## ABSTRACT

PATEL, PARITA MUKESH. Classification and Modeling of Internet Applications. (Under the direction of Dr. Arne Nilsson)

The classification of Internet traffic is an active research topic due to its applicability in the areas like differentiated services and network security. The introduction of voice, video and other real-time applications to the Internet has resulted in increasing demand for service differentiation and has triggered the need for change in traffic handling on the Internet. Traditionally, such a classification is done using the packet header field of 'port number,' which is a unique number associated with the application that generated the packet. In addition to adding complexity and extra computation in traffic handling, certain recent developments in networking techniques have rendered port numbers unreliable for this purpose. This motivates our scheme of classification that uses the distribution of packet sizes in a buffer or collected during a short time interval at a switch or a router. We demonstrate that the applications can be classified by these distributions. This 'implicit' classification builds a foundation for estimation and prediction of traffic mix, which is a long-term goal of this research project.

# CLASSIFICATION AND MODELING OF INTERNET APPLICATIONS

by
**Parita Patel**

A thesis submitted to the
Electrical and Computer Engineering Department of
North Carolina State University
in partial fulfillment of the requirements for the Degree of
MASTER OF SCIENCE

***ELECTRICAL ENGINEERING***

Raleigh, North Carolina
2003

**APPROVED BY:**

Dr. Arne A. Nilsson
Chair of Advisory Committee
Professor
ECE Department

Dr H. Joel Trussell
Director of Graduate Programs
Professor
ECE Department

Dr. Mo-Yuen Chow
Professor
ECE Department

*To my family*

## BIOGRAPHY

Parita Patel was born at Indore, India in April of 1980, the year that witnessed the assassination of John Lennon and the birth of CNN and Post-It notes. She spent her early fun-filled years in Ahmedabad, India enrolled in the Amrit Jyoti High School and the Loyola Convent School.

She joined Nirma Institute of Technology, Gujarat University in 1997 and graduated Magna Cum Laude with Bachelor's degree in 2001 in the discipline of Electronics and Communication Engineering. During this time, she did her undergraduate project in the Institute for Plasma Research (IPR), Gandhinagar, India. Her love for communications and networking resulted in her joining the Masters in Electrical Engineering program at North Carolina State University in Fall '01. She worked as a Co-Op in FMC Corporation, Baltimore in Spring and Summer 2003. She is currently employed as a Teaching Assistant at N.C. State and hopes to graduate by Dec'03. She is also a Yoga Instructor and has a passion towards dancing.

## ACKNOWLEDGEMENTS

I would like to express my heartfelt thanks to all the people who made this research and my graduate study at N.C. State a wonderful experience for me.

I would like to thank my Advisor and the Chair of Thesis committee, Dr. Arne Nilsson, for guiding me to take the right courses throughout my Masters Program at N.C. State and for his invaluable guidance throughout my research. I would like to thank Dr. H. Joel Trussell for spending his valuable time towards making sure that I didn't lose my focus in our research work and for the meticulous comments on my thesis document. I would especially like to thank Dr. Trussell for his role as the Director of Graduate Programs (DGP). It was their sincere desire to help the students that resulted in my gaining the valuable Co-Op experience in the Spring-Summer of 2003. I would like to thank Dr. Mo-Yuen Chow for his knowledgeable discussions.

I would like to express my gratitude towards all the members of faculty that I happened to have the pleasure of interacting with. All the courses and projects that I was involved with got me more and more interested in my field of study. Without their involvement my graduate study would have certainly lacked a flavor.

I would like to thank Yi Wang for all the help that he lent to me during the long hours of running codes, data mining and charting graphs. I would also like to thank Mr. Dan Green and the entire ECE Help team for all their help whenever I had any system issues or any upgrade requests for my terminal.

I would like to thank all my co-workers from FMC Corporation, Center for Universal Design at N. C. State and the N. C. State Women's Center for giving me the chance to work, learn and have fun with them.

There were always a few people who I wanted to do this for and who make all this worthwhile. I would like to thank my family and all my friends far and near for just being there. Always.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# 1. Introduction

The traffic over the Internet continues to grow, both in terms of amount of traffic and in variety of applications. This ongoing growth of the Internet traffic demands a better performance from the network and efficient use of its resources. The Internet Protocol (IP) with its related technologies has emerged for interconnecting different network technologies and providing a consistent view of the transport infrastructure. It is expected that next generation communication networks will be IP-based, integrating diverse services such as voice, multimedia, and data, each with the appropriate quality. To enable such multi-service network architectures, various mechanisms and methodologies have been developed, which demand Quality of Service (QoS) from the Internet. To provide different qualities of service to different applications, routers need to implement new mechanisms such as admission control, per-flow queuing, resource reservation and fair scheduling. This requires some kind of classification of the traffic on the Internet.

The existing routing policies in the Internet do not support priority-based or differentiated routing service. Currently, routing is done based on best route information at the network layer i.e. IP layer for TCP/IP networks. However, for differentiated services based on the application, we need some additional information like source and destination port numbers, which is obtained from the transport layer, i.e. the TCP layer in TCP/IP networks. This adds extra computation and packet handling at the routers, since the routers have to process information from TCP layer as opposed to the conventional routing that is done at the IP layer. Even if we assume that the Moore's law will hold until the next decade and hence complexity is not a major issue, there are other reliability issues in using the header information of the TCP layer for routing purposes.

Internet is being increasingly used to carry out a lot of important and highly confidential transactions, thus requiring increased security measure. For increased network security, it is possible that in future the headers will be also encrypted before transmission, as compared to only data encryption that is done today. In that case, it would be very difficult and complex to interpret them. With the introduction of concepts like Network Address Port Translation (NAPT), the port numbers are no longer a trustworthy source of information regarding the source application. If the port numbers are used for differentiated services, it is very likely that other applications would spoof port numbers in order to gain better service. Since, in a free environment like the Internet, it is not mandatory for the applications to use specific port numbers.

Conventional router solutions tend to make forwarding decisions such as whether to forward or filter a packet, which router port (if any) to forward it to, what class of service it should receive or how much should be charged for transporting it. These decisions are made for each packet separately even if the decisions for most of the packets would be identical. Information is queued to the routing processor on IP level packet by packet depending on the class of service specified for the packet. For each packet in the queue, the router first looks up the destination address using routing tables and then, sends the packet to the appropriate network interface according to the routing information obtained from the routing tables. Hence, a router inherently utilizes statistical multiplexing since its resources are shared between all users. In the current Internet, the number of router hops between two locations may be dozens or even more. This affects routing, especially the delay, which comes from propagating through a large number of queues and route processors. This problem has become much worse due to the increased packet arrival rates associated with high speed links achieved by technological advances, the complexity of forwarding lookup mechanism and the large size of forwarding tables. Routers are an essential part of the Internet and their limitations are creating bottlenecks in the network.

These bottlenecks motivate our approach in classifying and modeling the network traffic. Flows are specified by rules applied to incoming packets and a collection of these rules is called a classifier [1]. Each rule specifies a flow to which a packet may belong based on some criteria applied to the packet header. All packets belonging to the same flow obey a predefined rule and are processed in a similar manner by the router. Packets are identified

in the network traffic by determining which application an incoming packet belongs to. In this thesis, we present a unique scheme to classify Internet traffic flows over a time frame, based on the applications to which the packets in the flow belong. One key aspect of our scheme is that it does not involve reading the OSI (Open Systems Interconnection reference model) layer 4 i.e. the TCP layer header to determine the application or the flow. We try to classify the IP traffic into different application types on the basis of packet size distributions obtainable from the packet size field at OSI layer 3 i.e. the IP layer. The packet size distribution of an application depends on the software characteristics of the application and the protocols it uses. Hence, it is difficult to spoof or disguise it without degrading performance. Also, it should be noted that we do not classify the traffic on a per-packet basis; rather the traffic flows are classified as containing packets from various types of applications that have different needs for timed service. The routing of the packets would have to be done per packet; however, the class of service may be assigned per flow. This 'implicit' classification builds a foundation for estimation and prediction of traffic mix, which is a long-term goal of this research project. If one is able to estimate the contents of a traffic mix in a given time frame without prying into the TCP header, and from it predicts what the traffic composition is going to be like in the successive time frame, it is possible to configure the schedulers in routers to provide a priority-based service. The traffic mix in a time frame can be considered as the traffic mix in the input buffers i.e. the input lines in Fig 1.1, and the scheduling can be achieved by assigning priorities to the input line.

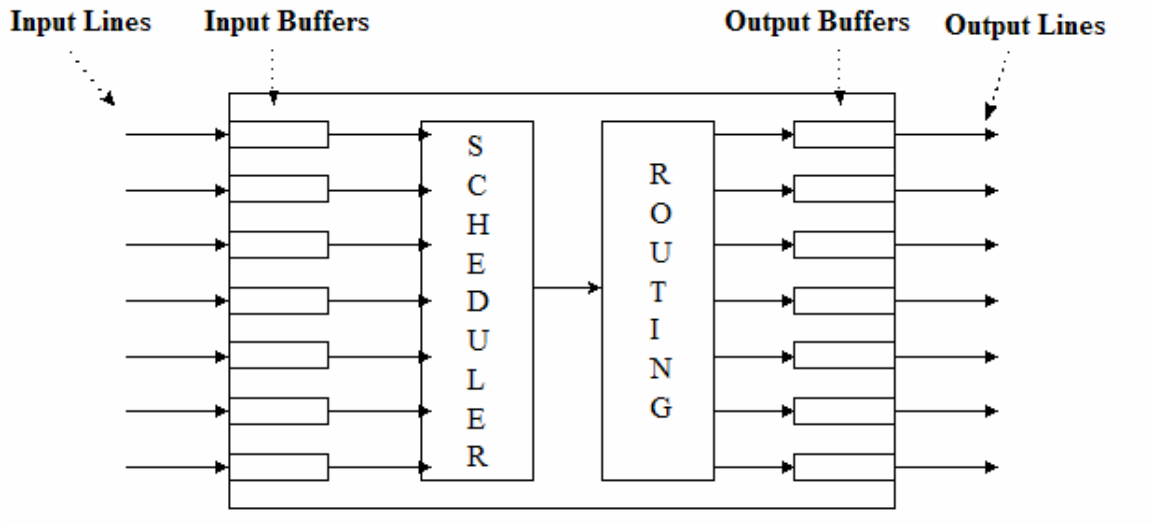



**Figure 1.1 Router Scheduling**

In order to understand this better, let us consider a simplistic case in which the service provider wants to give priority to RTP (Real-time Transport Protocol) packets over all the other packets. Consider the fictitious histograms shown in Fig 1.2 and labeled histogram 1, histogram 2 and histogram 3 be obtained from the packets in input buffers 1, 2 and 3 in Fig 1.1 respectively. The X-axis in these figures represents the histogram bins with some pre-defined bin size. These histograms represent the actual traffic on the Internet in a time frame denoted by $h$ in Equation 1.1. We propose to estimate the proportions of the major applications ($a_i$) in the actual traffic given the average histograms of the applications of interest denoted by $\overline{B}_i$ in Equation 1.1.

$$h = \sum_i a_i \overline{B}_i \tag{1.1}$$

Suppose depending on our estimations, we find that the histogram 1 contains 60% RTP and 40% HTTP packets, histogram 2 contains 30% FTP and 70% HTTP packets and histogram 3 contains 50% Kazaa and 50% eDonkey packets. We can then provide higher

priority to input line 1. Better still, if we are able to predict that we would receive RTP packets on input line 1 in the successive time frame, we can configure the scheduler to provide priority to this line in the next time frame. The scheduler can reserve the space in the queue so that these packets would get the specified priority. The routing is then performed per packet. This idea was introduced in paper [2] and is investigated in further detail in this thesis. The results generated in this thesis indicate that this approach is promising.

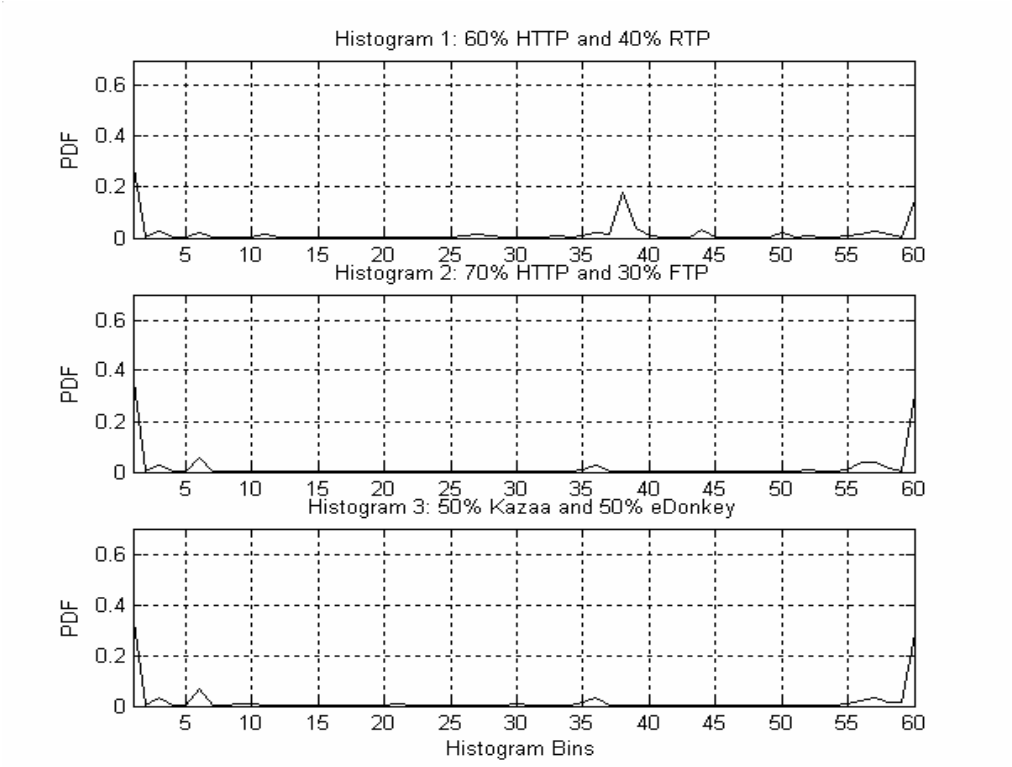



**Figure 1.2 Histogram of actual traffic on three input buffers**

# 2. Networking Background

In this research, we are trying to classify and model the Internet traffic flow, which is comprised of packets. Hence, it is important to understand how the packets are formed and how they are transmitted from one point to another in a network to understand its characteristics. This chapter reviews some basic concepts of networking that are related to this thesis research and defines some of the networking terms that are used in this thesis. It discusses how the packets are formed and why there exist different packet sizes. It also describes the characteristics of the applications and how can these applications be identified based on certain characteristics unique to them. In the end, this chapter gives a background on the use of port numbers by different applications, how the port numbers are assigned to different applications and how the packets are identified as belonging to certain application depending on these port numbers.

## *A. Networking Basics*

When a Source Host has data to be sent over the Internet, it forms a datagram, shown in Figure 2.1, and sends it over the network to the nearest router. The datagrams or the packets are message units that are handled by the Internet Protocol. The intermediate routers forward the datagram to the next router "closer" to the destination and the final router delivers the datagram to the destination host.

Thus a datagram consists of

- header which determines how the datagram is to be treated and gives some information about the data in the datagram and
- the actual data that is required to be transmitted from source to destination.

Each datagram is treated as data by the underlying layers. A datagram is encapsulated in a frame, shown in Figure 2.1, for transmission across a physical network. A frame is usually transmitted between network points bit by bit. The destination address in the frame is the address of the next hop to which the datagram should be sent; the address is obtained by translating the IP address of the next hop to an equivalent hardware address. Whereas the datagram address is the IP address of the final destination. The datagram survives entire trip across Internet whereas the frame survives only one hop because each hop extracts the datagram and discards the frame [4].

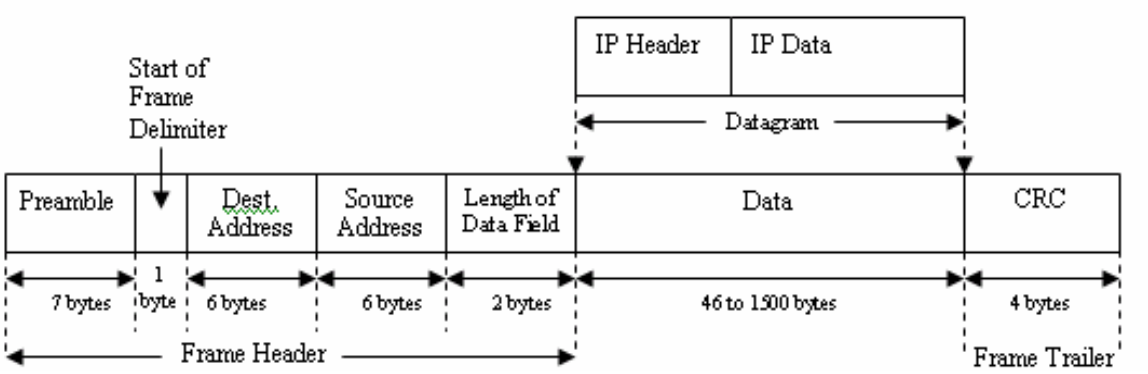



**Figure 2.1 Frame Format**

Modern networks are designed as series of layers where each layer is built using interface of layer below it and is responsible for the different facet of the communications. The purpose of each *layer* is to offer certain services to the higher layers, shielding those layers from the details of how the offered services are actually implemented. Separating network functionality into layers in this manner reduces complexity of network design, enables layers to be implemented relatively independently and allows swapping of functionally equivalent implementations. The set of layers and protocols is called *network architecture* and the list of protocols, one per layer, is called a *protocol stack*. Layers can offer connection-oriented or connectionless service. *Connection-oriented* service is like the telephone where the user establishes connection, sends and receives messages along route of this connection and finally releases connection. *Connectionless* service is like post where communication is split into a series of messages addressed and sent separately and messages may travel to other end by variety of routes [3].

The 7-layer OSI model and the 4-layer TCP/IP protocol suite shown in Figure 2.2 are two widely used layering systems. When the data is moving down the protocol stack, each

layer adds its own header information to the data and when the data is moving up the protocol stack, each layer strips the data of the header field meant for its layer and sends the remaining data to the upper layer. Thus, a new header is added at each layer. This process is called as Encapsulation/ Decapsulation as illustrated graphically in Figure 2.3.

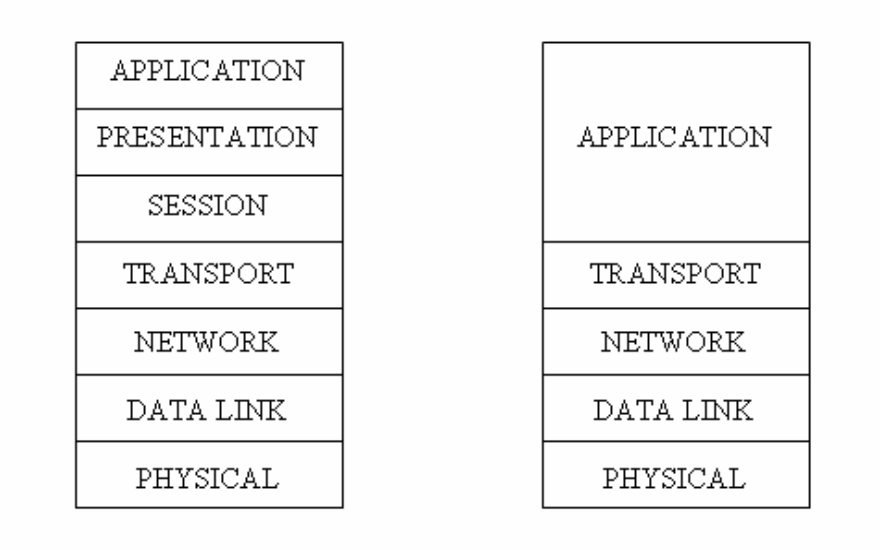



**Figure 2.2 (a) OSI Model** **(b) TCP/IP Model**

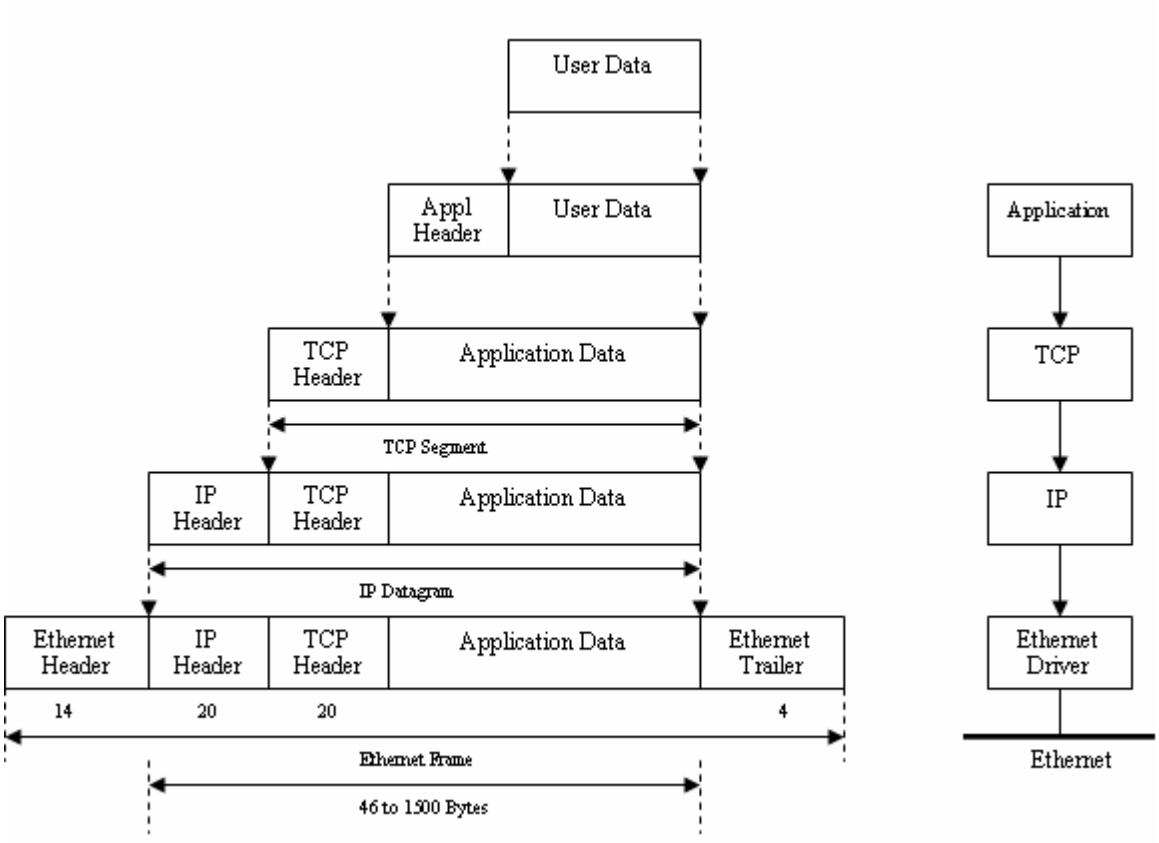



**Figure 2.3 Encapsulation/Decapsulation**

*Why do we have different packet sizes?*

- Network Topology

*Topology* refers to the shape of the network or the network's layout and it determines how the different nodes in the network are connected to each other and how they communicate with each other. Some of the most common network topologies are shown in Figure 2.4. The topology used in the network design depends upon achieving the specified performance from the network at a minimal cost.

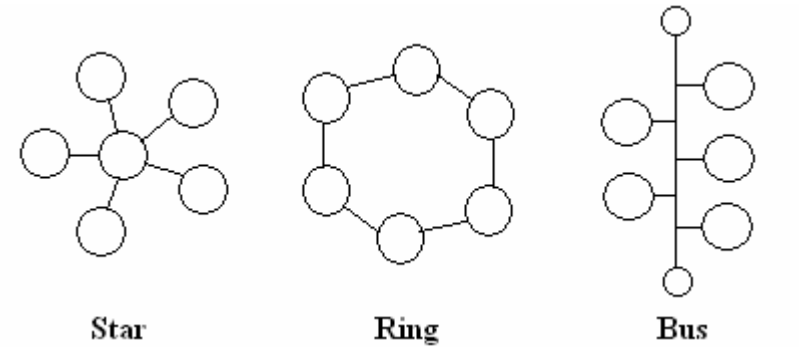



**Figure 2.4 Network Topology**

Each network topology imposes a maximum frame size i.e. the maximum number of bytes in a frame payload called as Maximum Transfer Unit (MTU). The MTUs differ for different network topologies depending upon various network parameters and design requirements. IEEE 802.3 has a restriction on the maximum packet size, which depends on the transmission rate to prevent one node from utilizing the channel for a long time. However, the frame length of IEEE 802.4 token bus may be much larger since the timers can be used to control channel utilization [3]. The maximum frame length of IEEE 802.5 token ring networks depends upon the maximum time a node may hold the token, the data signaling rate and the number of nodes on the ring [5]. MTU (Maximum Transfer Unit) for various underlying networks are shown in Table 2.1 [6]. Table 2.1 shows the maximum IP datagram packet size, which refers to the packet size from IP header to application data (inclusive) i.e. it does not take the Ethernet header or trailer into account (Figure 2.3). IEEE 802.3 has a restriction on the minimum packet size, which depends upon the transmission rate and the maximum distance between any two stations. The minimum frame length of IEEE 802.3 is 64 bytes, from destination to checksum [3].

**Table 2.1 Maximum Transfer Unit (MTU) for different network topologies**

| Network | Default IP Protocol MTU (bytes) |
|---|---|
| Hyperchannel | 65535 |
| 16 Mbps Token Ring (IBM) | 17914 |
| IEEE 802.4 | 8166 |
| IEEE 802.5 (4 Mbps Token Ring) | 4464 |
| FDDI | 4352 |
| Wideband Network | 2048 |
| Ethernet | 1500 |
| IEEE 802.3 | 1492 |
| X.25 | 576 |
| IEEE 802/Source Route Bridge | 508 |
| Point-to-point (low delay) | 296 |

The Internet consists of heterogeneous technologies that have different MTU and one packet might have to traverse through different network topologies before it finally reaches its destination. In order to accommodate the multiple MTUs, fragmentation is performed at the routers as shown in the Figure 2.5. If the IP packet arriving at the router from network 1 is larger than the MTU of the network 2, the router fragments the packet into smaller IP packets. Each fragment carries some data from the original datagram, and has an IP header similar to the original datagram that indicates where the fragment fits in the original datagram, as shown in Figure 2.6. The original packet may be fragmented multiple times along its route; however, defragmentation is performed only once and hence is carried out at the Internet Protocol (IP) layer of the destination. The destination collects all the incoming fragments and reassembles them when all fragments arrive. The receiver does not know the identity of the router that performed fragmentation and hence

it cannot request missing pieces. Thus, loss of one fragment means entire datagram is lost.

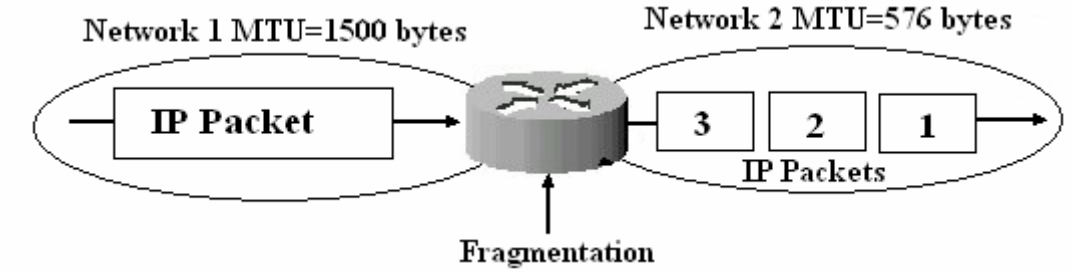



**Figure 2.5 IP Fragmentation at Routers**

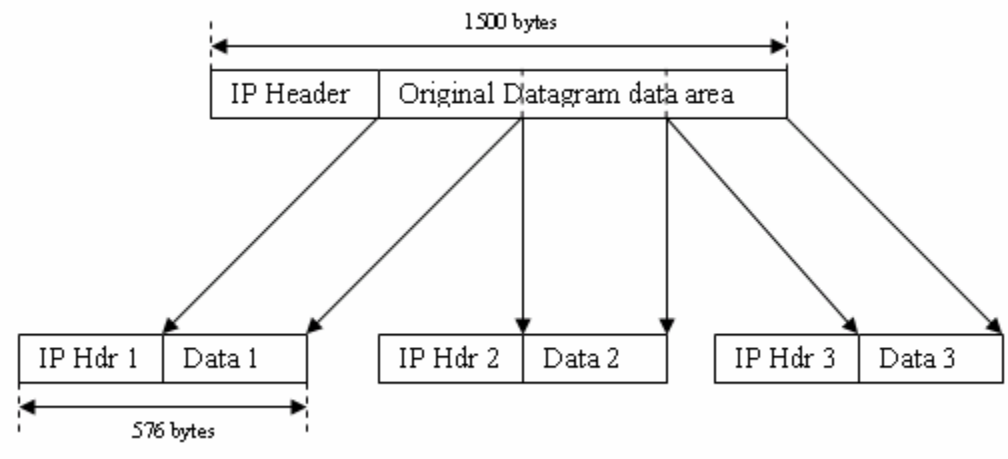



**Figure 2.6 Fragmentation**

For this research, we have collected data from an Ethernet network, so we shall discuss the packet sizes of an Ethernet network. The minimum packet size of the Ethernet network is 60 bytes and the maximum size is 1514 bytes from MAC header to data i.e. discarding the checksum. Figure 2.8, in the next section, shows the Ethernet packet format.

- Payload

Normally, you would expect that the packet sizes of the most of the packets on the Ethernet network would be either the minimum size i.e. 60 bytes corresponding to the acknowledgements or the maximum size i.e. 1514 bytes for data. Some small percentage of packets, corresponding to the last data packet in a particular transmission, might be smaller than 1514 bytes as the data was too large to fit in one packet. Hence you would expect to see peaks at packet size 60 bytes and 1514 bytes. But looking at the packet size distributions of some applications, it is evident that some applications use packet size less than 1514 bytes most times and this results in significantly different packet size distributions.

- Application

The packet size chosen by an application may depend on the transport protocol used by that application, for example, TCP or UDP, and the normal function of that application. Certain applications care more about overhead and throughput and hence send larger packets whereas certain others are more concerned about sending the information immediately to achieve low latency and hence send small packets. The packet size may also depend on whether the application requires reliable transmission or can tolerate errors in transmission. For example, the online games such as Half Life require a high degree of interactivity and low latency between the client and server. The real time action and coordinate information needs to be sent back and forth between clients and server. Such traffic tends to employ small, highly periodic, UDP packets since the clients typically send packets at an interval that is much shorter than the time it would take to

retransmit lost packets [7]. Packets are small since the application requires extremely low latencies, which make message aggregation impractical. Packets are highly periodic as a result of the game's dynamic requirement of frequent, predictable state updates amongst clients and servers. As seen from the packet size distribution for Half Life in Figure 2.7, where the horizontal axis indicates the packet size in bytes and the vertical axis indicates the probability, the packets for this application are usually in the range of 0-250 bytes.

