header information. The header information has a fixed format, so it is easy to extract information from the header. The internal layered structure of a packet in Wireshark[45] is shown in the figure 4.2. The values are in HEX format and header is at the start of the packet so it is easy to traverse header fields.
- vii. *Classification/Categorization*: Finally traffic is categorized in proper categories in various internet traffic based on application which generates the packet. The payload protocol along with packets is logged into the database for further analysis.
- viii. *Update the connection set*: Update the flow by adding newly detected protocol to flow/connection set priority list. So that if next time same protocol encounters then it can be searched within priority list. The connection set provides the priority to the system.

The Packet splitter used to split the packets into header and payload information. It is used to extract desired header field in order to maintain connection list. The payload contains the data of user applications. It is searched in order to find which application generates the data. The connection set is used to maintain the priority of active protocol between any communications. Connection set contains a list of all active flows between any two communicating machines. The flow can be defined using following five tuples:

- i. *Source IP address*: The IP address of machine which generates the packets. Packet may be for the request of a particular service/data or it may be a reply packet.
- ii. *Destination IP address*: The IP address of target machine for that packets. Packet may be for the request of a particular service/data or it may be a reply packet.
- iii. *Source port number*: The port number of machine which generates the packet.
- iv. *Destination port number*: The port number of target machine for that packet.

- v. Internet protocol: The internet protocol i.e. UDP or TCP are used to narrow the search for payload.

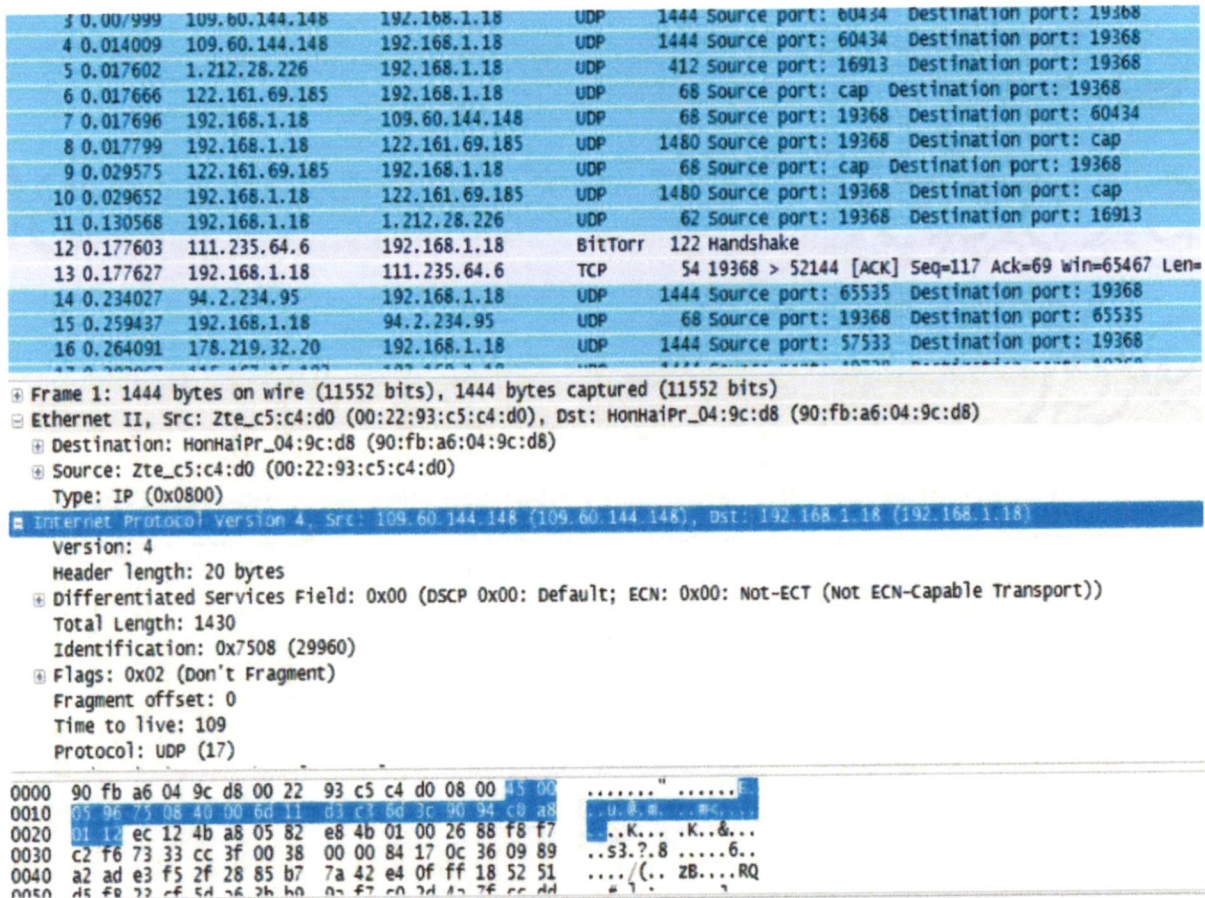


Figure 4.2 Internal Structures of the Packets

4.1.1 The Priority Based Bandwidth Management Algorithm (PB2MA)

The priority based bandwidth management algorithm is designed to implement the protocol search technique more efficiently. It introduces the concept of Priority by maintaining the list of already detected protocol and connection sets. This list helps to narrow the overall search process and hence it improves the execution time efficiently. Figure 4.3 shows the flow of priority based protocol matching and bandwidth utilization by protocol algorithm.

As shown in figure 4.3 we maintain three lists for the protocol matching:

- i. *list.protocol*: It contains the complete exhaustive list protocols for which signatures are available and we can use them for the matching. The *list.protocol* is

```

'list.protocol': {Complete Protocol list}
'flow.protocol': {Protocol already detected in that flow}
'flow.remaining.protocol': {{ flow.protocol}-{ list.protocol}}
'Interarrival_time': Time between two successive packet with same
protocol
'flow.pi_count': count of specific protocol in each set
'Packet.total_count': Total packet in that pcap file
foreach packet P in Pcap file or Captured do
  foreach protocol pi in flow.protocol do
    if Pi.protocol = flow.protocoli then
      Increase flow.pi_count;
      Increase Packet.total_count;
      write Interarrival_time[flow.Protocol_count]=
        packet.timestamp-Old_timestamp;
      write Old_timestamp=packet.timestamp;
      write total_Interarrival_time + =
        Interarrival_time[flow.Protocol_count];
    end
  end
  foreach protocol pi in flow.remaining.protocol do
    if Pi.Protocol = flow.remaining.protocoli then
      write Pi.protocol into flow.protocol;
      Increase flow.pi_count;
      Increase Packet.total_count;
      write Interarrival_time[flow.Protocol_count]=
        packet.timestamp-Old_timestamp;
      write Old_timestamp=packet.timestamp;
      write total_Interarrival_time + =
        Interarrival_time[Protocol_count];
    end
  end
  foreach protocol pi in protocol list do
    if Pi.Protocol = list.protocoli then
      write Pi.protocol into flow.protocol;
      Increase flow.pi_count;
      Increase Packet.total_count;
      write Interarrival_time[flow.Protocol_count]=
        packet.timestamp-Old_timestamp;
      write Old_timestamp=packet.timestamp;
      write total_Interarrival_time + =
        Interarrival_time[flow.Protocol_count];
    end
  end
  end
  foreach protocol pi in flow.protocol do
    Avg_Interarrival_time ← total_Interarrival_time ÷ flow.pi_count;
    packeti_ratio ← flow.pi_count ÷ Packet.total_count;
    BW_consume=
    Fuzzy_lib{total_Interarrival_time, flow.pi_count, packeti_ratio};
    if BW_consume ≥ Threshold_limit then
      | write Dropping packets for pi;
    end
  end
end

```

Figure 4.3 Priority Based Bandwidth Management Algorithm (PB2MA)

This list is only updated when any new protocol is added to the database. If any protocol found here then it added to the *flow.protocol* list.

- ii. *flow.protocol*: It is the complete list of the protocols which are detected for the particular connection set. This list is most frequently searched list. If there is any active connection set and it is using 10 protocols then all these 10 protocol are listed here for that connection set. The *flow.list* helps to improve the search time as most of the time protocols are found here only.
- iii. *flow.remaining*: It is the remaining list of protocols that are not present in the *flow.protocol* list and it is needed to be searched only if the protocol is not found in the *flow.protocol* list. Once the protocol is found here, it added to the *flow.list* for next time search and is removed from this list. It contains only those protocols which are available in database for the active connection set, but still not added to the *flow.protocol* list.

$$\text{flow.remaining list} = \text{list.protocol list} - \text{flow.protocol list}$$

Other parameters of the algorithms are:

- i. *Inter-arrival time*: It is the mean time between two successive packets of same protocol. This value shows the rate of generation of the packets by an application. The higher value of inter-arrival time means the application is less active and generates less number of packets with its protocol.
- ii. *Flow pi_count or protocol packet count*: It is the number of protocol packets present in 'pcap' file. The higher value of protocol packets means the application is generating more packets which may flood the overall network hence increase the congestion.
- iii. *Total packet count*: It is the number of packet present in the trace file. The total packets passing through the interface during the surveillance time.

