2011

Network traffic data analysis

Harish Kapri
Louisiana State University and Agricultural and Mechanical College

NETWORK TRAFFIC DATA ANALYSIS

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
In partial fulfillment of the
Requirements for the degree of
Master of Science in Electrical Engineering

in

The Department of Electrical and Computer Engineering

by
Harish Kapri
Bachelor of Engineering in Electronics and Telecommunication Engineering,
Rashtrasant Tukadoji Maharaj Nagpur University, 2006
Nagpur, India
December, 2011

ACKNOWLEDGEMENTS

I would like to express gratitude to my major professor, Dr. Suresh Rai who was abundantly helpful and offered invaluable assistance, support and guidance throughout this work. His suggestions, discussions and constant encouragement have helped me to get a deep insight in the field of networking. I would also like to thank my committee members, Dr. Ramachandran Vaidyanathan and Dr. Xue-Bin Liang, for sparing their valuable time to be a part of my thesis advisory committee. I feel honored that they gave me their consent and time to review my thesis.

I would like to thank the MAWI [1] repository (providing publicly available anonymized traces), without it, this research wouldn't have been possible. I appreciate Chris Holbrook, for been a constant motivator. My parents have been a constant source of love and support, without who's encouragement; I wouldn't have pursued my Master's Degree.

I would also like to thank Pallavi Rao Malempati, for making toughest situations comforting. Last but not the least; I would like to thank Jasmine. No words would be enough to say how grateful I am to know her. Her smile and sole presence were my best motivation towards the completion of this thesis.

Finally, I would like to thank everybody who has directly or indirectly contributed to the success of this thesis; I express my sincere apology that I could not mention them personally.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

CAIDA	Cooperative Assosciation for Internet Data Analysis
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
ICMP	Internet Control Message Protocol
Perl	Practical Extraction and Report Language
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
BGP	Border Gateway Protocol
ICMP	Internet Control Message Protocol
OSPF	Open Shortest Path First
EGP	Exterior Gateway Protocol
RIP	Routing Information Protocol
ARP	Address Resolution Protocol
RARP	Reverse Address Resolution Protocol
DHCP	Dynamic Host Configuration Protocol

ABSTRACT

The desire to conceptualize network traffic in a prevailing communication network is a facet for many types of network research studies. In this research, real traffic traces collected over trans-Pacific backbone links (the MAWI repository, providing publicly available anonymized traces) are analyzed to study the underlying traffic patterns. All data analysis and visualization is carried out using *Matlab*[1]. At packet level, we first measure parameters such as distribution of packet lengths, distribution of protocol types, and then fit following analytical models. Next, the concept of flow is introduced and flow based analysis is studied. We consider flow related parameters such as top ports seen, duration of the flow, distribution of flow lengths, and number of flows with different timeout values and provide analytical models to fit the flow lengths. Further, we study the amount of data flowing between source-destination pairs. Finally, we focus on TCP-specific aspects of captured traces such as retransmissions and packet round-trip times. From the results obtained, we infer the Zipf-type nature of distribution for number of flows, heavy-tailness of flow sizes and the contribution of well-known ports at packet and flow level. Our study helps a network analyst to further the knowledge and optimize the network resources, while performing efficient traffic engineering.

[1] Matlab® is a registered trademark of *The Mathworks, Inc.*

CHAPTER 1: INTRODUCTION

1.1 Background

The Internet is a network of networks which carries a vast range of information resources and services, such as the exchange of documents of the *World Wide Web* (WWW) and the framework to support emails and other applications. The desire to conceptualize network traffic in a prevailing communication network is a facet for many types of network research studies. Internet has long been studied to tackle vast gamut of problems, including security, attacks and monitoring general health of the network. Network analysis is a process of capturing network traffic and inspecting it closely to determine what is happening in the network [2]. It is also known by several other names: network analysis, protocol analysis, and packet sniffing and packet analysis to name a few. We consider network traffic analysis to be a set of methods that successively is utilized to understand the nature of traffic on per packet or per flow level basis.

The traffic data encompasses the time and duration of the communication, the intricate shape of the communication streams, the identities of the groups communicating, and their location. The analysis of network traffic provides information about the user behavioral patterns thereby enabling network operators to understand the underlying traffic phenomenon. Various parameters are studied or closely monitored based on this phenomenon. Maximum utilization must be obtained from the capital investment in network infrastructure but at the same time the operator must be aware of capacity constraints within the network and plans to meet future demands in a timely fashion [3]. Analysis of network activity is an important undertaking leading to valuable contributions both to the research community and to the commercial sector. For instance, researchers and *Internet service Providers* (ISPs) are constantly seeking ways to

optimize the network resources and perform efficient traffic engineering. Traffic analysis not only provides means of identifying a fault in the network, but also helps in understanding the root cause of the fault and its impact on communication between the users of a network. Authors, in [4] describe the main difficulties awaiting those who are trying to simulate the Internet and study its properties. Nonetheless, accomplished researchers share their knowledge of how to pursue this apprehensive task [5].

In this thesis, we pursue the analysis of traffic statistics, for traces collected every day, for 15 minutes, from 2001 to 2011, over trans-Pacific backbone links (the MAWI repository, providing publicly available anonymized traces [6]). Based on the dataset, there is a valuable contribution aiming at the longitudinal study of the evolution of the traffic, where long term characteristics are investigated both at TCP/IP layers (packet and flow attributes) and application usages [7]. The field of traffic analysis is very vast which encompasses data collection, statistical analysis, and prediction and pattern recognition analysis. The data to be analyzed depends on the availability of network traffic measurements. Traffic measurements in operational networks aid in understanding the traffic characteristics in deployed networks, developing traffic models and evaluate performance of protocols and applications. Computer network measurements provide network operations, its development and research with information regarding the network behavior. The reliability and credibility of this information directly affects the quality of these activities, and thus the perception of the network and its services [8, 9]. Traffic characterization has been utilized in the telephone network since the beginning of the 20th century, where Erlang established foundation exists for modern traffic theory [10].

Traffic prediction is another field which is closely related to traffic analysis. Here, one could assess the future network capacity requirements and the proactively plan the future network

developments. Several machine learning approaches have been applied to the network traffic recently. The approaches differ in the sense of time domain to signal domain and also making the use of wavelet analysis. The network traffic matrices are very huge and it's impossible to manually navigate through it. Thus in [11], the authors tried to closely study the Origin-Destination (OD) flow time-series, taken from two different backbone networks and incorporated the use of Principal Component Analysis (PCA) [12], thereby reducing the dimensionality of the dataset to the order of small number. PCA is one of the dimension reducing techniques for data sets of higher dimensions, which has been utilized in numerous fields such as face recognition and image compression and now finding its way in network traffic research. It is a technique for finding patterns in data of high dimensions since it is difficult to locate patterns in data of high dimensions preventing the data from been visualized.

The approach of analyzing the data could further introduce or subtract the errors found in network measurements. Hence, by ignoring the issues that arise from the collected data, the error introduced in the results can be articulated by the selected analysis method. Due to the issues in the measurement tools, properties of the measured parameter or the network variations, jitter, may arise in the output from the analysis. The approach to reduce jitter is to average the existing data over large intervals [13]. If the jitter does not contain any bias, increasing the interval size will lead to smoother value, but with the presence of jitter.

Network traffic is a cornucopia of data and the approaches involved in measuring, analyzing, predicting and modeling solely depends on the end use of the data. However, acquiring the data at the packet, flow and router level is a strenuous task. While working on this thesis, we tried to collect the data from a live campus, but were unable to do so as there was no provisioning for data measurement and collection. We had to test and formulate our theory based on the

anonymized (de-privatized) dataset obtained from the internet. Since the addresses are anonymized, it only affects the topology of the data and not the results.

1.2 Objective

The objective of this thesis is to analyze and visualize the network traffic data both at packet and flow connection levels. We try to address various concerns related to the percentage of TCP/UDP traffic, protocol categorization, load vs throughput, distribution of the flows and packets, length of the flows and TCP statistics such as *Round Trip Time* (RTT) and the retransmissions. The traffic traces utilized in the research are obtained from the online traffic data repository maintained by the *Measurement and Analysis on the WIDE Internet* (MAWI) working group of the WIDE Project. The traffic traces are further processed in a human readable form and analyzed using Matlab to reveal the underlying statistics. The traffic in the MAWI dataset is subject to bandwidth changes, to congestions and to a variety of anomalies. This allows us to compare the impacts of anomalies on the traffic statistics. Our approach encorporates the use of simple freely available tools which put together helps to accomplish interesting results. This thesis presents the ways of utilizing available resources to analyse traffic using standard free packet capture and analysis software. We try to manually inspect the traces for network nuances and filter it accordingly, so that it does not negate the underlying truth. Given the dataset, we try to investigate traffic volume mundane patterns and most common applications in use. Further, we tried to employ aggeration techniques, e.g. aggregation by origin-destination pairs and by user. Moreover, we try to identify the general nature of the traffic according to the nature of distribution, heavy-tailness and correlation patterns. The results obtained after performing different preprocessing steps at various levels were analyzed and visualized.

1.3 Layout of the Thesis

The thesis is organized as follows:

Chapter 1 provides an insight about the approaches involved in network traffic data analysis and visualization. It discusses different notions to analyze the data at signal, packet and flow level. It provides insight about the challenges faced in analyzing the traffic data followed by the objective and layout of the thesis.

Chapter 2 covers the IP network traffic and provides a review of various statistical distributions utilized throughout this research. It provides information about the 5-layer TCP/IP model and its relevance to the research. The protocols involved at Network and Transport layer are discussed. The chapter presents the heuristics adopted in filtering TCP/UDP ports from raw data and stresses on the IP and TCP/UDP packet header structure. It introduces the concept of flow with respect to network traffic. The analytical models incorporated to fit the traffic data distribution are also discussed in detail. The definition and interpretation of box plot when applied to a large data set is elaborated.

Chapter 3 introduces the tool's utilized in the analysing and visualizing the network traffic measurements. Most of the tools used are free, except for Matlab which is a commercial product. Further, it shows how these tools unite together to aid in carrying out efficient passive network analysis of a raw data. This chapter shows the usage of the tools and their contribution to the thesis.

Chapter 4 provides the methodology involved in analyzing the raw traffic data. It describes about the source of traffic data and its details. It discusses about the approach utilized for analyzing the

traffic data namely flow, packet, source-destination pair and TCP relevant statistics. The outputs generated at the aforementioned levels are discussed in detail.

Chapter 5 presents the results, including the traffic model and network performance parameters. The statistics of aggregated packets or byte counts distribution at the TCP/IP layer are analyzed. For the analysis performed in this section, we have tried to evaluate several analytical models that best describes the dataset. Further, the results obtained are substantiated where ever applicable.

Finally, we summarize the thesis in Chapter 6, followed by a scope for further study of our work presented in this thesis.

CHAPTER 2: IP NETWORK TRAFFIC AND REVIEW OF STATISTICS

This chapter provides an overview of the Internet model followed by the protocols used at the network and transport layers. Further, we elaborate on the approach employed to filter TCP/IP ports from raw packets. The concept of flow is introduced and discussed in detail. We, first, provide a definition for flow, and then elaborate on how flows are affected with different timeout values. Flows can be applied in many areas [14, 15], including packet filtering and IP routing. Here we focus on the application of flow in Internet traffic. Our goal in defining flow attributes is to properly identify and measure the dynamic traffic pattern of the Internet. We also describe the long and short tail distributions and their relevance to our research. Further, we provide the definition for various types of distribution implemented throughout this research. The process of distinguishing different application protcols in a packet traffic, is out of scope of the thesis. Finally, we conclude this chapter by providing a discussion, how the material presented here is relevant for the next chapter.

2.1 The 5-layer TCP/IP Model

The phenomenal achievement of the Internet has led to the rapid approbation of the *Internet Protocol* (IP) technology to build all types of communication networks, including private commercial networks (intranets), military communication networks, private home networks, smart devices (smart phones, Internet TV), and the emerging *Fourth-generation* (4G) cellular networks. The traffic in the modern IP networks is disparate and most of the applications used in the networking environment prefer IP as a medium to transmit the data. The traffic generated by these applications have highly deviating characteristics and the compulsion to peruse the protocols seperately becomes credible. Several models have been developed to categorize the

communication protocols into distinct hierarchial structure. The 7-layered OSI model [16] and the 5-layered TCP/IP model [17] are most commonly used in IP network studies. In this thesis, the 5-layered TCP/IP model, illustrated in Figure 2.1, is considered. Furthermore, we limit our study to the two layers: network and transport layer protocols. The first (Physical) and the second (Data link) layers in the model consist protocols for maintaining physical and link-level connectivity and ensuring bit-level integrity in data transfer. They do not contribute much from the analysis point of view of the network when inspecting packet traffic. In order to perform packet/flow traffic analysis, one needs to be aware of the protocols and its header structure at higher layers.

<table>
<tr><th>Layer</th><th>7-Layer OSI</th><th>5-Layer TCP/IP</th><th>Example Protocols and Specifications</th></tr>
<tr><td>L7</td><td>Application</td><td rowspan="3">Application</td><td rowspan="3">Telnet, HTTP, FTP, SMTP,POP3, VOIP, SNMP</td></tr>
<tr><td>L6</td><td>Presentation</td></tr>
<tr><td>L5</td><td>Session</td></tr>
<tr><td>L4</td><td>Transport</td><td>Transport</td><td>TCP, UDP</td></tr>
<tr><td>L3</td><td>Network</td><td>Network</td><td>IP</td></tr>
<tr><td>L2</td><td>Data Link</td><td>Data Link</td><td>Ethernet(IEEE 802.3), HDLC, Frame Relay, PPP</td></tr>
<tr><td>L1</td><td>Physical</td><td>Physical</td><td>RJ-45, Ethernet(IEEE 802.3)</td></tr>
</table>

Figure 2.1: Comparison of 5-layer TCP/IP and 7-Layer OSI models

2.1.1 Network and Transport Layers

Note that, the network layer (L3) is responsible for the transfer of data in the form of packets in an interconnected network. Refer to Figure 2.2 (a) for the header format of the IP layer [18].The study of network traffic starts from this network layer, as layers 1 through 2 do not contribute much from the aspects of end-to-end traffic study. The main function of network protocols is to

establish connectivity between all L3 routers and hosts in the network, thereby allowing communication between the direct or indirect connections.