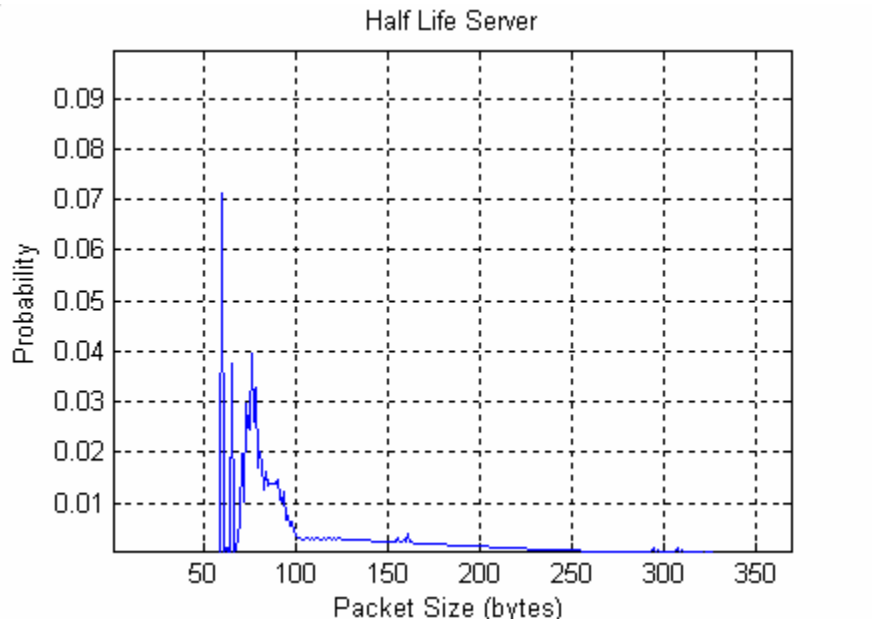



**Figure 2.7 Packet Size Distribution for Half Life**

## *B. Applications*

This section explains the function of many of the top 20 applications in the collected data. This is not an exhaustive list of applications in the data but serves to characterize many applications and relates their functions to their packet size distributions. The Table 2.2

gives the top 20 applications along with the port numbers associated with the application [8], the source and destination packets, the total packets of the application, the percentage of the application in the total network traffic and the protocol most commonly used by the application. Note that a packet might have different source and destination port numbers corresponding to different applications. Also note that the protocols listed are most commonly used by the application. The application may use protocols other than those listed in certain cases. In order to find the major applications in the traffic, the counters corresponding to both the source and destination port numbers are incremented. Hence, the same packet is considered twice for finding the major applications in the traffic. The actual traffic, as seen from Table 2.2, would be about twice the traffic calculated in this manner. Once we know the major applications in the network traffic, the actual percentage of the application can be calculated by incrementing the counters of the applications of interest.

**Table 2.2 Major Applications in Data Set 3**

| | Application | Port # | Source | Destination | Total | % | Actual % | Protocol |
|---|---|---|---|---|---|---|---|---|
| 1 | HTTP | 80 | 116002335 | 90618503 | 206620798 | 7.83% | 15.28% | TCP |
| 2 | Kazaa | 1214 | 27005836 | 31901668 | 58903521 | 2.23% | 4.35% | UDP |
| 3 | FTP | 20 | 18772578 | 18094580 | 36867158 | 1.40% | 2.73% | TCP |
| 4 | Gnutella | 6346 | 10635614 | 13331009 | 23875908 | 0.90% | 1.77% | TCP |
| 5 | Unassigned | 3933 | 9764395 | 6840237 | 16604630 | 0.63% | 1.23% | |
| 6 | RTP | 6970 | 2210188 | 13041141 | 15250628 | 0.58% | 1.13% | UDP |
| 7 | Napster | 6699 | 7437414 | 7155462 | 14592876 | 0.55% | 1.08% | TCP |
| 8 | eDonkey | 4662 | 5926373 | 7143020 | 13069364 | 0.50% | 0.97% | TCP/UDP |
| 9 | AOL | 5190 | 6242588 | 6260680 | 12503268 | 0.47% | 0.92% | TCP/UDP |
| 10 | Multicast | 16384 | 3677 | 12477952 | 12481629 | 0.47% | 0.92% | |
| 11 | Half Life Server | 27015 | 5028609 | 6725843 | 11754452 | 0.45% | 0.87% | UDP |
| 12 | Plethora | 3480 | 6064115 | 4414098 | 10478077 | 0.40% | 0.77% | TCP/UDP |
| 13 | Reserved | 0 | 8509252 | 8507311 | 8509359 | 0.32% | 0.63% | |
| 14 | IRC | 6667 | 3847334 | 4553535 | 8400869 | 0.32% | 0.62% | TCP/UDP |
| 15 | MS-OLAP | 2394 | 4480154 | 3149574 | 7629700 | 0.29% | 0.56% | TCP/UDP |
| 16 | SMTP | 25 | 3270275 | 4089714 | 7359989 | 0.28% | 0.54% | TCP |
| 17 | Half Life Client | 27005 | 4266992 | 3049172 | 7316164 | 0.28% | 0.54% | UDP |
| 18 | ICAP | 1344 | 4254704 | 2769622 | 7023631 | 0.27% | 0.52% | TCP |
| 19 | Tragic | 2642 | 4140301 | 2805480 | 6945227 | 0.26% | 0.51% | TCP/UDP |
| 20 | mloadd | 1427 | 3951642 | 2703181 | 6641561 | 0.25% | 0.49% | TCP/UDP |
| 21 | Other | | 1352564781 | 794585335 | 2147150116 | 81.33% | 63.56% | |

All the applications that use TCP/IP protocol have the packet format shown in Figure 2.8 and the applications that use UDP/IP protocol have the packet format shown in Figure 2.9.

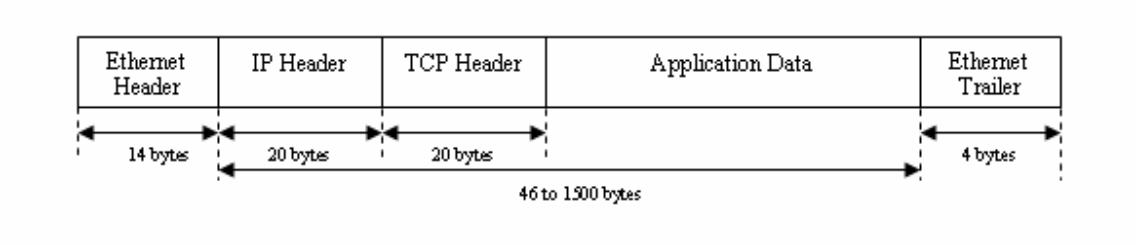



**Figure 2.8 TCP Packet Format**

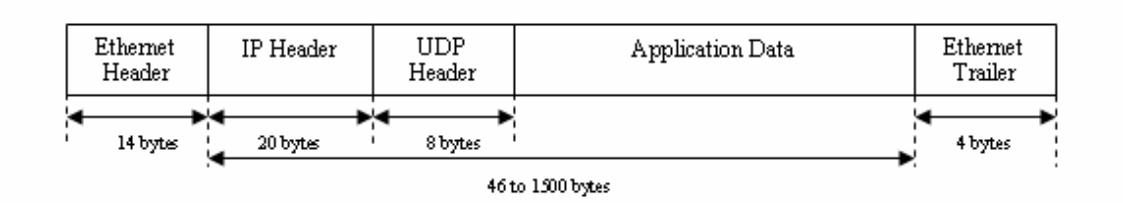



**Figure 2.9 UDP Packet Format**

The average packet size distributions and its variance shown in Fig 2.10 to Fig 2.29 are plotted considering the 50-bin histogram scheme. The bin size and range for the 50 bins are tabulated in Table 4.1.

- *HTTP (Hypertext Transfer Protocol)*

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a client-server TCP/IP protocol used on the World-Wide Web for exchanging HTML documents. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. The Web server that uses HTTP conventionally uses port 80. HTTP requires reliable transmission of data and hence uses TCP at the transport

layer. The packet format of HTTP is same as Figure 2.8. Figure 2.10 (a) and (b) shows the packet size distribution (PSD) of the packets originating and destined to port 80 respectively, which is associated mostly with HTTP server. HTTP packets destined to the server are usually small, about 60 bytes size, corresponding to the acknowledgements sent from the client to the server as seen in Fig 2.10 (b). HTTP packets originating from server are usually large, about 1514 bytes size, corresponding to the data sent from server to the clients as seen in Fig 2.10 (a).

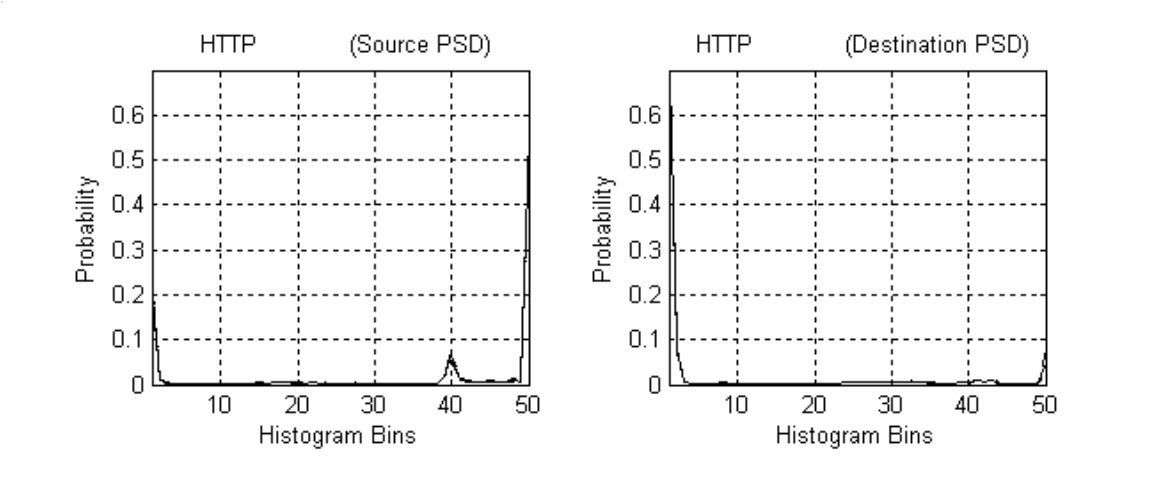



**Figure 2.10 (a) HTTP Source Port PSD (b) HTTP Destination Port PSD**

- *Kazaa*

Kazaa is a peer-to-peer (P2P) application that uses distributed architecture for file sharing. *Peer-to-peer or P2P* is a communications model in which each party has the same capabilities and either party can initiate a communication session. Kazaa runs on a decentralized network and each client must connect to a Supernode to be able to search

for files. The Supernode contains a list of some of the files made available by other Kazaa users and where they are located. When a search for a particular file is performed, Kazaa first searches the nearest Supernode, which sends the immediate results and then refers the search to other Supernodes, which send their results. Once the required file is located, it is sourced for downloading directly from the user who has it, resulting in a direct connection between the peers. It mostly uses UDP as the transport layer protocol but may use TCP. The default port associated with Kazaa is 1214 but the most recent versions of Kazaa can use other ports if port 1214 is not available. The source and the destination port packet size distributions (PSD) for port 1214 are shown in the Fig 2.11 (a) and (b) respectively. The PSDs show peaks in both 1$^{st}$ and 50$^{th}$ bin for source and destination. This is because Kazaa does not have client-server architecture and hence a client on port 1214 can receive large packets when it is downloading and send small acknowledgements back or it may receive small packets for acknowledgements when it is uploading and send large data packets.

**Figure 2.11 (a) Kazaa Source Port PSD (b) Kazaa Destination Port PSD**

- *FTP (File Transfer Protocol)*

The File Transfer Protocol is an application level protocol that uses Internet's TCP/IP protocols. The primary function of FTP is defined as transferring files efficiently and reliably among hosts on the Internet and allowing the convenient use of remote file storage capabilities. Traditionally, FTP uses port 20 for the data port and port 21 for the

command port [8]. Since FTP port 20 is associated with the server, the source and destination PSD for port 20 shown in Fig 2.12 (a) and (b) respectively are as viewed from the server. The source PSD has a large peak at 50$^{th}$ bin due to the data packets sent by the data port of the server to the data port of the client and the destination PSD has a peak at the 1$^{st}$ bin due to the acknowledgements sent by the data port of the client to the data port of the server once they receive the data packets. The peak at the 50$^{th}$ bin in destination PSD due to some files uploaded by the client to the server and the peak at 1$^{st}$ bin in source PSD is due to the acknowledgements sent by the server to the client for the uploaded files. However, depending on the purpose of the FTP server, there might be more uploads to the server as compared to downloads from the server. In this case there might be a higher peak at the 50$^{th}$ bin in the destination PSD as compared to the peak at the 50$^{th}$ bin in the source PSD.

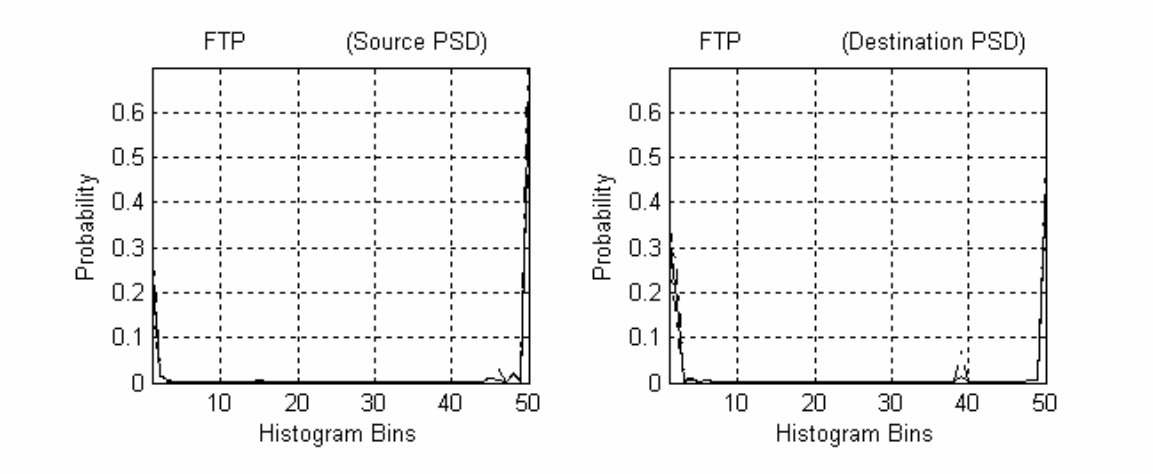



**Figure 2.12 (a) FTP Source Port PSD** **(b) FTP Destination Port PSD**

- *Gnutella*

The Gnutella protocol is a decentralized open source protocol designed for sharing and searching files in a distributed network and it defines the ways in which the Gnutella clients communicate over the network. The Gnutella protocol was designed by AOL's Nullsoft division to surpass the file sharing capabilities of Napster, but was soon released to the public domain. It allows a user to share any type of file from his computer and make it available to anyone using the Gnutella network. Every machine in the Gnutella network is connected to every other machine and no single server is responsible for distributing all of the content. This makes it highly fault-tolerant, as operation of the network will not be interrupted if a subset of *servents*, the application clients or programs that access the Gnutella network, go offline. There are many Gnutella clients or servents such as BearShare, Gnucleus, Limewire etc. and they perform the tasks normally associated with both clients and servers. They provide client-side interfaces through which users can issue queries and view search results, while at the same time they also accept queries from other servents, check for matches against their local data set, and respond with applicable results. When a servent responds to a search message, it includes all of the information needed to retrieve the file, including the IP address and the port on which it is listening for connections, then a TCP/IP or UDP/IP connection is created to the servent. Gnutella clients commonly use port 6346 or 6347. The source and destination PSD for port 6346 is shown in Fig 2.13 (a) and (b) respectively.

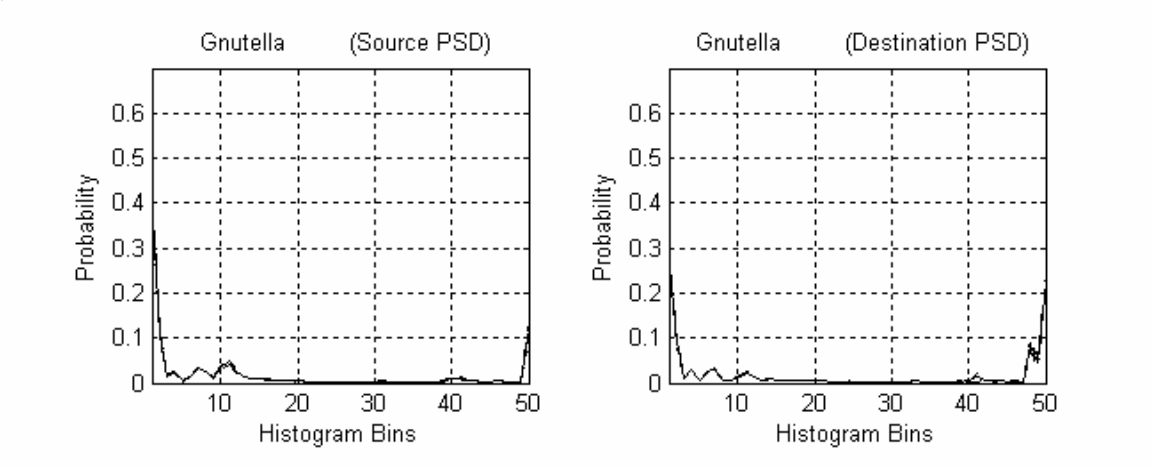



**Figure 2.13 (a) Gnutella Source Port PSD (b) Gnutella Destination Port PSD**

- *Unassigned*

Port 3933 is not registered with IANA [8] for any particular application. Hence, we cannot determine for sure what application is accessing this port and hence cannot explain clearly the characteristics of the PSD. The source and destination PSD for port 3933 is shown in Fig 2.14 (a) and (b) respectively. From the PSD plots, it seems that this application transfers large data files and receives small files, possibly corresponding to the acknowledgements.

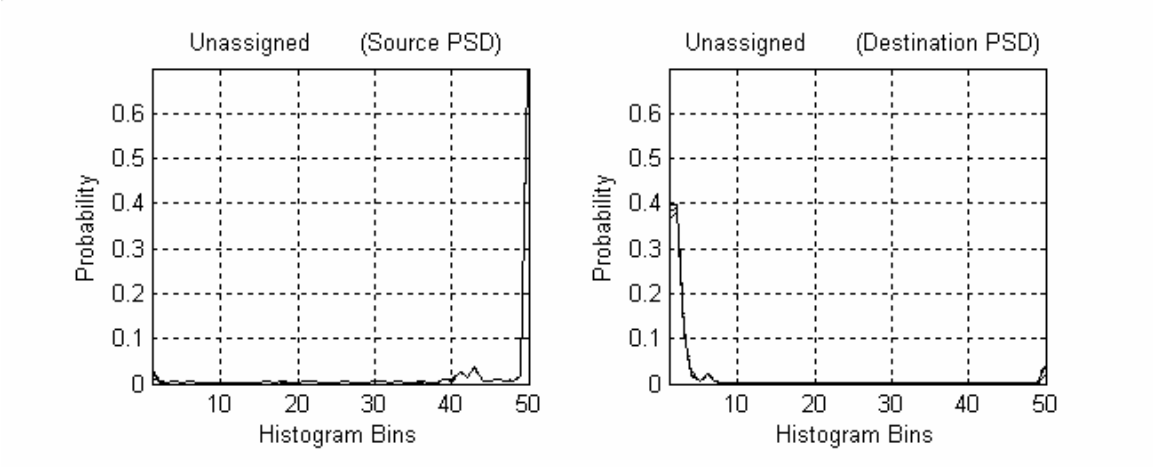



**Figure 2.14 (a) Unassigned Source Port PSD (b) Unassigned Destination Port PSD**

- *RTP (Real-Time Transport Protocol)*

RTP is an Internet protocol standard that provides end-to-end delivery services for multimedia data with real-time characteristics, such as audio and video, over either unicast or multicast network services. RTP itself does not guarantee real-time delivery of data (since this is dependent on network characteristics), but it does provide mechanisms for the sending and receiving applications to support streaming data. Typically, RTP runs on top of the UDP protocol, although the specification is general enough to support other transport protocols. If the entire audio stream were packed in one packet, the packet will be too large and we shall not be able to receive it at "real" time. Therefore, for practical purposes, the real audio or video stream is divided into 'chunks' that are transmitted using several packets, which are synchronized by the timestamp information in its header. RTP data packets contain no length field or other delineation; therefore RTP relies on the underlying protocol(s) to provide a length indication. The maximum length of RTP packets is limited only by the underlying protocols [9]. Ports 6970-6999 are generally

used by the RTP based applications such as QuickTime and RealAudio. Real Time Streaming Protocol (RTSP) which uses port 554 serves as the control protocol for RTP data. The source and destination PSD for port 6970 are shown in Fig 2.15 (a) and (b) respectively. In the collected data, there were many packets in the range 601-682 bytes destined to port 6970 resulting in a peak at the 41st bin in the destination PSD figure. Especially, there were many packets of 652 bytes size, which could be due to the underlying protocols or due to the requirement of the RTP application using that port.

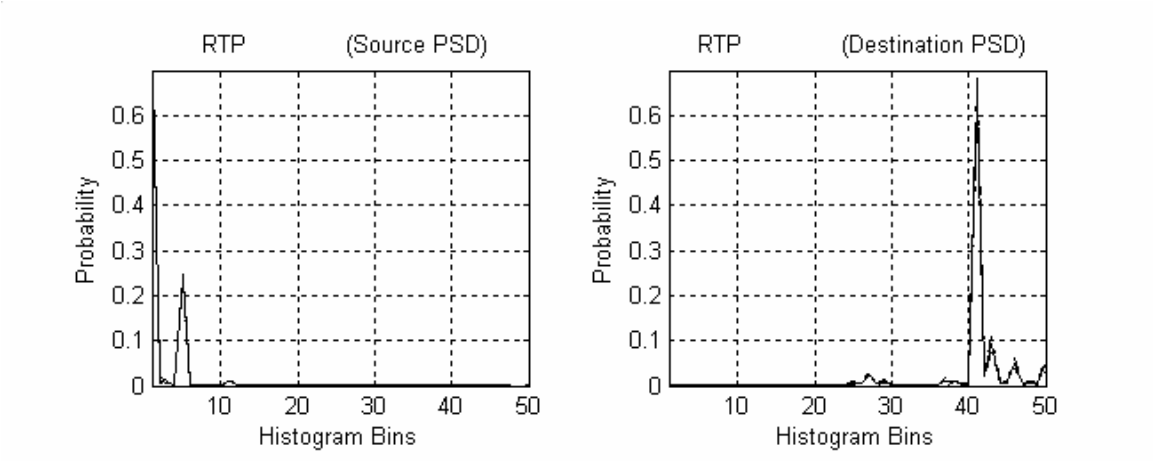



**Figure 2.15 (a) RTP Source Port PSD (b) RTP Destination Port PSD**

- *Napster*

Napster is a centralized peer-to-peer file sharing system, which means that Napster provides a server to the clients that organizes and allows access to all MP3 files that may be on other Napster client computers. Napster uses a server to keep a file index and handle client search requests but once the required file is located, the clients download files directly from its peers holding the file, called hosts. After the file is downloaded, the

host computer breaks the connection with the client. When clients connect to a server for the first time, they send a list of the files that they're serving up to the server; these files are added to the server's index. Napster uses TCP for client to server communication. The client wishing to download a file makes a TCP connection to the remote Napster client holding the file on their TCP port 6699, 6700 or 6701. In order to allow the users to bypass firewalls that allow only well-known traffic to pass through, Napster also uses well-known ports 21, 23 and 80 associated with FTP, Telnet and HTTP respectively. Since port 6699 is associated with remote Napster client from which file is downloaded, and due to the operation of the Napster, it receives several acknowledgements from the server and the Napster client downloading the files. Thus, the destination PSD has a peak in 1st bin. The source and destination PSD for port 6699 are shown in Fig 2.16 (a) and (b) respectively.

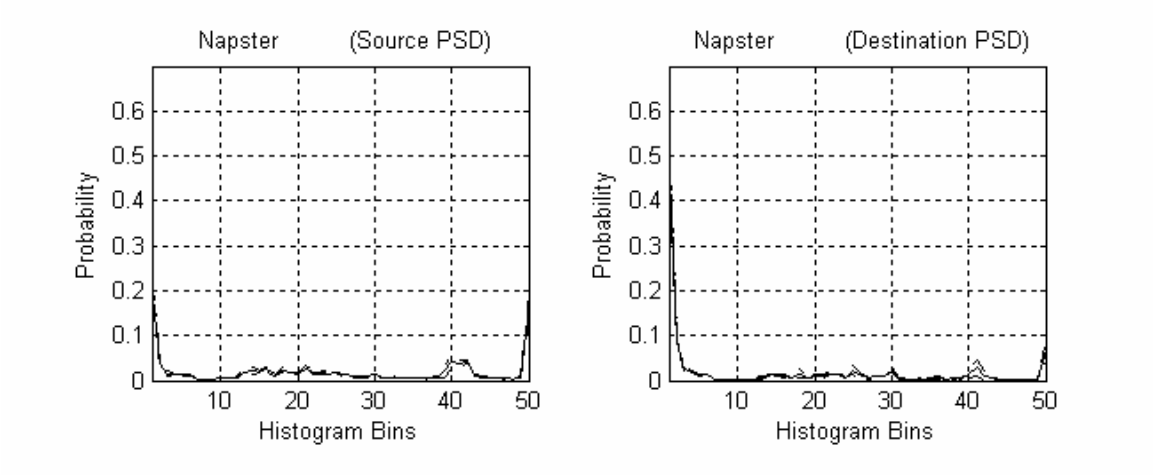



**Figure 2.16 (a) Napster Source Port PSD (b) Napster Destination Port PSD**

- *eDonkey*

eDonkey uses a distributed file-sharing network instead of one central server. Every client on the network is linked with every other client, possibly through a variety of intermediate clients. eDonkey uses Multisource File Transfer Protocol (MFTP) to download the same file from several sources concurrently, resulting in a combined higher speed. There are two applications that work together to create an eDonkey network, the client and the server. The eDonkey client is what people use to share and download files. The eDonkey server is what the clients connect to in order to search and find other users to download from. The eDonkey client searches the network for users sharing the file in question, and downloads different chunks of the file from each of them. The chunks are then combined to recreate the original file. Chunks are simply 9500KB portions of the file (for files smaller than that, or for the last chunk of a file, smaller chunks are used). eDonkey can use both TCP and UDP as transport protocols. It generally uses port 4661, 4662 or 4665. The operation of eDonkey is similar to Kazaa and Gnutella resulting in packet size distribution similar to these applications. It is seen from the clustering results in Chapter 5 that Kazaa and eDonkey fall in the same cluster for 50-bin histogram scheme and eDonkey and Gnutella fall in the same cluster for 60-bin histogram scheme. The source and destination port PSD for port 4662 are shown in Fig 2.17 (a) and (b) respectively.

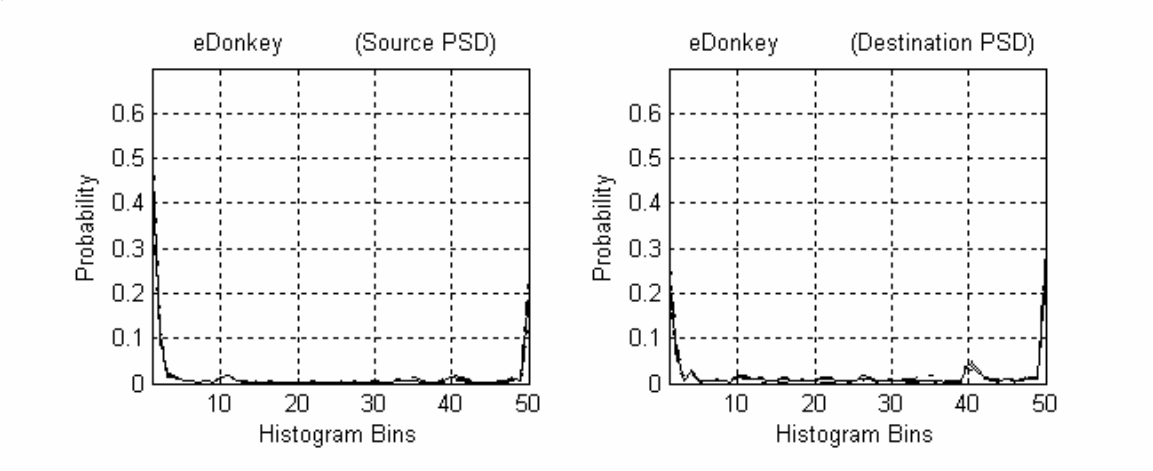



**Figure 2.17 (a) eDonkey Source Port PSD (b) eDonkey Destination Port PSD**

- *AOL (America Online)*

America Online (AOL) servers generally run on TCP port 5190. It provides a variety of services such as instant messaging referred to as AIM (America Online's Instant Messenger), e-mail services, AOL browser and personalized web page among others. Typically, upon connecting to America Online, the client first sends an identification packet that includes the version of the AOL client being used. On success, the server will send back a go-ahead packet and then the client will send Screenname/Password. On failure, the AOL server will disconnect the client. Port 5190 is also used by Apple's iChat, which has a built-in compatibility with AIM. The source and destination PSD for port 5190 is shown in Fig 2.18 (a) and (b) respectively.