The packets are processed in such way that first the *flow.list* is searched, if connection set is found there then the all detected protocol are extracted. These protocols are passed for payload matching. The signatures of all detected protocol are matched against packet payload and if it found into the packet payload then it updates the values and returns. If the connection set is present but the protocol is not in the *flow.list* then for remaining protocol the *flow.remaining* list is searched because we already searched few protocol in *flow.list* and hence only we need to search remaining protocols.

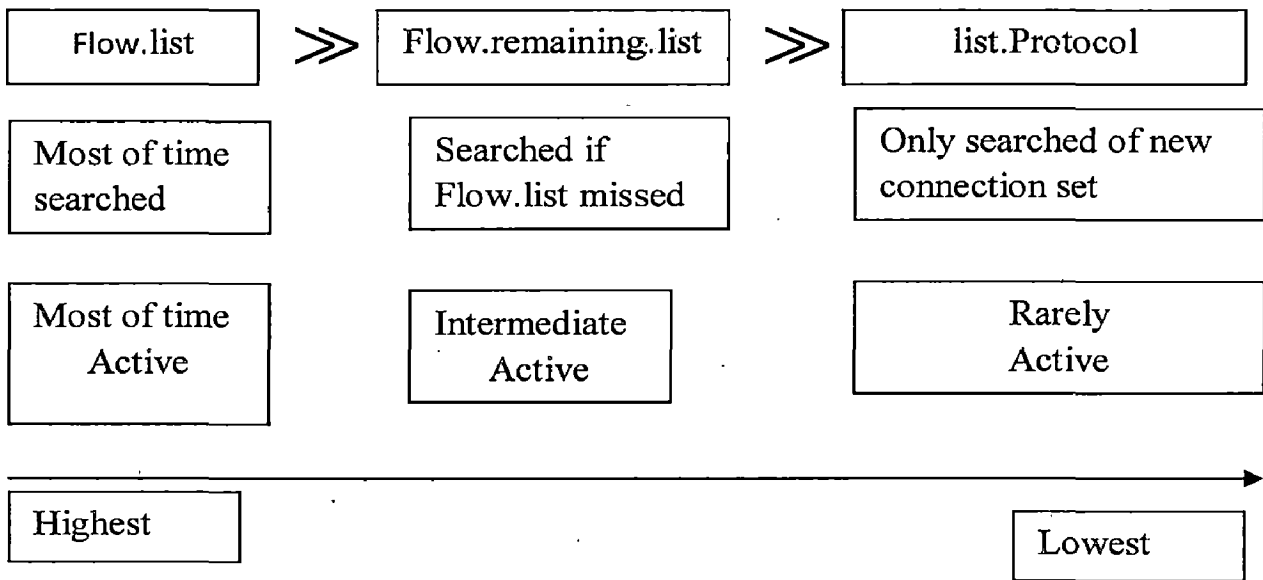


Figure 4.4 Order of List Searched in PB2MA

If the connection set is not present in the *flow.list* then the exhaustive list of *list.protocol* is searched and the protocol is added to the *flow.list* if it found here. If the protocol is not found here, it is marked as unknown, as the signature of this protocol is not present in our database. The time searching time of this list is high, but only newly added connection sets are searched in this list.

The Active execution time and impact of lists are shown in figure 4.4 as most of the time only *flow.list* is active and it contains only 5- 10 protocol, so it lowers the overall execution time of the algorithm. The *list.protocol* is only searched if any new connection set is setup so it is rarely searched, as a result although it take highest time but due to less active its contribution in overall execution time is very less.

4.2 Preprocessing and Tuning of Initial Results

The role of preprocessing phase is to generate input for the fuzzy controller. It processes the output of the priority based signature matching algorithm and extract only relevant field for the fuzzy controller. The logged results of payload matching phase are processed:

1. To keep only active protocols and remove all other protocols which are inactive as they do not contribute in current trace cycle. This step helps to remove unwanted protocols and it lower the execution time of fuzzy controller.
2. To calculate the average mean time between protocols packets from trace file. The mean time is used by fuzzy controller. Average mean time between packets are calculated as:

$$\text{Average mean time} = \frac{(\text{Timestamp of last packet} - \text{timestamp of first packet})}{\text{total number of packet}}$$

3. Total number of packets: It is the number of total packets in 'pcap' file i.e. packet passed through the interface.
4. Protocol ratio: As the total packet which is output of payload based approach is same for all protocol so it is not used in fuzzy controller but protocol ratio is used. It is calculated as follows:

$$\text{Protocol ratio} = \frac{\text{protocol packet}}{\text{total packets}}$$

The result of this phase is again logged into mysql so that it can be used by fuzzy controller. It is just intermediary phase between the fuzzy controller and payload signature matching.

4.3 Implementation of Fuzzy Controller

4.3.1 The Fuzzy Controller Parameters

The Fuzzy controller is based on fuzzy logic. The fuzzy controller works on three input parameters:

1. Average inter-arrival time (ART) of different packet is the average of mean time between two consecutive packets arriving at interface during surveillance time.

$$\text{ART} = \text{Protocol packets} / \text{time between first and last packets}$$

2. Number of Protocol packets (NPP) is the number of packet of a specific protocol detected at any interface during surveillance time.

3. Protocol packet ratio (PPR) is the ratio of number of protocol packets to the total packets arriving at any interface during the surveillance time.

$$\text{PPR} = \text{Protocol packets} / \text{total number of packets}$$

There is only one output of the fuzzy controller which is the percentage bandwidth consumption by each protocol. The percentage bandwidth consumption gives the channel capacity utilized by the protocol as compared to all other protocol active during the surveillance time. The Bandwidth consumption is calculated by processing the input value of ART, NPP, PPR using mamdani[40] inference. There are twenty seven rules set which fired on the input value of ART, PPR, and NPP. These rule set are simple if-then-else type of rules written in simple English language. We get real value of bandwidth consumption only after defuzzification of the output of inference engine. For the process of defuzzification we have used center of gravity defuzzification method.

The algorithm of fuzzy controller implementation is shown in figure 4.5. It has basic three phases:

- i. Fuzzify the input parameter: The values of input parameter are fuzzified in low medium and high.

- ii. Inference and Rule processing: The input parameters are processed by mamdani fuzzy inference[40] engine. Then the output is also a fuzzy value.
- iii. Defuzzification: The value of bandwidth at output are defuzzified using Center of Gravity[42] method, as it is one of the most accurate method.

```

'Initialization Average inter arrival time': {Initialize the low, mid,
high values of Average inter arrival time}
'Initialization protocol packets': {Initialize the low, mid, high values of
protocol packets} 'Initialization protocol packet ratio': {Initialize the
low, mid, high values of protocol packet ratio}
'Disign the Fuzzy Rulese using MATLAB': {Create rule data set}
foreach protocol set  $P_i$  in Pcap file or Captured do
    'FUZZIFY the Average inter arrival time': { Convert from crisp
to fuzzy value} 'FUZZIFY the protocol packets': { Convert from
crisp to fuzzy value} 'FUZZIFY the packet ratio': { Convert from
crisp to fuzzy value}
end
foreach protocol protocol set  $P_i$  in Pcap file or Captured do
    'APPLY Rules on Average inter arrival time': { To calculate the
fuzzy value} 'APPLY Rules on protocol packets': { To calculate the
fuzzy value} 'APPLY Rules on packet ratio': { To calculate the
fuzzy value}
end
foreach protocol protocol set  $P_i$  in Pcap file or Captured do
    'DEFUZZIFY the Average inter arrival time': { Convert from
crisp to fuzzy value} 'DEFUZZIFY the protocol packets': {
Convert from crisp to fuzzy value} 'DEFUZZIFY the packet ratio':
{ Convert from crisp to fuzzy value} write
    Return Bandwidth Consumption by that protocol;
end

```

Figure 4.5 Algorithm for Fuzzy Controller

The Initial threshold values of the parameter are calculated by performing several experiments in Information Security lab, Department of computer and Electronics, Indian Institute of Technology, Roorkee. The Initial values for three intensity levels are shown in table 4.1. The total bandwidth consumptions depends on the initial values of Average

inter-arrival time (ART), Protocol packet ratio (PPR), and number of protocol packets (NPP).

Table 4.1 Threshold Values of ART, NPP and PPR

Parameters/level	low (Normal)	medium (Medium/Moderate)	high (Alarming)
ART (sec)	36.25	22.5	13.5
NPP (number)	1260	5500	11250
PPR (number)	0.075	0.2	0.35

When the ART value is "low" we used a large number, this means that the time between two received packets is large, therefore the level of bandwidth consumption is normal. Similarly, when the ART value is "high" we used a small number which means that there exists a high probability of having high bandwidth consumption because the time between two received packets is small. Values in the "medium" column mean that the time between received packets is medium, so level of high bandwidth consumption is moderate. Looking at NPP parameters, the level of bandwidth consumption reflects the number of sent packets by source and the number of received packets by destination and PPR is the packet ratio of protocol packet to total packet. Larger the value of NPP and PPR, higher is the bandwidth consumptions by the protocols. These three parameters are extracted from Priority based algorithm and after preprocessing, fed to the fuzzy logic controller. The controller then combines them in an intelligent way and produces a single number indicating the bandwidth consumption. In the next section, we discuss the fuzzy logic controller.