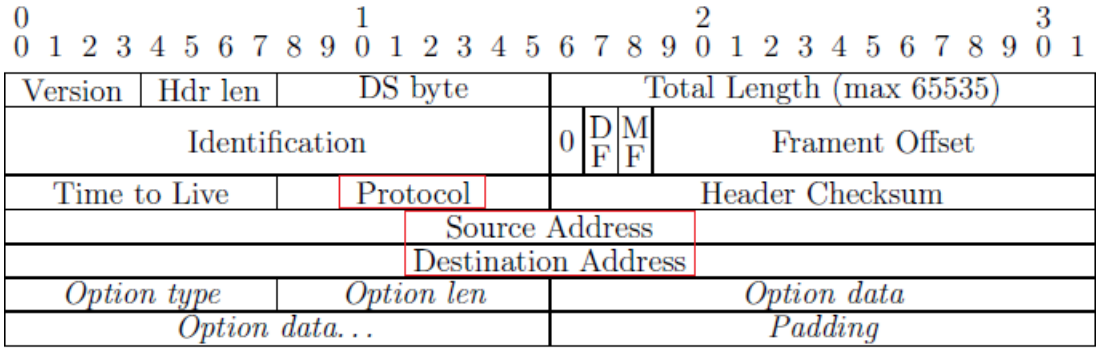

Figure 2.2(a): IP header format [18]

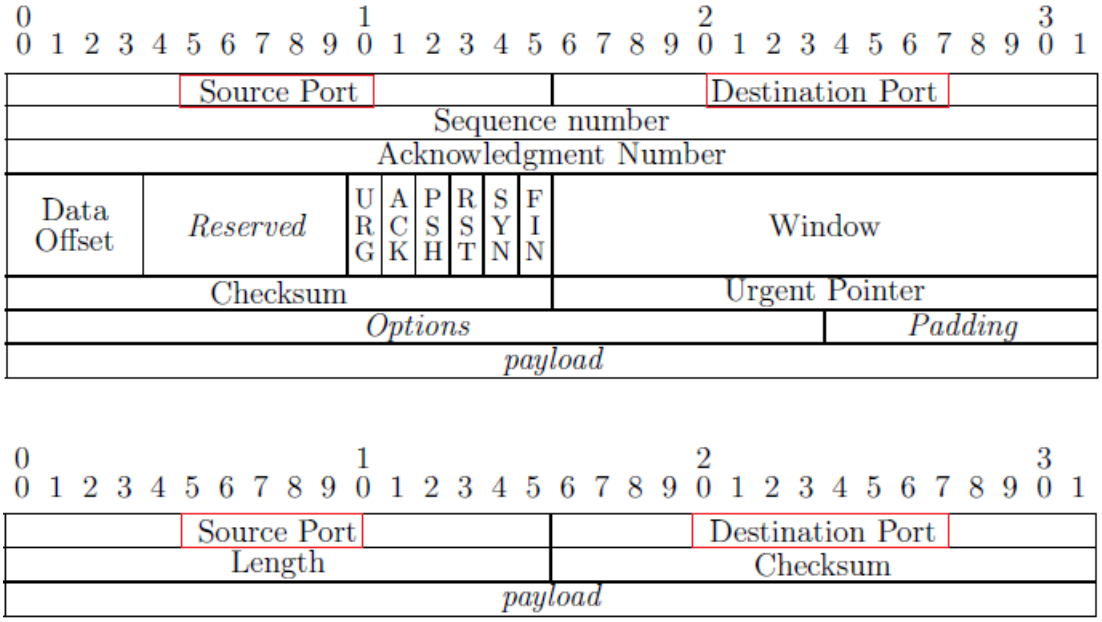

Figure 2.2(b): TCP/UDP header format [19, 20]

The main function of network protocols is to establish connectivity between all layer 3 routers and hosts in the network, thereby allowing communication between the direct or indirect connections. It performs network routing functions, and might also perform fragmentation and reassembly, and report delivery errors. The *Internet Protocol* is the heart of the TCP/IP protocol

suite and corresponds to the network layer. In Internet, IP protocol- currently, IPv6 [14] along with IPv4 [21] is responsible for transfer of packets between end machines on a hop-by-hop basis.

It offers a connectionless and best-effort delivery service to the transport layer (layer 4). A connectionless service does not require a virtual circuit to be established prior to the process of data transfer thereby offering a packet switched transfer medium where fidelity of the data is not guranteed. The connectivity between the end-hosts is established by incorporating the routing protocols, such as EGP [22], OSPF [23], RIP [24] and BGP [25]. There are a number of layered management protocols belonging to the network layer. The task of resolving host's physical and IP addresses is managed by ARP [26] and RARP [27]. Maximising the usage of IP address space allocation to the hosts can be automated by designing a DHCP [28] environment to the network. Error reporting and delivery of control messages is handled by ICMP [29]. It runs on top of the IP and is considered to be in the same layer as IP.

2.1.2 Transport Protocols

Transport protocols utilizes the services offered by the underlying network layer protocols and runs on top of the IP. TCP [20] and UDP [19], built on the best-effort service provided by IP are two most frequently used protocols to support a wide range of applications. Figure 2.2 (b) illustrates the header format for TCP/UDP packet segments.

TCP provides reliable, connection-oriented stream service over IP. It sets up a logical full-duplex connection between two end-hosts across a datagram network. It also provides flow and congestion control, thereby allowing receivers to control the rate at which sender transmits information preventing buffers from overflowing. Source IP, destination IP, source port and

destination port are the four tuple attributes which helps TCP in uniquely identifying each connection. TCP ensures connection establishment using three-way handshake by incorporating bidirectional connection which involves the exchange of SYN, SYN+ACK and ACK packets, respectively. It also needs proper use of the flags. It is most commonly known as a three-way handshake procedure for connection establishment and termination. TCP provides for a harmoniuos close that involves the independent termination of each direction of the connection, by sending a FIN packet which is reciprocated with an ACK packet. Another alternative for connection termination provided by TCP is through reset (RST) segments. Data reliability is achieved by the use of retransmission timeouts. Hence, TCP uses acknowledge packets and retransmissions to gurantee succesful transmission of the data.

Unlike TCP, UDP provides a much simpler and bare minimum service to the applications. The UDP is an unreliable, connection-less and not stream-oriented transport layer protocol. It is a very simple protocol which aids in demultiplexing and error checking on the data. It is specified by source and destination ports to identify the connection. Flow and congestion control mechanisms are not utilized by UDP. Since, UDP is connection-less, it does not implement connection establishment and termination procedure. The absence of ACK and retransmission mechanisms makes it an unreliable protocol. Further, there is no rate control mechanism to adjust the transmission rate and the applications using UDP do not require the heavyweight service of TCP. However, UDP proves to be very helpful in multicasting, network management, routing table updates and real-time multimedia as opposed to TCP.

2.1.3 TCP/UDP Ports

The study of applications and services incorporated in a network is of prime importance to

network operators. The elementary way to conduct this study is to consider transport layer port numbers. The service port number and an ephemeral port number are the two port numbers used by TCP and UDP packets, as highlighted in Figure 2.2 (b). The service port number or well known port number is a destination port number encountered in TCP SYN packet (first packet for UDP) which aids in identifying applications unquestionably. Since, each TCP and UDP packet has both source and destination ports. The protocol implemented to distinguish application port numbers is:

a) If one of the source port numbers is less than 1024 or if it is present in the list of well-known ports, consider it as service port to distinguish the application.

b) If neither is true, perform above step for the destination port.

c) Finally, if the above two steps fail, the smaller port number is considered having a higher probability of been a service port.

Internet Assigned Numbers Authority (IANA) [30] port assignment catalog is utilized for mapping ports to applications. The port numbers are broadly categorized as follows: well-known ports (0-1023), registered ports (1024-49151) and dynamic or private or ephemeral ports (49152-65535).

2.2 Traffic Flows

The nature of internet traffic can better be understood by knowing the concept of the flow. Flow is the sequence of packets or a packet that belonged to certain network sessions (conversation) between two hosts but delimited by the setting of flow generation or analyzing tool [31]. Alternatively, the definition of flow may also be coined as, a series of packets that share the

same *source IP*, *destination IP*, *source port*, *destination port* and the *protocol*. This is most commonly known as *five-tuple* IP flow, which is an aggregation of individual flows. The most important thing to remember is that all traffic in a flow is unidirectional. Network flow data symbolizes a summary of sessions between two end hosts. It further aids in network analysis and security issues. Flow data is independent of packet payloads. The flow tool or analyzer is dependent on the amount of information collected from packet headers and its relevant metrics. In addition, the network flow data deeply enhances the visualization of discrete network events such as protocol analysis or length distribution without the need for payload analysis. Further, the knowledge of flow data aids in understanding how different flows compete in a network to acquire network resources. Packets having similar *five-tuple* information belong to the same flow. The conversation between two end points in the Internet layer is directional. A network flow can be considered either as unidirectional flow or bidirectional flow. In unidirectional flow, the flow attribute is characterized in one direction i.e. from source to destination or vice versa. Whereas, in a bidirectional flow, the attributes are characterized considering both directions. Considering the flows to be unidirectional in our case is justified, as we can always combine unidirectional flows to form bidirectional flows in a trace file. However, from the results obtained, it is inferred that there is a strong correlation between the number of flows detected and the length of the timeout value, as corroborated in [32]. The paper shows that the number of flows varies inversely with the length of the timeout value. Flow identification is achieved by labeling each flow information uniquely. As our intention in the research, is to analyze the amount of traffic between two hosts achieved by focussing on source and destination IP. The Figure 2.3 shows the impact of timeout values on flows.

Each flow is identified using the *five-tuple* attribute with the help of *crl_flow* [33] tool (part of CoralReef [33] software). The duration of the flow is defined by the time for which the flow survived without timeouts. The period determines the start and end times of each flow. Apart from the *five-tuple* attribute, flow timeout is an important parameter when defining a flow. It refers to the duration during which flow can be idle before considering it as dead. Thus, a flow time out value cannot be too short or too long , as a short timeout would lead to dissemination of flow into several small flows therby impacting the workload on the system and a long timeout value may discard several short flows, thereby preventing us from tracking real traffic. Further, an extremely long timeout value prevents a flow from getting expired because of which the number of flows keep increasing. Thus, a flow time out value cannot be too short or too long , as a short timeout would lead to dissemination of flow into several small flows therby impacting the workload on the system and a long timeout value may discard several short flows, thereby preventing us from tracking real traffic. Further, an extremely long timeout value prevents a flow from getting expired because of which the number of flows keep increasing. Thus, a flow time out value cannot be too short or too long , as a short timeout would lead to dissemination of flow into several small flows therby impacting the workload on the system and a long timeout value may discard several short flows, thereby preventing us from tracking real traffic. Further, an extremely long timeout value prevents a flow from getting expired because of which the number of flows keep increasing.

According to the *Realtime Traffic Flow Measurement* (RFTM) architecture [34], a flow has attributes that are derived from endpoint attribute values, metrics values like packet and byte counts, time values as well as summary information like mean, mean or average values, jitters

and distributiuonal information. Our goal in defining flow attributes is, proper identification and measurement.

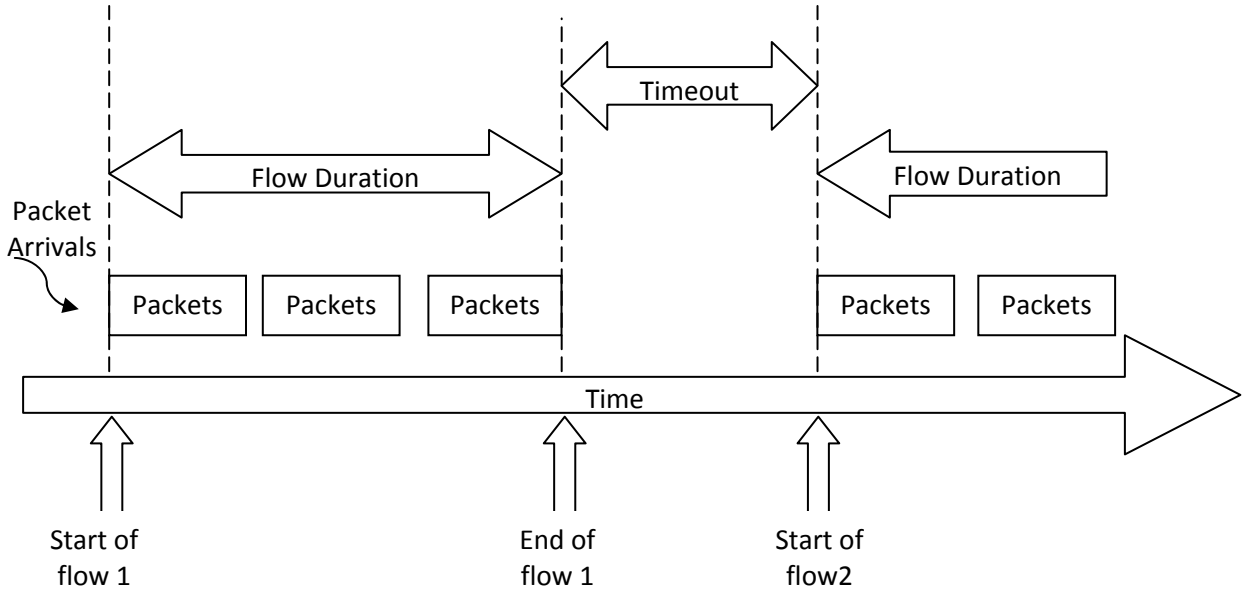

Figure 2.3: Flow model: Impact of timeout value

A TCP flow would start with a SYN packet and end with a FIN or RST packet. Whereas in UDP, it is difficult to define a flow, because UDP is a connection-less, unreliable datagram delivery protocol. Hence, there is no concept of a connection in UDP. However, a flow could still be defined by the controlling application specifying source or destination port number. Since, each application has its own header structure, so we tried to utilize ***crl_flow*** [33] tool along with manual inspection of the flow data. A flow can be defined in terms of different granularity levels, which inturn depends on the accuracy required for the analysis. The use of different granularity levels and their relevance to network analysis is discussed in [32]. We are using the most commonly used granularity which is the *five-tuple* attribute, as illustrated in Figures 2.2(a) and 2.2(b), illustrating the format of an IP and TCP/UDP header structure, respectively.

2.3 Probability Distributions

This section considers two types of distributions. The tail of the distribution, defined as $P(X>x)$ is, used to predict rare events in several areas of probability applications, including networking. Here, we consider typical distributions that are useful in modeling flow arrival time, flow length and packet distribution.

2.3.1 Long-tail Distributions

Distributions where the data values in the tails are more spread or shows decay in the tail, based on power law as $x^{-\alpha}$,are referred to as *long-tailed* or heavy-tailed distributions. The distributions analyzed in this thesis, include *Pareto, Log-normal* and *Weibull* distributions, as shown in Figure 2.4. Note, a heavy-tailed or *long-tailed* distribution [35], follows:

$$\text{Tail Distribution: } P[X > x] \sim x^{-\alpha}, \quad as\ x \rightarrow \infty, \quad 0 < \alpha < 2$$

This means that regardless of the distribution for small values of the random variable, if the asymptotic shape of the distribution is hyperbolic, it is heavy tailed [35]. We can further conclude that a heavy-tailed distribution has infinite variance and if α is less than unity, the mean tends to infinity. The probability of generating large values becomes almost negligible with the use of heavy-tailed distribution.

Pareto distribution [36] , is the simplest heavy-tailed distribution with probability density and cumulative distribution function given by the equation:

$$\text{PDF}_{\text{Pareto}} = P(x) = \alpha k^{\alpha} x^{-\alpha-1}, \qquad \alpha, k > 0, \qquad x \geq k.$$

$$\text{CDF}_{\text{Pareto}} = F(x) = P[X \leq x] = 1 - (k/x)^{\alpha}, \quad \alpha, k > 0, \quad x \geq k$$

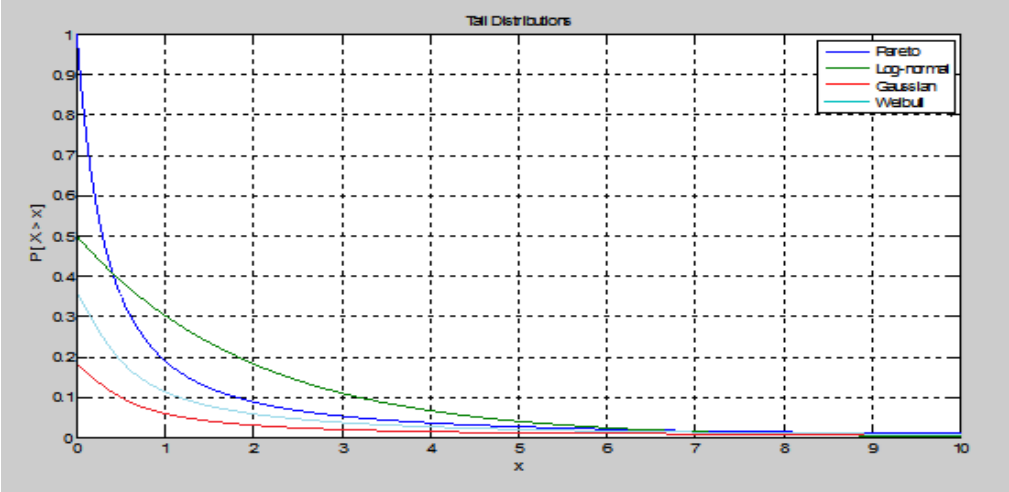

Figure 2.4: Tail Distributions: P[X > x] for various distributions

The parameter *k* is the lower limit for the random numbers generated by a Pareto random number generator. It is a power law probability distribution. It is given in terms of *cumulative distribution function* (CDF), i.e. the number of events larger than *x*, is an inverse power of *x*. Table 2.1 provides the computed mean and variance for various distributions based on the shape (α) and scale (β) parameters. From Figure 2.4, we infer that log-normal distribution is flatter than Pareto and Weibull distribution lies between Pareto and Gaussian.