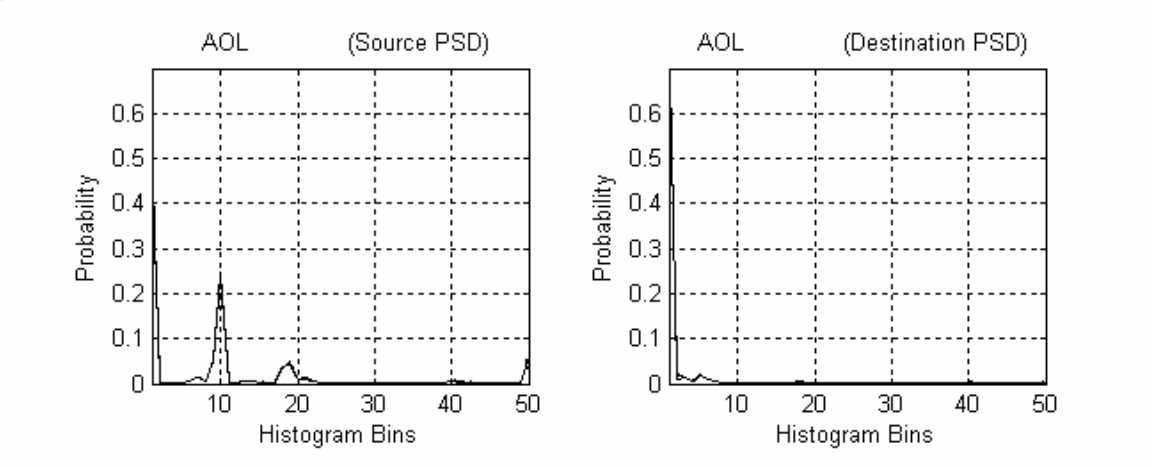



**Figure 2.18 (a) AOL Source Port PSD** **(b) AOL Destination Port PSD**

- *Multicast*

Multicast is a bandwidth-conserving technology that reduces traffic by simultaneously delivering a single stream of information to thousands of users who have subscribed to receive the multicast stream instead of sending an individual copy of the stream to the users. Applications that take advantage of multicast technologies include video conferencing, corporate communications, distance learning, and distribution of software, stock quotes, and news. The multicast streams are destined to IP addresses in the Class D [3] address range i.e. 224.0.0.0 to 239.255.255.255, which is reserved for multicast. When a host wishes to join a group, it sends an IGMP (Internet Group Management Protocol) message to the multicast router, indicating the sessions it wishes to receive. The multicast router begins broadcasting the sessions requested to the member's subnet, and the member adds the group ID address to its interface to begin reception. The source and destination PSD of port 16384 is shown in Fig 2.19 (a) and (b) respectively. The destination PSD of multicast has peaks in 3$^{rd}$, 17$^{th}$ and 42$^{nd}$ bin due to packets of size 74,

214 and 694 bytes. These peaks may be due to certain applications on the NC State backbone network that use Multicast and transmit packets of certain size only.

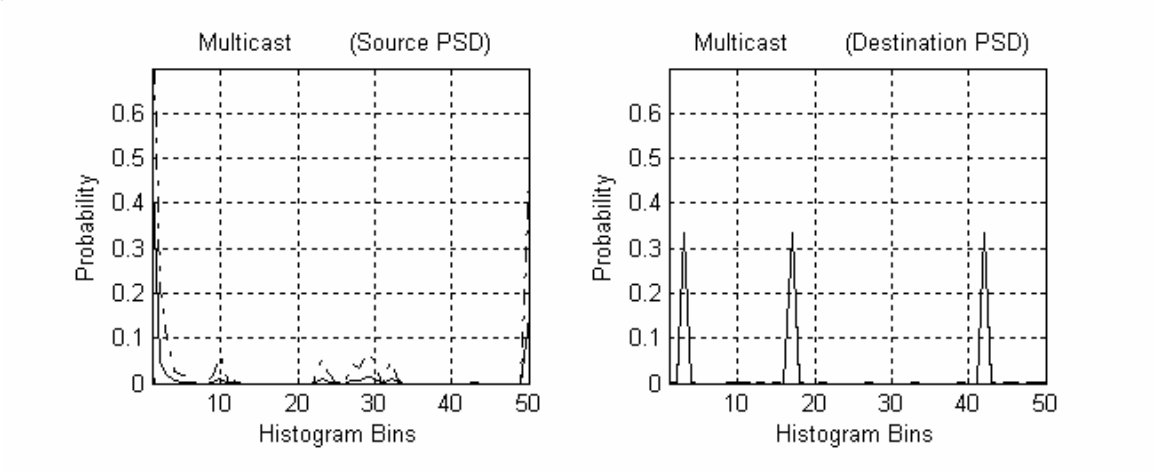



**Figure 2.19 (a) Multicast Source Port PSD (b) Multicast Destination Port PSD**

- *Half Life Server*

Port 27015 is most commonly used by Half Life server, which hosts online games such as Half Life and Counterstrike. These are the most popular online games developed by Valve. Online games require low-latency point-to-point communication as well as directed broadcast to facilitate its real-time game logic. Hence it uses small UDP packets, usually less than 512 bytes, to communicate between clients and servers as UDP is a less expensive and bandwidth intensive protocol due to its connection-less property and low overhead. Moreover, the clients typically send packets at an interval that is much shorter than the time it would take to retransmit lost packets [7]. Packets are small since the application requires extremely low latencies, which make message aggregation

impractical. The client datagram packets i.e. the packets received by the Half Life Server contains information such as co-ordinates of player's position, player status changes, etc. As shown in Figure 2.20 (b), it receives packets in the range 0-105 bytes (corresponding to 1 to 9 bins) with a peak in the range 76-80 bytes (corresponding to bin 2). The Half Life Server broadcasts the co-ordinates of other players on the current map, server status changes, etc. It mostly sends packets in the range 105-345 bytes (corresponding to 9 to 25 bins) with a peak in the range 106-120 bytes (corresponding to bin 10). The source and destination PSD for port 27015 is shown in Fig 2.20 (a) and (b) respectively.

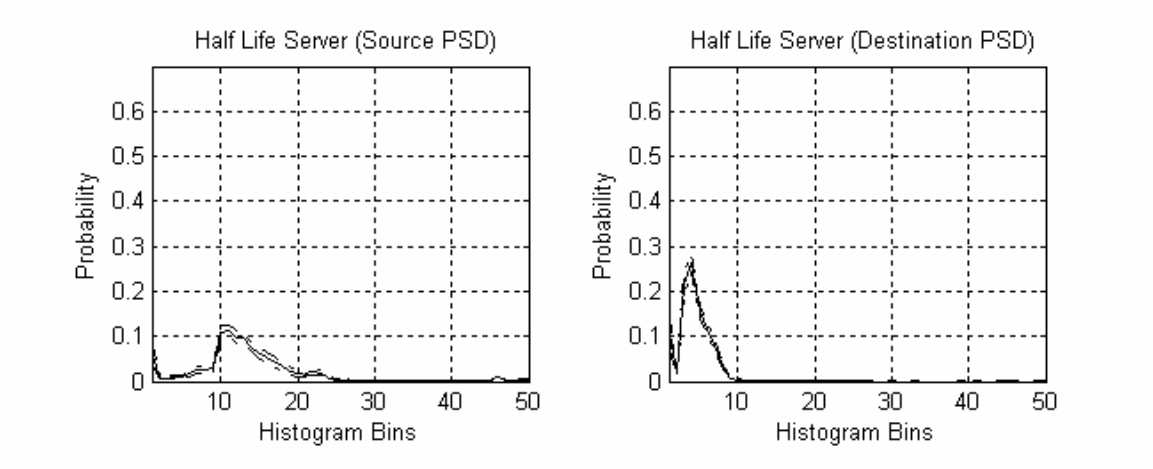



**Figure 2.20 (a) Half Life Server Source Port PSD (b) Half Life Server Destination Port PSD**

- *Plethora*

Plethora refers to a software program developed by Plethora Technology [10], which delivers a comprehensive remote access solution. It provides a secure virtual workspace which is a connected business computing platform combining secure remote access and

VPN (Virtual Private Network) technology with secure real-time communication, like instant messaging, and collaboration features. Some other communications features Plethora has added are application sharing, voice conferencing, and an electronic whiteboard. It sets up a virtual Peer-to-Peer (P2P) environment, while utilizing a physical client-server infrastructure. This means the employees get simple connectivity between any Internet-connected PC, and other users and corporate resources, while the business maintains control at the server, behind its firewall [10]. It has registered port 3480 with IANA [8] for this application and can use either TCP or UDP at the transport layer. The source and destination PSD for port 3480 are shown in Fig 2.21 (a) and (b) respectively.

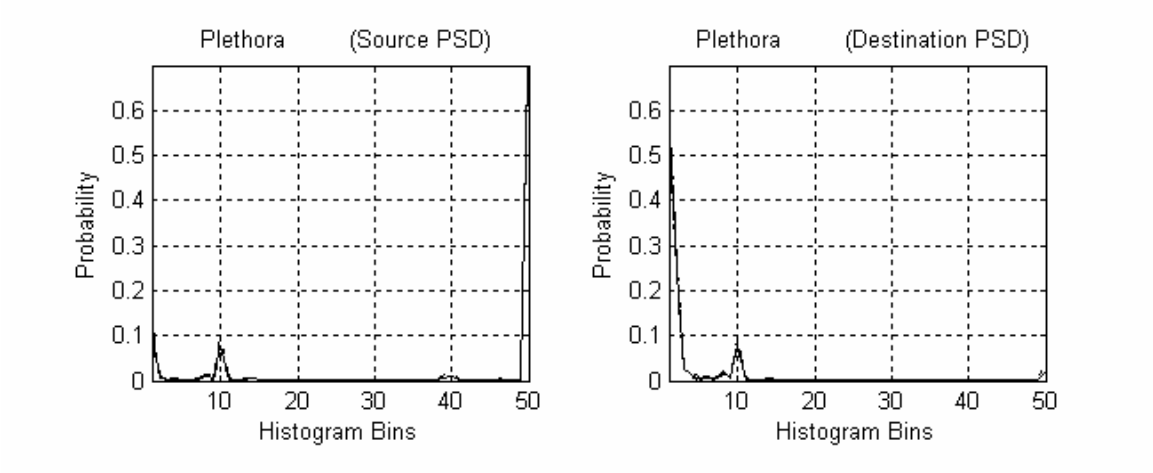



**Figure 2.21 (a) Plethora Source Port PSD (b) Plethora Destination Port PSD**

- *Reserved*

Port 0 is a reserved port in TCP/IP networking, meaning that it should not be used for any TCP or UDP network communications. However, port 0 sometimes takes on a special meaning in network programming, particularly Unix socket programming. In this

environment, port 0 is a programming technique for specifying system-allocated (dynamic) ports. Instead of "hard-coding" a particular port number, or writing code that searches for an open port, Unix programmers simply specify port 0 as a connection parameter. That triggers the operating system to automatically search for and return the next available port in the dynamic port number range. The source and destination PSD for port 0 are shown in Fig 2.22 (a) and (b) respectively.

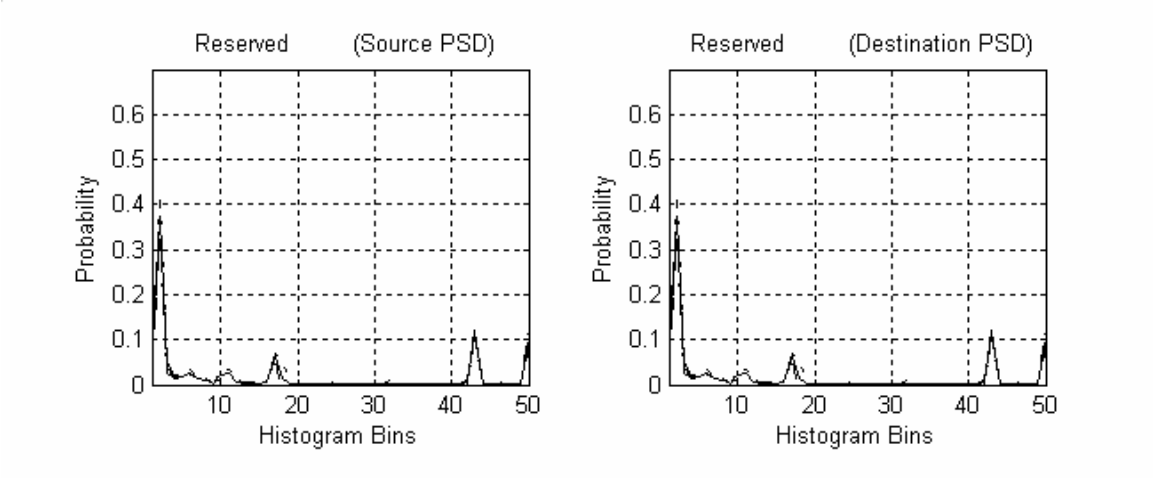



**Figure 2.22 (a) Reserved Source Port PSD (b) Reserved Destination Port PSD**

- *IRC (Internet Relay Chat)*

IRC or Internet Relay Chat is a text-based TCP/IP protocol with the simplest client being any socket program capable of connecting to the server. IRC is a multi-user chat system, where people meet on “channels” (rooms, virtual places, usually with a certain topic of conversation) to talk in groups, or privately. There is no restriction to the number of people that can participate in a given discussion, or the number of channels that can be formed on IRC. The user runs a client program, which connects to an IRC server,

typically on port 6667, in an IRC network. The server forms the backbone of IRC, providing a point to which clients may connect to talk to each other, and a point for other servers to connect to, forming an IRC network. The source and destination PSD for port 6667 are shown in Fig 2.23 (a) and (b) respectively.

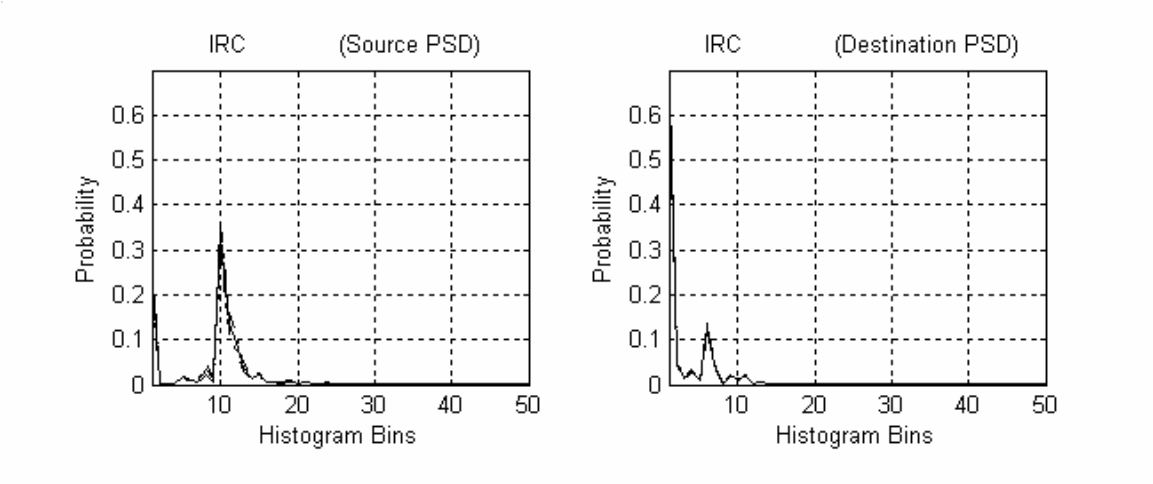



**Figure 2.23 (a) IRC Source Port PSD** **(b) IRC Destination Port PSD**

- *MS-OLAP (Microsoft Online Analytical Processing)*

MS-OLAP stands for Microsoft Online Analytical Processing and it refers to a category of software tools that provides analysis of data stored in a database. OLAP tools enable users to analyze different dimensions of multidimensional data. OLAP is often used in data mining. The chief component of OLAP is the OLAP server, which sits between a client and database management systems (DBMS) and uses ports 2393 and 2394. The OLAP server understands how data is organized in the database and has special functions for analyzing the data. OLAP tools are the foundation for a range of essential business

applications, including sales and marketing analysis, planning, budgeting, statutory consolidation, profitability analysis, balanced scorecard, performance measurement and data warehouse reporting. The source and destination PSD for port 2394 are shown in Fig 2.24 (a) and (b) respectively.

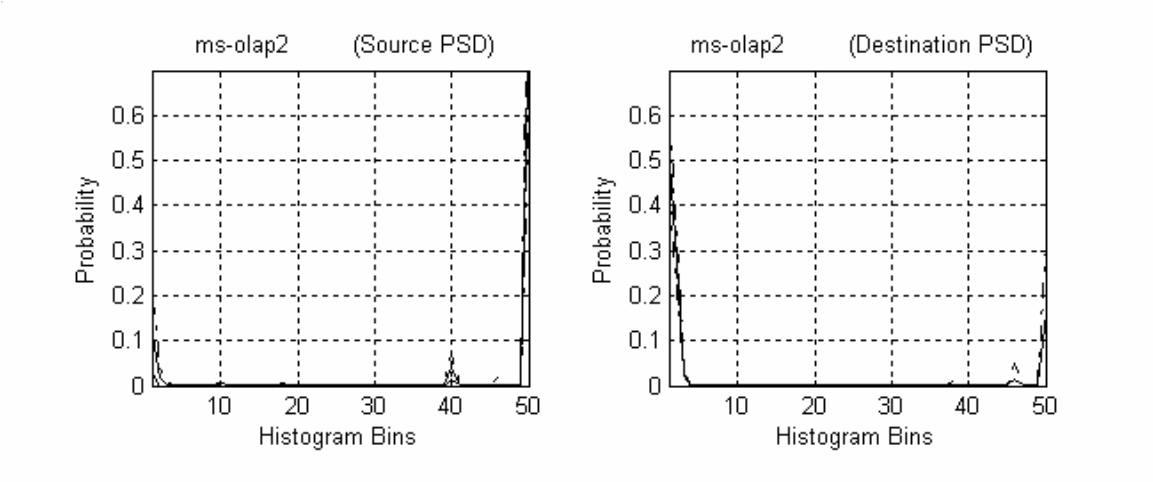



**Figure 2.24 a) MS-OLAP Source Port PSD b) MS-OLAP Destination Port PSD**

- *SMTP (Simple Mail Transfer Protocol)*

Simple Mail Transfer Protocol (SMTP) is Internet's standard host-to-host mail transfer protocol whose objective is to transfer e-mail reliably and efficiently. Traditionally, it operates over TCP port 25. When the user sends a mail request, the sender-SMTP establishes a two-way transmission channel to a receiver-SMTP. The receiver-SMTP may be either the ultimate destination or an intermediate. SMTP commands are generated by the sender-SMTP and sent to the receiver-SMTP. SMTP replies are sent from the receiver-SMTP to the sender-SMTP in response to the commands. The source and destination PSD for port 25 are shown in Fig 2.25 (a) and (b) respectively.

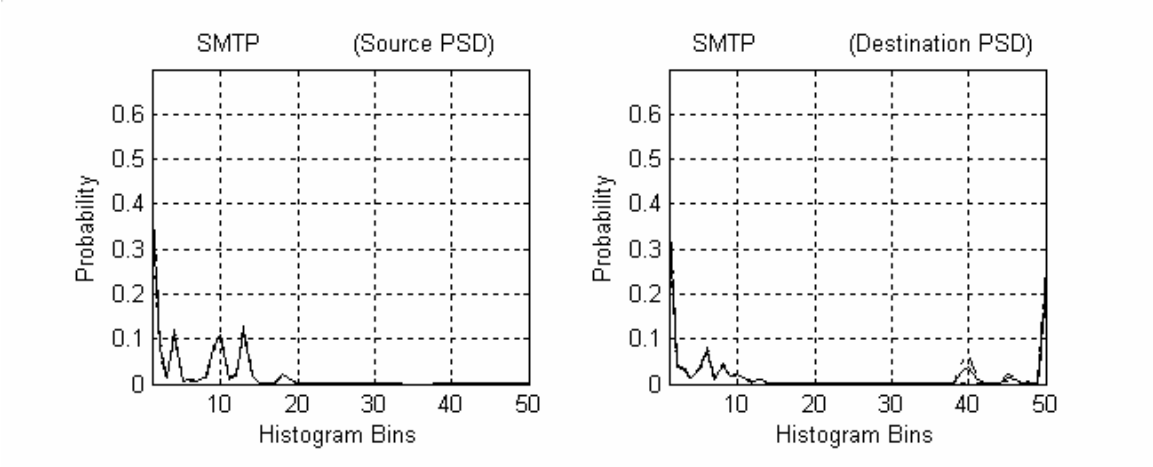



**Figure 2.25 (a) SMTP Source Port PSD (b) SMTP Destination Port PSD**

- *Half Life Client*

Half Life clients most commonly use port 27005 to connect to the Half Life server, which most commonly uses port 27015. As discussed earlier, Half Life clients send small UDP packets to the server that contains information such as co-ordinates of player's position, player status changes, etc. and it receives the broadcast packets from the server that contain the co-ordinates of other players on the current map, server status changes etc. The source and destination PSD for port 27005 is shown in Fig 2.26 (a) and (b) respectively. Since the characteristics of port 27005 and 27015 are complimentary to each other, the source PSD of port 27005 is similar to the destination PSD of port 27015 and vice versa. The port 27005 is also used by a software tool called FLEXlm, which is a popular license manager used in the software industry. FLEXlm is best known for its ability to allow software licenses to be available (or float) anywhere on a network, instead of being tied to specific machines.

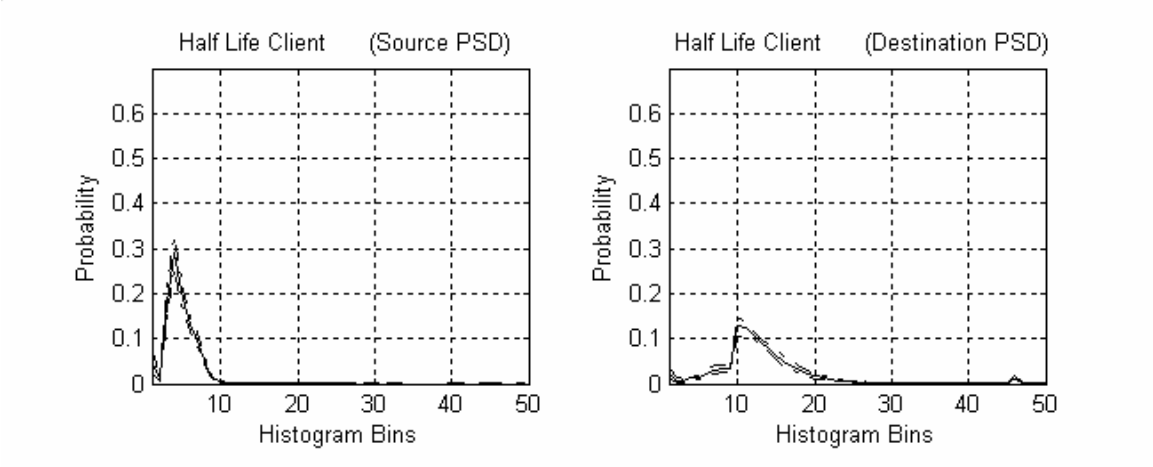



**Figure 2.26 (a) Half Life Client Source Port PSD (b) Half Life Client Destination Port PSD**

- *ICAP (Internet Content Adaptation Protocol)*

ICAP, the Internet Content Adaptation Protocol, is a protocol designed to off-load specific Internet-based content to dedicated servers, thereby freeing up resources and standardizing the way in which features are implemented. These ICAP servers are focused on a specific function, for example, ad insertion, virus scanning, content translation, language translation, or content filtering. A server that handles only language translation is inherently more efficient than any standard web server performing many additional tasks. ICAP is a request/response protocol similar in semantics and usage to HTTP. Despite the similarity, ICAP is not HTTP, nor is it an application protocol that runs over HTTP. ICAP uses TCP/IP as a transport protocol. The default port is 1344, but other ports may be used [11]. The ICAP clients pass HTTP requests to passively listening ICAP servers for some sort of transformation or other processing. The server executes its transformation service on messages and sends back HTTP responses to the client, usually

with modified messages [11]. The source and destination PSD of port 1344 are shown in Fig 2.27 (a) and (b) respectively.

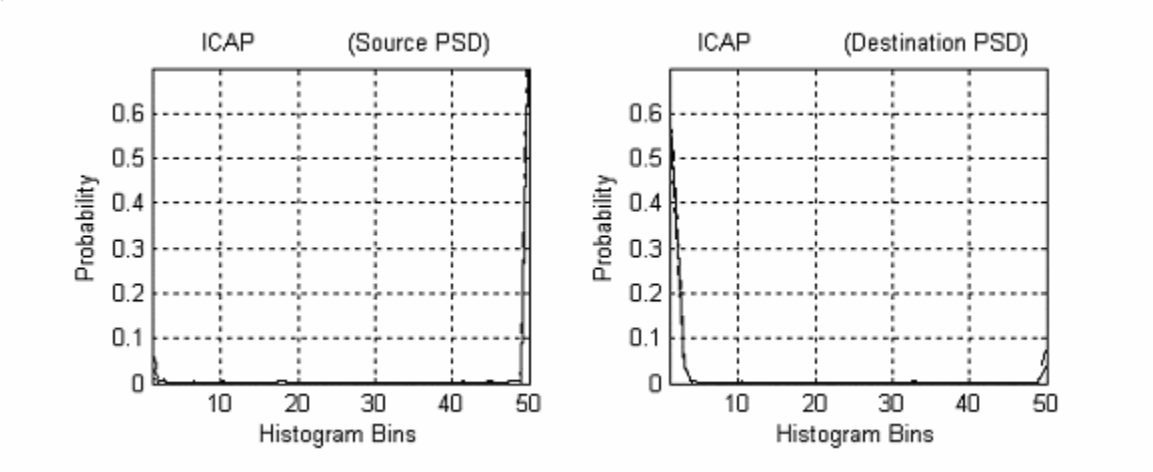



**Figure 2.27 (a) ICAP Source Port PSD** **(b) ICAP Destination Port PSD**

- *Tragic*

Port 2642 is registered with IANA [8] for an application called tragic. But apart from this there is not a lot of information on the Internet regarding the function of the application. The source and destination PSD for port 2642 are shown in Fig 2.28 (a) and (b) respectively. From the PSD plots, the application 'tragic' sends many large data packets and receives small packets, possibly corresponding to acknowledgements or other control packets.

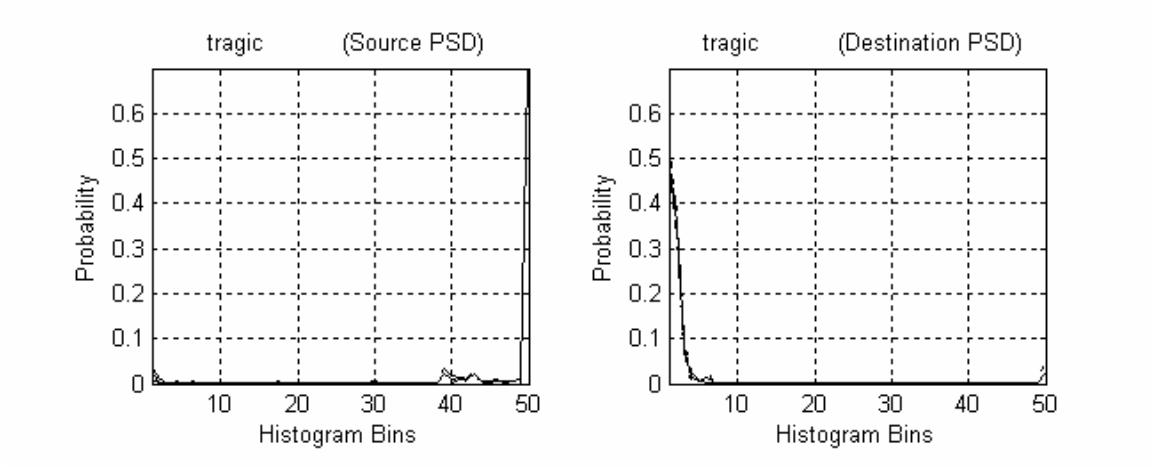



**Figure 2.28 (a) Tragic Source Port PSD (b) Tragic Destination Port PSD**

- *mloadd*

Port 1427 is registered with IANA [8] for an application called mloadd, which is a monitoring tool. But apart from this there is not a lot of information on the Internet regarding the function of the application. The source and destination PSD for port 2642 are shown in Fig 2.29 (a) and (b) respectively. From the PSD plots, the application 'mloadd' sends many large data packets and receives small packets, possibly corresponding to acknowledgements or other control packets.

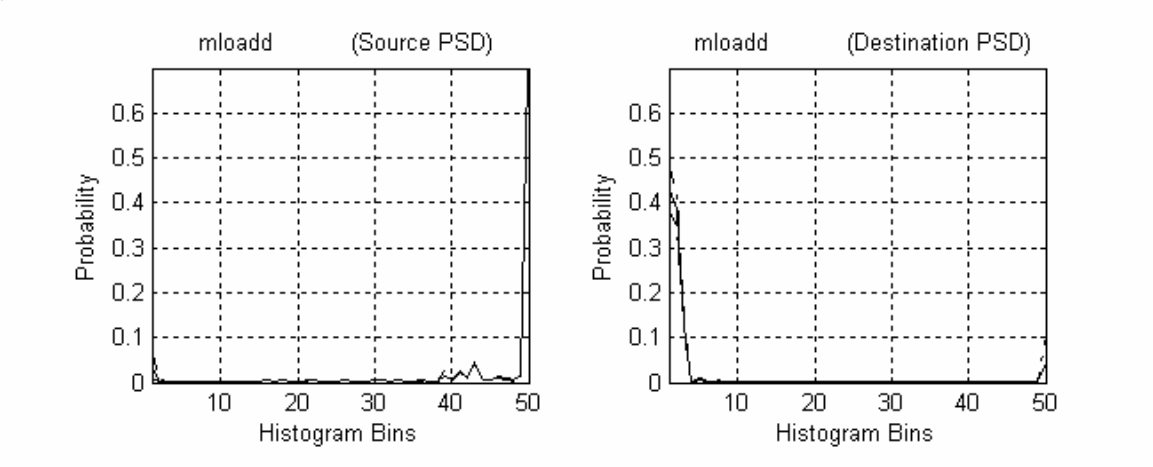



**Figure 2.29 (a) mloadd Source Port PSD (b) mloadd Destination Port PSD**

## *C. Use of Port Numbers*

A *port* is a logical channel or channel endpoint in a communications system. IP is concerned strictly with getting messages from one machine to another. But with multiple users/programs on machines, we need a more selective delivery mechanism to distinguish between different logical channels on the same network interface on the same computer. The transport layer protocols, TCP and UDP, identify applications using 16-bit port numbers; hence there can be 65536 ports [4]. Since we are collecting the source and the destination port numbers of the packet from the network, we can identify the application that the packet belongs to in the collected data. The destination port number of the packet often indicates the type of service accessed.

Port numbers are divided into three ranges [8]:

- The *Well-Known Ports* are those from 0 through 1023. These are tightly bound to services, and usually traffic on this port clearly indicates the protocol for that service. For example, port 80 virtually always indicates HTTP traffic.
- The *Registered Ports* are those from 1024 through 49151. These are loosely bound to services, which means that while there are numerous services "bound" to these ports, these ports may be used for many other purposes that have nothing to do with the official server. For example, port 27005 is registered with IANA for an application called FLEXlm but this port is also widely used by Half Life clients to connect to the servers. Registered ports are used by ordinary user processes or programs executed by ordinary users.
- The *Dynamic and/or Private Ports* are those from 49152 through 65535. Dynamic ports are not managed by any governing body like IANA and have no special usage restrictions. When an application does not bind a specific port number to its socket the TCP stack will assign the socket a port number from the dynamic port number range.

Servers are normally known by their well-known port number. For example, every TCP/IP implementation that provides an FTP service provides that service on TCP port 21. Every Telnet server is on port 23. Every implementation of TFTP (the Trivial File Transfer Protocol) is on UDP port 69. These programs run on the servers at the assigned ports waiting for the network messages requesting services. Those services that can be provided by any implementation of TCP/IP have well-known port numbers between 1 and 1023. The well-known port numbers are managed by the Internet Assigned Numbers Authority (IANA) [8].