4.3.1 Design of Fuzzy Logic Controller

For our implementation we use fuzzy controller designed on jFuzzyLogic: Open Source Fuzzy Logic (Java) [46].The fuzzy logic controller is designed using java Fuzzy library and which is composed of membership functions (for each input/output variable) and

fuzzy rules. In this section we discuss these two components. The values of the parameters were used to tune the fuzzy logic membership functions and to create the fuzzy logic rules. Three input parameters are used (ART, NPP, and PPR). For each input parameter, three trapezoidal membership functions were designed: Low, Med, and High. The output parameter also has three trapezoidal membership functions distributed in the range [0.0, 1.0].

4.3.1.1 Rule sets

The rule set contains following 27 rules based on the test experiments.

- [1]. if art is low and npp is high and ppr is high then attack is high;
- [2]. if art is low and npp is high and ppr is medium then attack is high;
- ...
- ...
- ...
- [26]. if art is medium and npp is high and ppr is high then attack is high;
- [27]. if art is high and npp is low and ppr is low then attack is low;

3.4.2 Boundary values

Once rules are defined, we define the boundary for low, medium and high for our fuzzy class

Avg_intarrival_time

TERM low := (5, 1) (15, 1) (22.5, 0);
TERM medium := (15, 0) (20,1) (25,1) (30,0);
TERM high := (22.5, 0) (30, 1) (50, 1);

Protocol_packet

TERM low := (50, 1) (1000, 1) (2500,0) ;
TERM medium := (1000,0) (2500, 1) (7500, 1) (10000, 0);

TERM high := (7500, 0) (10000, 1) (15000, 1);

Packet_ratio

TERM low := (0, 1) (0.1, 1) (0.3, 1) (0.5, 0);

TERM medium := (0.3, 0) (0.4, 1) (0.6, 1) (0.7, 0);

TERM high := (0.5, 0) (0.7, 1) (1, 1);

Then output is defuzzified to get real values

BW_Consume

TERM low := (0, 1) (20, 1) (40, 0);

TERM medium := (20, 0) (40, 1) (50, 1) (65, 0);

TERM high := (55, 0) (65, 1) (100, 1);

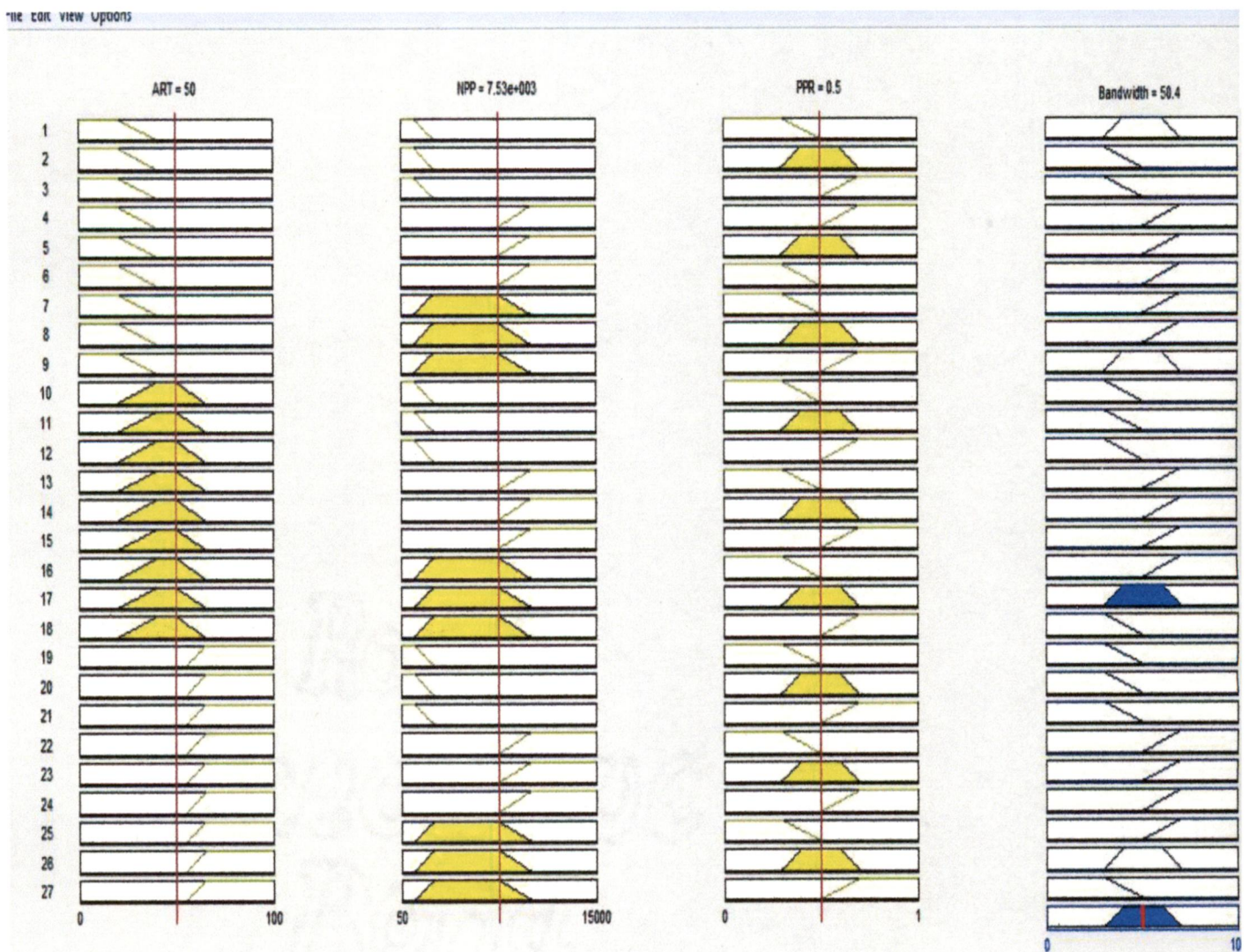


Figure 4.6 Fuzzy Rules and Inference

Chapter 5

Experimental Results and Discussions

5.1 Packet Inspection Mechanisms

This section shows results of port based discussed in introduction and literature review and payload based matching mechanisms for Bittorrent protocol. First we look at the signature used by both techniques and then test these signatures in our test bed.

5.1.1 Bittorrent Protocol Signatures

The detection system is developed in Ubuntu 9.10. The protocol signature of Bittorrent protocol is shown in table 5.1.

Table 5.1. Signature Pattern of Bittorrent Protocol Used for Matching

S.No	Signature	Used for	Short coming
1	Port no. 2710, 6881–6887, 6888, 6889–6890, 6891–6900, 6901, 6902–6968, 6969, 6970–6999, 7000, 30301 [47]	Port Based Matching	These Ports are used by other protocol also and they are not assign by (IANA) [29]
2	First 20 bytes of payload contains 19(0x13) and string "Bittorrent protocol" /* test for match 0x13+"Bittorrent protocol" */ if (packet->payload_packet_len > 20) { check if (packet->payload[0] == 0x13) { //Check whether <i>Bittorrent protocol</i> key word present or not if (memcmp(&packet->payload[1], "Bittorrent protocol", 19) == 0) Return <i>Bittorrent protocol</i> ; } } [44]	Payload based Matching	As the payload is matched against the The signature hence the chance of mismatching is very less.

5.1.2 Packet Detection Module

In the port number based approach, well known port numbers of applications assigned by the Internet Assigned Numbers Authority (IANA) [29] are matched against the port numbers in packet data. As no official port numbers are assigned to Bittorrent protocol by the IANA [29] and generally some well known port numbers are used by Bittorrent. For test purpose, we take complete pool of frequently used port numbers available at the Wikipedia [47]. These port numbers are listed in table 5.1. The frequently used Bittorrent client “µTorrent [48]” is not limited to these port numbers and we can use any dynamic port number i.e. 1024-65535 or 216 so that screening process can be easily bypassed.

The complete results of matching against various trace file is shown in Table 5.2.