Table 2.1: Shape (α) and scale (β) parameters and moments of distributions

Distributions	α	β	Mean	Variance
Gaussian	0	2	0	4
Pareto	1.3	2	15	∞
Log-normal	1	2	22.874	22183.27
Weibull	0.6	2	5	75

Weibull distribution [37] , is another type of heavy-tailed distribution with probability and cumulative density functions given by the equations

$$\text{PDF}_{\text{Weibull}} = P_{(\alpha,\beta)}(x) = \alpha\beta^{-\alpha}x^{\alpha-1}e^{-(x/\beta)^{\wedge}\alpha}, \qquad \alpha, \beta > 0$$

$$CDF_{Weibull} = P_{(\alpha, \beta)}(X \leq x) = 1 - e^{-(x/\beta)^{\alpha}}, \quad \alpha, \beta > 0$$

The above equations are for the standard *Weibull distribution* where, μ (the location parameter) is zero with $\alpha = 1$. This distribution is used extensively in reliability applications to model failure times. The complementary cumulative distribution function is a stretched exponential function. The goodness of fit of data to a *Weibull distribution* can be visually assessed using a *Weibull plot*.

Lognormal distribution [37], is another distribution taken into consideration. A variable X is log- normally distributed if $Y = LN(X)$ is normally distributed with "*LN*" denoting the natural logarithm. The case where α (the location parameter) and β (the scale parameter) are zero and one, respectively, is known as the *standard lognormal distribution* whose equation is given by

$$PDF_{lognormal} = P_{(\alpha, \beta)}(x) = \left(1/\left(x\alpha\sqrt{(2\pi)}\right)\right) e^{-0.5(\ln(x-\beta)/\alpha)^{2}}, \qquad x > 0$$

The *lognormal* and *Weibull distributions* are probably the most commonly used distributions in reliability applications.

2.3.2 Short-tail Distributions

Distributions where the data values in the tails are less spread or has exponentially decaying tail are known as *short-tailed* distributions, as shown in Figure 2.4. Apart from the heavy-tailed distribution, there are other distributions fitted to the data. It is formidable to prelude which distributions best describes the dataset. There are various parameters in the data which need to be modeled precisely and different distributions may be used for different parameters. Other distributions capitalized for modeling the statistical phenomena are *lognormal, normal, Rayleigh* and *Weibull distributions*.

Rayleigh distribution [37], may be considered as a special case of *Weibull distribution*. In communication theory, it is used to model scattered signals that reach a receiver by multiple paths. The probability density and cumulative distribution function are as follows:

$$\text{PDF}_{\text{Rayleigh}} = P_{(\alpha)}(x) = (x/\alpha^2)\, e^{-(x\text{^}2/(2\alpha\text{^}2))}, \qquad 0 \le x \le \infty$$

$$\text{CDF}_{\text{Rayleigh}} = F[x] = 1 - e^{(-0.5(x/\alpha)\text{^}2)}, \quad 0 \le x \le \infty$$

Normal distribution [37], is a theoretical function commonly used in inferential statistics as an approximation to sampling distributions. It is very important class of statistical distribution. In general, the normal distribution provides a good model for a random variable, when, there is a strong tendency for the variable to take a central value, positive and negative deviations from this central value are equally likely and the frequency of deviations falls of rapidly as the deviations become larger [38]. The probability density functions is given by

$$\text{PDF}_{\text{normal}} = P(x) = (1/(\sqrt{2\pi\alpha^2})\, e^{-((x-\beta)\text{^}2/(2\alpha\text{^}2))}, \qquad x \in (-\infty\ \infty)$$

2.3.3 Zipf's Distribution

The Internet is comprised of networks on many levels, and some of the most exciting consequences have been discovered in this area [39]. *Zipf-type* plot delineates the dependency between the number of occurrences of a variable and its rank. The Zipf's law states that for each following rank, the number of occurrences is α times less than the previous rank. Zipf's distribution is often correlated with *Power-law* and *Pareto*. *Zipf's law* usually refers to the size *'y'* of an occurrence of an event relative to its rank *'r'*. George Kingsley Zipf, a Harvard linguistics professor, sought to determine the size of the 3rd or 8th or 100th most common word. Size here denotes the frequency if use of the word in English text, and not the length of the word

itself. It further states that the size of the r'th largest occurrence of the event is inversely proportional to its rank. $Y \sim r^{-b}$, with b close to unity. A *power law distribution* tells us not how many people had an income greater than x, but the number of people whose income is exactly x. Claiming a *Zipf's law* in a dataset seems to be simple enough: if n values, x_i $(i = 1, 2 \ldots n)$, are ranked by $x1 \geq x2 \geq \ldots . x_r \ldots . \geq x_n$, *Zipf's law* states,

$$X_{(r)} = C * r^{-\alpha}$$

Where the parameter value, α , is usually close to 1, implies that 'x_r' versus 'r' plot on a log-log scale will be a straight line with a negative slope α close to -1. Hence, we assumed 'x_r' as the data (random variable), from statistical modeling point of view, *Zipf's law* is a model of average of 'x_r' or *log('x_r')* as a linear function (linear regression) of log(r) (with c = log(c)):

$$E (\log (x_r)) = c - \alpha \log(r)$$

2.4 Box plot

Apart from the distributions mentioned in section 2.3, we will study the *five-number summary*, the mean and the *standard deviation* (SD) for the data set. This kind of data representation is known as *box plot*. Its definition and usage is reviewed here. Box plots [40] are an excellent tool for conveying location and variation information in datasets, particularly for detecting and illustrating location and variation changes between different groups of data. When conducting a distribution study, the shape, the centre, and the variablility of the distribution are of primary concern [41]. The shape of the distribution can either be symmetric or skewed. The left and right portions about the middle value are symmetric, in a symmetric distribution. However, in real life, symmetric distributions rarely occur but the distribution might be considered as approximately symmetric. On the other hand, a skewed distribution is the one that tails off gradually but slowly

on one side than other. The distribution could either be skewed towards high or low values. The centre of distribution is commonly measured by the *mean,* the *mode*, and the *median*. The mean is calculated by dividing the sum of observations in the data set by the frequency of observations. The mode is the most frequent occuring observation in the data set. The median is the central value which splits the data exactly in two halves. A significant difference between mean and median is that the mean is more susceptible to extreme values than the median. Thus, median is more robust as it is less sensitive to outliers in the data set. We have considered three measures of variability: the *range*, the *interquartile range* (IQR), and the *standard deviation* (SD). The range is the width of the distribution which is calculated as the difference between the largest and the smallest value. It is sensitive to outliers. The *interquartile range* is the width of the distribution measured from lower quartile to the upper quartile of a distribution. In other words, the lower quartile may be considered as the median of the lower half of distribution ranging from lower extreme to the median of original distribution, and the upper quartile is vice versa. The interquartile range follows the median and unlike the range, it is more robust against outliers. The *standard deviation* is the widely used measure of variability.

The *five-number* summary consists of the median, the two quartiles, and the two extremes for the box plot as shown in Figure 2.5. The median indicates the centre of the data, quartiles indicate variability, and their difference is the interquartile range. The extremes provide information about the outliers and the bounds for data. The distance of the median to the quartile gives an indication of whether the distribution is skewed or symmetrical [41]. The box plot preserves all the information of the data set, there by distinguishing outliers. Since, it plots all the outliers in a large data set, the plot obtained might not be correct due to the large number of outliers. The box plot utilized in the data set is slightly modified in order to show two extreme values in the plot.

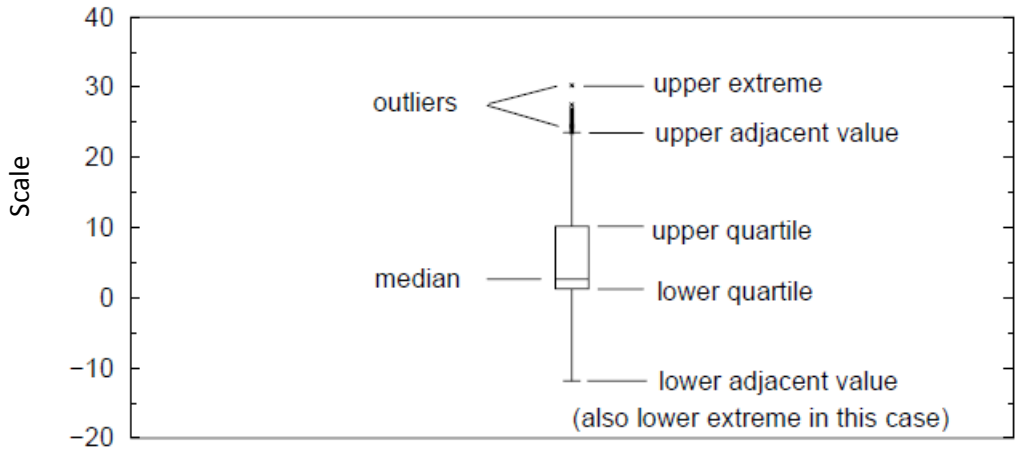

Figure 2.5: Box plot parameters

2.6 Conclusion

In this chapter, we discussed the importance of the 5-layer TCP/IP model. Further, the IP network traffic was categorized according to the 5-layer TCP/IP model. We studied the protocols and its importance at the network and transport layers. In order to filter TCP/UDP ports, we introduced the algorithm used in our research to filter the required ports without ambiguity. We provide a basic understanding of the general features of the flow. Studies suggest that the number of flows share an inverse relationship with timeout values, which is proved in chapter 5. Later, we discussed short and long tail distributions and its relevance to the network traffic, which is further described in Chapter 5. Apart from the above distributions, we discussed the use of box plot and Zipf's distribution. The terms studied under this chapter, immensely aids in understanding the underlying distribution of network traffic. The next chapter discusses the various tools utilized in our research.

CHAPTER 3: ANALYSIS TOOLS

The tools used in our study are mainly open source and free with the exception of *Matlab*, which is a commercial product. The programs used throughout this thesis are: ***tcpdump*** [42] (for reading captured packet traffic), ***CoralReef*** [33] (for pre-processing the captured traffic traces), ***tcptrace*** [43] (for analysis of TCP dump files), ***tcpstat*** [44] (for packet statistics), ***Perl*** (for scripting to normalize the data), and ***Matlab*** (for statistical analysis and visualization of the results).

3.1 Software utilized

3.1.1 Tcpdump

The ***tcpdump*** [42] program was written by Van Jacobson, Craig Leres, and Steven McCanne and all of the Lawrence Berkeley National Laboratory at the University of California at Berkeley. ***Tcpdump*** is one of the most popular network sniffing and analyzing command line tool. In addition, the choice of using ***tcpdump*** was favored by the fact that we can further analyze the traces. Further, ***tcpdump*** can also be used for network monitoring, protocol debugging and data acquisition. The major function of ***tcpdump*** is to monitor packets on the attached network and dump headers and payloads of packets to a human-readable format. In order to monitor the network traffic, ***tcpdump*** sets the network interface in the promiscuous mode to capture all the packets on the attached network. Here promiscuous mode means receiving a copy of every frame. Normally, our network interface will ignore any packet which is not addressed to our system. It uses *Packet Capture Library* (libpcap), which was originally developed by Van Jacobson et al., to retrieve packets from the network interface. After this, the packet passes

through Berkley Packet Filter (BPF), which can then be further analyzed by inclusion or exclusion of packets based on particular criteria.

Tcpdump is a simple and easy-to-use tool. It supports many command line options for flexibility and ease of convenience. Typically, the number of bytes parsed in the header of a captured packet determines the level of information we can gather. The longer the parsing length, the more we can discover. We can specify the parsing length i.e. the number of bytes (from the beginning of the packet). Since we are interested only in the headers of the packet, ***tcpdump*** can be set to capture only the first N bytes of a packet. By default the length is 68, which is adequate for *Internet Protocol* (IP), *Internet Control Message Protocol* (ICMP), *Transmission Control Protocol* (TCP) and *User Datagram Protocol* (UDP). Figure 3.1 shows the various command line options for ***tcpdump***.

```
tcpdump [ -AdDefIKlLnNOpqRStuUvxX ] [ -B buffer_size ] [ -c count ] [ -C file_size ]

[ -G rotate_seconds ] [ -F file ] [ -i interface ] [ -m module ] [ -M secret ] [ -r file ] [ -s snaplen ]

[ -T type ] [ -w file ][ -W filecount ][ -E spi@ipaddr algo:secret,... ][ -y datalinktype ]

[ -z postrotate-command ] [ -Z user ] [ expression ]
```

Figure 3.1: Command line options for ***tcpdump*** [45]

It is very easy to run this tool but can only be initiated with administrative privileges. Figure 3.2 shows raw output of tcpdump, passed with additional parameters as shown below.

tcpdump -ttttnnr 200608241400.dmp

2006-08-24 00:00:00.794858 IP 192.114.235.26.43924 > 192.77.128.86.22755: Flags [.], seq 2951648224:2951649672, ack 1347586929, win 32768, options [nop,nop,TS val 778726674 ecr 4213600849], length 1448

2006-08-24 00:00:00.794864 IP 192.114.235.26.43924 > 192.77.128.86.22755: Flags [.], seq 1448:2896, ack 1, win 32768, options [nop,nop,TS val 778726674 ecr 4213600849], length 1448

2006-08-24 00:00:00.794867 IP 145.120.109.103.1541 > 214.206.48.97.80: Flags [.], ack 1998939210, win 65535, length 0

2006-08-24 00:00:00.794870 IP 192.114.235.26.43924 > 192.77.128.86.22755: Flags [.], seq 2896:4344, ack 1, win 32768, options [nop,nop,TS val 778726674 ecr 4213600849], length 1448

Figure 3.2: *tcpdump* output format [45]

-tttt	Print a delta (micro-second resolution) between current and first line on each dump line.
-nn	Don't convert addresses (i.e., host addresses, port numbers, etc.) to names.
-r	Read packets from file (which was created with the -w option).