A client usually doesn't care what port number it uses on its end. All it needs to be certain of is that whatever port number it uses is unique on its host. Client port numbers are called ephemeral ports (i.e. short lived). This is because a client typically exists only as long as the host is active. IANA has allotted dynamic port range for ephemeral use.

When we see a packet destined to port 80, it usually means a packet sent by the client to the server running an HTTP application and the packet originating from port 80 usually implies packet sent by the server to the client. Depending on the application a server runs, it might send out more packets than it receives or it might usually send out large data packets but might receive small acknowledgements. For example, the HTTP application at port 80 of the server usually needs to send out data packets of about 1514 bytes and it receives small acknowledgement packets of about 60 bytes from the client. Also, it sends out many data packets and may receive one acknowledgement for multiple packets. Hence there are more source packets as compared to the destination packets in Table 2.2. The Multicast application at port 16384 sends out multicast packets but does not receive any packets from client as seen in Table 2.2. The small number of destination packets might be due to port scanning or 16384 might be an ephemeral port for the client for some application.

Since each application and hence the associated port number has significantly different PSD, the applications can be identified using the port numbers. But since the port numbers normally assigned to an application or program may also receive the packets not

belonging to that application and it is not necessary that the user follow the port assignments set by IANA, we may not be able to capture the characteristic of that application alone. For example, many Trojan Horses and viruses are sent to the well-known port numbers to decrease the probability of detection. Moreover, the registered ports (1024-49151) are loosely bound to the applications and hence, there may be instances of applications that are missed because they are using a non-standard port number.

# 3. Clustering of Applications

The first section in this chapter defines clustering and discusses various terms related to it. It also discusses the reasons for using the clustering techniques in this research. The subsequent sections in this chapter describe the steps involved in a clustering process in general. Section C describes one step in the clustering process, which involves choosing the distance metric and the clustering technique for performing clustering. Lastly, Section D discusses some of the problems encountered in the clustering process.

## *A. Heuristics*

*Cluster analysis* is an analytical technique whose primary purpose is to group objects or patterns (usually represented as a vector of measurements or a point in a multidimensional space) into a small number of mutually exclusive groups based on their similarity. Intuitively, the resulting clusters of patterns should then exhibit high internal (within-cluster) homogeneity and high external (between-cluster) heterogeneity [12]. Thus, if classification is successful, the objects within clusters will be close together when plotted geometrically, and different clusters will be far apart as shown in Figure 3.1.

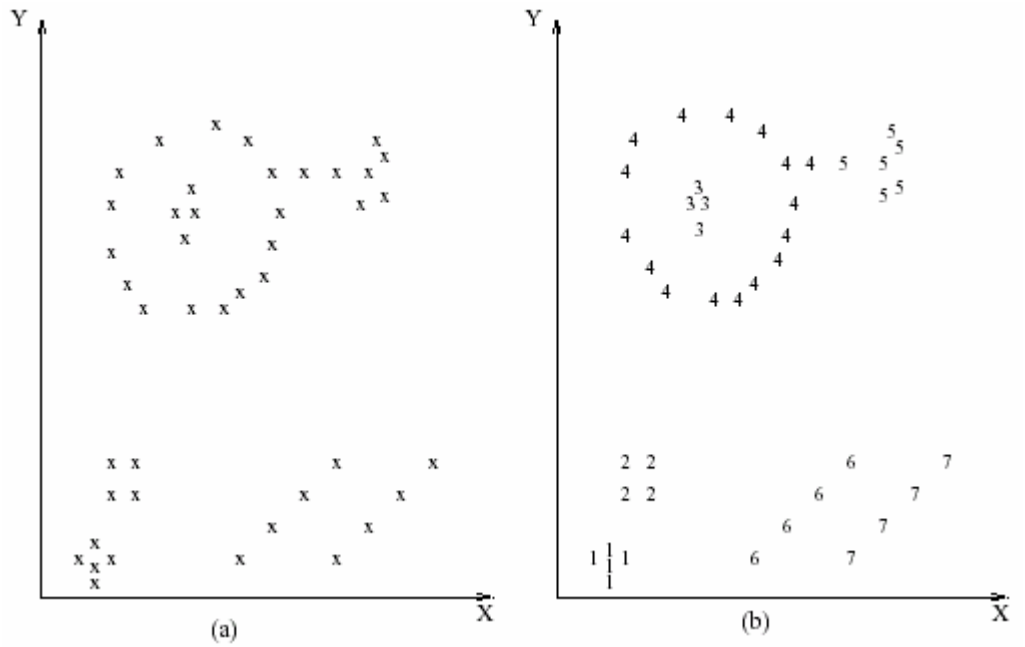



**Figure 3.1 Data Clustering**

Cluster analysis is appropriate for analyzing data that is described by many features and patterns, which results in high dimensionality of the data. The dimensionality of the data can be decreased if a subset of patterns or features that "best" represent the data are considered instead of all patterns or features [13].

Clustering is a special kind of classification. Figure 3.2 shows the relationship between clustering and classification and the terms are explained below [14]:

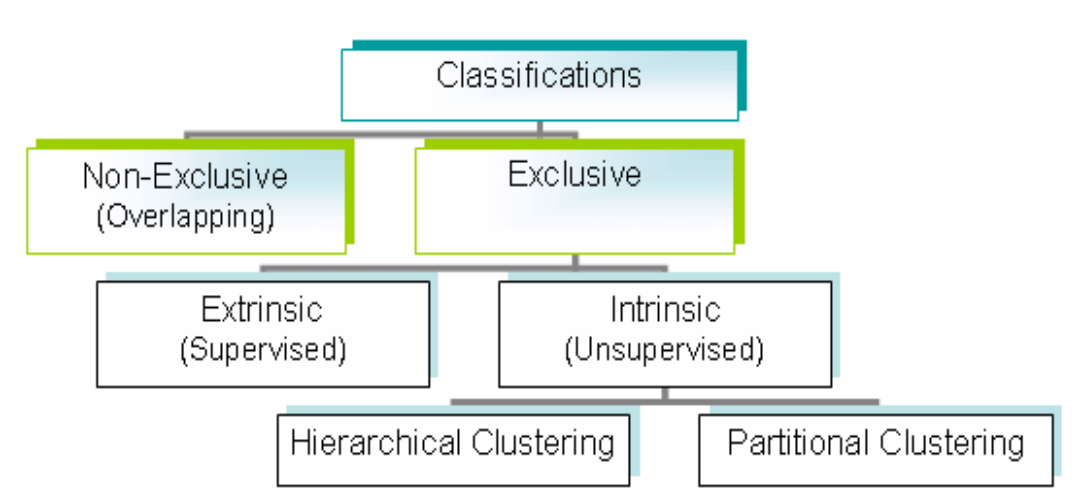



**Figure 3.2 Tree of classification types**

a. *Exclusive versus nonexclusive:* In exclusive classification, each object belongs to exactly one cluster. Nonexclusive, or overlapping, classification can assign an object to more than one cluster. Fuzzy clustering is a type of non-exclusive classification.
b. *Intrinsic versus extrinsic:* An intrinsic classification uses only the proximity measure i.e. the distance measures between pairs of patterns to perform the

classification. Intrinsic classification is also called "unsupervised learning" because the classes in which the data are to be grouped are also to be defined, although the number of classes may be set. In this case, the problem is to group a given collection of unlabeled patterns into meaningful clusters; it is entirely data driven. Extrinsic classification also referred to as "supervised learning," uses pre-defined classes on the patterns as well as the proximity measure. In supervised classification, we are provided with a collection of preclassified patterns that are used to learn the descriptions of classes, which in turn are used to classify a new pattern.

c. *Hierarchical versus partitional:* Exclusive, intrinsic classifications are subdivided into hierarchical and partitional classifications by the type of structure imposed on the data. Hierarchical clustering proceeds successively by either merging smaller clusters into larger ones, or by splitting larger clusters resulting in a nested sequence of partitions. Partitional clustering attempts to directly decompose the data set into a set of disjoint clusters resulting in a single partition. These methods of clustering will be explained in more detail in the next section.

A *cluster* is comprised of a number of similar objects collected or grouped together. Several definitions of a cluster have been proposed, depending on its application. Everitt [15] documents some of the following definitions of a cluster:

- A cluster is a set of entities which are alike, and entities from different clusters are not alike.

- A cluster is an aggregation of points in the test space such that the distance between any two points in the cluster is less than the distance between any point in the cluster and any point not in it.
- Clusters may be described as connected regions of a multi-dimensional space containing a relatively high density of points, separated from other such regions by a region containing relatively low density of points.

The definition of a cluster and the goals of clustering varies from case to case due to the fact that objects can be grouped into clusters with different purposes and goals in mind. Once this is realised, it is easier to see why such a variety of clustering techniques exist. Depending on the implementations of the clustering algorithms, the most common goals of the clustering techniques [16] and the reasons for using the clustering technique in this research are as now discussed.

*(i) Computation reduction*

Cluster analysis can contribute in compression of the information included in data. The amount of available data is very large in several cases and its processing becomes very demanding. Clustering can be used to partition data into a number of "interesting" clusters depending upon the metrics specified by the user. Then, instead of processing the data set as an entity, we adopt the representatives of the defined clusters in our process. In this research, the network traffic data consists of many applications that are quite intractable unless classified into manageable groups. Hence, in order to simplify the processing of the data, the applications are grouped into classes based on some similarity measure between the applications. Instead of treating each application separately, we can

consider each cluster formed by similar applications as one virtual or combined application. This results in data reduction while maintaing the important information in the data since we use the mean of the cluster instead of the measurements to all the members of the cluster in the computations.

*(ii) Hypothesis Testing*

Cluster analysis is used for the verification of the validity of a specific hypothesis. After studying the characteristics of some of the applications and examining their packet size distributions, it was evident that some applications in the traffic mixture behaved similarly whereas other applications behaved quite differently. This led us to believe that it could be possible to group the similar applications and separate the dissimlar ones into clusters. We tested this hypothesis using various clustering techniques for different number of clusters.

*(iii) Hypothesis Generation*

Cluster analysis is used here in order to infer some hypothesis concerning the data. Clustering is performed on the data to identify which applications in the network mixture fall together in the same cluster, indicating similar behaviour, and which applications fall in different clusters, indicating different behaviour. Thus, with clustering, we generate the hypothesis that these applications behave similarly and can be grouped together. This hypothesis can be also tested by collecting another set of data from the network, performing clustering on it and testing if the same applications fall in one cluster again. To provide a higher degree of certainity regarding the hypothesis, we have studied the

characteristics of the application and the protocols used by the application and how it works to ascertain if indeed they behave "similarly." For example, the peer-to-peer (P2P) applications that use decentralized server, such as, Kazaa, eDonkey, Gnutella, etc., behave "similarly."

*(iv) Prediction based on groups*

Cluster analysis is applied to the data set and the resulting clusters are characterized by the features of the patterns that belong to these clusters. Then, unknown patterns can be classified into specified clusters based on their similarity to the clusters' features. We have not performed prediction based on cluster groups in this research yet, but we plan to do so at a later point in the research. The basic idea is to model the clusters by the features of the applications that belong to those clusters. These models can then be applied to the unknown application in the next interval to predict what cluster it would fall under.

## *B. Clustering Process*

The basic steps involved in a clustering process are shown in Figure 3.3 and summarized as follows [14] [16] [17]:

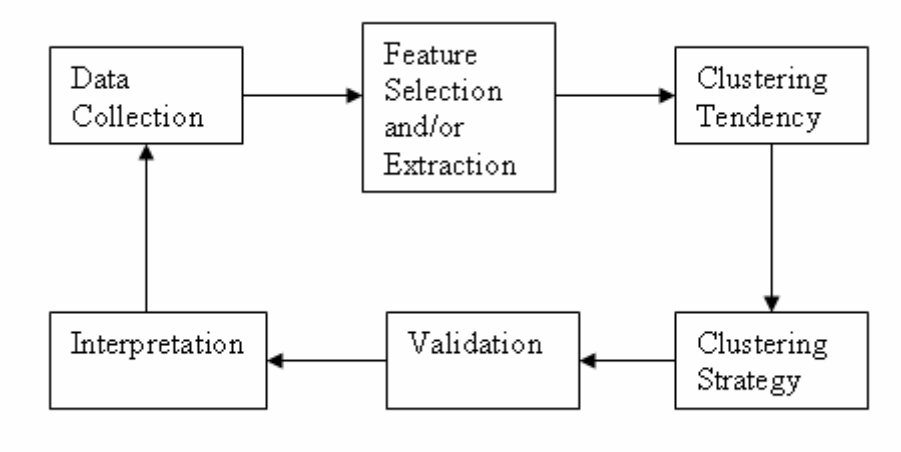



**Figure 3.3 Clustering Process**

*a. Data Collection*

The careful recording of data is very important in any exploratory data analysis. The amount and type of data will strongly influence the strategies available for analyzing the data. The results interpreted on one data set should be checked on other independent data sets for validating the results. The process of data collection, the amount of data collected and the trends in the data are explained in more detail in Chapter 4.

*b. Feature Selection and or feature extraction*

Feature selection is the process of identifying the most effective subset of the original features to use in clustering, so as to encode as much information as possible concerning the task of our interest. Feature extraction is the use of one or more transformations of the input features to produce new salient features. Either one or both of feature selection and feature extraction techniques can be used to obtain an appropriate set of features to use in clustering. Preprocessing of data may be necessary prior to their utilization in the

clustering task. As seen from the collected data format in Chapter 4, we are collecting a lot of information but only packet size and the port numbers are used to generate meaningful results. Port numbers are used to identify the application and the packet sizes are used to generate the packet size distributions for the applications of interest. The histograms of the packet size distributions of the applications are used as input to the clustering algorithms in order to identify the similarities and dissimilarities between applications. The histogram generation is explained in more detail in Chapter 4.

### *c. Clustering Tendency*

All clustering algorithms, when presented with data, produce clusters - regardless of whether the data contains clusters or not. Clustering tendency refers to the problem of deciding whether the data exhibits the tendency to fall into natural groups. The information gained from this step can not only prevent the inappropriate application of clustering algorithms, but can also provide information on the fundamental nature of the data. In order to prove that the different applications in the Internet traffic contain some similarities so that they can be grouped together, we have studied characteristics of the applications, as discussed in Chapter 2, and have recorded their packet size distributions, as shown in Chapter 2. There are several applications that have similar characteristics but use different application protocols such as FTP (File Transfer Protocol), MFTP (Multiple File Transfer Protocol), etc., depending upon the specific requirements. Hence, we see several different applications in the Internet traffic whose behavior is similar. From the packet size distributions of the applications, it is evident that there exists similarities in the behavior of some applications and hence they can be grouped into clusters.

*d. Clustering Strategy*

This step refers to the choice of an algorithm that results in the definition of a good clustering scheme for a data set. The pattern proximity measure and the clustering algorithm mainly characterize the clustering strategy as well as its efficiency to define a clustering scheme that fits the data. The next section discusses several clustering algorithms and the clustering criteria.

*i. Pattern Proximity Measure*

Proximity measure is a measure that quantifies how "similar" two data points are. It is usually measured by a distance function, such as Euclidean distance, defined on pairs of patterns.

*ii. Clustering Algorithm (Clustering or grouping)*

Having decided how to measure similarity (the distance measure) we must now choose the clustering algorithm, i.e. the rules which govern how distances are measured between clusters. The clustering algorithm can be hard or fuzzy, hierarchical or partitional, probabilistic or graph-theoretic. Processing a fixed data set with different cluster methods may result in a large number of different partitions of clusters. These partitions should be compared with each other to find the "most valid" or the "best" of them [13]. In some clustering algorithms, it is necessary to determine the number of clusters into which the data needs to be grouped. Since there is no particular method for determing the number of clusters, and it is not evident from looking at the data, we will perform clustering with variable number of clusters and use the most appropriate and best result determined using certain criteria discussed in Section D in this chapter.

*e. Validation of the results*

Since clustering algorithms define clusters that are not known a priori, irrespective of the clustering methods, the final partition of data requires some kind of evaluation in most applications. Cluster validity analysis is the assessment of the correctness of clustering algorithm results using appropriate criteria and techniques. A clustering structure is valid if it cannot reasonably have occurred by chance or as an artifact of a clustering algorithm. When statistical approaches to clustering are used, validation is accomplished by carefully applying statistical methods and testing hypotheses. The stability of the analysis can also be studied by perturbing the data slightly and repeating the analysis. In order to validate the clustering results, we have performed the entire clustering process with another independent data set and compared the results obtained from both the cases as discussed in Chapter 5. We also generated one of the applications artificially, as shown in Chapter 6, and checked to see whether we get similar results as those obtained earlier to validate the stability of the analysis.

*f. Interpretation of the results*

There is no standard method for interpreting the results obtained from various cluster analysis techniques and integrating the results with previous studies. In many cases, the experts in the application area have to integrate the clustering results with other experimental evidence and analysis, in order to draw the right conclusion. The results obtained from clustering and its interpretation will be explained in more detail in Chapter 5.

## *C. Clustering Techniques*

Several clustering methods have been proposed depending upon the goals of data classification. The clustering algorithms can be classified according to [16]:

1. the approach used for clustering the data
2. the similarity measure between the data points in a cluster or dissimilarity measure between the data points in different clusters.
3. the number of clusters in which an object can be classified

According to the method adopted to define clusters, the algorithms can be broadly classified into the following types :

- *Partitional Clustering* attempts to directly decompose the data set into a set of disjoint clusters.
- *Hierarchical Clustering* proceeds successively by either merging smaller clusters into larger ones, or by splitting larger clusters.
- *Density-based clustering* algorithms define clusters as dense regions of patterns or objects in the data space that are separated by low density regions. These algorithms have their origins in single linkage cluster analysis, which is a hierarchical clustering technique that operates by joining the two most similar objects not yet in the same cluster, at each step. It was developed in an attempt to overcome chaining, which is the main problem of single linkage technique that results in long chains of objects in a cluster [15].

- *Grid-based clustering* quantises the space into a finite number of cells and then does the operations on the quantised space.

Each of the above categories consists of a number of different algorithms for finding the clusters. Hierarchical and partitional are the most commonly used clustering algorithms and these are discussed later in this section. According to the approach used for clustering the data, the clustering algorithms can be classified as:

- *Statistical*, which are based on statistical analysis concepts, use similarity measures to partition objects and are limited to numeric data.
- *Conceptual*, which are used to cluster categorical data and cluster objects according to the concepts they carry [18]. Conceptual clustering summarizes and organizes data allowing for better inference ability. For example, given the animal descriptions, conceptual clustering tries to group them under classes, such as mammals, birds, reptiles, etc., in a most logical manner. It can also be used to design database schemas that link the related objects depending on their concepts.
- *Kohonen net clustering,* which uses self-organizing neural networks. The Kohonen Self-Organizing Map (SOM) [19] is a feedforward neural network approach that uses an unsupervised training algorithm and through a process called self-organization, configures the output units into a topological representation of the original data. The SOM is a data visualization technique which reduces the dimensions of data by producing a map of usually 1 or 2 dimensions that plots the similarities of the data by grouping similar data items together. It uses competitive learning algorithm which ensures that the most

highly activated node (winner of the competition) as well as its neighbors move towards a sample presented to the network. The connections between input nodes and the winner node are modified, increasing the likelihood that the same winner continues to win in future for input samples similar to the one that caused the adaptation. The networks are self-organizing in that the nodes tend to attain weight vectors that capture characteristics of the input vector space, with the neighborhood relation translating into proximity in Euclidean space, even if the initial values of the weight vectors are arbitrary. In clustering, the weight vectors associated with nodes in these networks are interpreted as cluster centroids.

We have used statistical data clustering in this research. According to the number of clusters in which an object can be classified, the clustering algorithms can be classified as:

- *Fuzzy Clustering* in which an object can be classified in more than one clusters.
- *Hard or crisp clustering* in which an object strictly belongs to one cluster only.

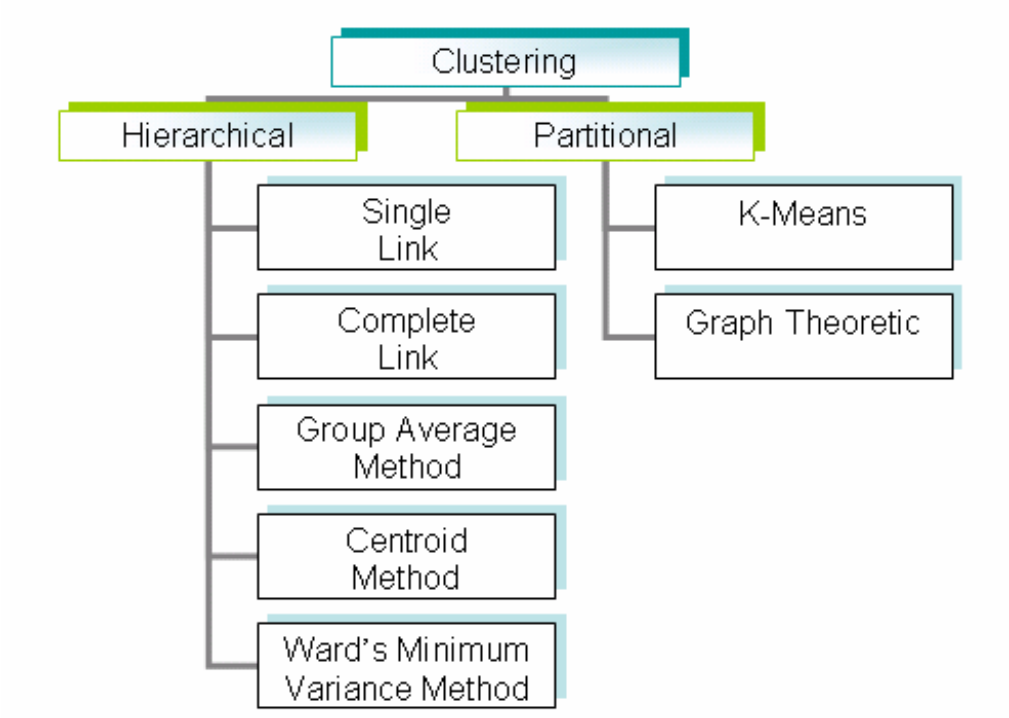



**Figure 3.4 Hierarchy of clustering algorithms**

Several approaches for clustering the data, which were considered in this work, can be categorized according to the hierarchy shown in Figure 3.4. Several algorithms can be proposed to express the same exclusive, intrinsic classification. The algorithmic options that affect different clustering techniques in Figure 3.4 are explained below [14] [17]:

1. *Agglomerative vs. divisive:* This aspect relates to algorithmic structure and operation and is associated mainly with hierarchical clustering. An agglomerative approach begins with each pattern in a distinct cluster, and successively merges clusters together until a stopping criterion is satisfied. This approach is data-driven, emphasizing similarities between patterns rather than their differences. A divisive method begins with all patterns in a single cluster and performs splitting until a stopping criterion is met. This approach is conceptually driven in that items

are discriminated along categorical boundaries and conceptual dimensions. It emphasizes the differences between patterns rather than their similarities.

2. *Monothetic vs. polythetic:* This aspect relates to the sequential or simultaneous use of features in the clustering process. In a polythetic clustering algorithm, all features are considered for computing the distances between patterns, and decisions are based on those distances. Most algorithms are polythetic. A monothetic clustering algorithm uses features sequentially to divide the given collection of patterns. As illustrated in Figure 3.5, the collection is divided into two groups by a vertical broken line V1 using feature x1. Each of these clusters is further divided independently using feature x2, as depicted by the broken lines H1 and H2. The major problem with this algorithm is that it generates 2d clusters where d is the dimensionality of the patterns. For large values of d, the number of clusters generated by this algorithm is so large that the data set is divided into uninterestingly small and fragmented clusters.

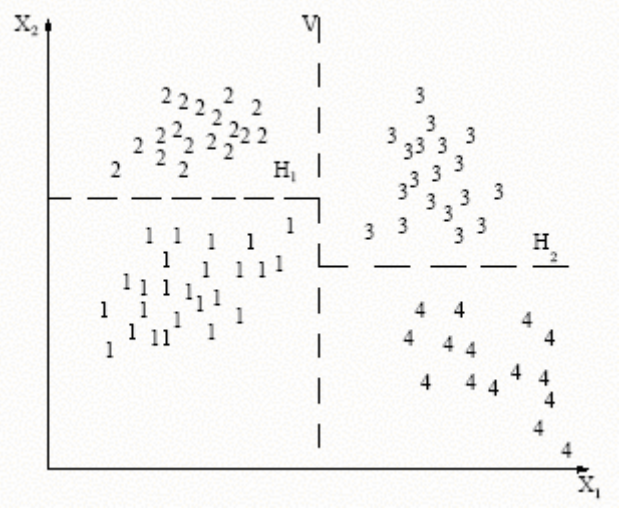



**Figure 3.5 Monothetic Partitional clustering**

3. *Hard vs. fuzzy:* A hard clustering algorithm allocates each pattern to a single cluster during its operation and in its output. A fuzzy clustering method assigns degrees of membership to each input pattern in the range [0,1] to several clusters. A fuzzy clustering can be converted to a hard clustering by assigning each pattern to the cluster with the largest measure of membership. Since each pattern may belong to more than one cluster in fuzzy clustering as shown in Figure 3.6, the results are more difficult to interpret as compared to hard clustering. The patterns denoted by '4' and '5' in the figure belong to the clusters F1 and F2 when fuzzy clustering is used. Considering the measure of membership of the pattern as the normalized distance from the cluster centroid, the patterns '4' and '5' belong to the cluster H1 when hard clustering is used. There are several models proposed to interpret the fuzzy clusters [20] but this is beyond the scope of this thesis.

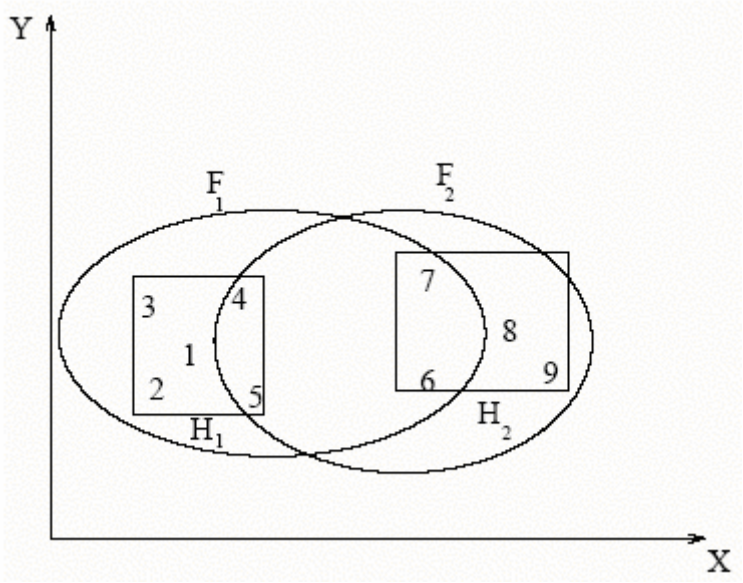



**Figure 3.6 Fuzzy Clusters**

4. *Deterministic vs. stochastic:* This issue is most relevant to partitional approaches designed to optimize a squared error function. This optimization can be accomplished using traditional techniques or through a random search of the state space consisting of all possible classifications. If all the steps in the clustering algorithm are deterministic, the method is deterministic. Some methods use randomized steps and fall in the other category. For example, the initial partitions in the partitional clustering method can be chosen randomly or it can be specified manually. The resulting clusters depend on the initial partition chosen and hence in the stochastic approach, we may not obtain the same clusters for the data set ever time.
5. *Incremental vs. non-incremental:* This issue arises when the pattern set to be clustered is large, and constraints on execution time or memory space affect the architecture of the algorithm. Large data sets have fostered the development of clustering algorithms that minimize the number of scans through the pattern set, reduce the number of patterns examined during execution, or reduce the size of data structures used in the algorithm's operations. Incremental learning algorithms complete the computations necessary to fit the respective models by processing one observation at a time, each time "refining" the solution; then, when all observations have been processed, only few additional computations are necessary to produce the final results. Non-incremental learning algorithms are those that need to process all observations in each iteration of an iterative procedure for refining a final solution. Obviously, incremental learning algorithms are usually much faster than non-incremental algorithms, and for

extremely large data sets, non-incremental algorithms may not be applicable at all (without sub-sampling first).

6. *Graph theory vs. matrix algebra:* This aspect relates to the appropriate mathematical form for expressing a clustering algorithm. Graph theory option uses properties such as connectedness and completeness to define classifications. Matrix algebra option expresses algorithms in terms of algebraic constructs, such as mean-square error. The choice of the option depends upon clarity, construct and personal choice. Some algorithms have convenient expressions under both options.

**Hierarchical Clustering Algorithms**

Hierarchical clustering proceeds successively by either merging smaller clusters into larger ones, or by splitting larger clusters. The hierarchical clustering methods differ by the metrics used to decide which two small clusters are merged or which large cluster is split. The end result of the algorithm is a tree of clusters called a dendrogram, which shows how the clusters are related. By cutting the dendrogram at a desired level, a clustering of the data items into disjoint groups is obtained [16]. The operation of the hierarchical clustering algorithm on a two-dimensional data set in Figure 3.7 (from [17]) generates three clusters {A,B,C},{D,E} and {F,G} as shown in the figure. A dendogram corresponding to this data set is shown in Figure 3.8.

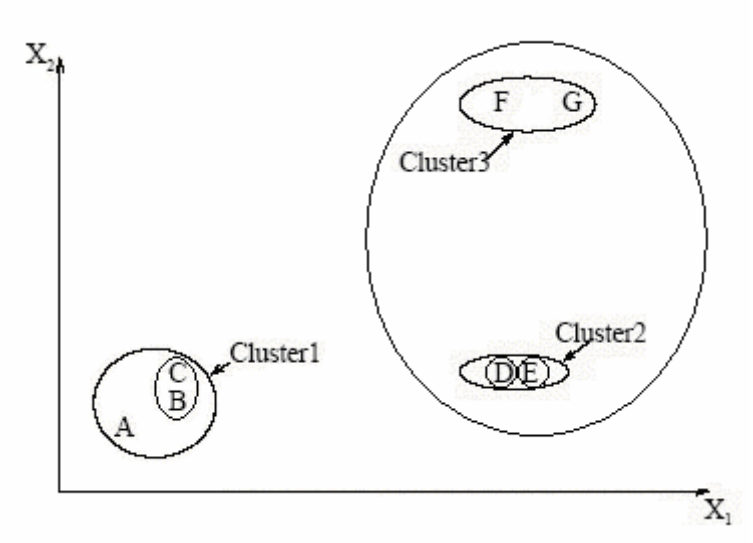



**Figure 3.7 Hierarchical Clustering**

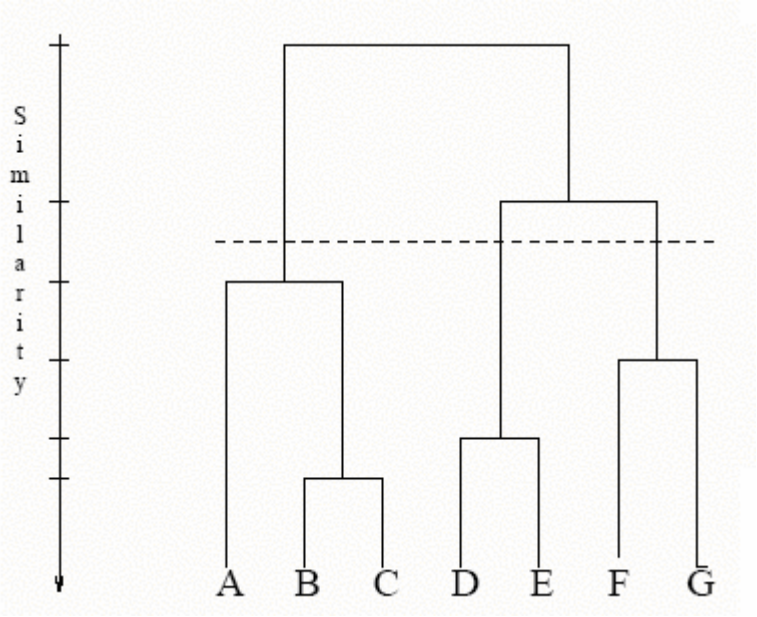



**Figure 3.8 The dendogram obtained using the single link algorithm**

Below we discuss the most popular hierarchical clusering algorithms. There are several other hierarchical clustering algorithms which are just the variants of those discussed below.