Table 5.2. Number of Packets Identified by Port Based versus Proposed Technique

S. No.	Packets in trace file (No.)	Bittorrent's packet (No.)	detected by our approach		port based technique		
			Packet (No.)	% packets identified	Packets (No.)	% packets identified	
1.	188472	13930	13020	93.46	258	1.85	<i>Packets correctly identified</i>
2.	182720	8068	7345	94.88	136	1.68	
3.	142792	7080	6566	92.74	196	2.76	
4.	89245	13460	13015	96.69	140	1.04	
5.	53547	6808	6566	96.44	56	0.82	
6.	27155	10346	10005	96.70	24	0.23	
7.	22563	4863	4637	95.35	790	16.24	
8.	17849	5009	4877	97.36	28	0.55	
9.	16838	8061	7923	98.28	412	5.11	
10.	13455	5209	5054	97.02	120	2.30	
11.	13132	6535	6368	97.44	189	0.29	
12.	4147	0	0	0	36	0.86	<i>Packets wrongly identified</i>
13.	2041	0	0	0	18	0.88	
14.	39944	0	0	0	414	1.03	

In payload matching, the signatures [44] are matched against the payload. Each Bittorrent client packet's payload contains these patterns and hence, these patterns can be matched against our signatures. As there is no way to bypass the matching process because each client must send these signal to the other client in order to communicate.

As shown by the table 5.2, the packets identified by our approach and the port based matching approach both. The packets correctly identified as Bittorrent's packets by our approach are much more than that of port based identification method. The approach used in port based technique is the matching of most frequently used port numbers. We try to cover exhaustive list of ports but still the result shows that by this technique we are only able to detect 10 % packets as compare to our approach or around 5% of total packets present in the trace. The main problem with port based approach is that in Bittorrent's client μ Torrent [48] we can change port numbers randomly and hence, we can use any port numbers from the list ranging from 1024 to 65535 . The payload based signature matching approach is much better as each client or server is needed to send the message in order to setup communicate with the server or the other client. Without this message, the server or the other client won't be able to recognize the packet and may discard the complete packet. If we match the payload of each packet against our signature, there is much better chance to detect the protocol accurately. This is due to the fact that approach used in our matching process is better than port based approach as it target the most essential port of Bittorrent communication. There is one more problem with port based matching technique is that the packet can be wrongly identified as Bittorrent packet. If we use port based approach for packet dropping then we might end up with dropping some wrong packet also because some other application can also falls in range of bittorrent's protocol ports range. Example of these are 6891–6900 (Windows Live Messenger (File transfer)), 6901 (Windows Live Messenger (Voice)), 6969 (acmsoda), 6697 (IRC SSL (Secure IRC)), 6699 (WinMX) [47].

5.1.3 Validation of Payload Signature Based Detection Module

The Testing and validation of payload signature based detection module is done on packets captured using the test bed setup at information security lab, Department of computer and Electronics Engineering department, Indian Institute of Technology,

Roorkee, India. These packets are captured on a local workstation. Then the variety of Internet applications were launched on the workstation and their trace are recorded. In particular, the trace traffic was generated by running P2P applications like BitTorrent, while also running an email client and accessing several Web sites over internet. The packets in the trace file were captured using Wireshark [3]. The capturing process is running in intervals which start with .5 minute and increment by 2-5 minute every time. There is random interval breaks between capturing in order to introduce the randomness in packet captured. We also try to break continuity or flow of packets of specific application especially bittorrent's in our captured trace for better assessment of payload based methodology or Deep packet inspection methodology.

Then these captures trace are examined offline using Wireshark to detect the flow of Bittorrent communication. The client and server sends "0x13" followed by "Bittorent protocol" in handshake packet of their communication. Also they exchange same signatures if any error occurs. We examine the packet to detect the flow of each communication. A flow is combination of source and destination IP address, source and destination port, and IP protocol. The flow can uniquely identify the communications between the Bittorrent client and server or clients. Then trace file is converted in text file using netsniff-ng as shown in figure 5.1.

```
netsniff-ng -i test1.pcap -X > test1.txt  
# searching of flows in test1.txt  
  
grep -c 'flow1' test1.txt //flow1 in HEX  
grep -c 'flow2' test1.txt //flow2 in HEX  
  
-i <pcap-file>: will be read in and printed to the console in order to perform an  
offline analysis.  
- X: Shows not only the payload in hexadecimal format, but the whole  
packet
```

Figure 5.1 Netsniff-ng Tools

Then this text file is searched for number of occurrence of each flow and is logged in a log file named as num. Finally packets for all BitTorrent flows are counted using shell script as shown figure 5.2.

```
SUM=0
while read NUM
do
    echo "SUM: $SUM";
    SUM=$((SUM + NUM));
    echo "+ $NUM: $SUM";
done < num
```

Figure 5.2 Shell Script for Counting Number of Packets

The table 5.2 provides summary information of the result. The signature method is able to classify around 95% of Bittorrent packets correctly, it missed only few packets or flows which had performed the handshaking before starting the capturing of particular trace. The signature method did not classify any non-P2P flow as P2P. Thus, the signature based or payload based method was quite accurate.

5.2 The Priority Based Bandwidth Management Algorithm (PB2MA)

The priority based bandwidth consumption algorithm depends on simple logic that between the two communicating machine there are only around 10 protocols which are frequently used. In the packet matching process we only compare the priority list before comparing the exhaustive list of protocol, once a packet is detected in the priority list then this packet is not searched in the exhaustive list.

The packet which is not identified by this priority list is searched against the complete exhaustive list and once we detect the protocol in the packet, it is added to the priority list for a connection set. If next time the same protocol is present in any of packets, it is first matched with the priority list.

The table 5.3 shows the need of priority based approach as only 5 – 10 application protocol is used by any machine to communicate with other machine in the network. It is also clear from table 5.3, that protocol used is not at all depends on the number of packets. If the number of packets increases rapidly, until and unless the user does not initiate any new application that uses the network for communication, the number of protocols is only limited to previously detected protocols.

Table 5.3 Number of Protocol Detected in Trace File

S. No.	No. of Packets in trace file	No. of Protocol found	Name of found protocols
1.	9031	3 valid protocols	a) Bittorrent b) NETBIOS c) ICMP
2.	9168	4 valid protocols	a) Bittorrent b) ICMP c) HTTP d) DNS
3.	9550	2 valid protocols	a) Bittorrent b) ICMP
4.	13728	6 valid protocols	a) Bittorrent b) NETBIOS c) ICMP d) HTTP e) DNS f) SSDP
5.	19029	4 valid protocols	a) Bittorrent b) ICMP c) HTTP d) DNS
6.	20403	4 valid protocols	a) Bittorrent b) ICMP c) HTTP d) DNS
7.	26471	6 valid protocols	a) Bittorrent b) NETBIOS c) ICMP d) HTTP e) DNS

In case where the application like Bittorrent client μ Torrent [48] which may generate huge network traffic and if the detection process is able to find out the protocol like

Bittorrent at first place without searching the complete list, the running time improves remarkably.

Figure 5.3 shows the running time of both priority based bandwidth consumption algorithm and non priority based bandwidth consumption algorithm. As the graph shows that the running time of the non priority based bandwidth is linear while the priority based algorithm steady with increase the number of packets, this is due to the fact that only 5-10 protocols are used at any instant. This is the underlying assumption of our priority based bandwidth consumption algorithm and this graph proves it.

The non priority based algorithms suffers as the number of packets increase because every time it searches the complete protocol list. The performance of non priority based