Figure 3.3: *tcpdump* options

Figure 3.3 explains the parameters passed with *tcpdump* command. We could see that the raw output always prints out the name of the network interface on which it is listening. It provides direction of the packet with symbol ">", timestamps in the form of YYYY-MM-DD with time resolution in milliseconds and shows IP addresses instead of names. The format is easy for user to understand, however it is difficult for our script to analyze, since the position of certain parameters may vary depending upon the packet. Instead, we select other options when running tcpdump to generate a better output. The command used in the thesis, is as follows:

tcpdump –nlq –r 200608241400.dmp

The new output format suitable for analyzing the traffic data is illustrated in Figure 3.4 with options described in Figure 3.5. Hence, we observe that the format is in the form of timestamp,

192.114.235.26.**43924**(*SourceIP*.***SourcePort***),

192.77.128.86.**22755**(*DestinationIP*.***DestinationPort***), protocol and the length of the data. Hence, the source port is 43924, the destination port is 22755, protocol is TCP and packet length is 1448. It is very easy to parse through the file output shown in Figure 3.4, to extract ports and packet lengths. The output generated is passed through a ***perl*** script to extract packet length, source and destination ports. Further, ***tcpdump*** has a default standard output based on the protocol. The main difference between the TCP formats from others is the TCP flags, sequence numbers, acknowledgements, acknowledgement numbers and TCP options.

```
00:00:00.794858 IP 192.114.235.26.43924 > 192.77.128.86.22755: tcp 1448
00:00:00.794864 IP 192.114.235.26.43924 > 192.77.128.86.22755: tcp 1448
00:00:00.794867 IP 145.120.109.103.rds2 > 214.206.48.97.http: tcp 0
00:00:00.794870 IP 192.114.235.26.43924 > 192.77.128.86.22755: tcp 1448
00:00:00.794873 IP 152.223.152.140.32819 > 201.196.236.252.domain: UDP, length 31
00:00:00.794963 IP 193.84.6.33.ott > 193.188.196.52.microsoft-ds: tcp 0
00:00:00.794967 IP 201.41.68.131.nntp > 197.117.111.242.informatik-lm: tcp 0
00:00:00.795089 IP 46.254.176.252.domain > 192.114.237.128.37044: UDP, length 151
00:00:00.795093 IP 161.32.137.50.64571 > 180.41.155.110.ftp-data: tcp 0
00:00:00.795340 IP 193.176.202.69.optilogic > 93.108.39.55.linogridengine: tcp 1440
```

Figure 3.4: Simplified ***tcpdump*** output format

-l	Make stdout line buffered. Useful if you want to see the data while capturing it.
-n	Don't convert addresses (i.e., host addresses, port numbers, etc.) to names.
-r	Read packets from file (which was created with the -w option).
-q	Quick (quiet?) output. Print less protocol information so output lines are shorter.

Figure 3.5: ***tcpdump*** options used

The output generated using ***tcpdump*** depends on the options used. It allows the user to extract particular kinds of network traffic. The basic syntax for a tcpdump filter is as follows:

<header>[<offset>:<length>] <relation> <value>

The filter syntax of tcpdump is very robust. We can employ the filters to extract connections involving specific networks, hosts, and ports. Hence, it is a great utility for studying network protocols. We could also analyze the set up phase of a TCP connection, which is generally referred to as a three-way handshake. Further, it can also be used to create a powerful, network-based Intrusion Detection System (IDS). In order to efficiently filter the required contents from a packet, one has to have a deep understanding of the header structure of the protocols.

3.1.2 Tcpstat

Tcpstat [44] periodically reports TCP-related statistics on a given network interface. These statistics include number of packets exchanged, average packet size of a TCP stream, standard deviation of packet size, bandwidth been used, and so on. It further provides information by either monitoring a specific network interface, or by reading a previously saved tcpdump data. Like ***tcpdump***, ***tcpstat*** also makes use of the libpcap library and can be used to analyze/capture packets on the specified network interface by setting the network interface to promiscuous mode similar to ***tcpdump***. We have utilized ***tcpstat*** to generate traffic statistics like average packet size, average bandwidth achieved from a ***tcpdump*** file. Further it is used as post processing tool for the raw data captured by ***tcpdump***. Figure 3.6 shows various command line options for ***tcpstat***.

The command utilized to classify packets according to their types, and compute packets per second statistics for each type is:

```
tcpstat –r 200608241400.dmp – o "%r %A %C %I %T %U %V %I %b\n" > 200608241400stat.log
```

```
tcpstat [-?haeFlp] [-B bps] [-b bps] [-f filter expr] [-i interface][-o output]
[-R seconds] [-r filename] [-s seconds] [interval]
```

Figure 3.6: Command line options for ***tcpstat*** [44]

2.500000 0 762 75870 64917 8786 405 1.00 74895836.80
7.500000 0 561 82176 71439 8782 352 0.93 80809864.00
12.500000 0 889 75968 65568 8234 427 0.94 78703870.40
17.500000 0 871 75957 65098 8932 432 0.94 77782824.00
22.500000 0 867 79878 70313 7765 511 0.92 82804331.20
27.500000 0 639 75596 66017 7923 411 0.92 77049424.00
32.500000 0 709 72233 62596 7712 487 0.92 73864659.20
37.500000 0 742 71456 61928 7729 494 0.92 75586312.00
42.500000 0 729 73476 63260 8284 548 0.92 77935491.20
47.500000 0 752 74439 64099 7990 262 0.92 79662100.80
52.500000 0 678 73975 64203 7630 450 0.93 78349310.40
57.500000 0 685 76483 66526 7888 434 0.93 81196268.80
62.500000 0 743 81304 70718 8417 457 0.93 81091350.40
67.500000 0 693 77268 67157 8276 695 0.94 79107052.80
72.500000 0 834 80176 69946 8118 444 0.93 84293918.40
77.500000 0 748 73671 64071 7646 379 0.93 78273076.80
82.500000 0 754 83826 73832 7949 314 0.89 88736915.20
87.500000 0 865 86414 75941 8244 356 0.86 92651355.20
92.500000 0 780 80411 70086 8179 260 0.87 84763854.40
97.500000 0 656 69963 60713 7441 279 0.86 74493004.80

Figure 3.7: ***tcpstat*** output format

-r	Read all data from filename ,which may be a regular file, a named pipe or "-" to read it's data from standard input
-o	Set the output format when displaying statistics
%A	The number of ARP packets
%C	The number of ICMP and ICMPv6 packets
%I	The number of IPv4 packets
%T	The number of TCP packets
%U	The number of UDP packets
%V	The number of IPv6 packets
%l	the network "load" over the last minute, like in uptime( )
%b	The number of bits per second

Figure 3.8: ***tcpstat*** options used

In this step, we make use of several tcpstat options, option '***–r'*** to read data file (in ***tcpdump*** format) and option '***–o'*** for specifying the output format suitable for the later processing by Matlab. Figure 3.8 shows the information about parameters passed along with the ***tcpstat*** command. The output could be in any format, but we preferred it as a log file. The log file was

read in Matlab to generate the relevant plot. The posts processing tasks performed in our analysis was to classify all packets found in the trace file according to their protocol types (TCP, UDP, etc.) and further derive various statistics of individual packet types. Finally, the results obtained are visualized using Matlab.

3.1.3 Tcptrace

Tcptrace [43] was developed by Shawn Ostermann at Ohio University, for the analysis of TCP dump files. It is a TCP connection analysis tool. By incorporating the use of ***tcptrace***, one can study the effect of retransmissions. ***Tcptrace*** understands various network dump file formats like ***tcpdump, snoop, ns, nlanr*** and others. We can pass multiple command-line options to perform various tasks. In order to study TCP connections reassembly, this tool is capable of providing output in terms of number of retransmitted data packets and average Round-Trip Time (RTT) value for each TCP connection. The command used is as follows:

tcptrace –l –r –n - - csv 200608241400.dmp > 200608241400tcp.csv

Figure 3.9 explains various parameters passed with ***tcptrace*** command. We confine the use of ***tcptrace*** in our analysis to retransmissions occurred during the captured time. The tool can produce detailed statistics of TCP connections from dump files when given the '–***l***' or the long output option. RTT (Round-Trip Time) statistics are generated when '–***r***' option is specified along with '–***l***' option.

-l	long output format
-r	print rtt statistics (slower for large files)
-n	no lookup of DNS names
--csv	display the long output as comma separated values

Figure 3.9: ***tcptrace*** options used

It can further report statistics on the estimated congestion window with the '*–W'* option when used in conjunction with '***–l***' option. Since there is no direct way to determine the congestion window at the TCP sender, the outstanding unacknowledged data is used to estimate the congestion window. One can also use the outputs generated by this tool in conjunction with Tim Shepard's xplot program in order to visualize different types of graphs illustrating various parameters of a TCP connection. The tool comes with various command-line options to filter the data and tailored it according to the needs. When tcptrace is run trivially on a dump file, it generates the output similar to the one discussed in Chapter 4.

3.1.4 CoralReef

CoralReef [33] is a versatile open source packet collector/analyzer software suite developed by *Cooperative Association for Internet Data Analysis* (CAIDA). It can collect and analyze data those obtained by either active/passive monitoring. It is designed to be flexible and a highly configurable internet traffic data collection and analysis tool. The ***CoralReef*** suite is completely a passive monitoring system which does not require any additional network infrastructure/resources and does not interfere with network traffic or other network devices. The usage of this tool is bolstered by the fact that it is used and tested extensively by researchers in the open source community which projects its reliability and credibility to the analysis results. ***CoralReef*** is a software suite which is built in layers. It is able to achieve the flexibility and efficiency by incorporating a 3-level layered design [46]:

a) API

It can read from live pcap interfaces and contains the code for physical devices such as ATM, DAG and POS cards .Further, a low level API is included with these devices to offer an interface for traffic analysis applications.

b) Traffic applications

It includes various tools for accessing the raw API for performing the most frequently used operations, such as tracing individual packets and packet flows as well as providing traffic rate information.

c) Traffic Flow applications are used to analyze the packet and flow traces. It can further be used to convert the captured pcap data to traffic matrices

Our analysis is confined to the third layer from the above mentioned layers. The most important aspect of the software is its ability to convert raw traces to flow data and thereby enhancing the flow based analysis. The features of the first layer are not utilized in this thesis as it comprises of mainly the hardware components and its configuration since we obtained the captured traces from online. The main CoralReef software tools utilized are ***crl_flow***, ***t2_convert*** and ***IP_matrix***. The option ***crl_flow*** is used to convert packet data to flow data; ***t2_convert*** along with ***IP_matrix*** is used to convert flow data to an aggregated IP matrix for each origin-destination pair.

3.1.5 Matlab and Perl[2]

Matlab is a very powerful mathematical tool which can be used for performing tasks involving high speed numerical calculations and visualization. Further, it provides its own scripting

[2] Perl is a scripting language and details can be found on http://www.perl.org/

language for automating complicated operations. There are various special purpose toolboxes available which further enhance the ease of use of the software. It is highly customizable and we can write various scripts tailored to the needs. Matlab is the only commercial tool used in our study. The usage of Matlab as an analysis tool was influenced by the fact that, we already had the campus license and further it suits our needs. There are other statistical applications available in the market, some of them are commercial and others are open source/free. The use of the application depends on the availability and familiarity with the software.

Practical Extraction and Reporting Language (***Perl***), is an interpreted scripting language which facilitates data manipulation and rapid application development. It is very powerful tool when it comes to data parsing and navigation. We have utilized various ***perl*** scripts for preprocessing data and further making it suitable for analysis with Matlab.

3.2 Workflow

Our analysis is mainly categorized into following sets: per-packet, per-flow and per-connection

levels.Our analysis is mainly categorized into following sets: per-packet, per-flow and per-connection levels. Network traffic related parameters are derived from the data gathered from the real network. Several freely availabe tools such as ***tcpdump***, ***tcpstat***, ***tcptrace, CoralReef*** software suite and ***perl*** have been utilized to process raw packet captures. The raw data is further processed in order to understand the traffic parameters both at packet and flow level. The overall principle of data gathering and result analysis is illustrated in Figure 3.10. First, the raw ***tcpdump*** files are processed using ***tcpdump*** along with ***perl*** to generate data at packet and flow level. The data is gathered seperately for each level of analysis: packets, flows, origin-destination pairs and TCP-specific statistics. The data is further tailored in suitable format for analyzing statistics and

further discuss about the observations on fitting theoretical distributions. Matlab is utilized to carryout all the analysis and generate the plots.

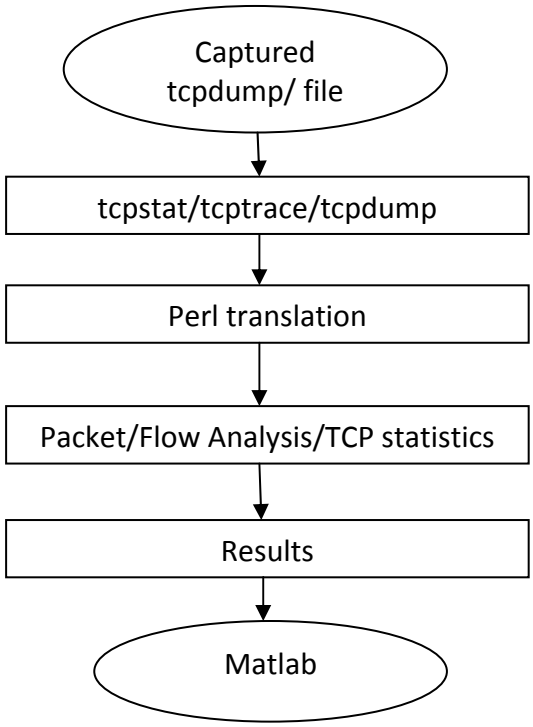

Figure 3.10: Workflow of research

3.3 Conclusion

We have studied various tools utilized in our research and their contribution in deriving network traffic related parameters. The results achieved by incorporating these tools are visualized in Chapter 5. The knowledge gained by understanding the syntax of these tools, aids in generating the relevant output from a raw dump file. Chapter 4 elaborates the approach adopted at different granularity levels – packets, flows, origin-destination pairs and TCP connections. We have utilized different preprocessing steps at different levels of traffic data. Further, Chapter 4 discusses the structure of the data considered and traffic analysis at different levels.

CHAPTER 4: METHODOLOGY

This chapter provides an insight about the network whose traffic traces are taken into account. Further, we pursue the analysis of the data from different perspectives, meaning different preprocessing steps at various levels of granularity. After each step, we try to provide an insight on how the data appears and how it is encorporated in our research. Finally, we provide a workflow of our research showing various tools utilized in combination to analyze relevant statistics.

4.1 Network Data

The traffic data to be analyzed is from a backbone network with 100Mbps link. The dataset consists of a set of labels locating traffic anomalies in the MAWI [1] archive (sample points B and F). The labels available in the dataset are sample points[3] A,B,C,D,E and F, respectively. The MAWI traffic repository archives the traffic data collected from the WIDE backbone networks. The WIDE network (AS2500) is a Japanese academic network connecting universities and research institutes. It has been provoding anonymized packet traces since 1999 (total volume of available data exceeds 1TB). The data used here are all publicly available on the website. The packet traces are captured daily, from 2:00 pm to 2:15 pm (Japanese standard Time, UTC+9). The traces have anonymized IP addresses and no payloads, and are made available to public along with a summary information web page about the traffic. The topology of the network is

[3] All sample point names are used to match the description provided on http://tracer.csl.sony.co.jp/mawi/

unknown and it does not affect the analysis performed as we have confined our research to one sample point, sample point F. Table 4.1 shows the summary of a sample trace file:

Table 4.1: Summary of sample trace file

Traffic Trace Info
Dump File: 200608241400.dump
File Size: 1009.67MB
Id: 200608241400
Start Time: Thu Aug 24 14:00:00 2006
End Time: Thu Aug 24 14:15:01 2006
Total Time: 900.22 seconds
Total Capsize: 792.02MB Cap Len: 96 bytes
of packets: 14259282 (9342.30MB)
AvgRate: 87.06Mbps stddev:13.45M
IP flow (unique src/dst pair) Information
of flows: 363070 (avg. 39.27 pkts/flow)
Top 10 big flow sizes (bytes/total in %):
5.3% 2.2% 2.0% 1.8% 1.6% 1.6% 1.5% 1.4% 1.4% 1.2%
IP address Information
of IPv4 addresses: 189342
Top 10 bandwidth usage (bytes/total in %):
22.0% 6.7% 5.3% 4.8% 4.6% 4.4% 2.3% 2.1% 2.1% 2.0%
of IPv6 addresses: 1463
Top 10 bandwidth usage (bytes/total in %):
39.8% 39.5% 12.5% 6.8% 5.9% 3.8% 3.6% 3.3% 3.3% 3.0%

The sample point F consists of daily traces at another trans-Pacific line which further includes 48/78/83/96 long hour traces. Moreover the dataset is daily updated to include new traffic from upcoming applications and anomalies. Several papers have been published using the above mentioned dataset [1, 7, 47]. The traces collected in the corresponding sample point F last over a period of five months and each month has traces collected on different days. Of all the traces, we analyse seven of them contributing to total of one hour and forty five minutes. The traffic traces

are captured using ***tcpdump*** and the IP addresses in the traces have been anonymized by a modified version of ***tcpdriv*** [48] in order to preserve privacy of the users.