The following notation is used in order to determine the distance metrics :

$n$ number of observations

$v$ number of variables if data are coordinates

$x_i$ $i$th observation

$\bar{\mathrm{x}}$ sample mean vector

$C_K$ $K$th cluster, subset of $\{x_1, x_2, \ldots, x_n\}$

$N_K$ number of observations in $C_K$

$\|x\|$ Euclidean length of the vector **x,** which is square root of sum of squares of the elements of **x**

$$\|\mathrm{x}\| = \sqrt{\left(x_{i_1} - x_{j_1}\right)^2 + \left(x_{i_2} - x_{j_2}\right)^2 + \ldots + \left(x_{i_n} - x_{j_n}\right)^2}$$

$\bar{\mathbf{x}}_K$ mean vector for cluster $C_K$

$d(\mathbf{x},\mathbf{y})$ any distance or dissimilarity measure between observations or vectors **x** and **y**

$D_{KL}$ any distance or dissimilarity measure between clusters $C_K$ and $C_L$

$W_K$ $\sum_{i \in C_k} \left\| x_i - \bar{\mathbf{x}}_K \right\|^2$

- *Single Link (SLINK) Method*

The single link method operates by joining, at each step, the two most similar objects, which are not yet in the same cluster. In this method, the distance between two clusters is the minimum distance between an observation in one cluster and an observation in the other cluster [21]. This method is also called “nearest neighbor” clustering method. This

method produces long chains of objects called *chaining* resulting in loose, straggly clusters. Chaining means that the objects join a cluster easily if there exists intermediate objects that bridge the cluster to other objects. The bigger the cluster, the easier it is for new elements to join, because the probability of finding intermediate objects increases [22].

$$D_{KL} = \min_{i \in C_K} \min_{j \in C_L} d(x_i, x_j)$$

- *Complete Link (CLINK) Method*

The complete link method is exactly the opposite of the single link method, since it uses the least similar pair between two clusters to determine the inter-cluster similarity. In this method, the distance between two clusters is the maximum distance between an observation in one cluster and an observation in the other cluster [21]. This method is also called "furthest neighbor" clustering method. This method tends to produce very tight clusters of similar cases. In a growing cluster, new objects have little chance to join, because they must be close to all objects in the cluster [22].

$$D_{KL} = \max_{i \in C_K} \max_{j \in C_L} d(x_i, x_j)$$

- *Group Average Method*

The group average method relies on the average value between all pairs of individuals in the two clusters. Since all objects in a cluster contribute to the inter-cluster similarity, each object is, on average more like every other member of its own cluster than the objects in any other cluster. In average linkage, the distance between two clusters is the average distance between pairs of observations, one in each cluster [21]. Average linkage tends to join clusters with small variances, and it is slightly biased toward producing clusters with the same variance.

$$D_{KL} = \frac{1}{N_K N_L} \sum_{i \in C_K} \sum_{j \in C_L} d(x_i, x_j)$$

- *Centroid Method*

In the centroid method, the distance between two clusters is defined as the (squared) Euclidean distance between their centroids or means.

$$D_{KL} = \left\| \bar{\mathbf{x}}_K - \bar{\mathbf{x}}_L \right\|^2$$

- *Ward's Minimum Variance Method*

In Ward's minimum-variance method, the distance between two clusters is the sum of squares between the two clusters added up over all the observations. The sums of squares

are easier to interpret when they are divided by the total sum of squares to give proportions of variance [21].

$$D_{KL} = \frac{\left\| \bar{\mathbf{x}}_K - \bar{\mathbf{x}}_L \right\|^2}{\frac{1}{N_K} + \frac{1}{N_L}}$$

Ward [23] proposed a clustering procedure seeking to form the groups in a manner that minimizes the information loss associated with each grouping and to quantify that loss in a form that is readily interpretable. The information loss is defined by Ward in terms of an error sum-of-squares (ESS) criterion.

$$\mathbf{ESS} = \sum_{\mathbf{i}=1}^{\mathbf{n}} \mathbf{x}_\mathbf{i}^2 - \frac{1}{\mathbf{n}} \left( \sum_{\mathbf{i}=1}^{\mathbf{n}} \mathbf{x}_\mathbf{i} \right)^2$$

It starts by placing each observation in a single cluster. At each step in the analysis, the union of every possible pair of clusters is considered and the two clusters whose fusion results in the minimum increase in the error sum of squares are combined.

In this research, we have tried to group the data using different distance metrics and clustering algorithms and by comparing the results we found that for our purpose, the Ward's minimum variance method generates the best results. Ward's method groups all the samples of an application together in one cluster for most of the applications. We have performed the clustering with the number of clusters ranging from 7 to 13. The results of clustering using Ward's Method are discussed in Chapter 5.

**Partitional Clustering Algorithms**

Partitional clustering attempts to directly decompose the data set into a set of disjoint clusters. More specifically, they try to determine an integeral number of partitions that optimize a certain criterion function defined either locally (on a subset of the patterns) or globally (over all of the patterns). The criterion function that the clustering algorithm tries to minimize may emphasize the local structure of the data, as by assigning clusters to peaks in the probability density function, or the global structure. Typically the global criterion involves minimizing some measure of dissimilarity in the observations within each cluster, while maximizing the dissimilarity of different clusters [16]. Partitional clustering algorithms differ from the hierarchical techniques by allowing relocation of the observations, thus allowing poor initial partitions to be corrected at a later stage.

Unlike the hierarchical clustering methods, these algorithms do not create a tree structure to describe the groupings in the data, but rather creates a single level of clusters. Thus, partitional methods have advantages in applications involving large data sets for which the construction of a dendrogram is computationally prohibitive. Another difference is that partitional clustering uses the actual observations of objects or individuals in the data, and not just their proximities. A problem accompanying the use of a partitional algorithm is the choice of the number of desired output clusters [17].

- *K-Means Clustering Algorithm*

K-Means clustering is an algorithm for partitioning (or clustering) $n$ data points into $K$ disjoint subsets $\boldsymbol{C_j}$ containing $\boldsymbol{N_j}$ data points, so as to minimize the sum-of-squares criterion

$$J = \sum_{j=1}^{K} \sum_{i \in C_j} \left\| x_i - \mu_j \right\|^2$$

where, $\boldsymbol{x_i}$ is a vector representing the $i$th data point and $\boldsymbol{\mu_j}$ is the geometric centroid of the data points in $\boldsymbol{C_j}$ [24]. The algorithm consists of a simple re-estimation procedure. The K-means algorithm starts with a random initial partition of the patterns into K clusters. Subsequent steps modify the partition by reassigning patterns to the clusters to reduce the sum of the distances for each pattern from the mean of the cluster to which the pattern belongs. The modification consists of allocating each pattern to the nearest of the K means of the previous partition. This leads to a new partition for which the sum of distances is strictly smaller than before. There is a possibility that the improvement step leads to fewer than K partitions. In this situation, one of the partitions (generally the one with the largest sum of distances from the mean) is divided into two or more parts to reach the required number of K partitions. The K-means algorithm is popular because it is easy to implement, and its time complexity is O(n), where n is the number of patterns [17]. However, since the algorithm is sensitive to the initial partition, it is a good idea to rerun the algorithm with different randomly generated initial partitions to reduce the chances of the heuristic producing a poor solution. Generally, the number of "true" clusters in the data is not known. Therefore, it is a good idea to run the algorithm with different values of K that are near the number of clusters one expects from the data to see how the sum of distances reduces with increasing values of K.

- *Graph-Theoretic Clustering*

The best-known graph-theoretic divisive clustering algorithm is based on construction of the minimal spanning tree (MST) [25] of the data, and then deleting the MST edges with the largest lengths to generate clusters. Figure 3.9, from [17], depicts the MST obtained from nine two-dimensional points. The figure shows separation of these nine points into two clusters. Three clusters would result from deleting the edge with maximum length, which is between E and F. The hierarchical approaches are also related to graph-theoretic clustering [17].

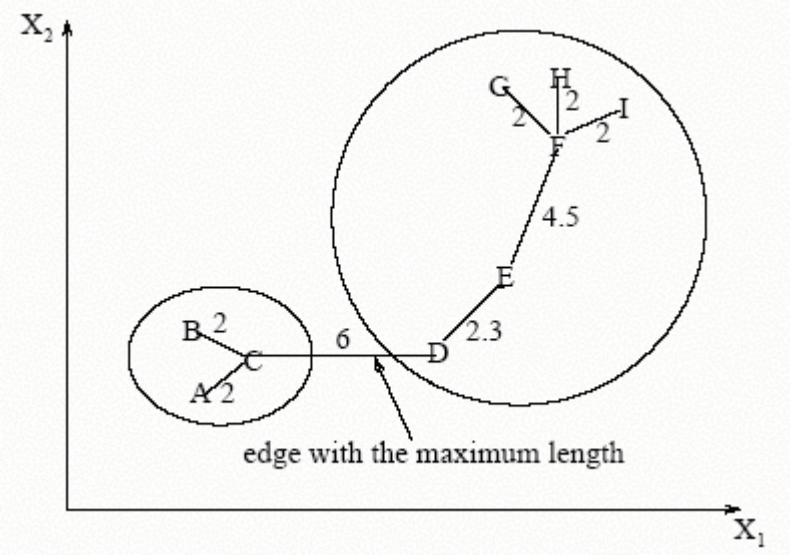



**Figure 3.9 Using the minimal spanning tree to form clusters**

## *D. Problems encountered in the clustering process*

### *a. Determination of the input vector size*

The packet sizes encountered in the data were in the range of 60-1514 bytes. In order to simplify the computation and to reduce the computational time, this range was divided into bins of varying sizes. The bin size of the histograms is important, since we want to differentiate the dissimilarities in different applications in order to identify them and generate "correct" clusters. Hence, we had to experiment with different histogram bin sizes and numbers to develop a scheme that groups the similar applications and differentiates the dissimilar ones, thus generating the "best" cluster partitions according to our requirements and goals. The histogram bin sizes depend upon the Internet traffic mixture and it is quite possible that if the Internet data shows different trends in the future, the histogram bin size might have to be changed accordingly. The histogram schemes for the different data sets and how the histograms were generated are discussed in Chapter 4.

### *b. Deciding the number of clusters present*

A problem common to all clustering techniques is the difficulty of deciding the number of clusters present in the data. In order to optimize the clustering criterion, it is generally suggested to plot the criterion value against the number of clusters, which will indicate the correct number of clusters to be considered by showing a sharp increase, if the criterion is being maximized; or a sharp decrease, if the criterion is being minimized [15]. In order to determine the optimal number of clusters, we generated a plot of root mean square standard deviation (RMSSTD) between the adjacent numbers of clusters against

the number of clusters. Based on this plot, we chose the number of clusters, which showed the least difference in the RMSSTD when compared to the adjacent higher number of clusters. Since we minimized the criteria, a sharp decrease indicates the correct number of clusters. We chose the number of clusters that had the lesser value of the two.

The root-mean-square standard deviation [21] of a cluster $C_K$ with $N_K$ observations is

$$\textbf{RMSSTD} = \sqrt{\frac{W_K}{v(N_K - 1)}}$$

where, $W_K = \sum_{i \in C_k} \left\| x_i - \bar{\mathbf{x}}_K \right\|^2$ and $v$ is the number of variables if data are coordinates.

# 4. Data Description

It is necessary to understand the method of data collection and the treatment of collected data for better understanding of the results generated in this research. This chapter describes the manner and the format in which the data was collected and stored. The next section describes the steps involved in histogram generation such as identifying the data, determining the bin numbers and sizes and finally, generating histograms. The last section describes the problems encountered in data collection and histogram generation. It also discusses some observed characteristics of the collected data.

## *A. Data Collection*

For this research, the data was collected from the North Carolina State University backbone network using a software tool called *Tcpdump*. Tcpdump is a powerful tool that allows us to sniff TCP/IP packets and display the contents of the Ethernet segment in a number of different formats. It allows us to see all the traffic and has options to record the part of the Ethernet segment that we are interested in. The data collected using tcpdump is stored in text format. Each line in the file, similar to the fictitious line shown below, represents one data packet.

11:30:00.000055,6,152.1.55.145,1321,189.142.29.77,64714,1514

The fields, separated by commas, in the data packet shown above from left to right are:

- Timestamp (in seconds)
- IP Protocol
- Source IP Address
- Source Port Number
- Destination IP Address
- Destination Port Number
- Packet Size (bytes)

We collected three independent sets of data from the NC State backbone network for this research. These data sets are discussed below and summarized in Table 4.1. The first two data sets can be found under the main directory 'C:/Research/DATA' and the third data set can be found in the directory 'F:/RawData.'

- The first data set was collected in 1999. It was collected for five days from June 1 to June 5, every 20-25 minutes for a period of 5 minutes or until 500,000 packets were collected. The size of this data set was about 10GB. Each of the five-minute intervals is stored in a separate text file, which can be found in the directory 'data_set_1'. The name of the file is of the form 'tcpdump_19990601.1216' which indicates the date and the time (tcpdump_yyyymmdd.hhmm) when the collection of the data stored in the file was started. The tcpdump data files for one day are archived together using the zip utility under a file name of the form 'data_19990601' (data_yyyymmdd).
- The second data set was collected in 2002 for two days – Feb 8 and Feb 9. It was collected at half hour intervals for a period of 5 minutes or until 500,000 packets were collected. The size of this data set was about 1GB. Each of these 5-minute intervals is stored in a separate text file which is then converted to zip format using the zip utility to reduce the file size. There are 48 such files, which can be found in the directory 'data_set_2'. The name of the text and the zipped file is of the form '02082002.0858', which indicates the date and the time (mmddyyyy.hhmm) when the collection of the data stored in the file was started.
- The third data set was collected continuously for about four hours on February 25, 2003 from 11:30:00 to 15:34:59 EST. The size of this data set was about 74GB.

In the third data set as well, the data collected in the five-minute intervals is stored in a separate text file, which is then converted to zip format using the zip utility. There are 48 such files, excluding the file 'log.040' that was corrupted during the process of data collection. These files are stored in the directory 'data_set_3'. The name of the text and the zipped file is of the form 'log.000', 'log.001' and so on.

**Table 4.1 Data Summary**

| Data Set | Date and Time | Size (approx.) | Interval of data collection | Characteristic of data |
|---|---|---|---|---|
| Data Set 1 | June 1, 1999 to June 5, 1999 | 10GB | 5 min interval with a period of 25 mins | Sampled |
| Data Set 2 | Feb 8, 2002 and Feb 9, 2002 | 1GB | 5 min interval with a period of 30 mins | Sampled |
| Data Set 3 | 25-Feb-03 | 74GB | Four hours | Continuous |

In this research discussion, we shall discuss the results based on the third data set and give comparisons with the results from first and second data sets.

## *B. Histogram Generation*

As seen from Table 4.1, the size of data sets was huge. It is very difficult, costly and time consuming to process such large data sets in real time. Feature selection that chooses the important original features is an effective data reduction technique. An important feature for a learning task can be defined as one whose removal from the computation degrades the learning accuracy [26]. By removing the unimportant features, data size is reduced,

while learning accuracy and comprehensibility does not degrade and may actually improve. We have collected a lot of information about the network data as seen from the collected data format in the previous section, but we are using just the source and destination port number and the packet size information in this research. The source and destination port numbers are used to identify the application that the packet belongs to and the packet size is used to generate the packet size distributions.

The process of histogram generation is summarized in the steps described below:

*a. Identifying the major applications*

The data collected was scanned to identify the major applications in the network traffic data. The applications were identified using the source and destination port numbers. For example, port 80 usually refers to HTTP application. Hence if the source or the destination port is 80, it implies that the packet belongs to HTTP flow. Similarly, port 20 is reserved for FTP. Hence, packets originating from or destined to port 20 belong to the FTP flow. The list of the most common port numbers and the corresponding applications is shown in Table 2.3. The twenty major applications in the data set 2 and 3, their associated port numbers and their percentage in the total traffic mixture are shown in the Table 4.2 in the order of their percentage in total traffic. All the other applications in the traffic mixture are grouped under 'Other'. Note that since we are incrementing the counters for both the source and destination ports in the packet, to know the major applications in the traffic mixture, the total number of packets would be double than the

actual number of packets. Hence, after determining the major applications in the data set, the actual percentage of the major applications in the data set can be determined by scanning the data again and incrementing the counters for the major applications only.

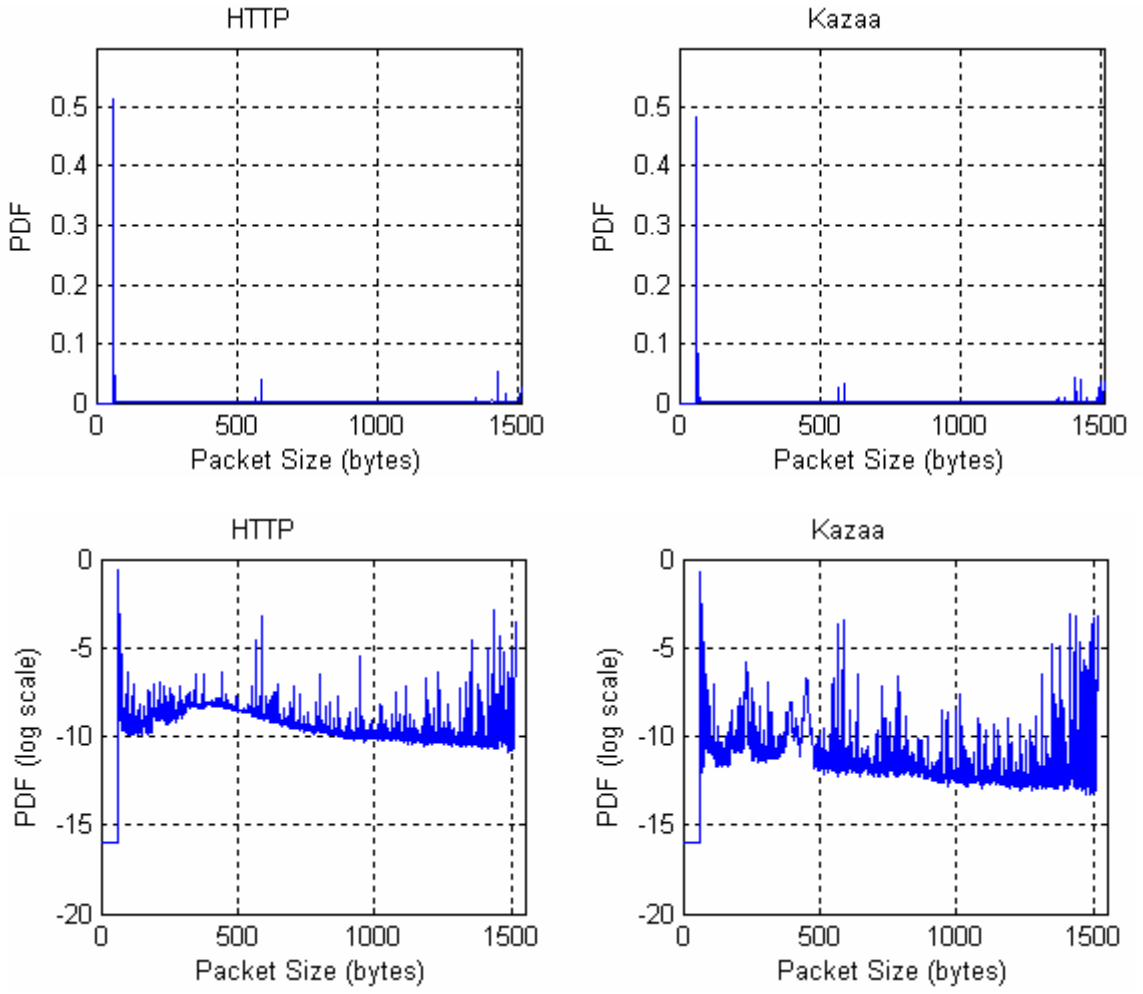
**Table 4.2 Major applications in data set 2 and 3**

| Data Set 2(2002) | | | Data Set 3 (2003) | | |
|---|---|---|---|---|---|
| **Application** | **Port** | **%** | **Application** | **Port** | **%** |
| Kazaa | 1214 | 12.60% | HTTP | 80 | 7.83% |
| HTTP | 80 | 3.79% | Kazaa | 1214 | 2.23% |
| Carracho | 6701 | 3.24% | FTP | 20 | 1.40% |
| eDonkey | 4662 | 2.65% | Gnutella | 6346 | 0.90% |
| Napster | 20 | 2.16% | Unassigned | 3933 | 0.63% |
| Hotline | 6699 | 1.84% | RTP | 6970 | 0.58% |
| FTP | 5501 | 1.70% | Napster | 6699 | 0.55% |
| MIT ML Device | 6346 | 1.00% | eDonkey | 4662 | 0.50% |
| Gnutella | 85 | 0.89% | AOL | 5190 | 0.47% |
| AOL | 3221 | 0.65% | Multicast | 16384 | 0.47% |
| AIMPP Port | 412 | 0.58% | Half Life Server | 27015 | 0.45% |
| NNTP | 2847 | 0.52% | Plethora | 3480 | 0.40% |
| DNSIX | 5190 | 0.50% | Reserved | 0 | 0.32% |
| X window | 90 | 0.33% | IRC | 6667 | 0.32% |
| FTP | 21 | 0.30% | ms-olap2 | 2394 | 0.29% |
| XML | 119 | 0.25% | SMTP | 25 | 0.28% |
| Trap Convention Port | 3909 | 0.24% | Half Life Client | 27005 | 0.28% |
| Network blackjack | 1025 | 0.21% | ICAP | 1344 | 0.27% |
| Gnutella | 3657 | 0.19% | tragic | 2642 | 0.26% |
| ipcd | 6348 | 0.18% | mloadd | 1427 | 0.25% |

*b. Determining the bin numbers and sizes*

As noted in Chapter 2, Ethernet packet size ranges from 60-1514 bytes. This is quite a large range when it comes to performing complex real time calculations on the data set. Hence, in order to reduce the dimensionality of the data, the Ethernet packet size range

was divided into a manageable number of bins. For initial analysis, bin size of one byte was considered and the data was scanned to identify the total number of packets in each bin and generate the packet size distribution for each major application. Therefore, initially the packet size distribution for each major application was generated with 1514 bins for 1-1514 bytes. It was observed that for most applications, the number of packets in the small packet-size range, between 60-80 bytes, and the large packet-size range, between 1420-1514 bytes, was large. In fact, it is so large that if a graph used linear scaling on Y-axis, the lines for other packet sizes become almost invisible. Thus, logarithmic scaling is used to display the packet size distributions. The bin sizes were determined based on these high resolution histograms. The bin sizes were chosen so that no two peaks in the logarithmic plot of an application fall in the same histogram, thus allowing us to capture distinctly the characteristics of different applications. The packet size distribution with 1514 bins and corresponding logarithmic plots for the 4 most prevalent applications in data set 3 are shown in Figure 4.1.


HTTP
Kazaa
PDF
0.5
0.4
0.3
0.2
0.1
0
500
1000
1500
Packet Size (bytes)
HTTP
Kazaa
PDF (log scale)
0
-5
-10
-15
-20
500
1000
1500
Packet Size (bytes)

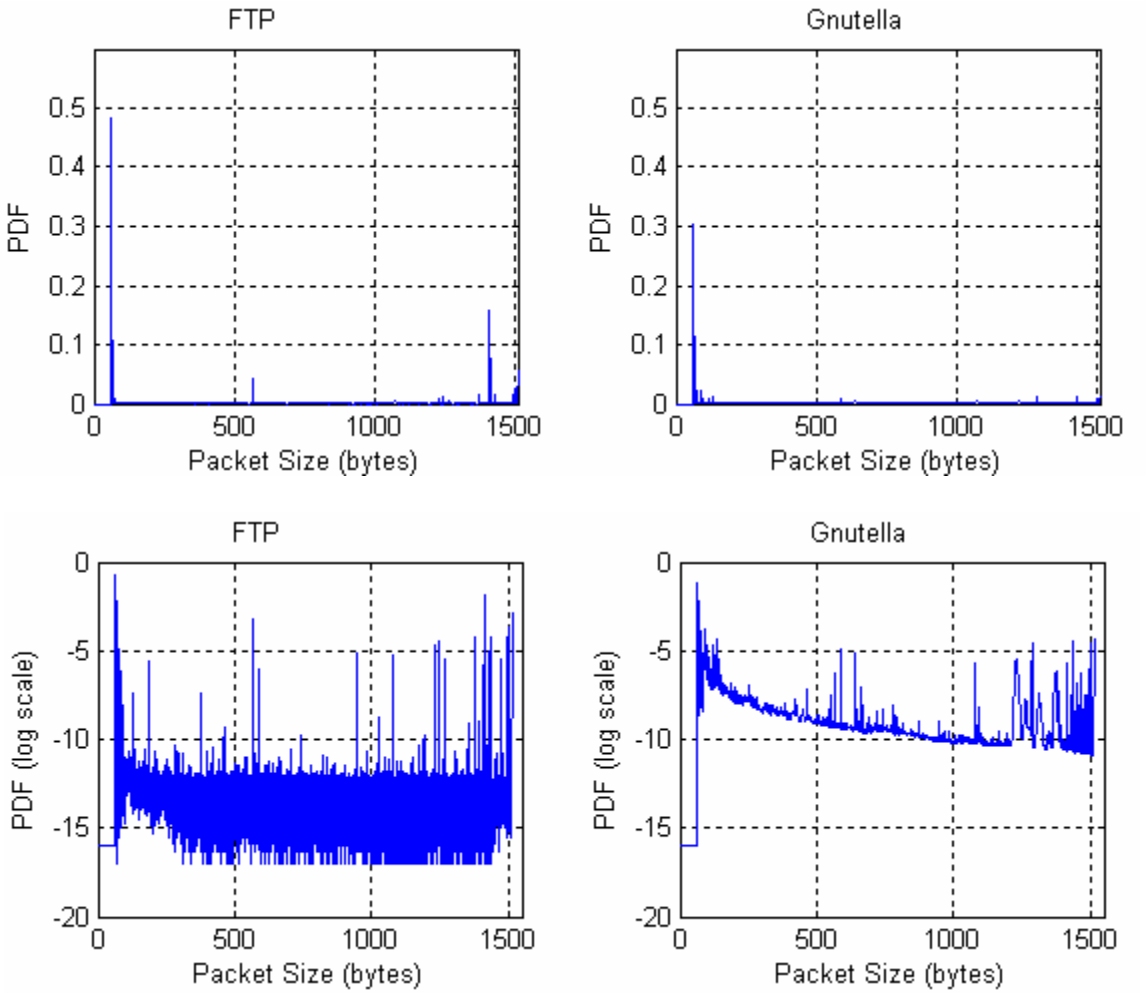



**Figure 4.1 Packet size distributions of 4 most prevalent applications in data set 3 (plotted on linear scale and log scale)**

*c. Generating histograms*

For generating the histograms for each application, each data packet in the collected data was scanned to identify the packet size and the bin range in which it falls, and the application that it belongs to. The counter for the corresponding application histogram bin was incremented. The total number of the packets in each bin was then normalized to generate the histograms or the packet size distributions of the major applications.

From the logarithmic plots of the ten major applications in data set 1, we developed a histogram scheme with 50 bins. The packet size distributions of major applications of data set 1, generated using this 50-bin scheme displayed significantly distinct characteristics. The classification performed on the network traffic with these 50 bins generated good results as shown in [2]. The packet size distributions of the twenty major applications of data set 2 that were generated using the 50-bin scheme used for the data set 1 also displayed significantly distinct characteristics and generated good classification results. So we adopted the same 50-bin scheme for data set 2. We generated the packet size distributions of the twenty major applications of data set 3 with the same 50-bin histogram scheme and performed classification with this scheme. However, this scheme did not distinguish the characteristics in the small packet size region, from 60-65 bytes, and in large packet size region, from 1330-1500 bytes, for some of the major applications. So we used a 60-bin histogram scheme with bin sizes shown in Table 4.3. The classification performed with the 60-bin histogram scheme generated better results as compared to the results generated by the 50-bin histogram scheme on data set 3, which will be shown in Chapter 5. In order to optimize the histogram scheme for data set 3, we generated the packet size distributions for the major applications of data set 3 with 1514 bins each of size one byte. For each major application, we noted the bins that had a weight of greater than 0.002 and then integrated this information from all the applications to develop a 40-bin histogram scheme. The bin ranges were chosen such that two peaks from one application did not fall in the same bin and the peaks would fall in the center of the bin range. The classification results generated with the 40-bin histogram scheme were

about the same as those generated with the 60-bin histogram scheme. The 40-bin histogram scheme did better in estimation and detection as compared to the 60-bin histogram scheme in some cases whereas it did worse in some other cases.