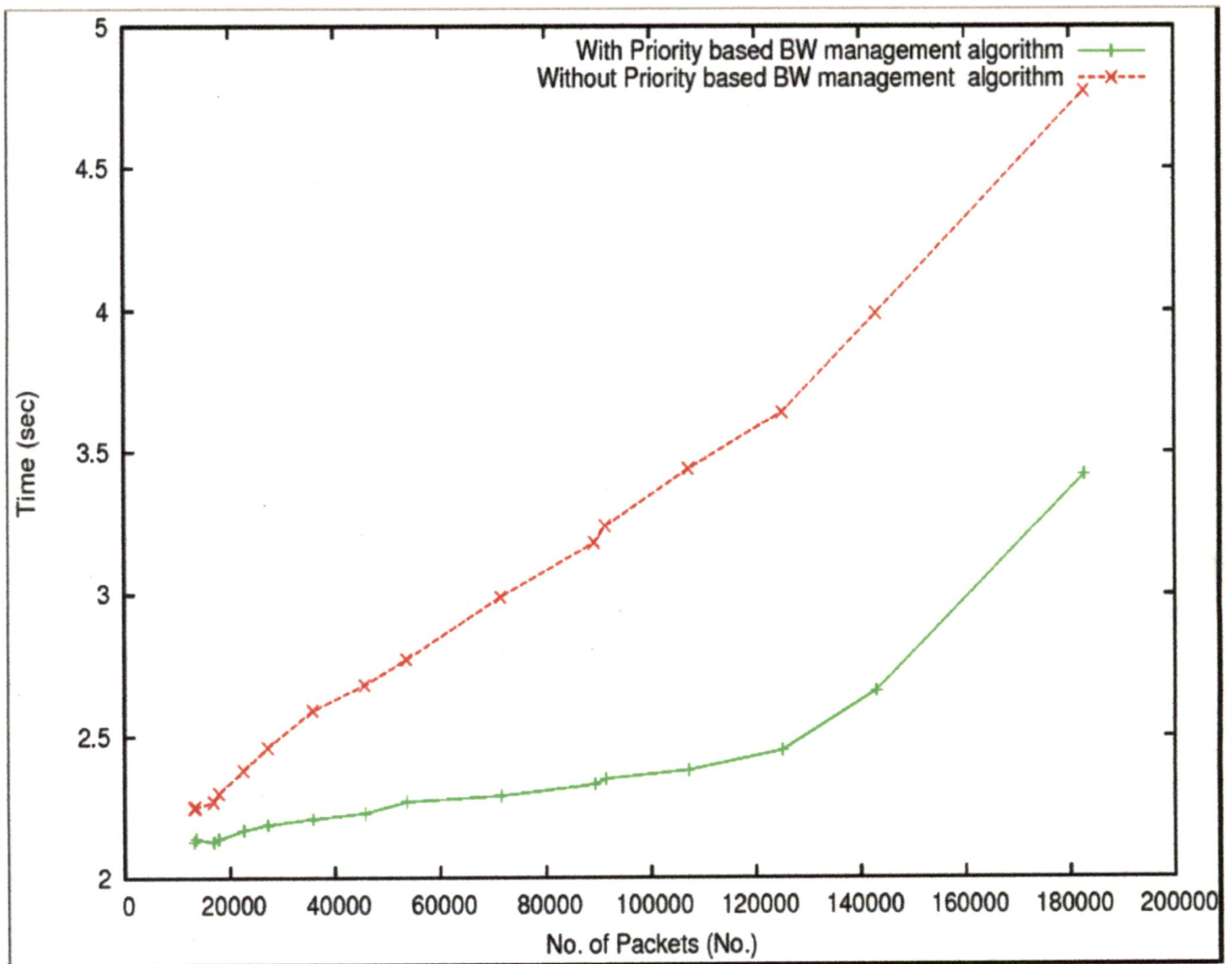


Figure 5.3 Running Time of Priority Versus Non Priority Based BWM Algorithm

algorithm is degraded many a fold if the target protocol is present at the bottom of the list hence it needs to go through the complete list in order to detect the protocol in payload. But our priority based algorithm won't suffer from any such issues, as it already maintains a list of previously detected protocol. Even if target protocol is present at the bottom of the list then only it needs to search once, and it is added to the priority list. Next time if same protocol is encountered, it searches the priority list first and returns the result. Hence there is no need to search the complete list for same protocol again and thus improves the performance. The complete list is searched only if the protocol is not found in the priority list.

The priority based bandwidth approach is combination of both the flow based approach and payload based approach, which is equivalent to the non priority based approach. Thus it give better result in term of performance as compared to other payload based approaches [32, 34, 49] which is mostly non priority based and limited to a single field i.e. either P2P or Live tv etc. But our approach can be expanded further and we can use it with other application as well. The overall frame work is always same for any application as long as we design the matching process of those application signatures.

5.3 Fuzzy Controller

5.3.1 Bandwidth Consumption by Various Protocols

The overall role of fuzzy controller is that it gives the projected percentile bandwidth consumption by various protocols in the network.

The fuzzy logic controller is placed at the end of our system and hence it uses the values given by the flow and payload matching process. The fuzzy controller uses the three input parameters and in result it returns the bandwidth consumed by all protocol during the period of surveillance. The three inputs to the fuzzy controller are:

- i) *Average interarrival time (ART)*: ART of different packet which is the average of mean time between two consecutive packets at interface during surveillance time.

$$\text{ART} = \text{Protocol packets} / \text{time between first and last packets}$$

- ii) *Number of Protocol packets (NPP)*: The number of packets of a specific protocol detected at any interface during surveillance time.
- iii) *Protocol packet ratio (PPR)*: the ratio of number of protocol packets to the total packets arriving at any interface during the surveillance time.

$$PPR = \text{Protocol packets} / \text{total number of packets}$$

The value of these three parameters is calculated using the output of the previous detection done using flow and payload based technique.

These values are then fuzzified and tested against the complete fuzzy rule sets. To do so we designed twenty rules using matlab [50]. Then the fuzzy controller is tuned using test data. The tuning is done to make system aware about low bandwidth consumption, mid bandwidth consumption, and high bandwidth consumption. The low bandwidth consumption means the application utilize minimum channel capacity during the time of active communication. The medium bandwidth consumption means the application uses moderate channel capacity during the time of active communication. The high bandwidth consumption means application exhausted the channel capacity during the time of active communication. Finally the defuzzification take place and we get a real output bandwidth consumption values.

The defuzzification is done by using the center of gravity or center of area defuzzification method. This is the very accurate and most prevalent of all the defuzzification methods [43].

The collected trace then first classifies using the priority based bandwidth consumption algorithm. These values are log into the log file and stored into mysql. The log data pre processed in order to find out the total number of packet for time of surveillance, the number of packet of the particular protocol arrived at interface during the surveillance time and the average interarrival time of the packet of the respective protocols. These values are used to find out protocols packets, timestamp of each packet, and packet ratio i.e. ratio of packet to total packet in pre processing phase. After pre processing the values of average inter arrival time, the packet ratio i.e. ratio of protocol are supplied as input for