4.2 Traffic Analysis

4.2.1 Packet Based Analysis

Packet analysis was performed by implementing a simple heuristic as discussed in Chapter 2.1.3. We analyzed the per-packet characteristics of the data confined to the transport layer protocols. For port distribution, the raw dump files were read using ***tcpdump*** [42] along with a small ***perl*** script in order to get port numbers from packet data. The command utilized to filter ports numbers from the packet data is as follows:

[root@MACBOOK~]# *tcpdump* -nlq -r *200608241400.dump* | perl -lne '/IP\s+ (\d+\.\d+\.\d+\.\d+) (\.(\d+))?\s+>\s+(\d+\.\d+\.\d+\.\d+) (\.(\d+))?\:\s+(\w+)/&&print $3," ",$6' > 200608241400ports

The file obtained after preprocessing it comprises of source and destination ports. For traffic volume, a ***perl*** script was utilized again to extract packet lengths from the dump files. Furthermore, packet data was extracted in similar way with the help of ***perl*** script resulting in selected timestamps and packet lengths. Since the packet length is extracted from the IP header, the 14 bytes of ethernet header do not contribute to the results. We confined out research to the distribution of most commonly used ports, contribution of different protocols, analysis of traffic volume as function of time, distribution of packet lengths and analytical fits for packet length. The traffic rate or traffic volume is analyzed as function of time. Further, the exracted packet lengths are anaylzed to fit different analytical distributions.

4.2.2 Flow Based Analysis

In this thesis, we confine our research to flow and packet level. In the flow based approach, we aggregated separate packets into flows with the help of ***crl_flow*** [33] tool. Flows are identified from the network traffic using the 5-tuple attribute of protocol and source and destination IPs and ports, as explained in Chapter 2. Flow can be considered as a sequence of packets or a packet which belongs to a network session(conversation) between two hosts but delimited by the setting of flow generation or analyzing tool [31]. Flow based analysis enhances the investigation of network parameters and security concerns. Flow data does not rely on packet payloads, unlike deep packet inspection. In addition, a network flow data provides better visibility of the network events without the need to perform payload analysis. We have analyzed the flow sizes, flow length distribution and flow data. The obtained raw data was converted to flow data using ***crl_flow*** [33]. The command used for converting the raw data to flow data is as follows:

[root@MACBOOK~]# *crl_flow* 200608241400.dump –cl –Tf60 –o *200608241400flow.t2*

By default, the application ***crl_flow*** [33] uses 300 second trace interval duration.We aggregated separate packets into flows known as flow aggregated data.We have used 60 seconds timeout values for inactive flows. Later on, we performed the breakdown of flows by port numbers, where only TCP and UDP flows (protocol numbers 6 and 17 respectively) were considered, because other protocols do not use port numbers. Flow length distributions and its empirical cumulative distribution are produced from flow data. The variation in the flow lengths was confirmed with the help of box plot. Next, the flow length density function was fitted to several analytical models as discussed in section 2.3. At last, we analyzed the effect of timeouts on the number of flows. Table 4.2 shows the flow data for a sample trace file.

Table 4.2: Sample Flow Data

source ip	destination ip	protocol	ok	src port	dst port	packets	bytes	flows	first	latest
145.120.188.50	117.139.135.89	17	1	34817	33466	1	40	1	1156395748	1156395748
198.53.192.94	193.20.182.105	17	1	22239	53	1	74	1	1156395818	1156395818
201.196.236.249	204.8.103.144	17	1	53	2496	1	141	1	1156395716	1156395716
192.114.232.238	44.78.174.12	6	1	80	8371	5	1589	1	1156395729	1156395729
199.98.58.160	145.120.188.51	1	1	11	0	2	112	1	1156395753	1156395759
209.19.254.208	193.20.182.96	17	1	58175	53	1	71	1	1156395607	1156395607
46.57.72.52	193.20.182.105	17	1	32874	53	1	83	1	1156395614	1156395614
164.66.234.200	46.224.123.88	6	1	80	60387	8	3823	1	1156395835	1156395839
193.86.115.104	193.20.182.96	17	1	57585	53	1	75	1	1156395718	1156395718
192.182.105.20	193.188.201.199	6	1	80	3325	9	7344	1	1156395697	1156395701
193.2.72.72	193.20.182.96	17	1	5206	53	1	72	1	1156395730	1156395730
145.120.188.58	117.149.83.99	17	1	58170	33473	1	40	1	1156395615	1156395615
49.185.46.44	193.20.182.105	17	1	22359	53	1	73	1	1156395629	1156395629
161.32.7.65	193.161.161.83	6	1	80	1442	32	17087	1	1156395830	1156395833
208.27.40.51	193.20.180.152	17	1	44558	53	1	72	1	1156395756	1156395756
192.114.235.164	215.246.130.148	1	1	0	0	1	84	1	1156395675	1156395675
46.181.145.211	193.20.182.96	17	1	40170	53	1	73	1	1156395811	1156395811
201.196.236.252	177.0.177.226	17	1	53	1024	1	130	1	1156395646	1156395646
44.206.252.198	145.120.33.176	6	1	80	2747	2	211	1	1156395779	1156395779

4.2.3 Origin-Destination Pairs Analysis

We study the amount of data flowing between pairs of IP address (Origin-Destination pairs). We compute the number of flows and sum of bytes transferred for each origin-destination pair. An important fact is that, the origin and destination IPs can interchange their positions from flow to flow. Further we realized that a connection initiated by IP_1 can be considered as (IP_1IP_2) flow and later a connection initiated in the reverse direction can be considered as another (IP_2IP_1) flow. Hence, we utilized a script which overcomes the aformentioned realization and counts both directions as a single origin-destination pair. Hence, the script outputs the number of flows for each origin-destination pair and the total bytes transferred between two hosts. The flow data was processed, so that first source IP, destination IP, amount of bytes, amount of packets and flow counts were transformed to an IP matrix. It was achieved by issuing the following command:

[root@MACBOOK~]# cat *.t2 | *t2_convert IP_Matrix > totalmatrix*

The command option ***t2_convert*** , converts the flow data with aggregation of the source and destination IP pairs, ports and protocols. Once the IP matrix is obtained, connections occuring more than once are summed up along with the removal of additional information in the converted IP matrix. The Table 4.3 shows the striped IP matrix after conversion comprising of origin-destination pairs, packets, bytes and flows. The results generated after processing are explained in Chapter 5.

Table 4.3: Striped Origin-Destination pair Matrix

#Origin-Destination Pairs		packets	bytes	flows
170.2.58.108	200.77.10.225	2	132	2
52.212.25.177	161.32.131.65	2	194	1
213.111.52.200	209.148.187.96	1	108	1
68.253.52.129	206.250.157.11	2	124	1
193.176.202.69	218.12.228.185	1	300	1
92.183.110.126	164.66.46.120	4	650	4
197.228.69.169	197.161.44.172	1	48	1

4.2.4 TCP Statistics

Finally, we focus on TCP-specific aspects of the obtained traces. In our research, we confine our analysis to retransmissions and packet Round Trip Times (RTT). Passively estimating RTT is useful in measuring the congestion window size and retransmission timeout of a connection, as well as the available bandwidth on a path [49]. This information can help determine factors that limit data flow rates and cause congestion [50]. Additionally, RTT can be used to improve node distribution in peer-to-peer and overlay networks [51]. In order to aid with TCP connections reassembly, we utilized ***tcptrace*** [43] tool which is adequate enough of providing the number of retransmitted data packets and average RTT. The command utilized to convert the raw data to TCP statistics is:

[root@MACBOOK~]# *tcptrace* –l –r –n --*csv* 200608241400.dump > 200608241400tcp.csv

Tcptrace command with option '*–l –r –n*' produces detailed TCP-specific statistics from dump files. The output obtained after issuing the above command is very huge comprising of 128 columns. We utilize a custom ***perl*** script to parse through the *CSV* file with the '*- - csv*' option. The script outputs the necessary fields from the *CSV* file. Table 4.4 shows the file output after using ***perl*** script. Our main aim in selecting the parameters is to investigate if there is any correlation between retransmissions and packet RTT. The tool calculates RTT values as difference between the sent data packets and received ACKS preventing from dubious cases such as retransmissions. Table 4.4 clearly shows that the tool provides RTT values considering both directions. However, determining the RTT value from the receivers' perspective is a very intricate task [52]. From the manual inspection of the obtained results, we infer that RTT's observed in both directions vary hysterically. As we mentioned earlier, estimating RTT's from receivers' perspective is a non-trivial procedure which requires more practical approaches as mentioned in [53]. First, the raw data was processed using ***tcptrace*** [43] tool to attain the required statistics. A custom ***perl*** script was utilized to further process the data resulting in a custom file providing all the statistics related to retransmissions and Round Trip Times (RTT). Table 4.4 shows relevant fields after processing. The number of retransmissions for a connection from host(a) to host(b) and from host(b) to host (a) are aggregated to obtain the results. Finally, we try to find correlation between TCP retransmissions with RTT's for each connection.

Table 4.4: TCP Statistics

connection	host_a	host_b	rexmt_data_pkts_a2b	rexmt_data_pkts_b2a	rexmt_data_bytes_a2b	rexmt_data_bytes_b2a	RTT_min_a2b	RTT_min_b2a	RTT_max_a2b	RTT_max_b2a	RTT_avg_a2b	RTT_avg_b2a	RTT_stdev_a2b
1	192.114.235.26	192.77.128.86	258	0	373584	0	0	0	0	0	0	0	0
2	145.120.109.103	214.206.48.97	0	0	0	0	129.2	9	129.2	207.5	129.2	13.1	0
3	193.84.6.33	193.188.196.52	1	0	1	0	0	0	0	0	0	0	0
4	201.41.68.131	197.117.111.242	1	0	54	0	0	0	0	0	0	0	0
5	161.32.137.50	180.41.155.110	0	0	0	0	0	0	0	0	0	0	0
6	193.176.202.69	93.108.39.55	54	0	61200	0	0	0	0	0	0	0	0
7	164.93.60.249	93.110.157.219	7	0	10080	0	0	0	0	0	0	0	0
8	88.165.95.49	164.88.48.135	0	0	0	0	0	0	0	0	0	0	0
9	88.164.75.117	192.114.235.26	0	0	0	0	0	0	0	0	0	0	0
10	46.51.255.42	192.114.235.26	0	0	0	0	8.2	113.1	108.2	227.4	31.3	129.2	40.8
11	193.176.202.69	41.82.216.4	84	0	92580	0	0	0	0	0	0	0	0
12	35.21.253.77	161.32.164.15	0	0	0	0	0	0	0	0	0	0	0
13	161.32.131.65	217.106.145.193	5339	0	111044	0	0	0	0	0	0	0	0
14	194.184.109.64	161.32.38.70	0	0	0	0	2.6	209.3	2.6	442	2.6	236	0
15	182.56.80.63	89.81.212.148	368	0	529920	0	0	0	0	0	0	0	0
16	192.20.47.7	192.114.235.26	0	0	0	0	0	0	0	0	0	0	0
17	209.179.216.226	192.114.227.142	0	223	0	322904	1.3	434.6	51.1	1223.8	12.6	789.5	16.2

4.3 Conclusion

This chapter discussed the nature of the traffic data. We learnt that the trace files are in the dump file format and last for 15 minutes each. Then, we described how the data is tailored in collaboration with the tools discussed in Chapter 3, according to various levels of traffic analysis. Further, we have seen the structure of the data and its variability according to the approach adopted. Finally, based on the data generated in this chapter, we try to analyze and visualize the underlying traffic characteristics in the coming Chapter 5.

CHAPTER 5: RESULTS AND DISCUSSION

5.1 Traffic Volume

We analyze the traffic pattern as a function of time. In this section, the amount of traffic volume analyzed is for one week at different resolutions of time. Since the traces in the data set are contiguous, it enables us to analyze patterns in traffic volume up to one week. Figure 5.1(a), shows the traffic volume as a function of time. In [54], the author presents the work on methodology of traffic analysis and strongly encourages the search for invariants, one of which is diurnal patterns. From the results, it infers the presence of lucid patterns of sinusoidal type.

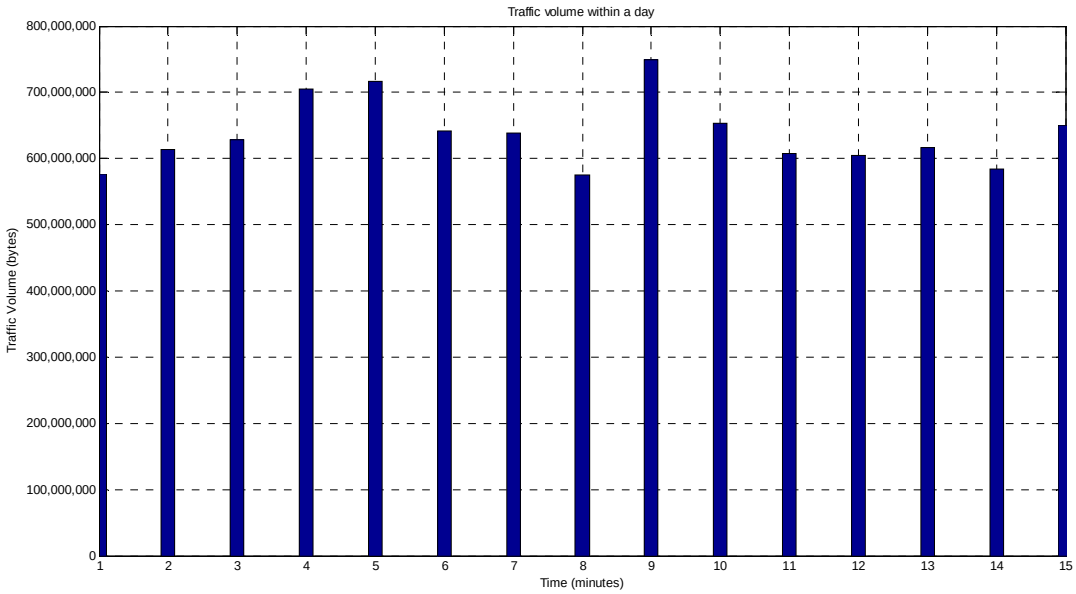

Figure 5.1(a): Traffic volume (1 day)

Figure 5.1(b), depicts the traffic volume as function of time for one week. It is necessary to check for variants in traffic from one day to one week. From the results, it infers that there exists a sinusoidal-type pattern for traffic volumes both for one day and one week.