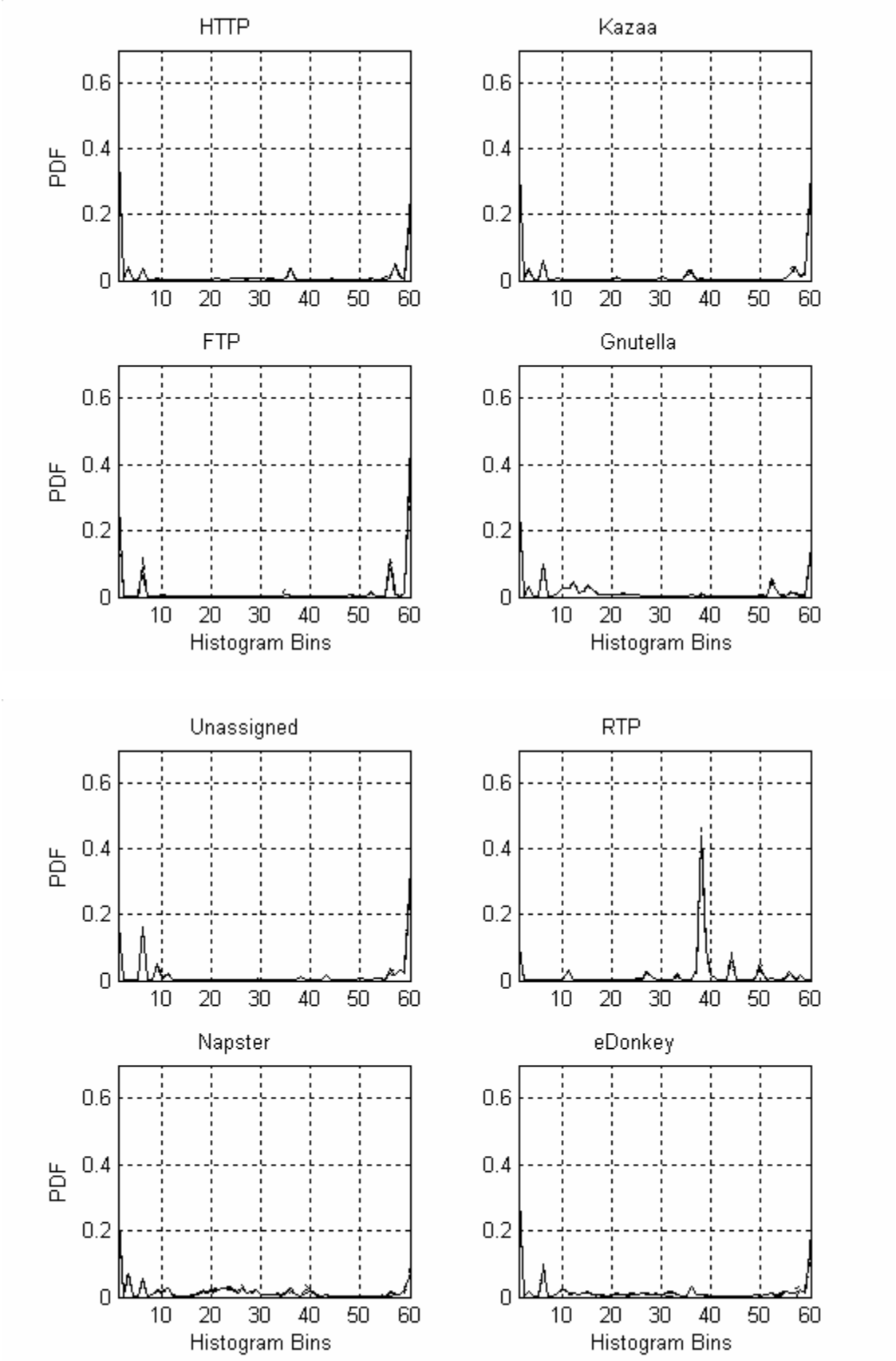
The packet size ranges of the 50-bin, 60-bin and 40-bin histogram schemes are shown in Table 4.3. The sizes of the bins are not equal across the entire range of packet-sizes. In fact, they are narrow in the small packet size region, where the packet size distribution is dense, and increase toward the large packet-size region.

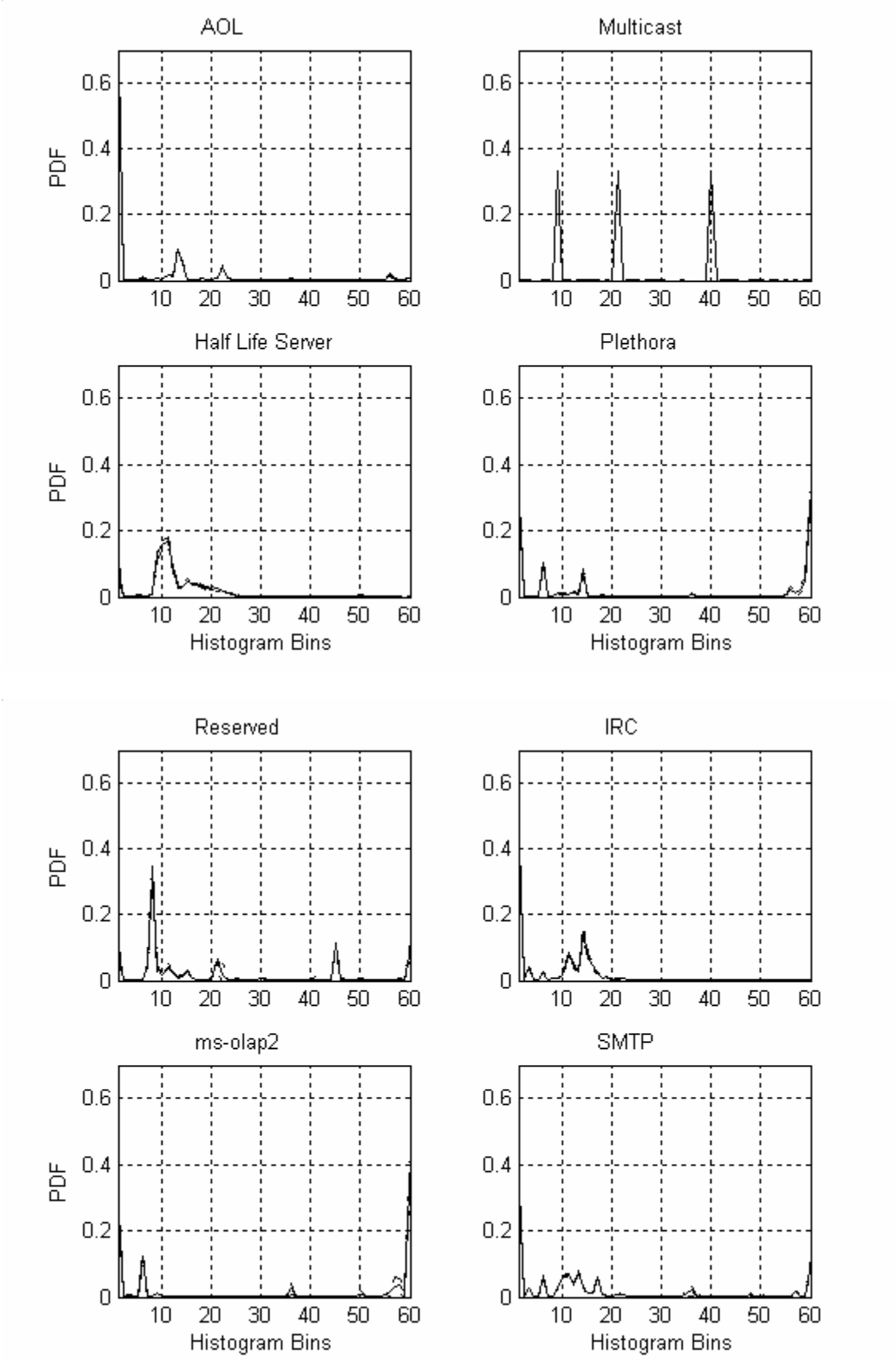
**Table 4.3 Bin sizes for 50-bin, 60-bin and 40-bin histogram schemes**

| Bins | Bin Sizes | | | Bins | Bin Sizes | | |
|---|---|---|---|---|---|---|---|
| | 50 Bins | 60 Bins | 40 Bins | | 50 Bins | 60 Bins | 40 Bins |
| 1 | 0-65 | 0-60 | 0-60 | 31 | 421-435 | 461-485 | 746-812 |
| 2 | 66-70 | 61 | 61-64 | 32 | 436-450 | 486-510 | 813-957 |
| 3 | 71-75 | 62 | 65-67 | 33 | 451-465 | 511-535 | 958-1231 |
| 4 | 76-80 | 63 | 68-69 | 34 | 466-480 | 536-560 | 1232-1395 |
| 5 | 81-85 | 64 | 70 | 35 | 481-495 | 561-580 | 1396-1424 |
| 6 | 86-90 | 65-66 | 71-72 | 36 | 496-510 | 581-600 | 1425-1444 |
| 7 | 91-95 | 67-68 | 73 | 37 | 511-525 | 601-630 | 1445-1474 |
| 8 | 96-100 | 69-70 | 74 | 38 | 526-540 | 631-655 | 1475-1500 |
| 9 | 101-105 | 71-75 | 75 | 39 | 541-570 | 656-680 | 1501-1510 |
| 10 | 106-120 | 76-80 | 76 | 40 | 571-600 | 681-705 | 1511-1514 |
| 11 | 121-135 | 81-90 | 77 | 41 | 601-682 | 706-735 | |
| 12 | 136-150 | 91-100 | 78 | 42 | 683-764 | 736-760 | |
| 13 | 151-165 | 101-110 | 79 | 43 | 765-846 | 761-785 | |
| 14 | 166-180 | 111-120 | 80 | 44 | 847-928 | 786-810 | |
| 15 | 181-195 | 121-135 | 81 | 45 | 929-1010 | 811-835 | |
| 16 | 196-210 | 136-150 | 82 | 46 | 1011-1092 | 836-860 | |
| 17 | 211-225 | 151-165 | 83-85 | 47 | 1093-1174 | 861-905 | |
| 18 | 226-240 | 166-180 | 86-89 | 48 | 1175-1256 | 906-950 | |
| 19 | 241-255 | 181-195 | 90 | 49 | 1257-1338 | 951-1015 | |
| 20 | 256-270 | 196-210 | 91-101 | 50 | 1339-1514 | 1016-1090 | |
| 21 | 271-285 | 211-235 | 102-115 | 51 | | 1091-1170 | |
| 22 | 286-300 | 236-260 | 116-167 | 52 | | 1171-1250 | |
| 23 | 301-315 | 261-285 | 168-219 | 53 | | 1251-1290 | |
| 24 | 316-330 | 286-310 | 220-395 | 54 | | 1291-1330 | |
| 25 | 331-345 | 311-335 | 396-578 | 55 | | 1331-1370 | |
| 26 | 346-360 | 336-360 | 579-621 | 56 | | 1371-1420 | |
| 27 | 361-375 | 361-385 | 622-656 | 57 | | 1421-1440 | |
| 28 | 376-390 | 386-410 | 657-667 | 58 | | 1441-1475 | |
| 29 | 391-405 | 411-435 | 668-684 | 59 | | 1476-1500 | |
| 30 | 406-420 | 436-460 | 685-745 | 60 | | 1501-1514 | |

The average packet size distributions of the twenty major applications of the data set 3 for 60-bin histogram scheme are shown in Figure 4.2. As discussed earlier, the bin sizes for the histogram schemes were chosen such that no two peaks in the logarithmic plots of an application, shown in Figure 4.1 for some applications, fall in the same bin. Comparing Figure 4.1 and Figure 4.2, we see that the peaks in the logarithmic plots in Figure 4.1 fall in separate bins in Figure 4.2 enabling us to capture the characteristics of the applications. From the packet size distributions in Figure 4.2, it is evident that some

applications behave quite differently from others and (display enough variations) they have certain unique characteristics in their packet size distributions, which can be explored to identify the applications. For example, the RTP application has a peak at the $38^{th}$ bin unlike any other application among the major applications. This enables us to identify RTP applications very easily from its packet size distribution. The packet size distribution of HTTP and FTP in data set 3 for 40-bin, 50-bin and 60-bin histogram scheme is shown in Figure 4.3 to illustrate the importance of the bin sizes in order to distinguish the characteristics of an application. The 50-bin scheme fails to capture the characteristics of HTTP and FTP in the small packet size region, whereas the 60-bin and 40-bin schemes are able to capture these characteristics. In order to distinguish the applications from one another during the clustering process, it is important to capture maximum information about the applications in the histograms. As seen from the clustering results in Chapter 5, the 60-bin histogram scheme gives better results as compared to the 50-bin since it captures all the distinct characteristics of the applications. For example, Kazaa and eDonkey fall together in one cluster in 50-bin histogram scheme but they fall in separate clusters in the 60-bin histogram scheme.


HTTP
Kazaa
FTP
Gnutella
Unassigned
RTP
Napster
eDonkey
PDF
Histogram Bins
0
0.2
0.4
0.6
10
20
30
40
50
60


AOL
Multicast
Half Life Server
Plethora
Reserved
IRC
ms-olap2
SMTP
PDF
Histogram Bins
0
0.2
0.4
0.6
10
20
30
40
50
60

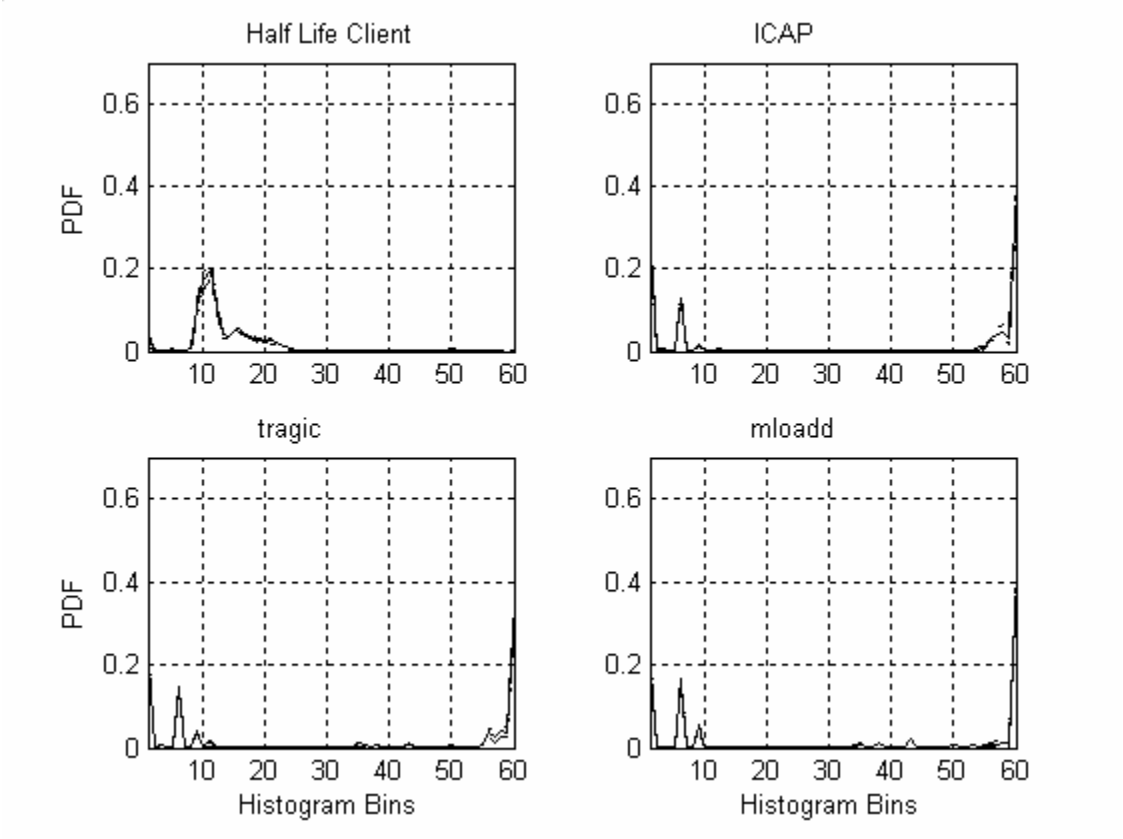
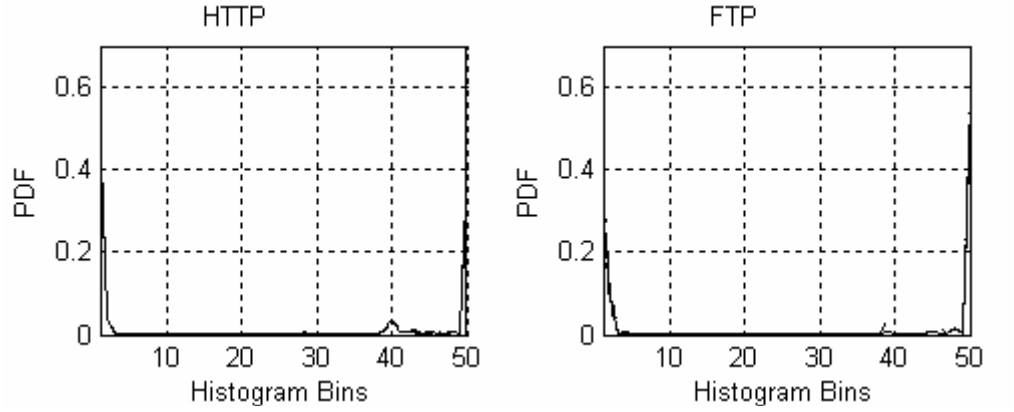



**Figure 4.2 Average packet size distributions of 20 major applications in data set 3 for 60-bin histogram scheme**



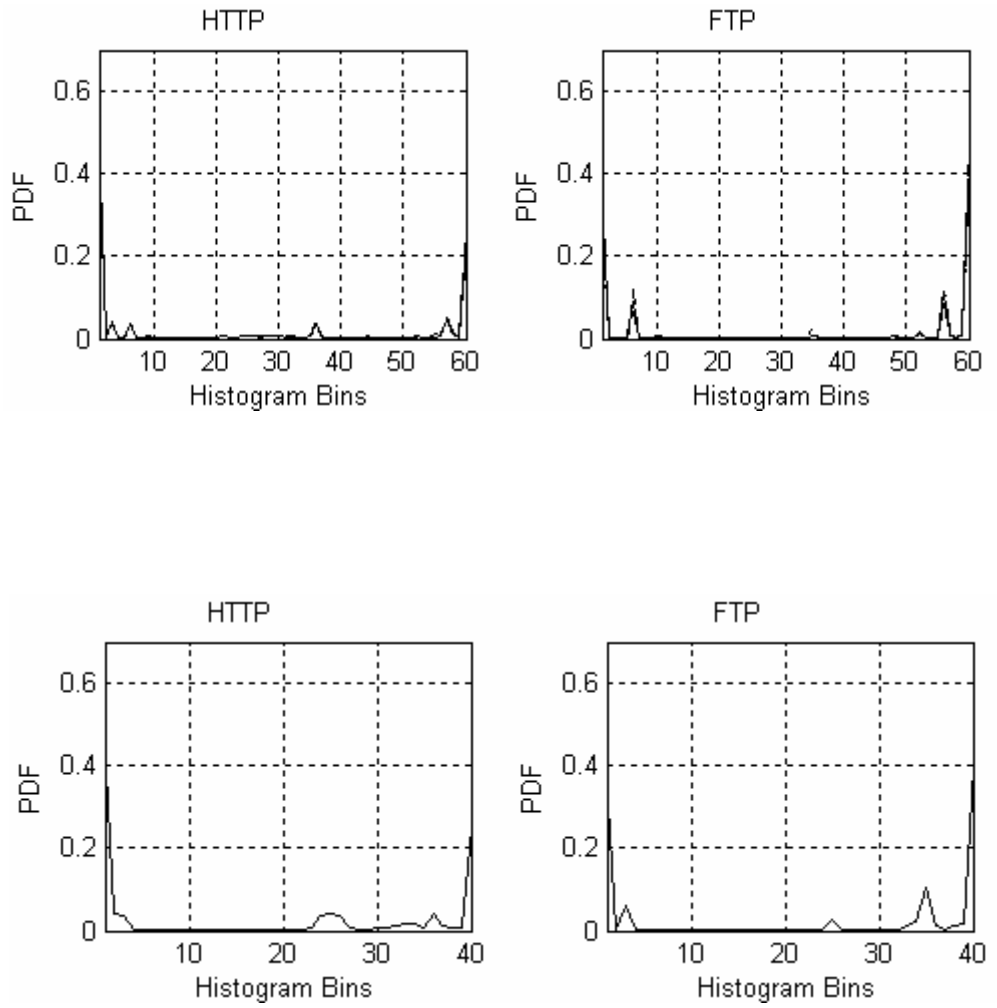



**Figure 4.3 PSD of HTTP and FTP for 50, 60 and 40 bin histogram scheme for data set 3**

## *C. Problems and Characteristics of data collection*

### *a. Evolution of Internet*

From the data sets collected at different times, it is evident that the Internet has evolved. Peer-to-peer (P2P) applications did not appear in the first data set; however there were

many peer-to-peer applications in the second and the third data set. In fact, it constitutes significant percentage of the total Internet traffic mixture as shown in Table 4.2.

Also looking at the packet size distributions of some applications like AOL, it is clear that the behavior of the application itself has changed over the years. The packet size distribution of AOL for data set 2 and data set 3 considering the 50-bin histogram scheme is shown in Figure 4.4.

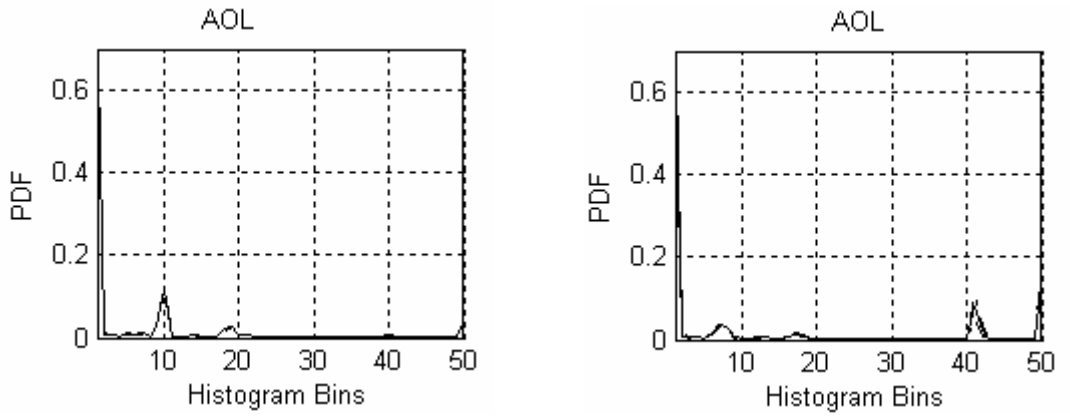



**Figure 4.4 PSD of AOL for data set 2 and 3 using 50-bin scheme**

*b. Histogram Scheme*

From the results generated using the different histogram schemes, it is clear that the selection of histogram scheme is important in data analysis. However, there is no set method to develop an optimal histogram scheme to generate good classification of the applications into clusters. The histogram scheme that we used in this research might fail

to capture the characteristics of the applications in an optimal manner in future. We have not investigated the duration of time for which the current histogram scheme may be valid. It may be required to vary the current scheme in future when it fails to capture the distinct characteristics between the applications of interest. This should not be surprising, considering the evolution and growth of the Internet.

### *c. Identifying the applications*

The application that a data packet belongs to is identified using the source and destination port numbers in the data packet. As discussed in Chapter 2, there are certain ambiguities regarding the port assignments. Especially, the port numbers in the range of 1025-65536 do not have an application reserved strictly for it. A port number in this range can be used by more than one application and some applications may use more than one port for sending and/or receiving data. For example, the RTP applications use the ports in the range 6170-6199. Certain Trojan horses also use the port 6170. Due to this, we may not be able to isolate some applications. The packet size distributions of the applications may appear different in the different data sets due to a different mixture.

# 5. Results of Clustering

The goal for using clustering in this research is primarily to verify that applications can be effectively characterized by their packet size distributions and to determine natural grouping of the applications based on their similarities and dissimilarities. This chapter shows the results obtained from clustering on data sets 2 and 3 for the different histogram schemes and then discusses the characteristics of the results for the 60-bin histogram scheme. The clustering results are generated using the Cluster Procedure in a software tool called SAS V8.02 [21]. SAS (Statistical Analysis System) is a widely used tool, which includes a wide range of statistical analyses, including analysis of variance, regression analysis, categorical data analysis, multivariate analysis, survival analysis, psychometric analysis, cluster analysis and nonparametric analysis.

## *A. Clustering Results and Comparison*

As noted in Chapter 3, in this research, clustering was performed using several methods such as K-Means method, Group Average method, Single Link method and Ward's Minimum Variance method. Of these, the Ward's method gave the best results and we shall discuss the results generated using this method in this Chapter.

The results of clustering with Ward's Minimum Variance method, which is discussed in Chapter 3, on data set 2 using the 50-bin histogram scheme for 12 clusters, are shown in Table 5.1. The results of clustering with Ward's Minimum Variance method on data set 3 using the 50-bin, 60-bin and 40-bin histogram scheme for 12 clusters are shown in Table 5.2, Table 5.3 and Table 5.4 respectively. The input to the clustering algorithm in each case is the normalized packet size distribution for all the applications to be clustered in each data file. Here, we are clustering 20 major applications in data set 2 and 3 and have grouped the rest of the applications under 'Other.' We have 48 data files of 5-minute intervals in data set 2 and 3 as discussed in Chapter 2. The input to the clustering algorithm will be 48 vectors for each of the 21 applications resulting in a total of 1008 input vectors to the clustering algorithm. In each of these tables, the first column lists the twenty major applications in order of their percentage in total traffic mixture and the first row indicates the name of the clusters in which the application falls. The names of the clusters are generated by the SAS software. The fields in the remaining cells of the table indicate the percentage of the application in a cluster. For example, if all 48 vectors of the application sample fall in one cluster only, then it will be indicated by 100%.

In Table 5.2, which shows clustering results on data set 3 using 50-bin histogram scheme, there are some applications, such as Unassigned, Half Life Server, MS-OLAP, ICAP, Tragic and mloadd, that don't fall entirely in one cluster. This is an indication that it would be difficult for us to identify these applications distinctly and hence we would not be able to estimate the traffic mixture accurately. As seen from Table 5.3 and 5.4, all the 48 vectors of an application fall in one cluster for the 40-bin and 60-bin histogram scheme. Hence, the results obtained using 60-bin and 40-bin histogram scheme on data set 3 are better than those obtained using 50-bin histogram scheme. This shows that choosing the correct histogram scheme is important to capture the characteristics of the applications and generate good estimations of the traffic mixture. There is not much difference between the results obtained using the 40-bin and 60-bin histogram scheme since for each application, 100% of the application falls in one cluster. However, some applications in the 40-bin histogram scheme do not merge with the same applications with which they merged in the 60-bin histogram scheme. For example, SMTP was in a singleton cluster in 60-bin scheme and Gnutella merged with AOL as shown in Table 5.3. In 40-bin scheme, shown in Table 5.4, SMTP merged with AOL whereas Gnutella was in a singleton cluster. However the fact that SMTP merges with Gnutella and AOL at the nine-cluster level in 60-bin scheme, as shown in Table 5.9, shows that there exists similarity between these applications. HTTP merges with Unassigned, eDonkey, Tragic and mloadd for 60-bin scheme, whereas, it merges with eDonkey alone for the 40-bin scheme.

### *B. Characteristics of the clustering results*

In this section, we discuss the characteristics of the clustering results for the 60-bin histogram scheme. The results of clustering with Ward's minimum variance method on data set 3 using the 60-bin histogram scheme for 13 clusters to 7 clusters is shown in Table 5.5 to Table 5.11 respectively. As we decrease the number of clusters in the data set, the clusters from the previous classification that are nearest to each other and whose fusion results in the minimum increase in the error sum of squares, merge together to form one cluster. Reducing the number of clusters in this manner indicate which applications are most similar. The degree of similarity can be inferred from the level at which they are merged into the same cluster. For example, Gnutella and AOL in clusters CL57 and CL27 respectively in Table 5.5 merge together to form CL12 in Table 5.6. The SMTP application is in a singleton cluster CL39 until ten-cluster level but is shown closer to Gnutella and AOL when it merges with them at the nine-cluster level to form CL9 as shown in Table 5.9. Note that the names of the clusters also indicate the level at which the application falls in that cluster. For example, Gnutella and AOL merge together at the 12-cluster level to form CL12, as shown in Table 5.6. In Table 5.5, Gnutella was the only application in CL57 which indicates that all the samples of Gnutella merge together in a cluster at 57-cluster level. Also, AOL was the only application in CL27, which indicates that all the samples of AOL merge together in a cluster at 27-cluster level. Some characteristics of the data and the clustering results that are worth noting are discussed below:

*a. Grouping of applications*

As discussed in the previous section, for 60-bin histogram scheme, all the 48 vectors corresponding to an application fall together in a single cluster for all the applications as shown in Table 5.5. This shows the similarity of the application. For example, 48 FTP samples corresponding to the 48 data files group together in cluster CL23 resulting in 100% in the corresponding cell in the table. Since, FTP is the only application in this cluster; it also indicates that all the FTP samples merge together in a cluster at 23-cluster level.

*b. Like applications fall together*

Likeness or the similarities between the applications is determined by the characteristics of the applications discussed in Chapter 2 and the similarities in their packet size distributions. As discussed in Chapter 2 and 3, certain applications like Half Life Client and Half Life Server have similar packet size distributions and hence they fall together in the same cluster (CL17) as seen in Table 5.5. Kazaa, MS-OLAP and ICAP fall in a cluster (CL15) as seen in Table 5.5. These applications fall together in a single cluster in the clustering results for 40-bin (Table 5.4) and 50-bin (Table 5.2) histogram scheme as well indicating a strong similarity between the applications. An interesting case is that of the peer-to-peer applications - Napster and eDonkey. The eDonkey application is similar to several applications, e.g., HTTP, Unassigned, Tragic and mloadd. Napster is sufficiently different from eDonkey that it is merged with eDonkey group only at the

eight-cluster level. This would indicate that Napster would be easier to detect than eDonkey, since eDonkey looks like several other applications, including the very prevalent HTTP. Thus, as we decrease the number of clusters, the applications that are most similar combine together in a cluster and diverse applications move farther away from one another. Similarly, the FTP application is sufficiently distinct from other applications in the network traffic and combines with the Kazaa group, which consists of Kazaa, Plethora, MS-OLAP, ICAP and 'Other' at the seven-cluster level as seen in Table 5.11.

*c. Diverse applications fall in different clusters*

The packet size distributions of certain applications are quite distinct when compared to the other applications and hence fall in a singleton cluster as seen in Table 5.11. For example, the packet size distribution of RTP (Real-time Transport Protocol) based applications, like QuickTime, shows a peak at bin 38, which is not seen in the other applications. This characteristic of the RTP based applications makes them distinct and easier to detect. Similarly, packet size distribution of multicast packets show three peaks which is not seen in any other application. Reserved also shows sufficiently distinct characteristics and falls in a singleton cluster as seen in Table 5.11. FTP and HTTP applications have different characteristics and hence they fall in separate clusters, CL8 and CL7, as seen in Table 5.11.