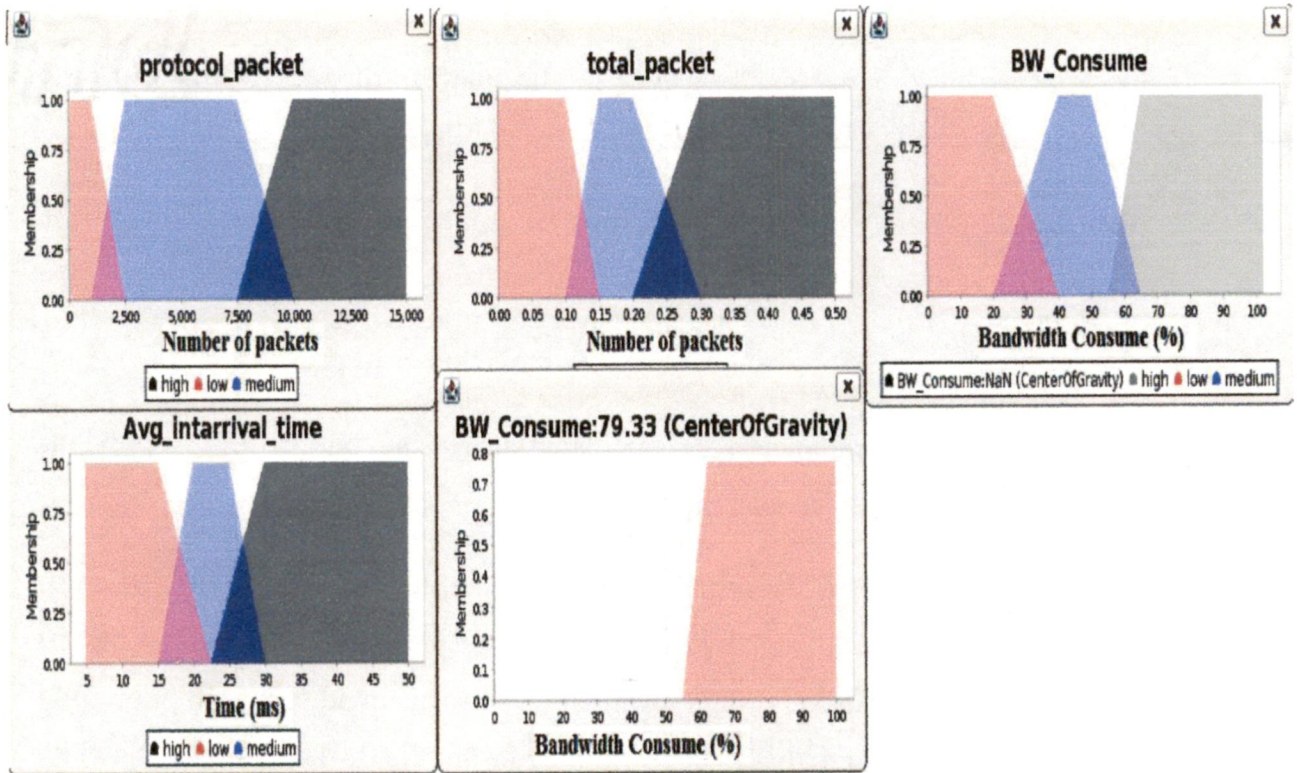


Figure 5.4 Calculation of High Bandwidth Consumption by BWM Algorithm

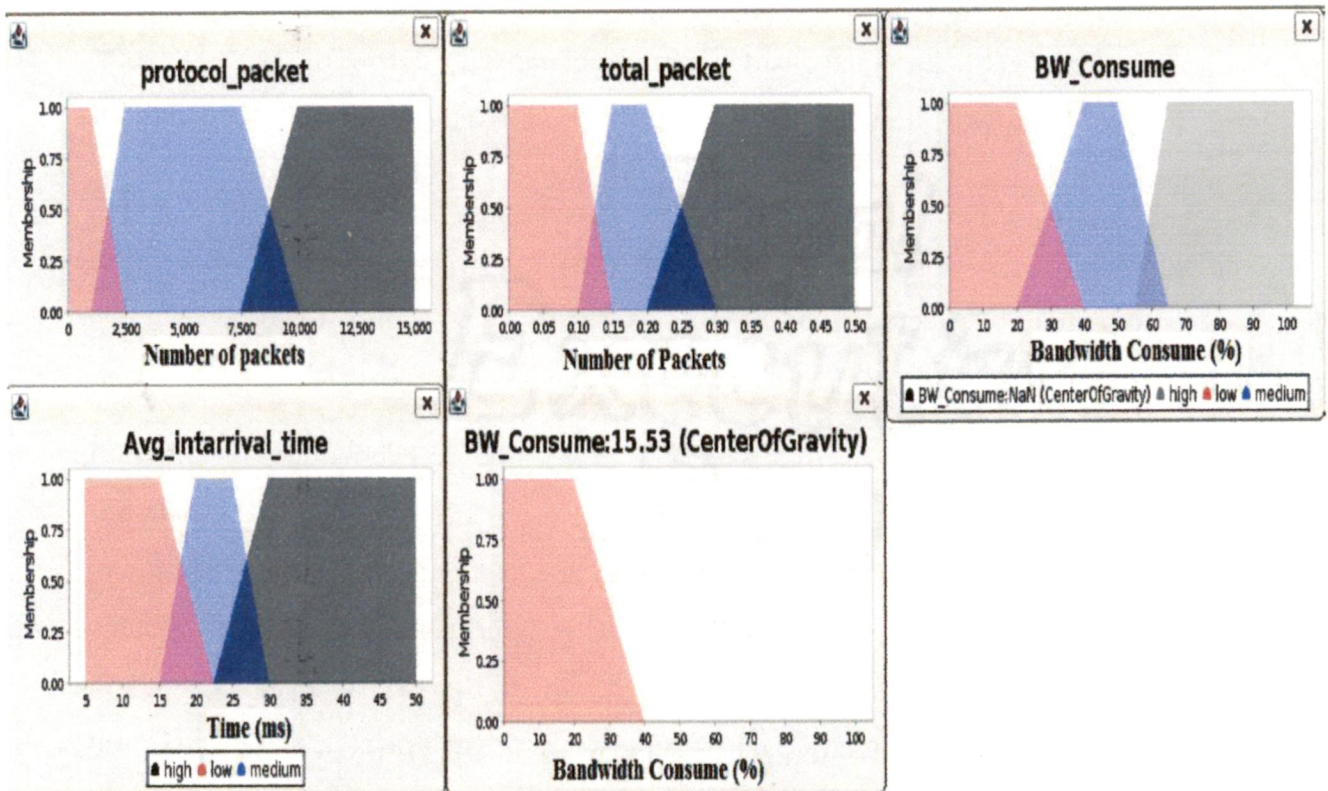


Figure 5.5 Calculation of Low Bandwidth Consumption by BWM Algorithm

the fuzzy controller. These parameters are processed by fuzzy controller and return the bandwidth consumption by individual protocol.

As shown in figure 5.4 and 5.5 the fuzzy controller use center of gravity method to calculate the values of bandwidth consumption. This value depends on the three input parameters average interarrival time, number of protocol packet, and protocol ratio.

Table 5.4 Bandwidth Consumption by Various Protocols

Trace file	protocol	Avg time (ms)	protocol_packet (Nos.)	total_packet (Nos.)	BW_consume (%)
1.	ICMP	18.83	13163	26471	79.33
	Bittorrent	34.39	7345	26471	43.32
	DNS	40.28	2483	26471	15.52
	HTTP	5.01	52	26471	15.52
	NETBIOS	31.26	3428	26471	43.16
2.	Bittorrent	45.34	4637	19029	43.13
	ICMP	14.37	13707	19029	79.87
	HTTP	176.62	377	19029	15.52
	DNS	196.75	308	19029	15.52
3.	ICMP	36.06	4285	13728	43.48
	Bittorrent	33.61	4877	13728	43.48
	HTTP	14.63	2491	13728	43.48
	SSDP	14.63	919	13728	15.52
	NETBIOS	19.74	1147	13728	15.90
	DNS	7.33	9	13728	15.52
4.	Bittorrent	23.83	6368	9168	43.48
	ICMP	51.82	1501	9168	26.00
	DNS	16.14	1265	9168	29.47
	HTTP	39.50	34	9168	15.52
5.	Bittorrent	23.20	5054	9550	43.48
	ICMP	13.21	4496	9550	79.87
6.	DNS	378.48	476	20403	15.52
	HTTP	42.57	973	20403	15.52
	ICMP	24.24	8949	20403	65.02
	Bittorrent	25.44	10005	20403	69.43
7.	Bittorrent	45.34	4637	19029	43.13
	ICMP	14.37	13707	19029	79.87
	HTTP	176.62	377	19029	15.52
	DNS	196.75	308	19029	15.52
8.	Bittorrent	16.59	7923	9031	69.59
	NETBIOS	18.16	103	9031	17.00
	ICMP	11.24	1003	9031	16.46

Once these three parameters are given as input fuzzy controller apply rule on them and then convert resultant fuzzy output into a real value and returns it to us. The bandwidth consume by various application protocol during surveillance time is shown in table 5.4. The graph of bandwidth consume by various application protocol during surveillance time is shown in figure 5.6. The Bittorrent is one of the highly active protocols during complete surveillance period. That is why most of time it tries to exhausts the available bandwidth capacity of the channel. The high value of Bittorrent protocol in all traces reflects that it is a bandwidth intensive protocol during each surveillance cycle. As Bittorrent is a time insensitive and bandwidth intensive application so now there is a need to design a marking threshold to reduce the impact of application like Bittorrent on overall network capacity.