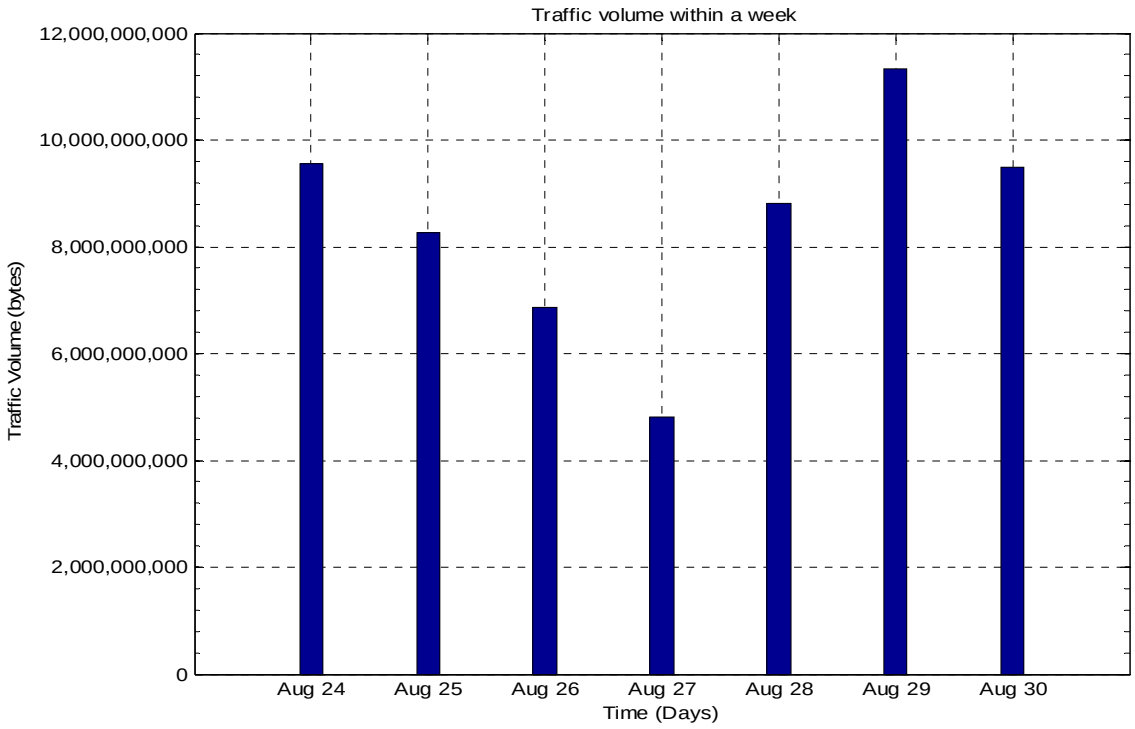

Figure 5.1(b): Traffic volume (1 week)

5.2 Packet Analysis

5.2.1 Packet Distribution by Ports

In this section, we discuss about the distribution of ports. While manually inspecting the port distribution, we could not find sufficient amount of well-known ports to plot its share in a pie chart. Even after applying the above method, we saw a lot of ephemeral ports in the data. There could be two reasons for the appearance of ephemeral ports. First, due to the epidemic of botnets, Internet worms and other malicious network programs that can use some of these ports for communication [55]. The second reason could be because of the use of peer-to-peer file sharing applications which can use both ports as ephemeral while initiating TCP connection. Thus, after implementing the heuristic mentioned in chapter 2.1 on the dump file, we get a list of service ports which are used to categorize applications and services used in the data. We manually

categorized the data into 3 classes: well-known ports (0-1023), registered ports (1024-49151) and dynamic or private or ephemeral ports (49152-65535). The Figure 5.2 (a), shows the categorization of the ports. The total amount of well-known ports found is very small (8) compared to ephemeral(5264) and registered ports(11478). The Figure 5.2(a), shows the bar plot of three classes of ports.

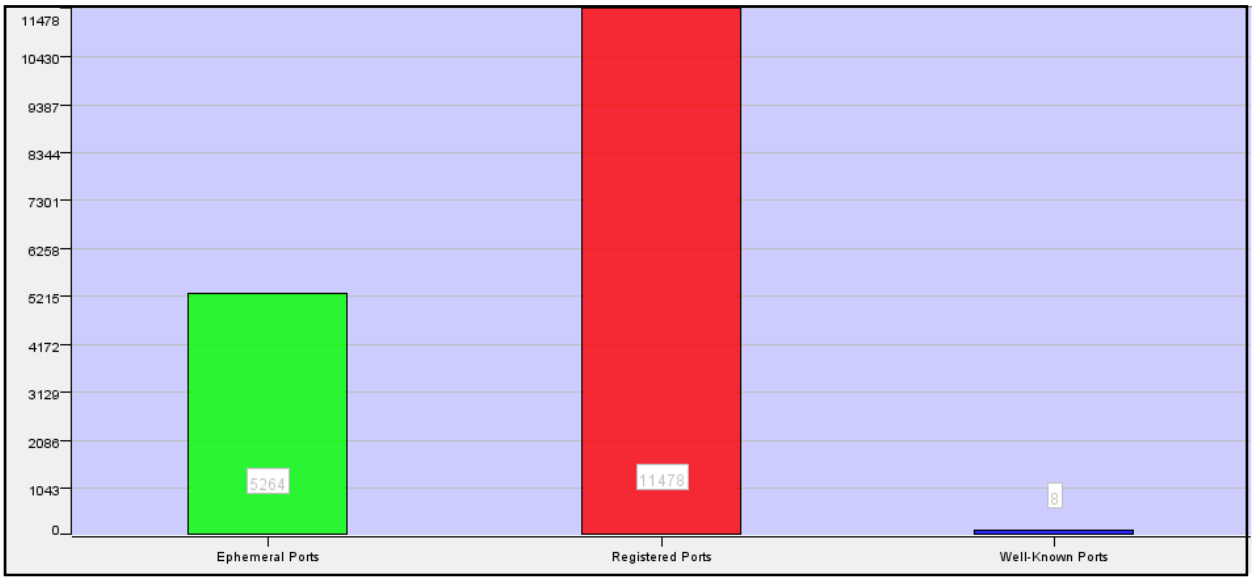

Figure 5.2(a): Packet distribution by ports

5.2.2 Packet Distribution by Protocols

Further, we classified packets according to their types and computed packets per second statistics for each type. The Figure 5.2(b), shows the number of ARP, ICMP and ICMP v6, IPv4, IPv6, UDP and TCP packets for every 5 seconds. From the results, it infers that the percentage of IPv4 packets is very high compared to TCP and UDP. The task is to determine how many packets of a particular protocol are observed in a given traffic file. Figure 5.2(c), shows the throughput and the load of the observed traffic. Throughput can be defined as the number of application bytes transferred in seconds. The effective throughput can reduce significantly either due to an

inefficient TCP algorithm or its implementation even if the underlying network provides a high speed communication channel.

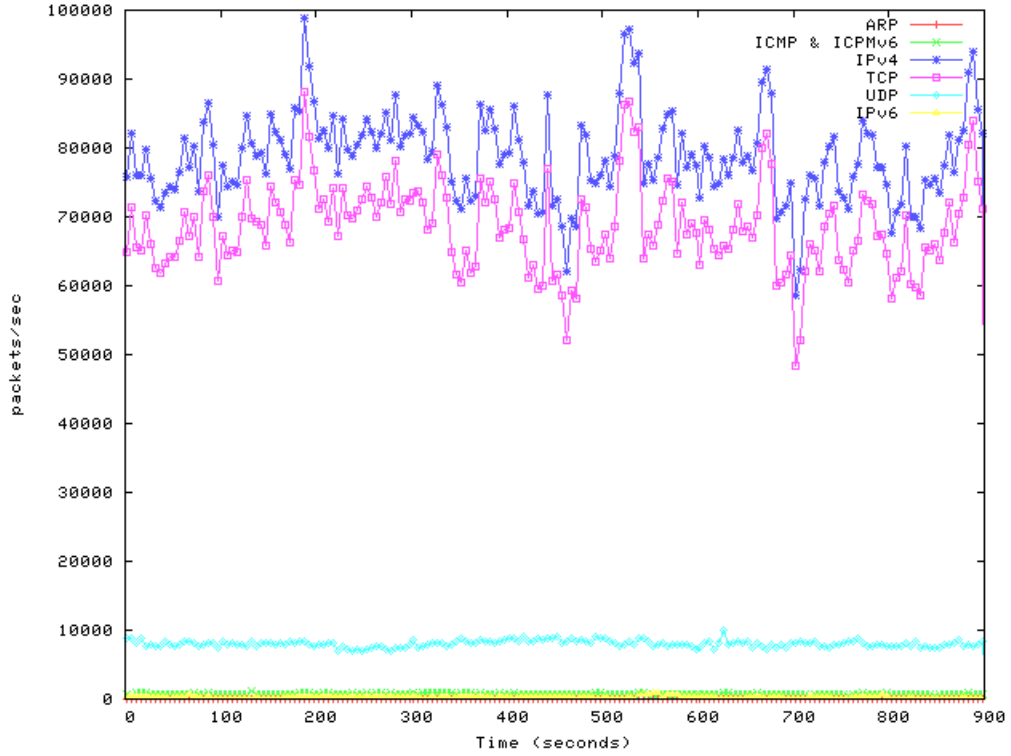

Figure 5.2(b): Packet distribution by protocols

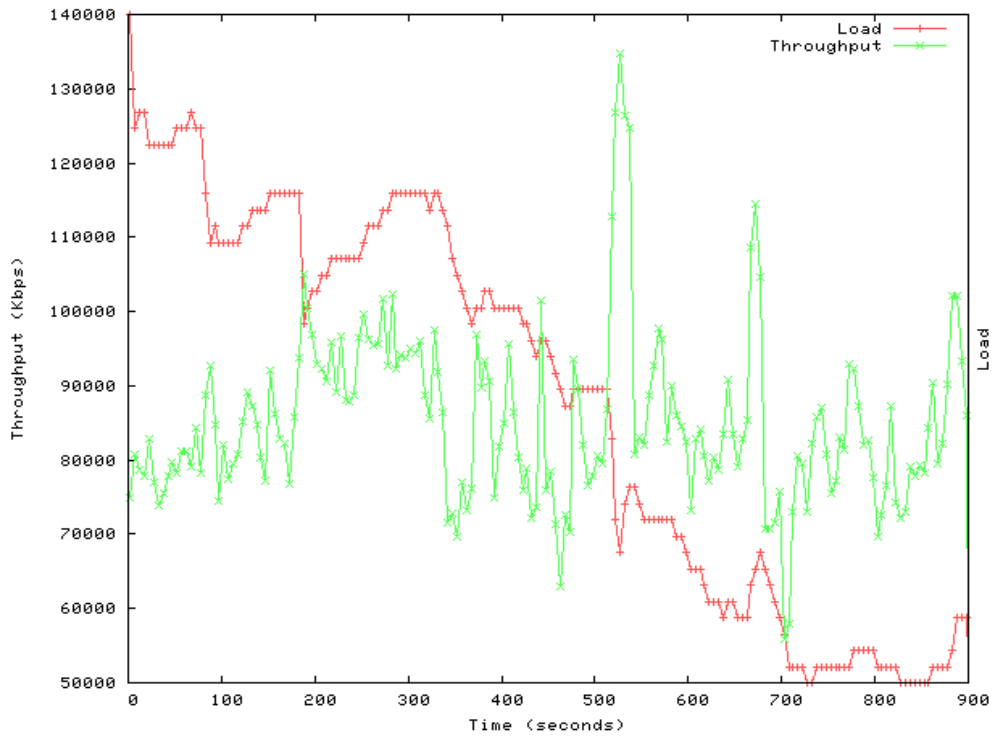

Figure 5.2(c): Load vs. Throughput

5.2.3 Packet Length Distribution

In this section, we investigated the distribution of packet lengths and tried to fit a theoretical probability distribution to it. The following plots in Figure 5.4, Figure 5.5 and Figure 5.6, describes the histogram of packet length distribution, its empirical *cumulative distribution function* (CDF) and the *five-number* summary statistics in the form of a box plot. In case of the histogram plot, we tried to plot the distribution using bins of width 1 byte in order to better understand the distribution. From the histogram plot, it infers that packets with lengths of 1500 bytes (the maximum size of IP packet in Ethernet) is quite prominent.

We utilized *Matlab's* ***dfittool*** to fit a theoretical distribution to the packet lengths data. The distributions we tried to fit were exponential and log-normal. Later on, we tried to validate the above mentioned distributions by plotting their probability distribution functions.

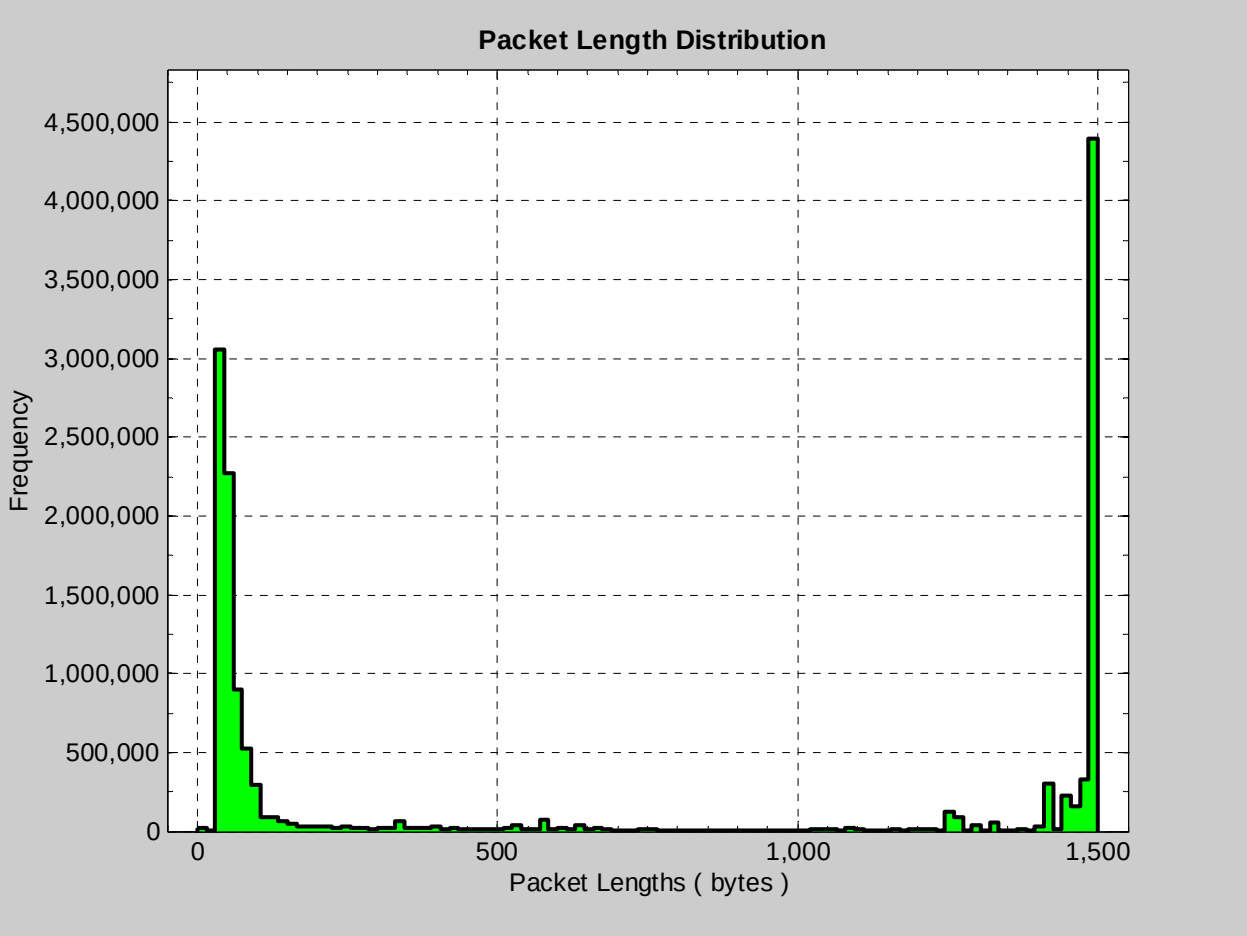

Figure 5.4: Distribution of packet length

From the results of the fit, it inferred that neither of the distributions fit the packet length data till any acceptable degree. We could have tried to cut the heavy tail of the distribution and fit a theoretical distribution to the remaining data, but that would not serve the purpose of properly fitting a distribution. Further, we plotted the emperical CDF of the packet lengths for different sets of traces. At last, we generate a box plot for the packet length data. The box plot provides the *five-number* summary for a sample trace file. From the box plot, we conclude that the number of outliers is very small (negligible), the upper quartile range is 1500 bytes with median of 112 bytes and lower quartile range is zero bytes.