**Table 5.1Clustering with Ward's minimum variance method on data set 2 using the 50-bin histogram scheme for 12 clusters**

| | Data Set 2 | 12 clusters - 50bins | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CL14 | CL20 | CL22 | CL15 | CL63 | CL13 | CL17 | CL23 | CL43 | CL12 | CL37 | CL18 |
| 1 | Kazaa | | | | | | 100.00% | | | | | | |
| 2 | HTTP | | | 100.00% | | | | | | | | | |
| 3 | Carracho | 100.00% | | | | | | | | | | | |
| 4 | eDonkey | | | 100.00% | | | | | | | | | |
| 5 | Napster | | | 100.00% | | | | | | | | | |
| 6 | Hotline | 100.00% | | | | | | | | | | | |
| 7 | FTP | | | 4.17% | | | | | | | 95.83% | | |
| 8 | MIT | | | | | | | | | 100.00% | | | |
| 9 | Gnutella | | | | | 100.00% | | | | | | | |
| 10 | AOL | | | | | | | 100.00% | | | | | |
| 11 | AIMPP | | | | | | | 100.00% | | | | | |
| 12 | NNTP | | | | 100.00% | | | | | | | | |
| 13 | DNS | 4.17% | 95.83% | | | | | | | | | | |
| 14 | X-win | | | | 25.00% | 12.50% | 62.50% | | | | | | |
| 15 | FTP | | | | 4.17% | | 20.83% | | | | | | 75.00% |
| 16 | xml | 25.00% | | | | | | | | | | 62.50% | 12.50% |
| 17 | Trap Convention Port | 12.50% | 12.50% | | | | | | 75.00% | | | | |
| 18 | Network Blakckjack | | | | | | 37.50% | | | | 62.50% | | |
| 19 | Gnutella | | | | | 100.00% | | | | | | | |
| 20 | IPCD | 100.00% | | | | | | | | | | | |
| 21 | other | | | | 100.00% | | | | | | | | |

**Table 5.2 Clustering with Ward's minimum variance method on data set 3 using the 50-bin histogram scheme for 12 clusters**

| | | 12 clusters - 50bins | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CL33 | CL12 | CL14 | CL29 | CL40 | CL13 | CL28 | CL25 | CL20 | CL19 | CL67 | CL26 |
| 1 | HTTP | | | 100.00% | | | | | | | | | |
| 2 | Kazaa | | | | | | 100.00% | | | | | | |
| 3 | FTP | | | | | | | | | 100.00% | | | |
| 4 | Gnutella | | 100.00% | | | | | | | | | | |
| 5 | Unassigned | | 31.25% | 68.75% | | | | | | | | | |
| 6 | RTP | | | | | | | | 100.00% | | | | |
| 7 | Napster | | | | 100.00% | | | | | | | | |
| 8 | eDonkey | | | | | | 100.00% | | | | | | |
| 9 | AOL | | | | | 100.00% | | | | | | | |
| 10 | Multicast | 100.00% | | | | | | | | | | | |
| 11 | Half Life Server | | 4.17% | | | | | | | | 95.83% | | |
| 12 | Plethora | | 100.00% | | | | | | | | | | |
| 13 | Reserved | | | | | | | | | | | | 100.00% |
| 14 | IRC | | | | | | | 100.00% | | | | | |
| 15 | ms-olap2 | | 60.42% | | | | 39.58% | | | | | | |
| 16 | SMTP | | | | | | | | | | | 100.00% | |
| 17 | Half Life Client | | | | | | | | | | 100.00% | | |
| 18 | ICAP | | 29.17% | | | | 70.83% | | | | | | |
| 19 | tragic | | | 35.42% | | | 64.58% | | | | | | |
| 20 | mloadd | | | 79.17% | | | 20.83% | | | | | | |
| 21 | Other | | 100.00% | | | | | | | | | | |

**Table 5.3 Clustering with Ward's minimum variance method on data set 3 using the 60-bin histogram scheme for 12 clusters**

| 12 clusters - 60bins | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CL28 | CL13 | CL12 | CL14 | CL36 | CL15 | CL39 | CL17 | CL34 | CL31 | CL23 | CL24 |
| 1 | HTTP | | 100% | | | | | | | | | | |
| 2 | Kazaa | | | | | | 100% | | | | | | |
| 3 | FTP | | | | | | | | | | | 100% | |
| 4 | Gnutella | | | 100% | | | | | | | | | |
| 5 | Unassigned | | 100% | | | | | | | | | | |
| 6 | RTP | | | | | 100% | | | | | | | |
| 7 | Napster | | | | | | | | | | 100% | | |
| 8 | eDonkey | | 100% | | | | | | | | | | |
| 9 | AOL | | | 100% | | | | | | | | | |
| 10 | Multicast | 100% | | | | | | | | | | | |
| 11 | Half Life Server | | | | | | | | 100% | | | | |
| 12 | Plethora | | | | 100% | | | | | | | | |
| 13 | Reserved | | | | | | | | | | | | 100% |
| 14 | IRC | | | | | | | | | 100% | | | |
| 15 | ms-olap2 | | | | | | 100% | | | | | | |
| 16 | SMTP | | | | | | | 100% | | | | | |
| 17 | Half Life Client | | | | | | | | 100% | | | | |
| 18 | ICAP | | | | | | 100% | | | | | | |
| 19 | tragic | | 100% | | | | | | | | | | |
| 20 | mloadd | | 100% | | | | | | | | | | |
| 21 | Other | | | | 100% | | | | | | | | |

**Table 5.4 Clustering with Ward's minimum variance method on data set 3 using the 40-bin histogram scheme for 12 clusters**

| 12 clusters - 40bins | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CL14 | CL13 | CL24 | CL47 | CL32 | CL29 | CL12 | CL27 | CL16 | CL19 | CL18 | CL37 |
| 1 | HTTP | 100% | | | | | | | | | | | |
| 2 | Kazaa | | 100% | | | | | | | | | | |
| 3 | FTP | | | 100% | | | | | | | | | |
| 4 | Gnutella | | | | 100% | | | | | | | | |
| 5 | Unassigned | | 100% | | | | | | | | | | |
| 6 | RTP | | | | | 100% | | | | | | | |
| 7 | Napster | | | | | | 100% | | | | | | |
| 8 | eDonkey | 100% | | | | | | | | | | | |
| 9 | AOL | | | | | | | 100% | | | | | |
| 10 | Multicast | | | | | | | | 100% | | | | |
| 11 | Half Life Server | | | | | | | | | 100% | | | |
| 12 | Plethora | | | | | | | | | | 100% | | |
| 13 | Reserved | | | | | | | | | | | 100% | |
| 14 | IRC | | | | | | | | | | | | 100% |
| 15 | ms-olap2 | | 100% | | | | | | | | | | |
| 16 | SMTP | | | | | | | 100% | | | | | |
| 17 | Half Life Client | | | | | | | | | 100% | | | |
| 18 | ICAP | | 100% | | | | | | | | | | |
| 19 | tragic | | 100% | | | | | | | | | | |
| 20 | mloadd | | 100% | | | | | | | | | | |
| 21 | Other | | | | | | | | | | 100% | | |

**Table 5.5 Clustering with Ward's minimum variance method on data set 3 using the 60-bin histogram scheme for 13 clusters**

| 13 clusters - 60bins | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CL28 | CL13 | CL57 | CL14 | CL36 | CL15 | CL27 | CL39 | CL17 | CL34 | CL31 | CL23 | CL24 |
| 1 | HTTP | | 100% | | | | | | | | | | | |
| 2 | Kazaa | | | | | | 100% | | | | | | | |
| 3 | FTP | | | | | | | | | | | | 100% | |
| 4 | Gnutella | | | 100% | | | | | | | | | | |
| 5 | Unassigned | | 100% | | | | | | | | | | | |
| 6 | RTP | | | | | 100% | | | | | | | | |
| 7 | Napster | | | | | | | | | | | 100% | | |
| 8 | eDonkey | | 100% | | | | | | | | | | | |
| 9 | AOL | | | | | | | 100% | | | | | | |
| 10 | Multicast | 100% | | | | | | | | | | | | |
| 11 | Half Life Server | | | | | | | | | 100% | | | | |
| 12 | Plethora | | | | 100% | | | | | | | | | |
| 13 | Reserved | | | | | | | | | | | | | 100% |
| 14 | IRC | | | | | | | | | | 100% | | | |
| 15 | ms-olap2 | | | | | | 100% | | | | | | | |
| 16 | SMTP | | | | | | | | 100% | | | | | |
| 17 | Half Life Client | | | | | | | | | 100% | | | | |
| 18 | ICAP | | | | | | 100% | | | | | | | |
| 19 | tragic | | 100% | | | | | | | | | | | |
| 20 | mloadd | | 100% | | | | | | | | | | | |
| 21 | Other | | | | 100% | | | | | | | | | |

**Table 5.6 Clustering with Ward's minimum variance method on data set 3 using the 60-bin histogram scheme for 12 clusters**

| 12 clusters - 60bins | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CL28 | CL13 | CL12 | CL14 | CL36 | CL15 | CL39 | CL17 | CL34 | CL31 | CL23 | CL24 |
| 1 | HTTP | | 100% | | | | | | | | | | |
| 2 | Kazaa | | | | | | 100% | | | | | | |
| 3 | FTP | | | | | | | | | | | 100% | |
| 4 | Gnutella | | | 100% | | | | | | | | | |
| 5 | Unassigned | | 100% | | | | | | | | | | |
| 6 | RTP | | | | | 100% | | | | | | | |
| 7 | Napster | | | | | | | | | | 100% | | |
| 8 | eDonkey | | 100% | | | | | | | | | | |
| 9 | AOL | | | 100% | | | | | | | | | |
| 10 | Multicast | 100% | | | | | | | | | | | |
| 11 | Half Life Server | | | | | | | | 100% | | | | |
| 12 | Plethora | | | | 100% | | | | | | | | |
| 13 | Reserved | | | | | | | | | | | | 100% |
| 14 | IRC | | | | | | | | | 100% | | | |
| 15 | ms-olap2 | | | | | | 100% | | | | | | |
| 16 | SMTP | | | | | | | 100% | | | | | |
| 17 | Half Life Client | | | | | | | | 100% | | | | |
| 18 | ICAP | | | | | | 100% | | | | | | |
| 19 | tragic | | 100% | | | | | | | | | | |
| 20 | mloadd | | 100% | | | | | | | | | | |
| 21 | Other | | | | 100% | | | | | | | | |

**Table 5.7 Clustering with Ward's minimum variance method on data set 3 using the 60-bin histogram scheme for 11 clusters**

| 11 clusters - 60bins | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CL28 | CL13 | CL12 | CL14 | CL36 | CL15 | CL39 | CL11 | CL31 | CL23 | CL24 |
| 1 | HTTP | | 100% | | | | | | | | | |
| 2 | Kazaa | | | | | | 100% | | | | | |
| 3 | FTP | | | | | | | | | | 100% | |
| 4 | Gnutella | | | 100% | | | | | | | | |
| 5 | Unassigned | | 100% | | | | | | | | | |
| 6 | RTP | | | | | 100% | | | | | | |
| 7 | Napster | | | | | | | | | 100% | | |
| 8 | eDonkey | | 100% | | | | | | | | | |
| 9 | AOL | | | 100% | | | | | | | | |
| 10 | Multicast | 100% | | | | | | | | | | |
| 11 | Half Life Server | | | | | | | | 100% | | | |
| 12 | Plethora | | | | 100% | | | | | | | |
| 13 | Reserved | | | | | | | | | | | 100% |
| 14 | IRC | | | | | | | | 100% | | | |
| 15 | ms-olap2 | | | | | | 100% | | | | | |
| 16 | SMTP | | | | | | | 100% | | | | |
| 17 | Half Life Client | | | | | | | | 100% | | | |
| 18 | ICAP | | | | | | 100% | | | | | |
| 19 | tragic | | 100% | | | | | | | | | |
| 20 | mloadd | | 100% | | | | | | | | | |
| 21 | Other | | | | 100% | | | | | | | |

**Table 5.8 Clustering with Ward's minimum variance method on data set 3 using the 60-bin histogram scheme for 10 clusters**

| 10 clusters - 60bins | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CL28 | CL13 | CL12 | CL10 | CL36 | CL39 | CL11 | CL31 | CL23 | CL24 |
| 1 | HTTP | | 100% | | | | | | | | |
| 2 | Kazaa | | | | 100% | | | | | | |
| 3 | FTP | | | | | | | | | 100% | |
| 4 | Gnutella | | | 100% | | | | | | | |
| 5 | Unassigned | | 100% | | | | | | | | |
| 6 | RTP | | | | | 100% | | | | | |
| 7 | Napster | | | | | | | | 100% | | |
| 8 | eDonkey | | 100% | | | | | | | | |
| 9 | AOL | | | 100% | | | | | | | |
| 10 | Multicast | 100% | | | | | | | | | |
| 11 | Half Life Server | | | | | | | 100% | | | |
| 12 | Plethora | | | | 100% | | | | | | |
| 13 | Reserved | | | | | | | | | | 100% |
| 14 | IRC | | | | | | | 100% | | | |
| 15 | ms-olap2 | | | | 100% | | | | | | |
| 16 | SMTP | | | | | | 100% | | | | |
| 17 | Half Life Client | | | | | | | 100% | | | |
| 18 | ICAP | | | | 100% | | | | | | |
| 19 | tragic | | 100% | | | | | | | | |
| 20 | mloadd | | 100% | | | | | | | | |
| 21 | Other | | | | 100% | | | | | | |

**Table 5.9 Clustering with Ward's minimum variance method on data set 3 using the 60-bin histogram scheme for 9 clusters**

| 9 clusters - 60bins | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | CL28 | CL13 | CL9 | CL10 | CL36 | CL11 | CL31 | CL23 | CL24 |
| 1 | HTTP | | 100% | | | | | | | |
| 2 | Kazaa | | | | 100% | | | | | |
| 3 | FTP | | | | | | | | 100% | |
| 4 | Gnutella | | | 100% | | | | | | |
| 5 | Unassigned | | 100% | | | | | | | |
| 6 | RTP | | | | | 100% | | | | |
| 7 | Napster | | | | | | | 100% | | |
| 8 | eDonkey | | 100% | | | | | | | |
| 9 | AOL | | | 100% | | | | | | |
| 10 | Multicast | 100% | | | | | | | | |
| 11 | Half Life Server | | | | | | 100% | | | |
| 12 | Plethora | | | | 100% | | | | | |
| 13 | Reserved | | | | | | | | | 100% |
| 14 | IRC | | | | | | 100% | | | |
| 15 | ms-olap2 | | | | 100% | | | | | |
| 16 | SMTP | | | 100% | | | | | | |
| 17 | Half Life Client | | | | | | 100% | | | |
| 18 | ICAP | | | | 100% | | | | | |
| 19 | tragic | | 100% | | | | | | | |
| 20 | mloadd | | 100% | | | | | | | |
| 21 | Other | | | | 100% | | | | | |

**Table 5.10 Clustering with Ward's minimum variance method on data set 3 using the 60-bin histogram scheme for 8 clusters**

| 8 clusters - 60bins | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | CL28 | CL8 | CL9 | CL10 | CL36 | CL11 | CL23 | CL24 |
| 1 | HTTP | | 100% | | | | | | |
| 2 | Kazaa | | | | 100% | | | | |
| 3 | FTP | | | | | | | 100% | |
| 4 | Gnutella | | | 100% | | | | | |
| 5 | Unassigned | | 100% | | | | | | |
| 6 | RTP | | | | | 100% | | | |
| 7 | Napster | | 100% | | | | | | |
| 8 | eDonkey | | 100% | | | | | | |
| 9 | AOL | | | 100% | | | | | |
| 10 | Multicast | 100% | | | | | | | |
| 11 | Half Life Server | | | | | | 100% | | |
| 12 | Plethora | | | | 100% | | | | |
| 13 | Reserved | | | | | | | | 100% |
| 14 | IRC | | | | | | 100% | | |
| 15 | ms-olap2 | | | | 100% | | | | |
| 16 | SMTP | | | 100% | | | | | |
| 17 | Half Life Client | | | | | | 100% | | |
| 18 | ICAP | | | | 100% | | | | |
| 19 | tragic | | 100% | | | | | | |
| 20 | mloadd | | 100% | | | | | | |
| 21 | Other | | | | 100% | | | | |

**Table 5.11 Clustering with Ward's minimum variance method on data set 3 using the 60-bin histogram scheme for 7 clusters**

| 7 clusters - 60bins | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | CL28 | CL8 | CL9 | CL7 | CL36 | CL11 | CL24 |
| 1 | HTTP | | 100% | | | | | |
| 2 | Kazaa | | | | 100% | | | |
| 3 | FTP | | | | 100% | | | |
| 4 | Gnutella | | | 100% | | | | |
| 5 | Unassigned | | 100% | | | | | |
| 6 | RTP | | | | | 100% | | |
| 7 | Napster | | 100% | | | | | |
| 8 | eDonkey | | 100% | | | | | |
| 9 | AOL | | | 100% | | | | |
| 10 | Multicast | 100% | | | | | | |
| 11 | Half Life Server | | | | | | 100% | |
| 12 | Plethora | | | | 100% | | | |
| 13 | Reserved | | | | | | | 100% |
| 14 | IRC | | | | | | 100% | |
| 15 | ms-olap2 | | | | 100% | | | |
| 16 | SMTP | | | 100% | | | | |
| 17 | Half Life Client | | | | | | 100% | |
| 18 | ICAP | | | | 100% | | | |
| 19 | tragic | | 100% | | | | | |
| 20 | mloadd | | 100% | | | | | |
| 21 | Other | | | | 100% | | | |

# 6. Mathematical Model

In this chapter we have suggested a method to develop the mathematical model for the packet size distribution of the applications using the example of the HTTP application. The first section describes Pareto distribution which is used to model the applications and the reason for using it for modeling. The next section describes how the model was generated. The last section uses the model to simulate the applications and checks the validity of the model.

## *A. Pareto Distribution*

The *Pareto distribution* named after the Italian economist Vilfredo Pareto is a power law distribution found in a large number of real-world situations. The Pareto principle, also known as 'the 80-20 rule,' states that for many phenomena 80% of consequences stem from 20% of the causes.

In contrast to the circuit switched voice networks which can be modeled well with Poisson model, the packet switched networks have very different characteristics. It exhibits long range dependencies i.e. the correlations that persist (do not degenerate) across large time scales. Recent work argues convincingly that Ethernet local area network (LAN) traffic, which is extremely bursty, is much better modeled using statistically self-similar processes [27]. Self-similarity [28] phenomenon, which have much different theoretical properties than Poisson processes, display structural similarities over a wide range of time-scales. Poisson based models assume that aggregate traffic becomes smoother (less bursty) as the number of traffic sources increases. In fact, the burstiness (degree of self-similarity) of LAN traffic typically intensifies as the number of active traffic sources increases. The Poisson process accurately describes the connection arrivals but it can be seen from [27] and [29] that it seriously underestimates the burstiness of the TCP traffic over a wide range of time scales. It has been shown in [30] that the superposition of many (strictly alternating) independent and identically distributed (i.i.d.) ON/OFF sources, each of which exhibits heavy tailed distribution, results in self similar aggregate traffic. Heavy-tailed

distributions, such as Pareto distribution, have the property that while the majority of tasks are very small, more than half of the total work is made up by a tiny fraction of the largest tasks. The sum of multiple Pareto ON/OFF processes can be used to model the packet size distribution of the network traffic.

If X is a random variable with Pareto distribution, then the probability and the cumulative distribution function of X is characterized as:

$$f(x) = \begin{cases} \dfrac{ab^a}{x^{a+1}} & for\ x \geq b \\ 0 & for\ x < b \end{cases} \tag{6.1}$$

$$F(x) = \begin{cases} 1 - \left(\dfrac{b}{x}\right)^a & for\ x \geq b \\ 0 & for\ x < b \end{cases} \tag{6.2}$$

where, $a$ is the shape parameter and $b$ is the location parameter.

The mean and the variance of Pareto distribution are

$$E(x) = \begin{cases} \infty & a \leq 1 \\ \dfrac{ab}{(1-a)} & a > 1 \end{cases} \tag{6.3}$$

$$Var(x) = \begin{cases} \infty & a \leq 2 \\ \dfrac{ab^2}{(a-1)^2(a-2)} & a > 2 \end{cases} \tag{6.4}$$

Note that for 1<a<2, Pareto distribution has finite mean and infinite variance. The lower the shape parameter *a*, the more pronounced is the heavy-tailed property. Pareto distributions are continuous probability distributions as shown in Figure 6.1.

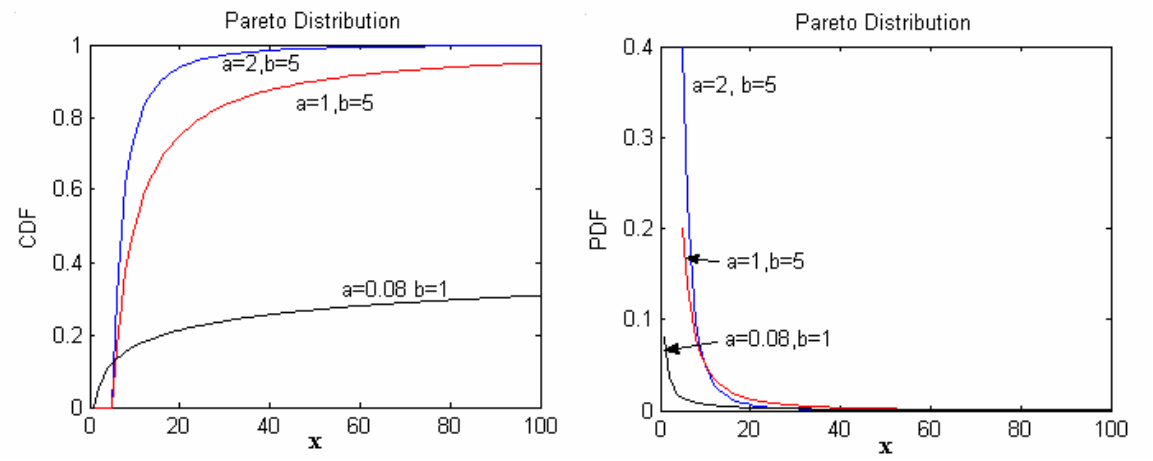



**Figure 6.1 PDF and CDF of Pareto for different values of a and b**

## *B. Modeling the applications*

The complementary cumulative distribution function (CCDF) of HTTP for data set 3, shown in Figure 6.2 for all packet sizes can be modeled as sum of Pareto distributions. CCDF for Pareto distribution is defined in Equation 6.5.

$$\boldsymbol{CCDF\ or\ F'(x) = 1 - F(x)} \qquad (6.5)$$

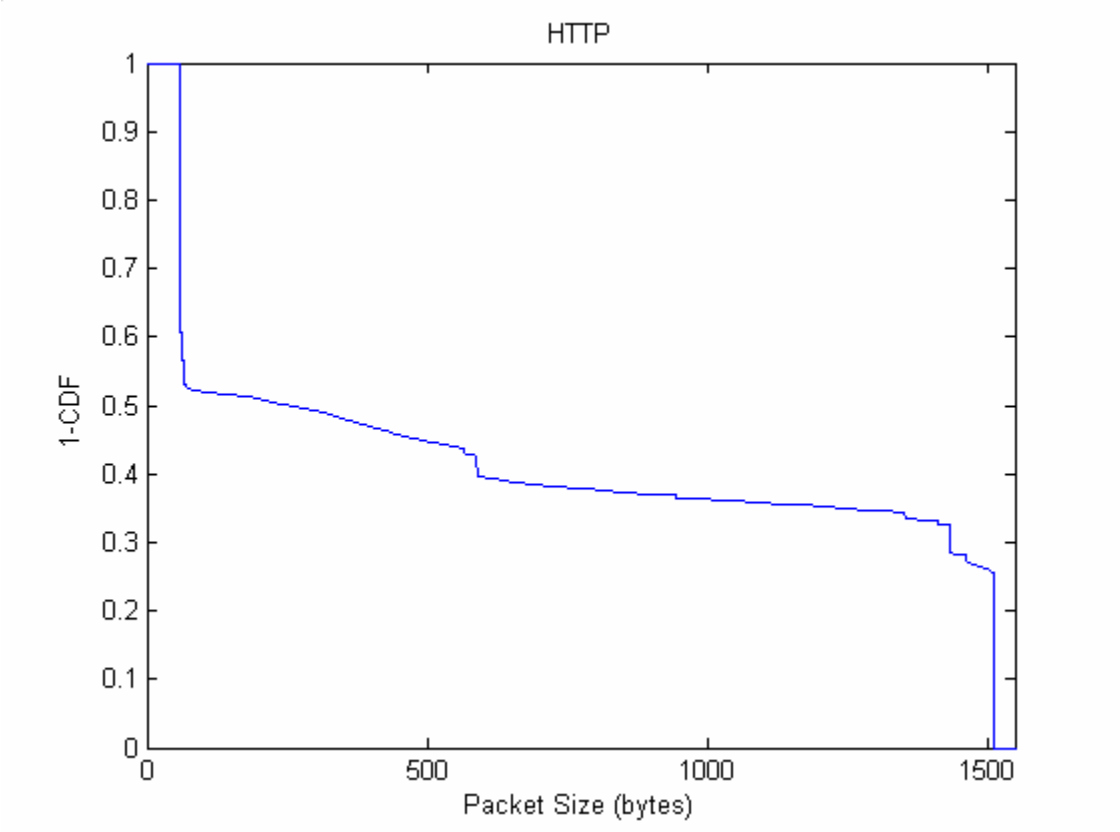



**Figure 6.2 Complementary CDF (CCDF) for HTTP in data set 3 for entire packet size range**

It is clear that the application can be modeled piecewise with Pareto distributions in different ranges as shown mathematically in Equation 6.6. We have modified the cumulative distribution function defined in Equation 6.2 for our model to include a scaling factor *c*.

$$F'(x) = \sum_i c_i \left( \frac{b_i}{x} \right)^{a_i} \quad where\ b_1 \le x < b_2 \tag{6.6}$$

Figure 6.3 shows the Pareto model in dotted line corresponding to the original plot in solid line. The plot is divided into three ranges and each part is modeled separately. The

Table 6.1 summarizes the range, the values of the parameters and the mean square error for each range.

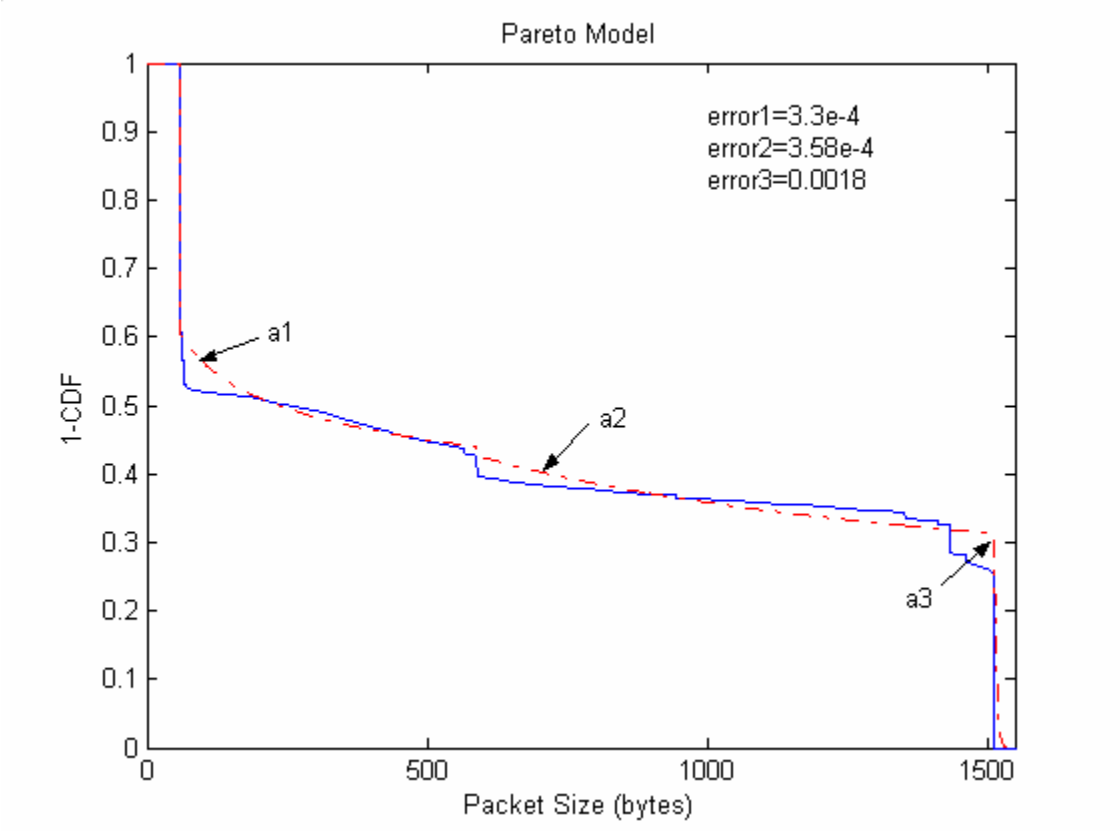



**Figure 6.3 Pareto Model**

**Table 6.1 Pareto: the range, the values of the parameters and the mean square error**

| Range (bins) | a | b | c | Mean Sq. Error |
|---|---|---|---|---|
| 60-589 | 0.14 | 60 | 0.60551 | 3.30E-04 |
| 590-1513 | 0.33 | 590 | 0.42684 | 3.58E-04 |
| 1514-1600 | 400 | 1514 | 0.25642 | 0.0018 |

The packet size distribution generated with the Pareto model is shown in Figure 6.4 with the solid line whereas the original packet size distribution is shown with dotted line. The error between the original packet size distribution and the one generated with the Pareto model is shown in Figure 6.5.