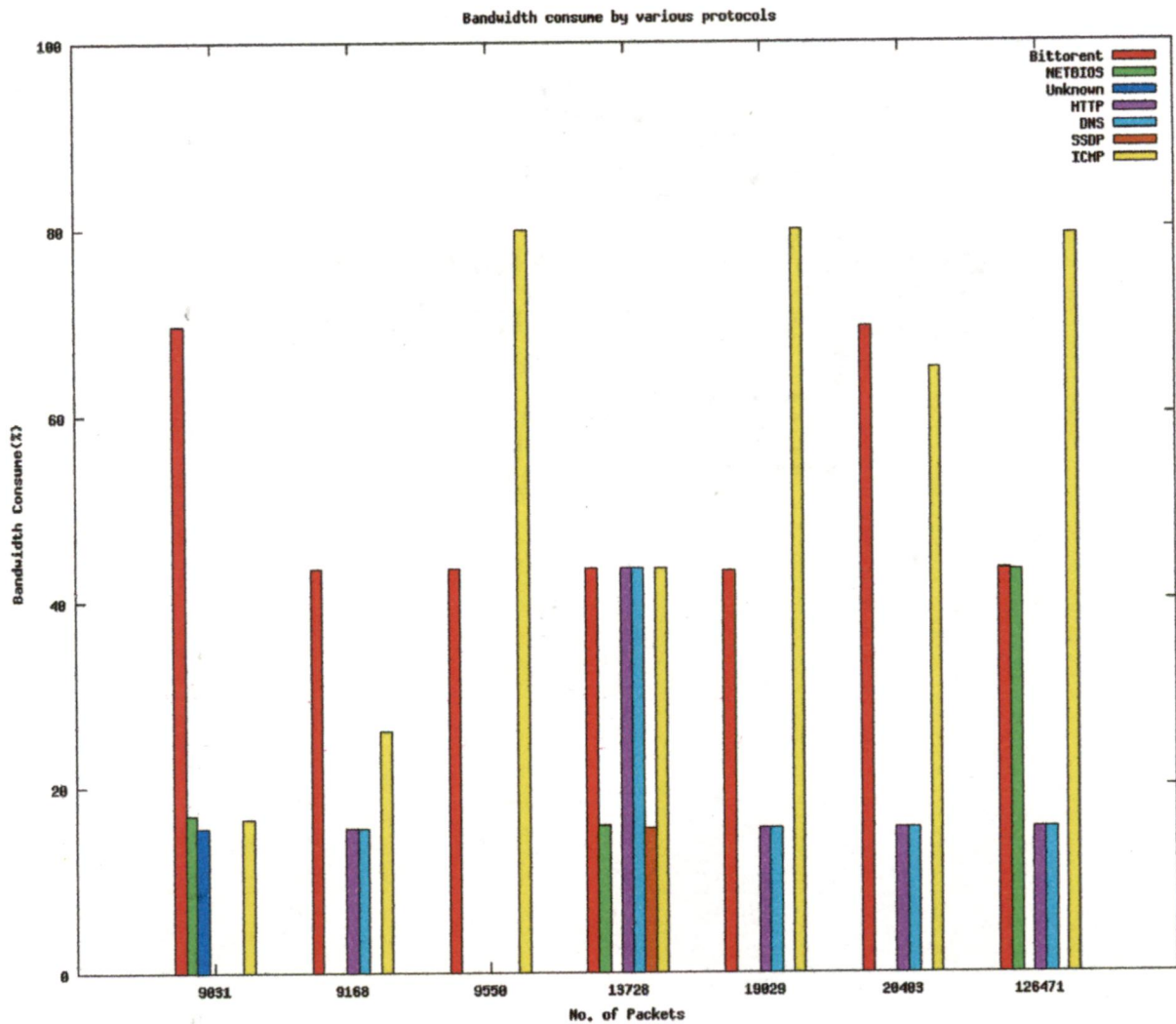


Figure 5.6 Bandwidth Consumption by Various Applications

5.4 Threshold Marking

Due to the high bandwidth consumptions by protocols like Bitorrent which may exhausts the channel capacity. There is a need of design some marking threshold in order limits the high bandwidth intensive and time insensitive applications. We use to mark policy by marked packet whose bandwidth consumption is more than that of 50% of channel capacity. Once packet is marked they can be dropped. The marking algorithm works if the overall bandwidth consumption in current surveillance cycle is more than 50% we can mark the packets for next surveillance cycle. The marking is done on every alternative packet as per our implementation which brings down the packet ratio and total packet of that protocol. The total packet of that protocol also lower the average inter arrival time. One Next surveillance cycle if the overall bandwidth consumption by the protocol falls below 35% we start marking every third protocol packet to allow some traffic by Bittorrent protocol. The overall policy keeps the traffic of time insensitive and bandwidth intensive application below 50%.

Table 5.5 shows the bandwidth consumption calculation of various protocols after implementation of marking algorithm.

Table 5.5 Bandwidth Consumption by Protocols after Marking Threshold

Trace file	protocol	Avg time (ms)	protocol_packet (Nos.)	total_packet (Nos.)	BW_consume (%)
1.	ICMP	37.67	6581	26471	43.08
2.	ICMP	28.75	6853	19029	43.37
3.	No marking is needed as every protocol consume bandwidth less than threshold				
4.	No marking is needed as every protocol consume bandwidth less than threshold				
5.	ICMP	26.43	2248	9550	38.56
6.	ICMP	48.49	4474	20403	43.12
	Bittorrent	50.89	5002	20403	43.35
7.	ICMP	28.75	6853	19029	43.37
8.	Bittorrent	33.18	3961	9031	43.4

In table 5.5 contains only protocol of table 5.4 which need marking due to their bandwidth consumption id more than that of threshold limit of bandwidth. In trace 1 and 2 ICMP is marked as its packet utilize around 70 % channel before marking. The 3 and 4 trace contains packet of each protocol below the threshold limit hence no need to mark any protocol. The trace 5, 6, and 7 contains ICMP packets which need to be marked as there packet flooded the channel. The trace 6 and 8 contains Bittorrent protocol which also consumes around 80 % of capacity so again marking is needed.

Figure 5.5 shows the bandwidth consumption by all protocol after implementation of marking algorithm.

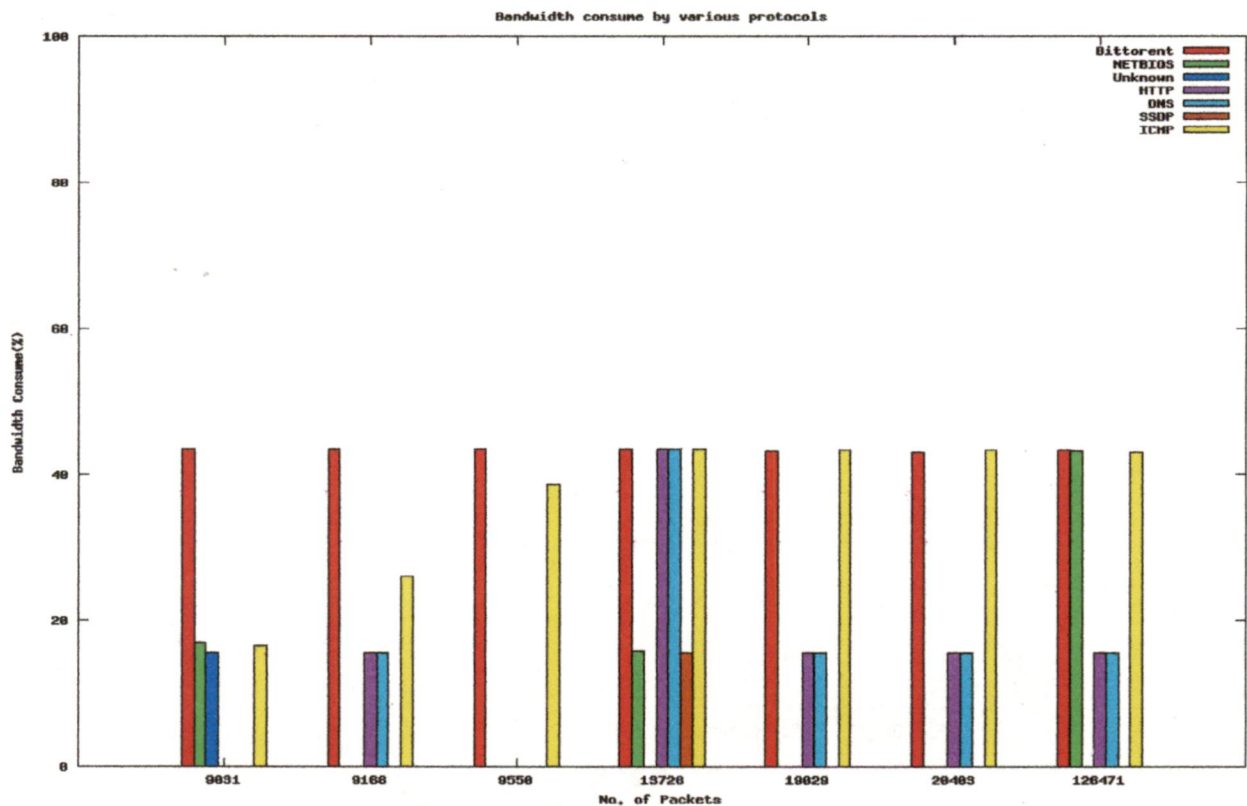


Figure 5.7 Bandwidth Consumption by Applications After Marking Threshold

Protocol like time insensitive and bandwidth intensive Bittorrent is consumes lots of bandwidth. Once we marked these protocols we create good space for other application and hence there is some spare bandwidth for new application or time sensitive application

which needed the real quality of service. The marking hardly impact the performance of Bittorrent protocol but it's really improves the channel capacity.

The overall design of our priority based bandwidth consumption algorithm is to provide quality of service. The algorithm is more accurate in term of detection and covers more number protocols as compare to other algorithms [32, 34, 49] .The algorithm uses both flow based approach for connection set which used for priority and application based approach for protocol detection. The algorithm is as it maintains the priority list of protocols. Generally protocol is easily detected in priority list so it improves overall execution time of algorithm.

5.5 Overview of the Results

The test results shows that the implemented approach limits the P2P and other non real time applications. It increases the channel capacity for real time applications. There are less than 10 protocols used by communicating machines during testing. The priority based algorithm is really fast in term of execution as compare to non priority based algorithm in which, for each packet, all protocols available in database are searched. The priority based bandwidth management algorithm also able to detect 90% - 95% packets correctly as compared to port based approach. Port based approach able to detect below 10% packets only and wrong classification is also done but by our approach not a single packet is wrongly classified.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

In this dissertation, an approach for providing QoS by network traffic classification and bandwidth management is implemented to optimize the bandwidth usage. The priority based bandwidth management algorithm (PB2MA) approach uses the concept that only a 10 to 15 protocols instead of all available protocols are frequently active between any two communicating machines and the most of the bandwidth is consumed by the time insensitive but bandwidth intensive applications only. The bandwidth management decision is taken in such a way that the QoS requirement of the real time traffic is not violate and non real time packets may drop or mark. The marking algorithm only limits the packets of time insensitive applications with high bandwidth consumptions.

The following conclusions can be made from the results obtained using the proposed PB2MA approach:

- This approach is able to limit the throughput of the non-real time connections. This limit in throughput is at the cost of providing bandwidth space for the real time connections to an extent which does not degrade their QoS requirements.
- The proposed approach is really fast in execution as compare to payload based approach, so it is more feasible in real condition.
- The use of fuzzy controller enhances the performance of algorithm, as it is able to do classification in real time.
- The marking algorithm helps to limit non real time bandwidth intensive protocol only, due to which there is sufficient availability of bandwidth space for critical applications.
- The marking approach helps network administrator to limit the packets based on network demand.

6.2 Future Works

The future work includes following areas:

1. The implementation so far is only marks the packets but there is a need to create ICMP message containing flow of high bandwidth consumption protocols for Router/Gateways so that packets can be dropped.
2. Although the algorithm is really fast as compare to payload based approach but its performance can be enhanced by implementing in parallel on Graphics processing Units (GPUs).
3. Further extension is needed in order to support all protocols used by network applications.