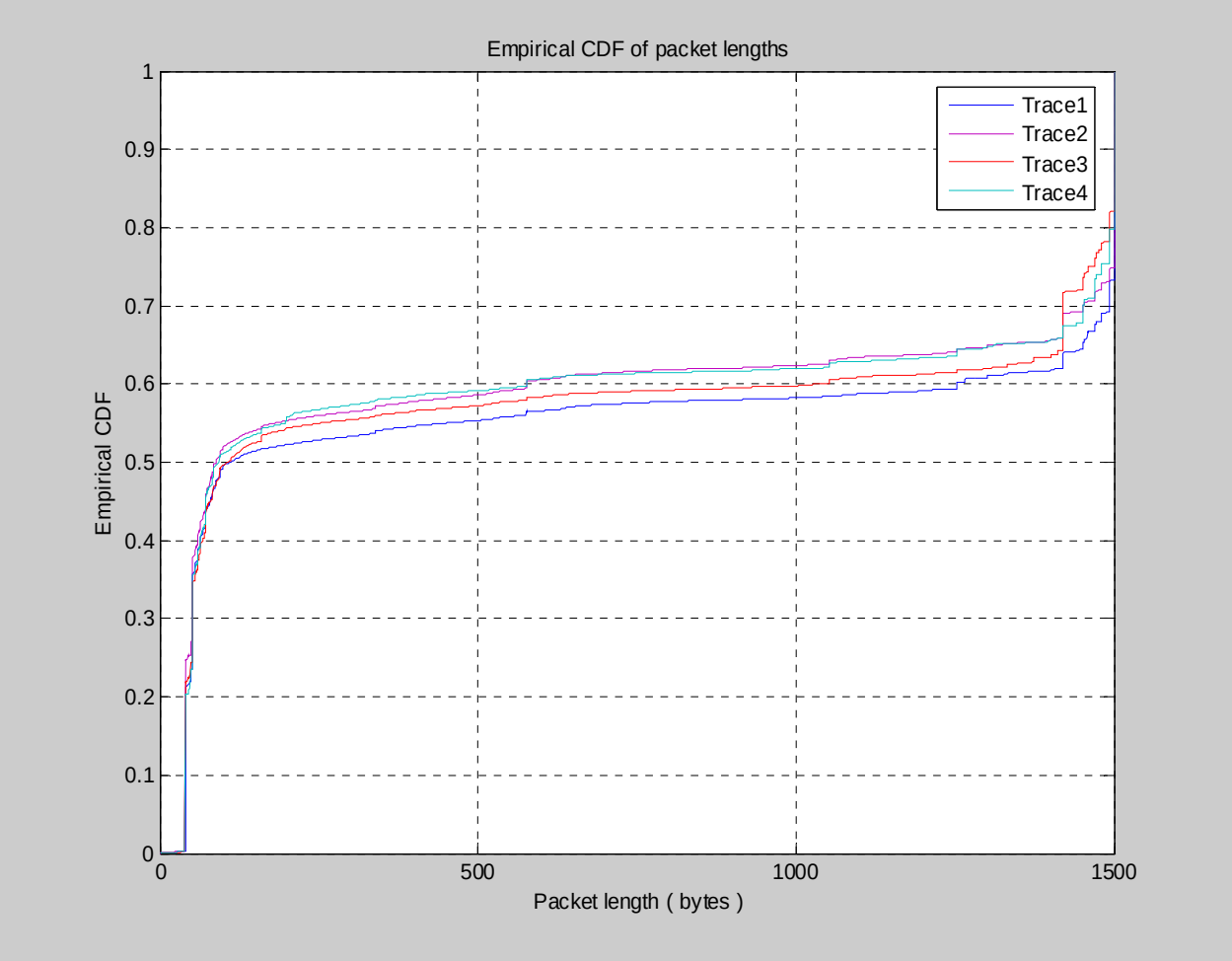

Figure 5.5: Empirical Distribution of packet lengths

Table 5.1 shows the most commonly seen packet lengths. It shows the top ten most frequent packet lengths. The highest value (1500 bytes) undoubtedly refers to the maximum size of IP packet in Ethernet. The packet lengths of size 52 bytes refer to the TCP pure ACKs due to the length of IP and TCP along with 12 bytes of options length in the TCP/IP stack. Another most common size of packet length is 40 bytes (29946), which is purely IP and TCP packet lengths.

Table 5.1: Top frequent packet lengths

Packet length (bytes)	Counts
1480	148781
56	149622
64	159864
60	166881
48	222679
1420	284414
1492	441591
52	1740113
40	2995486
1500	3814832

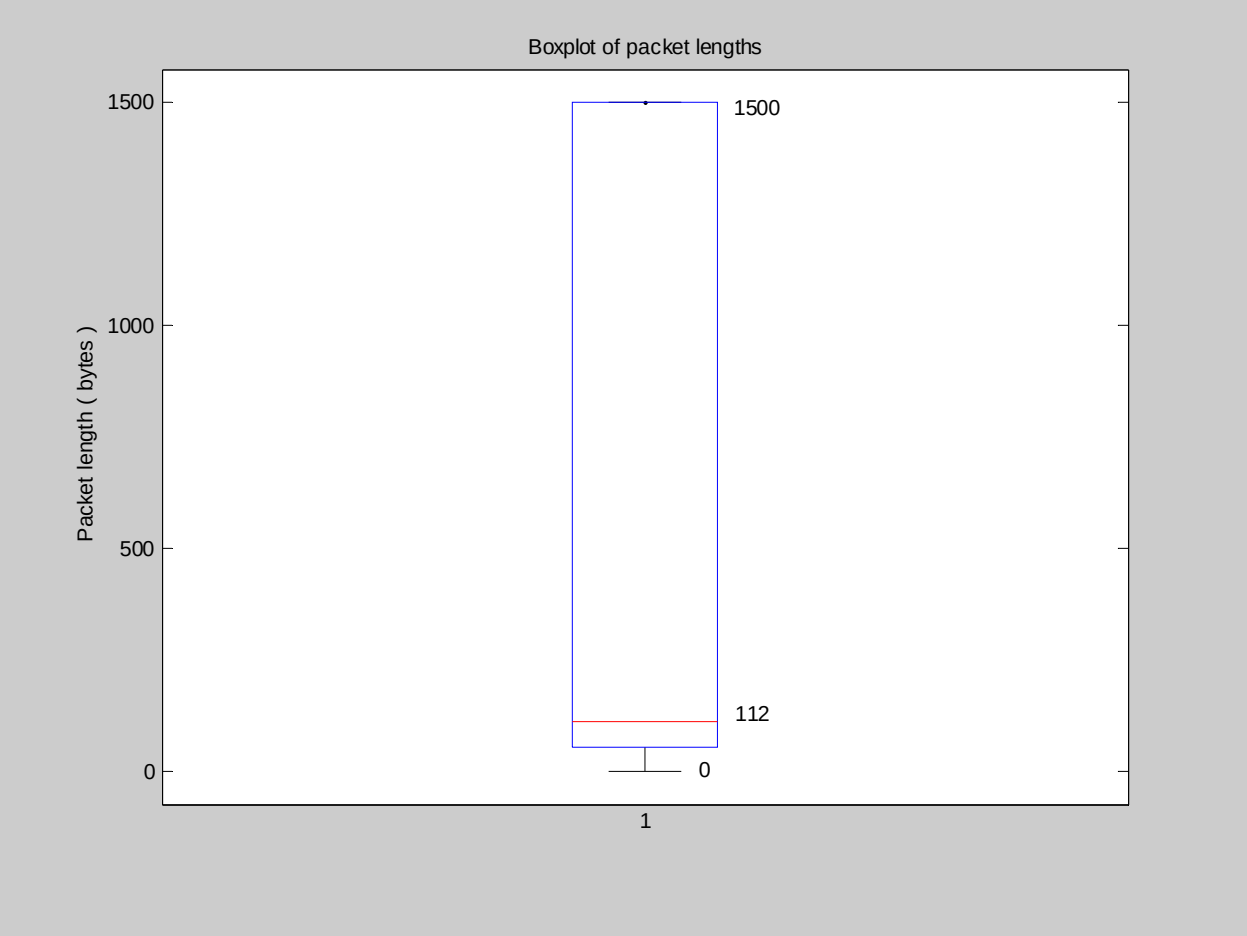

Figure 5.6: Box plot of packet lengths

5.3 Flow Analysis

5.3.1 Flow Distribution by Ports

This section analyzes the distribution of flows according to the ports utilized. The analysis is very similar to section 5.2.1. We consider only TCP and UDP flows which are identified by the protocol numbers 6 and 17 respectively in the data set, as other protocols do not utilize ports numbers. The Figure 5.7 shows the categorization of the ports into three categories namely, ephemeral ports (16376), registered ports (47770) and well-known ports (160). As seen earlier, the share of well-known ports is very small when compared to the other categories.

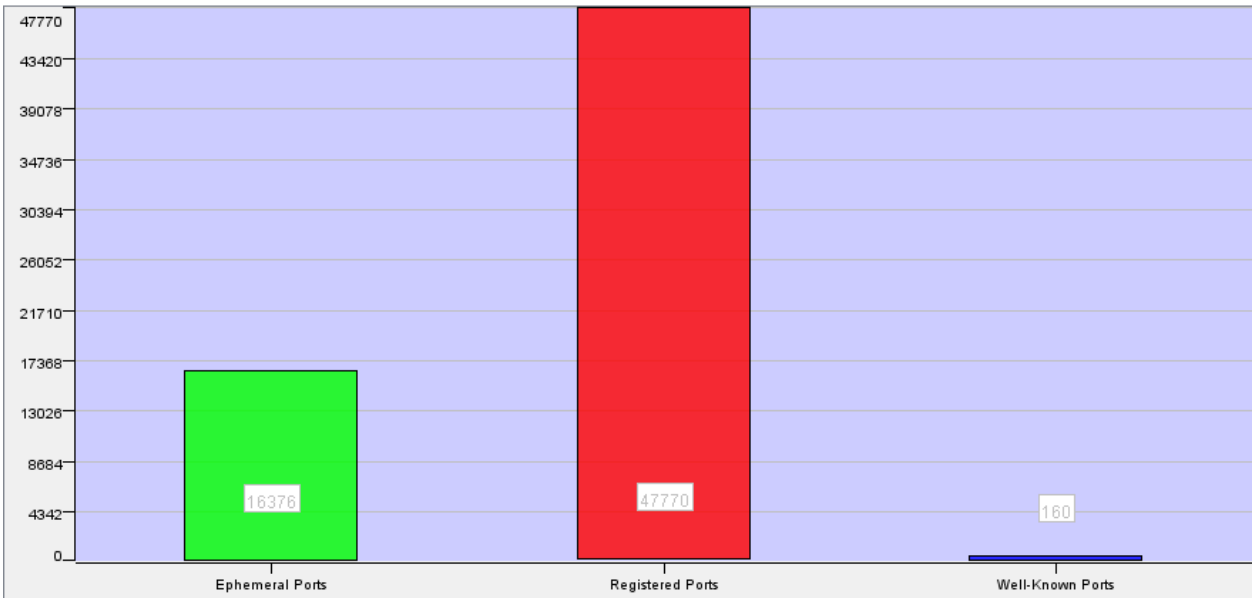

Figure 5.7: Flow distribution by ports

The figure depicts the pie chart for the top ports (well-known ports) seen in the flows. From the plot, it infers that HTTPS (45%) has the top share followed by NETBIOS (14%) and NTP (12%).The plots generated for port distributions at packet-level does not correlates with the one at flow level, because each flow may carry an arbitrary number of packets. Since, there were

very few packets utilizing the ports at packet level but these packets contributed a large number of flows.

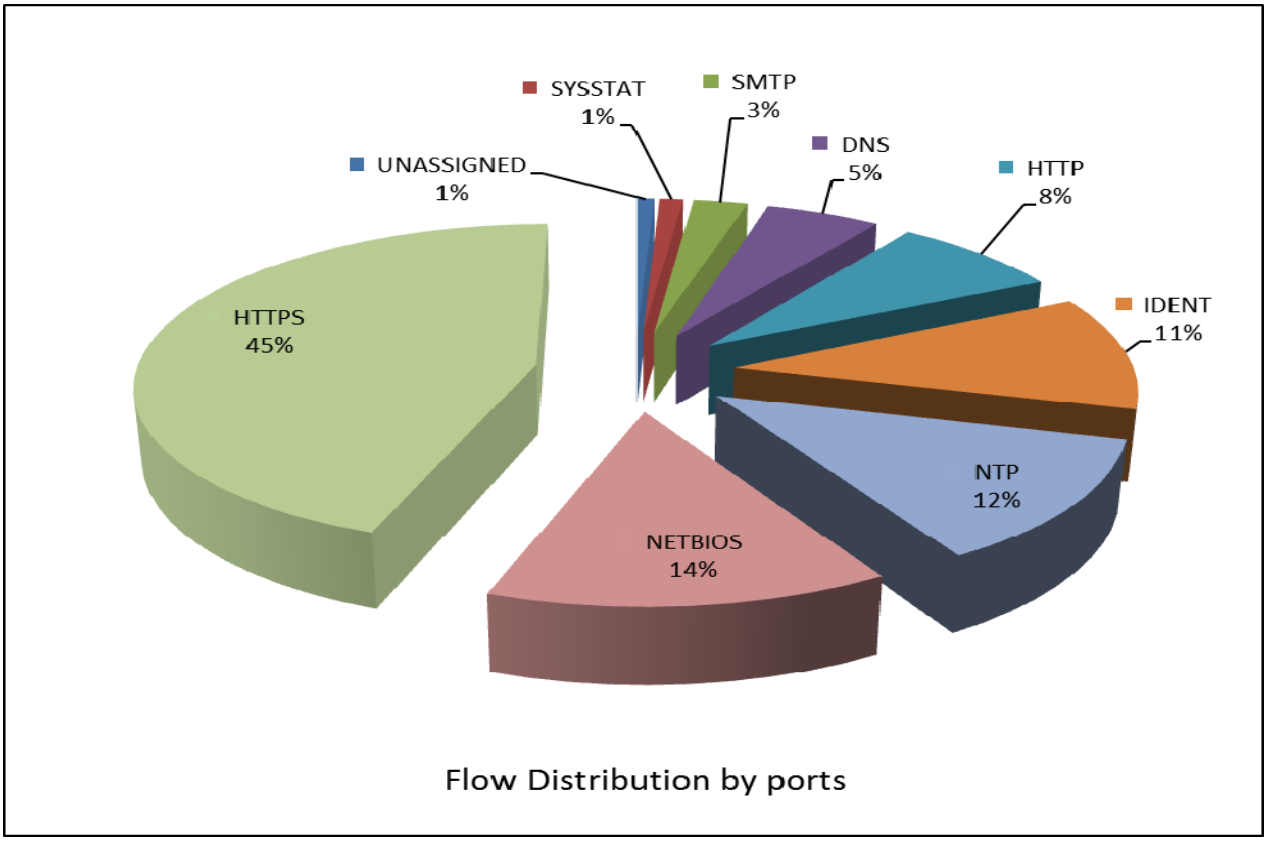

Figure 5.8: Flow distribution by ports (protocols)

5.3.2 Flow Length Distribution

In this section, we analyze the distribution of flow lengths for one complete week. The raw dump file was processed and merged to provide the relevant output. Our aim was not only to plot the distribuiton of flow lengths, but also to fit theoretical probability distributions. Figure 5.9 shows the histogram plot of flow lengths with logarithm of flow lengths on *x-axis* and its frequency on the *y-axis*. The plots depicting the histogram of flow length distribution, its empirical *cumulative*

distribution function (CDF) and the *five-number* summary statistics in the form of a box plot, as shown in Figure 5.9, Figure 5.10 and Figure 5.11 respectively.