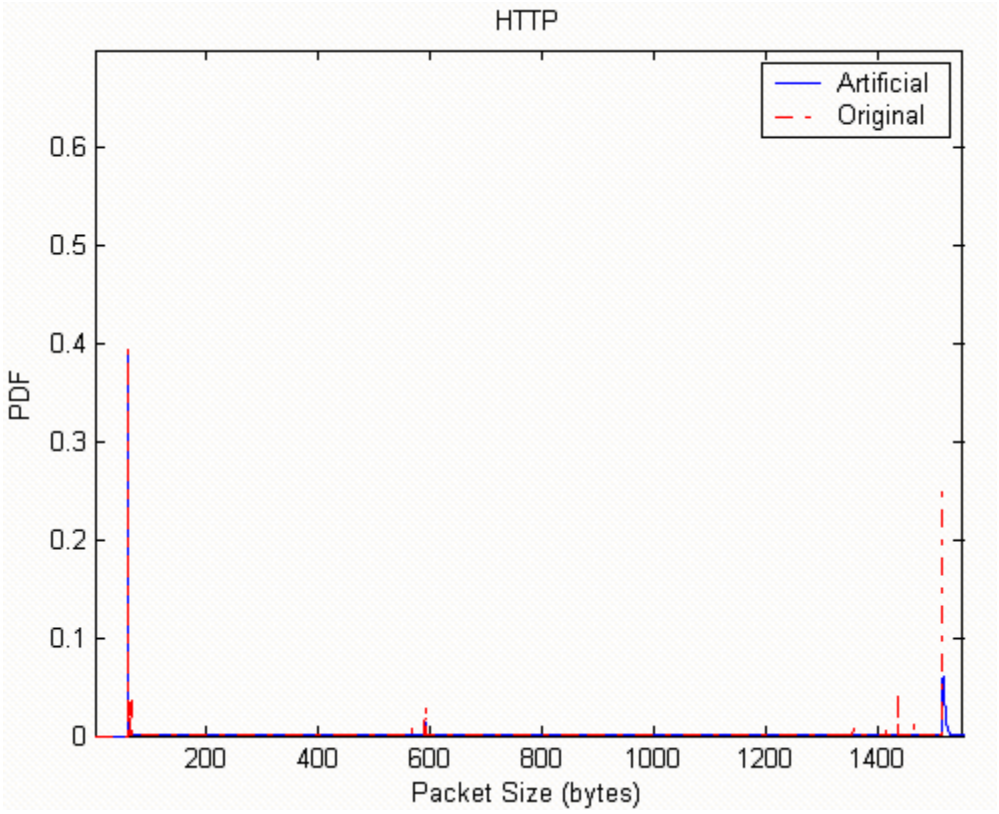



**Figure 6.4 The packet size distribution generated with the Pareto model**

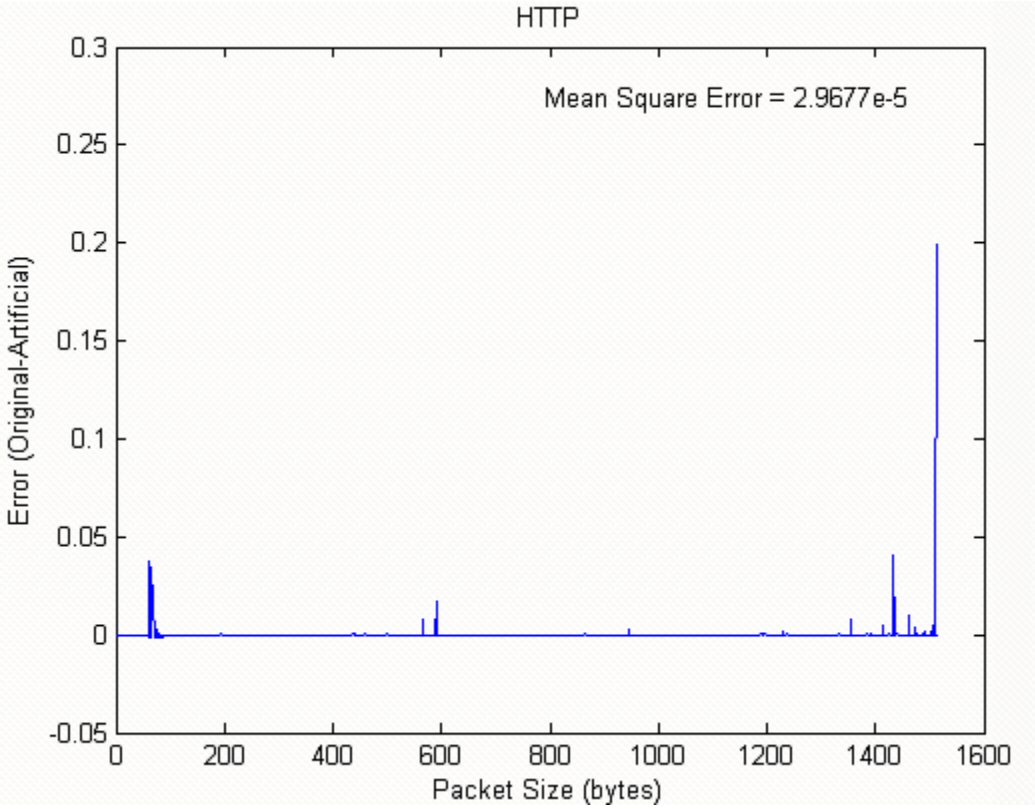



**Figure 6.5 Error between the original and artificial packet size distribution**

### *C. Simulating applications using Pareto model*

The Pareto model of HTTP is used to simulate the presence of the application in the network traffic. A random generator is used to generate a random number between 0 and 1. Depending on the range in which this number falls in the cumulative distribution plot, the counter of the corresponding histogram bin is incremented. Since the data set 3 consisted about 2 million HTTP packets in each data file, this process is repeated 2 million times for each data file to obtain a distribution with characteristics similar to the original data. The normalized average packet size distribution of the artificially generated

HTTP with 2 million HTTP packets in each of the data file is shown in Figure 6.7 and the original packet size distribution of HTTP is shown in Figure 6.6. The results of the classification performed by replacing the original HTTP traffic with artificially generated HTTP in data set 3 for 50-bin histogram scheme, keeping the remaining traffic as is were the almost the same as the results generated with the original HTTP traffic (Table 5.2). HTTP falls with Unassigned, tragic and mloadd in original traffic but in the case of artificially generated HTTP, it falls with eDonkey and FTP. However in both the cases, all the samples of HTTP fall together in one cluster. The groups of certain other applications are also different from the case of the original HTTP traffic. For example, the Napster application falls in a singleton cluster in the original HTTP case whereas it falls with 'Other' in the artificial HTTP case. The results of classification with artificial HTTP traffic for 12 clusters, 10 clusters and 8 clusters are shown in Table 6.2, Table 6.3 and Table 6.4 respectively.

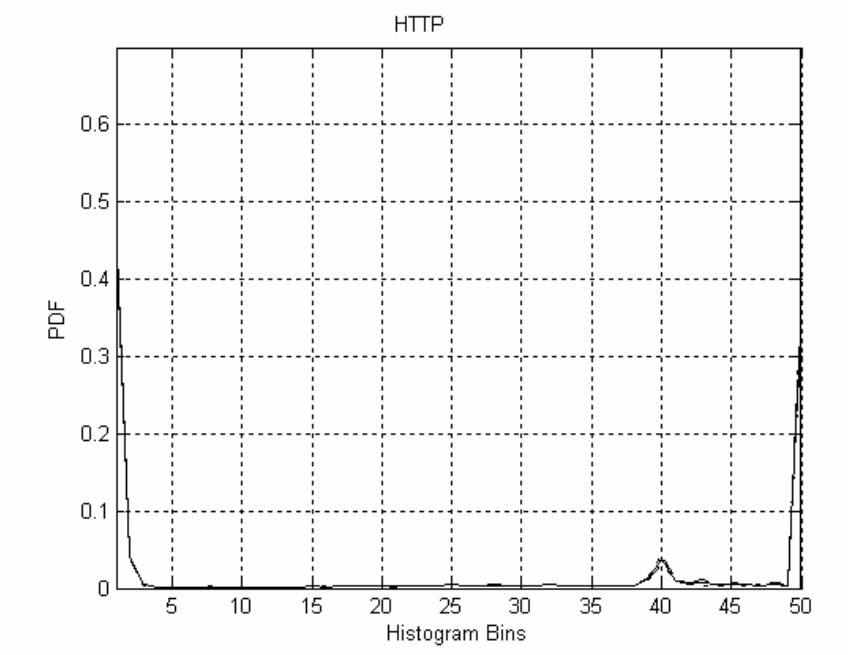



**Figure 6.6 Original HTTP**

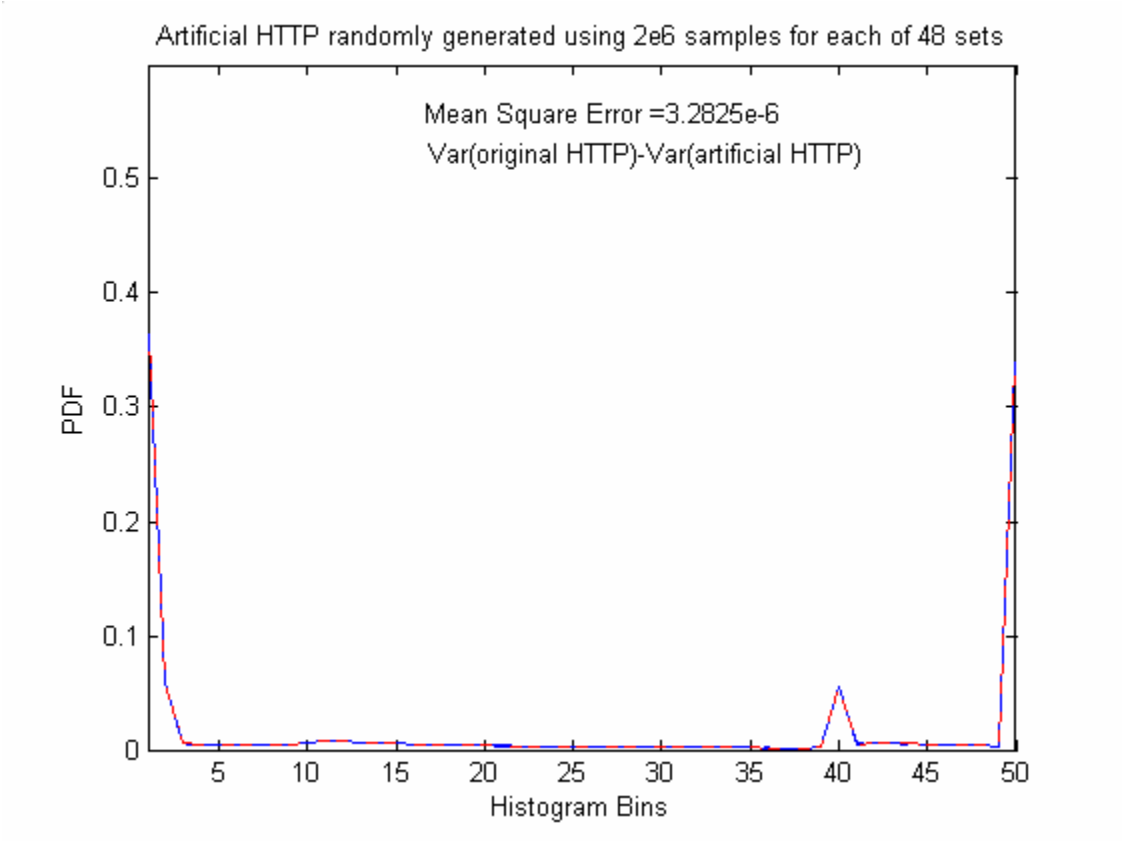



**Figure 6.7 Artificially generated HTTP**

**Table 6.2 Clustering with Ward's Method on data set 3 with artificial HTTP traffic using 50-bin histogram scheme for 12 clusters**

| app name | CL60 | CL26 | CL15 | CL91 | CL12 | CL77 | CL55 | CL64 | CL21 | CL13 | CL120 | CL46 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HTTP | | | | | 100.00% | | | | | | | |
| Kazaa | | | | | | | | | | | 100.00% | |
| FTP | | | | | 100.00% | | | | | | | |
| Gnutella | | | | | | | | | 100.00% | | | |
| Unassigned | | | | | | | | 68.75% | 31.25% | | | |
| RTP | | | | | | | 100.00% | | | | | |
| Napster | | | | 100.00% | | | | | | | | |
| eDonkey | | | | | 100.00% | | | | | | | |
| AOL | | | | | | | | | | 100.00% | | |
| Multicast | | | | | | 100.00% | | | | | | |
| Half Life Server | 95.83% | | | | | | | | 4.17% | | | |
| Plethora | | | | | | | | | 100.00% | | | |
| Reserved | | 100.00% | | | | | | | | | | |
| IRC | | | 100.00% | | | | | | | | | |
| ms-olap2 | | | | | | | | | 60.42% | | 39.58% | |
| SMTP | | | | | | | | | | | | 100.00% |
| Half Life Client | 100.00% | | | | | | | | | | | |
| ICAP | | | | | | | | | 29.17% | | 70.83% | |
| tragic | | | | | | | | 62.50% | | | 37.50% | |
| mloadd | | | | | | | | 79.17% | | | 20.83% | |
| Other | | | | 100.00% | | | | | | | | |

**Table 6.3 Clustering with Ward's Method on data set 3 with artificial HTTP traffic using 50-bin histogram scheme for 10 clusters**

| app name | CL10 | CL19 | CL26 | CL12 | CL13 | CL28 | CL11 | CL29 | CL44 | CL73 |
|---|---|---|---|---|---|---|---|---|---|---|
| HTTP | | | | | 100.00% | | | | | |
| Kazaa | | | | | | | 100.00% | | | |
| FTP | | | | | 100.00% | | | | | |
| Gnutella | 100.00% | | | | | | | | | |
| Unassigned | 31.25% | | | | | | 68.75% | | | |
| RTP | | | | | | | | 100.00% | | |
| Napster | | | | 100.00% | | | | | | |
| eDonkey | | | | | 100.00% | | | | | |
| AOL | | | | | | | | | 100.00% | |
| Multicast | | | | | | 100.00% | | | | |
| Half Life Server | 4.17% | 95.83% | | | | | | | | |
| Plethora | 100.00% | | | | | | | | | |
| Reserved | 100.00% | | | | | | | | | |
| IRC | | | 100.00% | | | | | | | |
| ms-olap2 | 60.42% | | | | | | 39.58% | | | |
| SMTP | | | | | | | | | | 100.00% |
| Half Life Client | | 100.00% | | | | | | | | |
| ICAP | 29.17% | | | | | | 70.83% | | | |
| tragic | | | | | | | 100.00% | | | |
| mloadd | | | | | | | 100.00% | | | |
| Other | | | | 100.00% | | | | | | |

**Table 6.4 Clustering with Ward's Method on data set 3 with artificial HTTP traffic using 50-bin histogram scheme for 8 clusters**

| app name | CL10 | CL19 | CL8 | CL12 | CL13 | CL28 | CL11 | CL29 |
|---|---|---|---|---|---|---|---|---|
| HTTP | | | | | 100.00% | | | |
| Kazaa | | | | | | | 100.00% | |
| FTP | | | | | 100.00% | | | |
| Gnutella | 100.00% | | | | | | | |
| Unassigned | 31.25% | | | | | | 68.75% | |
| RTP | | | | | | | | 100.00% |
| Napster | | | | 100.00% | | | | |
| eDonkey | | | | | 100.00% | | | |
| AOL | | | 100.00% | | | | | |
| Multicast | | | | | | 100.00% | | |
| Half Life Server | 4.17% | 95.83% | | | | | | |
| Plethora | 100.00% | | | | | | | |
| Reserved | 100.00% | | | | | | | |
| IRC | | | 100.00% | | | | | |
| ms-olap2 | 60.42% | | | | | | 39.58% | |
| SMTP | | | 100.00% | | | | | |
| Half Life Client | | 100.00% | | | | | | |
| ICAP | 29.17% | | | | | | 70.83% | |
| tragic | | | | | | | 100.00% | |
| mloadd | | | | | | | 100.00% | |
| Other | | | | 100.00% | | | | |

# 7. Summary

The first section in this chapter summarizes what we have learnt in this research based on the results stated in this thesis document. The next two sections discuss the future directions of the research. Based on the results generated in this research, the classification of the Internet traffic mixture based on the packet size distributions seems promising. The results stated in this chapter have been included in the papers "submitted for publication" [31] [32]. The flow classification can be applied to a number of network services, such as access-control in firewalls, policy-based routing, provision of differentiated qualities of service, and possibly, in future, traffic billing.

## *A. Conclusion*

We have shown that using the packet size distributions and the clustering algorithms, it is possible to classify the major applications of the Internet traffic. Different applications display different packet size distributions depending on the characteristics of the applications and the protocols that it uses, e.g., TCP, UDP, etc. Since our classification is based on the packet size distribution which can be generated from the packet size information at OSI layer 3 i.e. IP layer, we can conclude that the classification of the Internet traffic using features that are intrinsic characteristics of the packets, and features that cannot be spoofed so easily, is possible [31] [32]. We can generate good clustering results by carefully choosing the histogram bin sizes to capture the distinct characteristics of the applications of interest.

Pareto distribution can be used to model the packet size distributions of the applications sufficiently well. Pareto models for individual applications can be used to simulate the application traffic in the traffic mixture. The artificially generated applications using the Pareto model have characteristics similar to the original traffic and generate almost same clustering results as those generated by original traffic.

## *B. Estimation of mixtures*

The total distribution of packet sizes at a particular network node, e.g. switch, router, is the mixture of the distribution of the individual applications. If the proportion of each application can be estimated accurately or the presence of a certain specified application can be detected, then a higher quality of service (QoS) can be obtained by giving preference to the flows of interest. The simulations and results of estimation of mixtures were generated by Yi Wang as a part of this research project.

We can model the total network traffic as the linear combination of major applications, as

$$E\{\mathbf{h}\} = E\{\mathbf{B}\}\mathbf{a} \tag{7.1}$$

where $\boldsymbol{E}\{.\}$ is the expected value operator, $\mathbf{h}_{\mathbf{M}\times 1}$ is the total network traffic distribution, given as an M bins histogram, $\mathbf{B}_{\mathbf{M}\times\mathbf{N}}$ is the distribution of M bins of a histogram of each of the top N-1 applications and 'Other', and $\mathbf{a}_{\mathbf{N}\times 1}$ is the proportion of each of the N-1 applications and 'Other'. We know $\mathbf{h}$ by collecting packet samples over a fixed period of time and we know $\mathbf{B}$ in a statistical sense. Since the components of the mixture are probability distributions, the elements of $\mathbf{a}$ are constrained to the set $\mathbf{S}_\mathbf{a}$ defined in Equation 7.2.

$$\mathbf{S}_\mathbf{a} = \left\{ \mathbf{a} \in \mathbf{R}^\mathbf{N} \left| \sum_{\mathbf{k}=1}^{\mathbf{N}} \mathbf{a}_\mathbf{k} = 1, \quad 0 \le \mathbf{a}_\mathbf{k} \le 1 \right. \right\} \tag{7.2}$$

We can estimate $\boldsymbol{E}\{\boldsymbol{B}\}$, denoted as $\overline{\mathbf{B}}$, from observations. For any particular time interval, we can estimate $\mathbf{a}$ from

$$\mathbf{h} = \overline{\mathbf{B}}\mathbf{a} \tag{7.3}$$

We have used Constrained Linear Least Squares (CLLSQ), Projection onto Convex Sets (POCS) and artificial neural network methods to perform estimation of the traffic mixture. Initial results generated by Yi Wang show that, the neural networks method gives the best performance. The proportions of the applications in the network traffic could be estimated with average rms (root-mean square) error 0.0012 using the neural networks method [31] [32].

Detection of the presence of a single application in the network traffic mixture generates more accurate results. The neural networks method gives the best performance in detection as well. Initial results generated by Yi Wang show that, the presence of RTP in the network traffic mixture could be detected with 100% accuracy using the neural networks method [31] [32]. The proportion of RTP in the traffic mixture was varied before performing detection to generate reliable results.

### *C. Prediction of traffic*

In order to obtain the desired QoS, it is more useful if we are able to predict the proportions of the applications in the next time interval. Depending on the predicted

proportion of the applications of interest, the scheduler can be configured to reserve space in the queue and give priority to the desired flow.

Adaptive filters can be used for prediction. Adaptive filters are digital filters capable of self adjustment. These filters can change in accordance to their input signals. An adaptive filter has the ability to update its coefficients. New coefficients are sent to the filter from a coefficient generator, which is an adaptive algorithm that modifies the coefficients in response to an incoming signal.

Here we have proposed a method by which the proportion of the applications in the Internet traffic mixture can be predicted using adaptive filters. Figure 7.1 shows a block diagram of an adaptive filter model. There are two inputs to the model:

- The input signal *x(n)* which is the histogram of the actual traffic
- The reference input *a(n)* which is the proportion of the application in the traffic

The unknown system, which in this case is the proportion of the applications in traffic mixture is modeled by an FIR filter with adjustable coefficients. Both the unknown system and FIR filter model are excited by an input sequence *x(n),* which is the histogram of actual traffic. The adaptive FIR filter output $\hat{a}(n)$, which is the predicted value of the application proportions, is compared with the unknown system output *a(n),* which is the actual proportion of the application in the traffic, to produce the error signal *e(n).* The error signal represents the difference between the actual and the predicted application

proportions. The error *e(n)* is then used as the input to an adaptive control algorithm, which corrects the individual tap weights of the filter. This process is repeated through several iterations until the error signal *e(n)* becomes sufficiently small. The resultant FIR filter response now represents that of the previously unknown system.

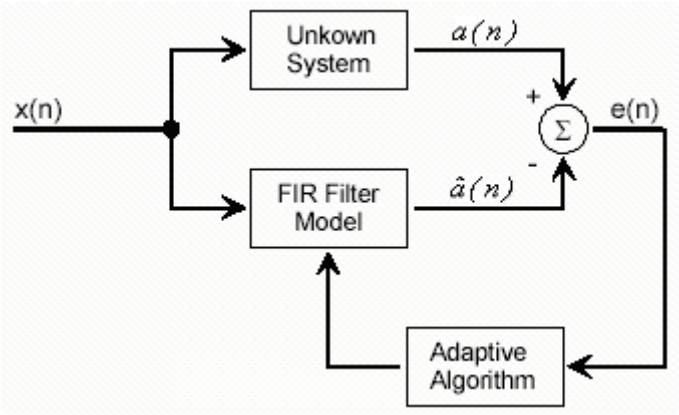



**Figure 7.1 Adaptive Filter Model**

As discussed earlier, we may not be able to use the port numbers information in future to generate the actual proportions of the applications in the traffic mixture due to NAPT and possible header encryption. In this case, we can use estimation to estimate the proportions of the applications based on the models that we have generated for the applications of interest. The FIR filter model in this case would try to predict the estimated value of the proportion of the applications.

## REFERENCES


[1] P. Gupta and N. McKeown, "Algorithms for packet classification," *IEEE Network*, pp. 24-32, Vol.15:2, 2001.

[2] Chintan Trivedi, H. Joel Trussell, Arne A. Nilsson and Mo-Yuen Chow, "Implicit traffic classification for service differentiation," *ITC Workshop*, July 24&25, 2002, Wurtzburg, Germany.

[3] Andrew S. Tanenbaum, *Computer Networks*, 3rd Edition, Prentice Hall 1996.

[4] W. Richard Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1st Edition, 1994.

[5] RFC 1042, "A Standard for the Transmission of IP Datagrams over IEEE 802 Networks." Available: http://www.ietf.org/rfc/rfc1042.txt

[6] RFC 1191, "Path MTU Discovery." Available: http://www.ietf.org/rfc/rfc1191.txt

[7] Wu-chang Feng, Francis Chang, Wu-chi Feng and Jonathan Walpole. "Provisioning on-line games: A traffic analysis of a busy counter-strike server," *In Proceedings of Internet Measurement Workshop (IMW)*, November 6-8, 2002, Marseille, France.

[8] Port Number Assignments by Internet Assigned Numbers Authority (IANA), Available: http://www.iana.org/assignments/port-numbers

[9] RFC 1889, "RTP: A Transport Protocol for Real-Time Applications." Available: http://www.ietf.org/rfc/rfc1889.txt

[10] Plethora Product Overview. Available: http://www.plethoratech.com/product/

[11] RFC 3507, "Internet Content Adaptation Protocol (ICAP)." Available: http://www.ietf.org/rfc/rfc3507.txt

[12] J. Hair, R. Anderson, R. Tatham, and W. Black, *Multivariate Data Analysis with Readings,* Prentice-Hall, 4th Edition, 1995.

[13] Ivan Popchev and Vania Peneva, "Cluster – A package for cluster analysis," *In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society,* November 1988.

[14] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[15] B. S. Everitt, *Cluster Analysis*, Heinemann Educational Books, 1974.

[16] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Clustering algorithms and validity measures," *In Proceedings of the Thirteenth International Conference on Scientific and Statistical Database Management (SSDBM)*, July 2001, Fairfax, Virginia.

[17] A. K. Jain, M. N. Murty and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys (CSUR)*, Vol. 31, No.3, pp 264-323, September 1999.

[18] D. H. Fisher, "Knowledge acquisition via incremental conceptual clustering," Machine Learning, Vol. 2, No. 2, pp. 139-172, September 1987.

[19] T. Kohonen, *Self-Organizing Maps*, 2nd Edition, Springer-Verlag, 1997.

[20] M. Sato-Ilic, "On clustering based on homogeneity," *Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, Vol. 5, pp. 2505-2510, July 2001.

[21] Cluster Procedure, SAS/STAT User's Guide, *SAS Online Documentation*, Ver. 8.

[22] Brain S. Everitt, *Cluster Analysis*, 3rd Edition, Edward Arnold, 1993.

[23] J. H. Ward, "Hierarchical groupings to optimize an objective function," *Journal of the American Statistical Association*, Vol.58, No. 301, pp.234--244, Mar 1963.

[24] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” *In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, pp. 281-298, 1967.

[25] C. T. Zahn, “Graph-theoretical methods for detecting and describing gestalt clusters,” *IEEE Transactions on Computers*, Vol. C-20, pp 68–86, 1971.

[26] M. Dash, K. Choi, P. Scheuermann and H. Liu, “Feature selection for clustering- A filter solution,” *In Proceedings of 2002 IEEE International Conference on Data Mining (ICDM)*, pp. 115-122, December 2002.

[27] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, “On the self-similar nature of Ethernet traffic (Extended Version),” *IEEE/ACM Transactions on Networking*, Vol. 2, No. 1, pp. 1-15, February 1994.

[28] B. B. Mandelbrot, J. W. Van Ness, "Fractional Brownian Motions, Fractional Noises and Applications", *SIAM Review*, Vol. 10, pp. 422-437, 1968.

[29] V. Paxson and S. Floyd, “Wide-area traffic: the failure of Poisson modeling,” *IEEE/ACM Transactions on Networking,* Vol. 3, No. 3, pp. 226-244, June 1995.

[30] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson. “Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level,” In *Proceedings of ACM SIGCOMM '95*, pp. 100-113, 1995.

[31] H. J. Trussell, Arne Nilsson, Parita Patel, Yi Wang, “Internet Application Traffic Detection and Estimation,” *IEEE Transactions on Neural Networks*.

[32] H. J. Trussell, Arne Nilsson, Parita Patel, Yi Wang, “Internet Application Traffic Detection and Estimation,” *ICASSP 2004*, 17-21 May 2004, Montreal.

# APPENDIX A

- ***SAS Code to generate clustering results***

```
/*
The input to the SAS code is generated by c:\research\matlab-files\clu_ps.m
This code uses Wards minimum method for clustering. It generates sas_results.txt as the
output file which shows in which clusters the applications fall.                    */


PROC IMPORT OUT= WORK.data
            DATAFILE= "C:\MyResearch\Cluster\sas_input.txt"
/* Directory Path of input file */
            DBMS=TAB REPLACE;
     GETNAMES=YES; /* Get Variable names from the file */
     DATAROW=2;
RUN;
proc print data=work.data;
title 'Cluster 12 : Data Set 3 (48 sets): Top 20 Apps + Other';
run;
proc cluster data=work.data method=ward outtree=tree;
        id name;
run;

proc tree N=12 out=out;
        id name;
run;

proc export data=work.out
outfile="C:\MyResearch\Cluster\sas_results.txt" replace;
/* Directory Path of output file */

run;
```

- ***Code to generate input to the SAS program***

```
% Hist.dat contains histograms of the packet size distributions of the applications of
%interest. This code massages the data and creates an ascii file (sas_input.txt) that is
%used as input to SAS.

clear all;close all;clc;

Hist = load('C:\MyResearch\HistData\50_bins\HistSrcDst.dat');

log_trans = 1;
```

```
ind_scale = 1;
bins=50;
apps=21;                                        % no. of applications of interest
for q=1:apps
  for p=1:48   % p= no. of sets
        hist_input = Hist(((p-1)*101 + q),:);
        tran = hist_input';
        sum_h = sum(tran);
        if sum_h~=0
           hist_input1=tran/sum_h;              % if statement added to remove NaN
        else
           hist_input1=0;
        end

        if log_trans == 1
           R_clip = max(hist_input1,10^(-5));           % set limit for range
           R_log = log10(R_clip);
           Rt = R_log;
        else
           Rt = hist_input1;
        end

        if ind_scale ==1
           Rmax = max(max(Rt));
           Rmin = min(min(Rt));
           if Rmax==Rmin                %if statement to remove NaN due to divide by 0
              R_out = 0;
           else
              R_out = (Rt - Rmin)*255/(Rmax - Rmin);
           end
        else
           R_out = Rt;
        end

       R_out1((q-1)*48+p,:) = R_out';
   end
end
disp('Done')
save sas_input.txt R_out1 -ascii -tabs;
```

- ***Code to generate the Pareto Model and the packet size distribution based on the Pareto model***

*% First part of this code generates the Pareto approximation for the Complementary CDF % of the actual application traffic and generates a plot to compare the two.*

```
% Second part of this code generates the artificial packet size distribution based on the
Pareto model and compares it with actual packet size distribution

clc; clf; clear all;
load C:\MyResearch\HistData\1500_bins\HistSrcDst.dat

Range=[61,590,1514,1601];  % Model Pareto separately in  this packet size ranges
alphaRange=[0.01, 0.01, 1; 0.1, 0.01, 1; 300, 0.1, 400]
n=1;                       % HTTP
   app(n,1514)=0;          % Initialize
   for i=1:48              % no. of data sets = 48
      app(n,:) = app(n,:) + HistSrcDst(n + (21*(i-1)),:);
   end
   sumapp(n,:)=sum(app(n,:));
   PSD(n,:)=app(n,:)/sumapp(n,:);          % Packet Size Distribution
   CDF(n,1)=PSD(n,1);
   for j=2:1514
      CDF(n,j)=PSD(n,j)+CDF(n,j-1);        % CDF
   end

CCDF = 1-CDF(n,:);                         % Complementary CDF
CCDF(n,1516:1600)=zeros;

% Packet Size Range 60 to 589, 590 to 1513 and 1514 to 1600
% Calculate the values of alpha, beta and c.

beta=[60, 590, 1514];
c=[CCDF(n,Range(1)), CCDF(n,Range(2)-1), CCDF(n,Range(3)-1)]
for k=1:3
   %beta(k)=Range(k);
   %c(k)=CCDF(n,Range(k)-1);
   min_error(k)=100;
   for alpha = alphaRange(k,1):alphaRange(k,2):alphaRange(k,3)
      error=0;
      for i=Range(k):Range(k+1)-1
         error = error + (CCDF(n,i)-c(k)*((beta(k)/i)^alpha))^2;
      end
      if (error < min_error(k))
         min_error(k)=error;
         Alpha(k)=alpha;
      end
   end
   min_error(k)=min_error(k)/(Range(k+1)-Range(k));
end

% Generate Pareto Model based on values of alpha, beta and c
```

```
ParetoCCDF(n,1:60)=1;
for k=1:3
   for i=Range(k):Range(k+1)-1
      ParetoCCDF(n,i)=c(k)*((beta(k)/i)^Alpha(k));
   end
end

figure(1)
plot(CCDF(1:1600))
axis([0 1550 0 1])
hold
plot(ParetoCCDF(1:1600),'-.r')
xlabel('Packet Size')
ylabel('1-CDF')
title('HTTP : Pareto Model')

% Generate artificial packet size distribution based on the Pareto model
ParetoPSD(1)=ParetoCCDF(1);
for i=2:1600
   ParetoPSD(i)=ParetoCCDF(i)-ParetoCCDF(i-1);
end

figure(2)
plot(ParetoPSD);
axis([1 1550 0 0.7]);
hold
plot(PSD,'-.r')
xlabel('Packet Size (bytes)')
ylabel('PDF')
Title('HTTP')
legend('Artificial','Original');
```