REFERENCES

- [1] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *Communications Surveys & Tutorials, IEEE*, vol. 10, pp. 56-76, 2008.
- [2] D. Ferguson, "P2P File Sharing –The Evolving Distribution Chain," CacheLogic, WDC, 2006.
- [3] A. Madhukar and C. Williamson, "A Longitudinal Study of P2P Traffic Classification," in *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2006. MASCOTS 2006. 14th IEEE International Symposium on*, 2006, pp. 179-188.
- [4] Z. Z. Kuai Xu, and Supratik Bhattacharyya, "Profiling Internet Backbone Traffic: Behavior Models and Applications," in *SIGCOMM 2005*, Philadelphia, Pennsylvania, USA, 2005.
- [5] Y. M. Gaogang Xie, Dafang Zhang, "Router Performance Analysis Based on a Periodical Logarithmic Traffic Model," *Chinese Journal of Computers*, vol. Vol.25, No.12, Dec.2002.
- [6] M. C. A. Lakhina, C. Diot, "Characterization of Network-Wide Traffic Anomalies in Traffic Flows," in *Proc. Of ACM SIGCOMM Internet Measurement Conference*, 2004.
- [7] G. V. Cristian Estan "New directions in traffic measurement and accounting," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, Pittsburgh, Pennsylvania, USA, 2002.
- [8] Esoft, "Modern Network Security: The Migration to Deep Packet Inspection," 2005.
- [9] M. Mohsin, W. Wong, and Y. Bhatt, "Support for real-time traffic in the Internet, and QoS issues," in *A survey of QoS protocols for the Internet*: Department of Computer Science, University of Texas at Dallas, , March 30, 2002.
- [10] J. Saltzer, D. Reed, and D. Clark, "End-to-end arguments in system design," *ACM Transactions on Computer Systems (TOCS)*, vol. 2, p. 288, 1984.
- [11] B. E. Carpenter, "RFC 1958:Architectural Principles of the Internet. ," Fremont/CA IAB Network Working Group 1996.

- [12] P. Ferguson and G. Huston, "Quality of Service in the Internet: Fact, fiction, or Compromise," in *Proceedings of AUUGN 1998*, NSW, Australia, Sept., 16-18, 1998, pp. 231-256.
- [13] T. AbuHmed, A. Mohaisen, and D. H. Nyang, "Deep Packet Inspection for Intrusion Detection Systems: A Survey," Technical Report, Security Research Laboratory, Inha University, Information Incheon 402-751, Korea 2007.
- [14] SNORT, "Network intrusion detection system," Available: www.snort.org.
- [15] A. Spyros, G. A. Kostas, and P. M. Evangelos, "Generating realistic workloads for network intrusion detection systems," in *Proceedings of the 4th international workshop on Software and performance* Redwood Shores, California: ACM, 2004, pp. 207-251.
- [16] M. Roesch, "Snort-lightweight intrusion detection for networks," in *Proceedings of LISA'99: 13th Systems Administration Conference*, 1999, pp. 229-238.
- [17] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," *Communications of the ACM*, vol. 18, p. 340, 1976.
- [18] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Commun. ACM*, vol. 20, pp. 762-772, 1977.
- [19] S. Wu and U. Manber, "A fast algorithm for multi-pattern searching," *Technical Report TR-94-17, Department of Computer Science, University of Arizona*, 1994.
- [20] I. D. S. Bro, "Homepage: <http://www.bro-ids.org>," Sept., 2010.
- [21] L.-f. c. p. team, "L7-filter Classifier." vol. [Accessed: Sept.,2010]: Available:<http://l7-filter.sourceforge.net/>, 2010.
- [22] N. Tuck, T. Sherwood, B. Calder, and G. Varghese, "Deterministic memory-efficient string matching algorithms for intrusion detection," in *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, 2004, pp. 2628-2639 vol.4.
- [23] M. Fisk and G. Varghese, "An analysis of fast string matching applied to content-based forwarding and intrusion detection," Technical Report CS2001-0670 (updated version), University of California - San Diego, 2002.
- [24] T. Raita, "Tuning the Boyer-Moore-Horspool string searching algorithm," *Software: Practice and Experience*, vol. 22, pp. 879-884, 1992.
- [25] A. N. M. Rafiq, M. W. El-Kharashi, and F. Gebali, "A fast string search algorithm for deep packet classification," *Computer Communications*, vol. 27, pp. 1524-1538, 2004.

- [26] L. Rong-Tai, H. Nen-Fu, K. Chia-Nan, C. Chih-Hao, and C. Chi-Chieh, "A fast pattern-match engine for network processor-based network intrusion detection system," in *Proceedings. ITCC 2004. International Conference on Information Technology: Coding and Computing, 2004.*, 2004, pp. 97-101 Vol.1.
- [27] A. Chaudhary and A. Sardana, "Software Based Implementation Methodologies for Deep Packet Inspection," in *2011 International Conference on Information Science and Applications (ICISA)*, Jeju Island, South Korea, 2011, pp. 1-10.
- [28] I. E. T. Force, "RFC 2475, An Architecture for Differentiated Services" section 2.3.3.3 - definition of "Shaper", " Available at: <http://tools.ietf.org/html/rfc2475#section-2.3.3.3>. Last updated: December, 1998 [Last accessed: June 5, 2011].
- [29] IANA, "The Internet Assigned Numbers Authority port-numbers," <http://www.iana.org/assignments/port-numbers>, 2011.
- [30] G. Szabo, I. Szabo, and D. Orincsay, "Accurate Traffic Classification," in *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, 2007, pp. 1-8.
- [31] J. Erman, A. Mahanti, and M. Arlitt, "Byte me: a case for byte accuracy in traffic classification," 2007, p. 38.
- [32] M. Iliofotou, K. Hyun-chul, M. Faloutsos, M. Mitzenmacher, P. Pappu, and G. Varghese, "Graph-Based P2P Traffic Classification at the Internet Backbone," in *IEEE INFOCOM Workshops 2009*, , 2009, pp. 1-6.
- [33] W. Nigel, Z. Sebastian, and A. Grenville, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *SIGCOMM Comput. Commun. Rev.*, vol. 36, pp. 5-16, 2006.
- [34] S. Subhabrata, S. Oliver, and W. Dongmei, "Accurate, scalable in-network identification of p2p traffic using application signatures," in *Proceedings of the 13th international conference on World Wide Web* New York, NY, USA: ACM, 2004.
- [35] L. A. Zadeh, "Fuzzy sets*," *Information and control*, vol. 8, pp. 338-353, 1965.
- [36] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning--I* 1," *Information sciences*, vol. 8, pp. 199-249, 1975.
- [37] L. A. Zadeh, "Quantitative fuzzy semantics*," *Information sciences*, vol. 3, pp. 159-176, 1971.
- [38] L. A. Zadeh, *Fuzzy algorithms*: World Scientific Publishing Co., Inc., 1996.

- [39] Botskool, "Autopilot Control System," [Available at]: <http://www.botskool.com/tutorials/fourth-year-projects/autopilot-control-system/autopilot-control-system-page-4>, 2011.
- [40] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, pp. 1-13, 1975.
- [41] H. Hellendoorn and C. Thomas, "Defuzzification in fuzzy controllers," *Journal of Intelligent and Fuzzy Systems*, vol. 1, pp. 109-123, 1993.
- [42] T. J. Ross, *Fuzzy logic with engineering applications*, 2 ed.: John Wiley & Sons Ltd, West Sussex, England, 2004.
- [43] M. Sugeno, "An introductory survey of fuzzy control," *Inf. Sci.*, vol. 36, pp. 59-83., 1985.
- [44] opendpi, "OpenDPI," <http://www.opendpi.org/>, 2011.
- [45] W. Foundation, "Wireshark," <http://www.wireshark.org/>, 2011.
- [46] Sourceforge.net/, "jFuzzyLogic: Open Source Fuzzy Logic (Java)," www.jfuzzylogic.sourceforge.net/, 2011.
- [47] Wikipedia, "List of TCP and UDP port numbers," http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers, 2011.
- [48] μ Torrent, " μ Torrent, a (very) tiny BitTorrent client," <http://www.utorrent.com/>, 2011.
- [49] D. A. Carvalho, M. Pereira, and M. M. Freire, "Detection of Peer-to-Peer TV Traffic Through Deep Packet Inspection."
- [50] M. U. Guide, "The MathWorks," *Inc., Natick, MA*, vol. 5, 1998.

LIST OF PUBLICATIONS

- [1]. **Ajay Chaudhary**, Manoj Misra, and Anjali Sardana, "An efficient fuzzy controller based technique for network traffic classification to improve QoS," in *Seventh International Conference on Information Systems Security (ICISS 2011)*, 5-19 December 2011, Jadavpur, India (Communicated).
- [2]. **Ajay Chaudhary** and Anjali Sardana, "Software Based Implementation Methodologies for Deep Packet Inspection," in *2011 International Conference on Information Science and Applications (ICISA)*, Jeju Island, South Korea, 2011, pp. 1-10.