The histogram of flow length distribution, with logarithm of flow lengths on x-axis and its frequency on y-axis, has been produced using log scale, because of the high variability of the data set. Further, the plot on log scale aids in better visualization of the data set. From the plot, it infers that the distribution of the flow lengths is heavy-tailed.

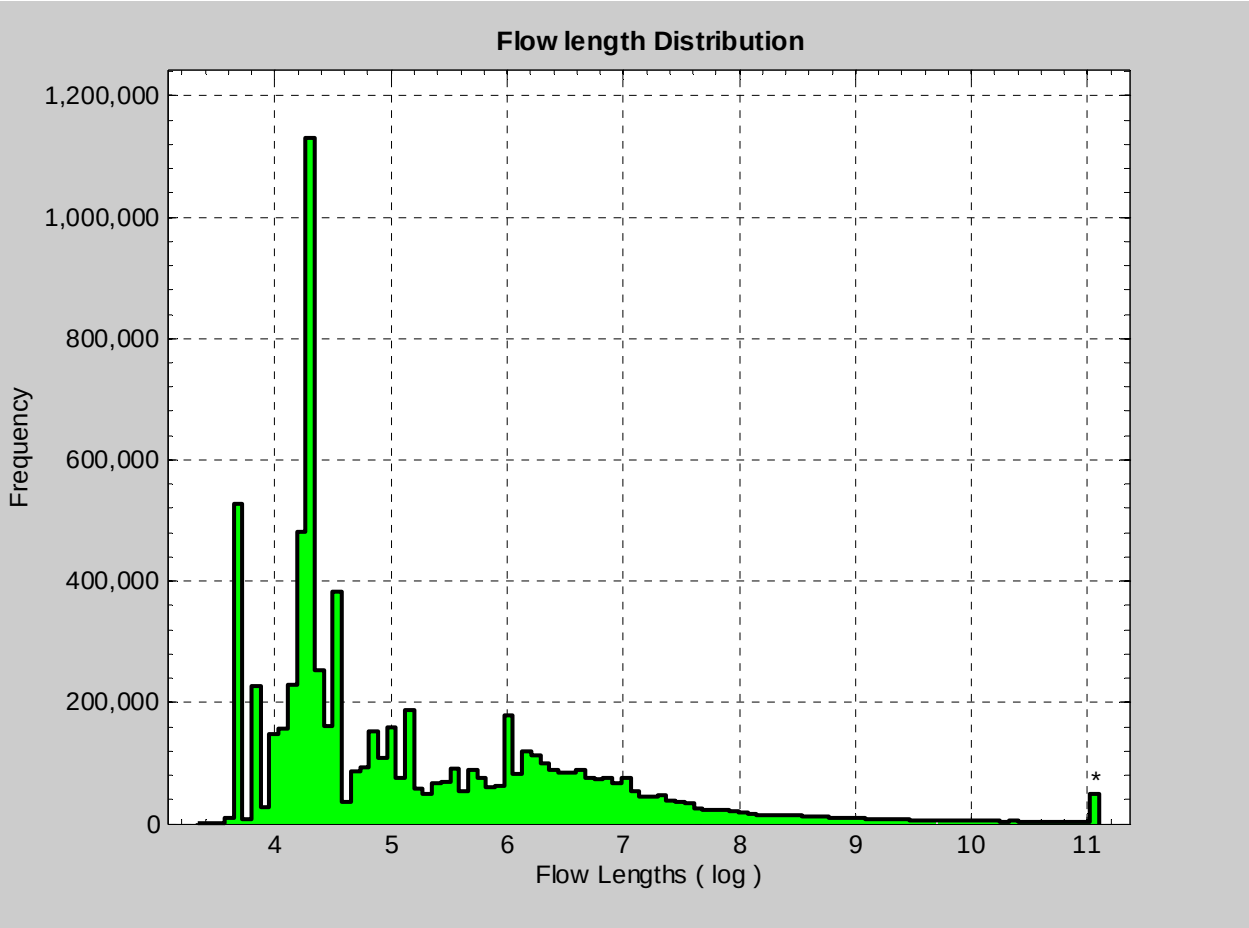

Figure 5.9: Flow length distribution (log scale)

The Figure 5.10, shows the logarithm of ECDF of the flow lengths. At last, we generate a box plot for the flow length data. The box plot provides the *five-number* summary of the flow lengths

for an entire week. From the box plot, we conclude that the number of outliers is very high and the median of the flow lengths is very close to the lower quartile.

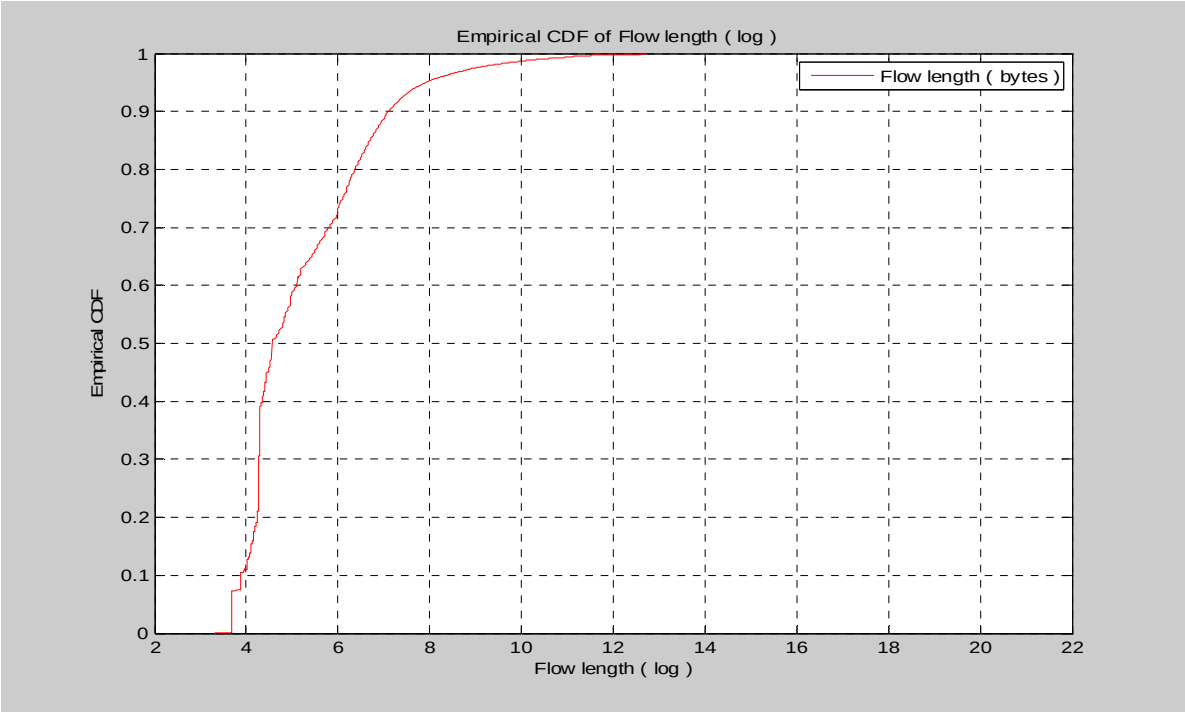

Figure 5.10: Empirical distribution of flow lengths (log scale)

Finally, we have tried to fit a theoretical distribution to the flow lengths data using Matlab. The distributions we tried to fit were Log-normal, Normal, Rayleigh and Weibull. The distribution parameters are described in Table 5.2. Further, the distributions are confirmed by plotting the density of flow lengths on a log scale. From the density plot (Figure 5.12), it is inferred that the density of flow lengths on a log scale. From the density plot (Figure 5.12), it is inferred that the log-normal fit is the closest one to the distribution of flow lengths, compared to other distributions.

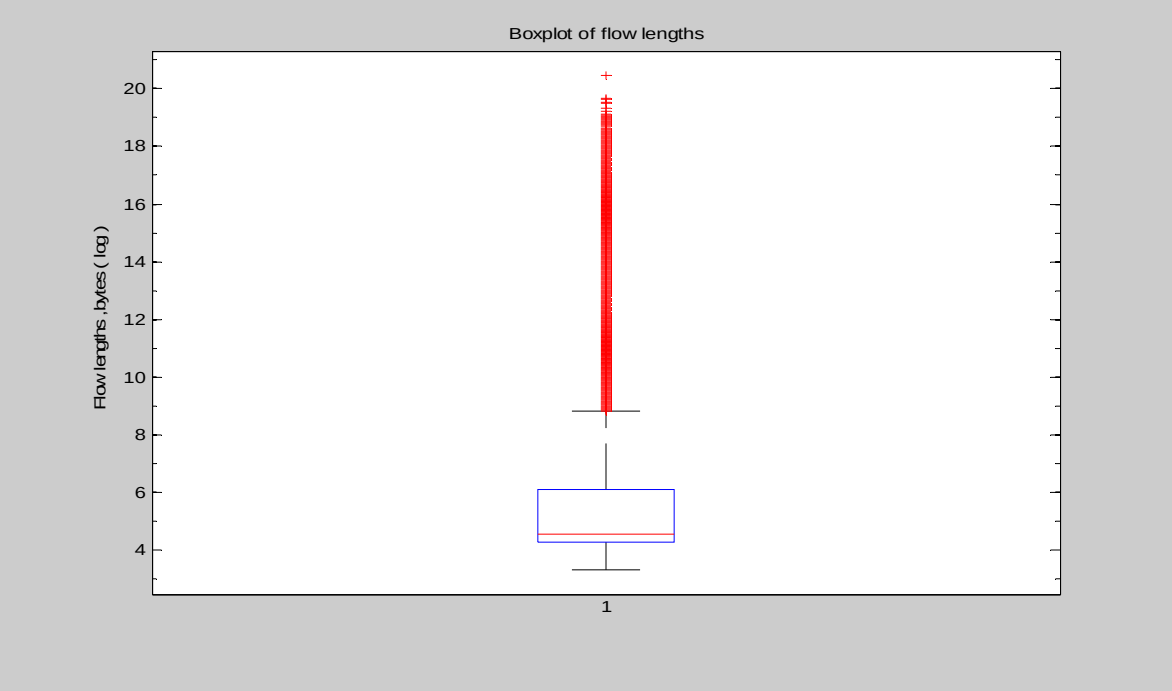

Figure 5.11: Box plot of flow lengths

Also, we tried to plot the cumulative distribution function for the logarithm of flow lengths. From the Figure 5.13, we infer that the log-normal fit is the closest fit to the flow data, which corroborates our hypothesis. We tried to validate the above fitted distributions by plotting their probability and cumulative distribution functions and respective QQ-plots. On the contrary, from the results, it is understood that neither of the distributions fit the flow length data till any satisfactory degree. Apparently, this could happen because of the three outlying bins having high frequencies. We could both try to remove the outlier bins having high frequencies or cut the heavy tail of the distribution and fit a theoretical distribution to the remaining data, but that would not serve the purpose of studying the underlying traffic behavior. Finally, we further extend the fit of the distributions by plotting the cumulative distribution function of the logarithm of flow lengths on a quantile-quantile (Q-Q) plot. The quantile-quantile (Q-Q) plot is a graphical technique for determining if two data sets come from populations with a common distribution

[38]. From, the Q-Q plot (Figure 5.14), it is inferred that the log-normal fit is the closest fit to the flow lengths data.

Table 5.2: Distribution parameters (PDF)

Distributions	Mean	Variance
Normal	5.23941	2.14742
Log-normal	5.22684	1.70637
Rayleigh	4.8215	6.35196
Weibull	5.20102	2.82613

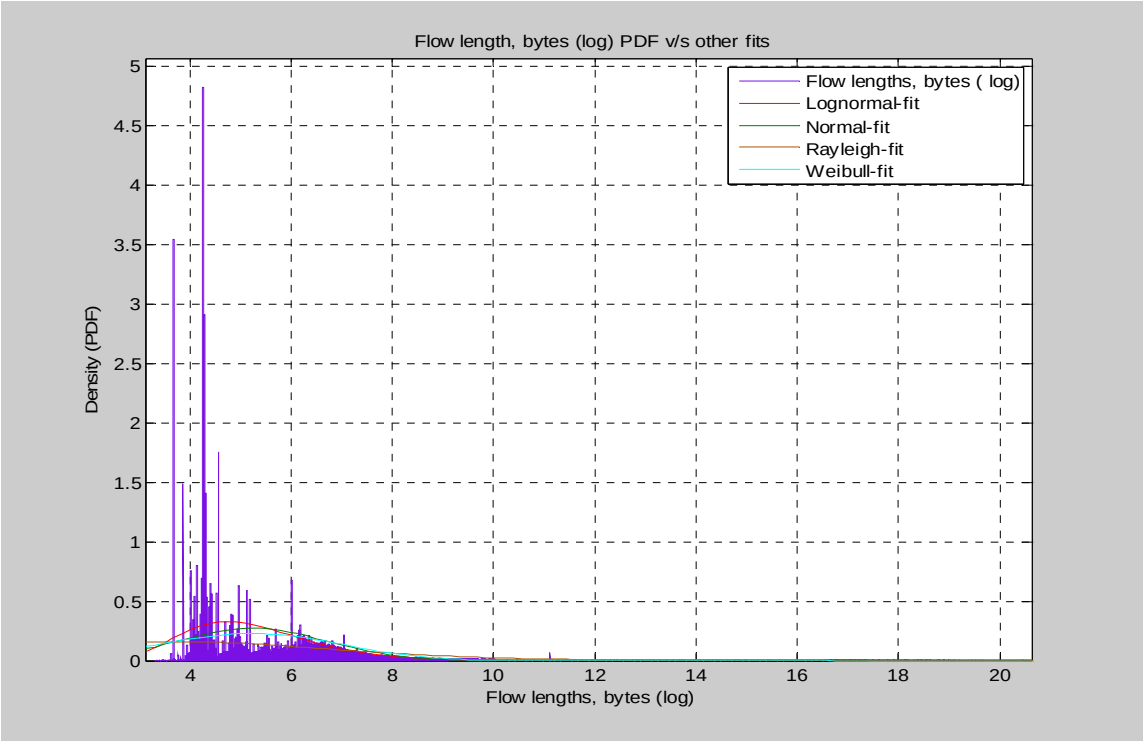

Figure 5.12: Density of flow lengths (log scale)

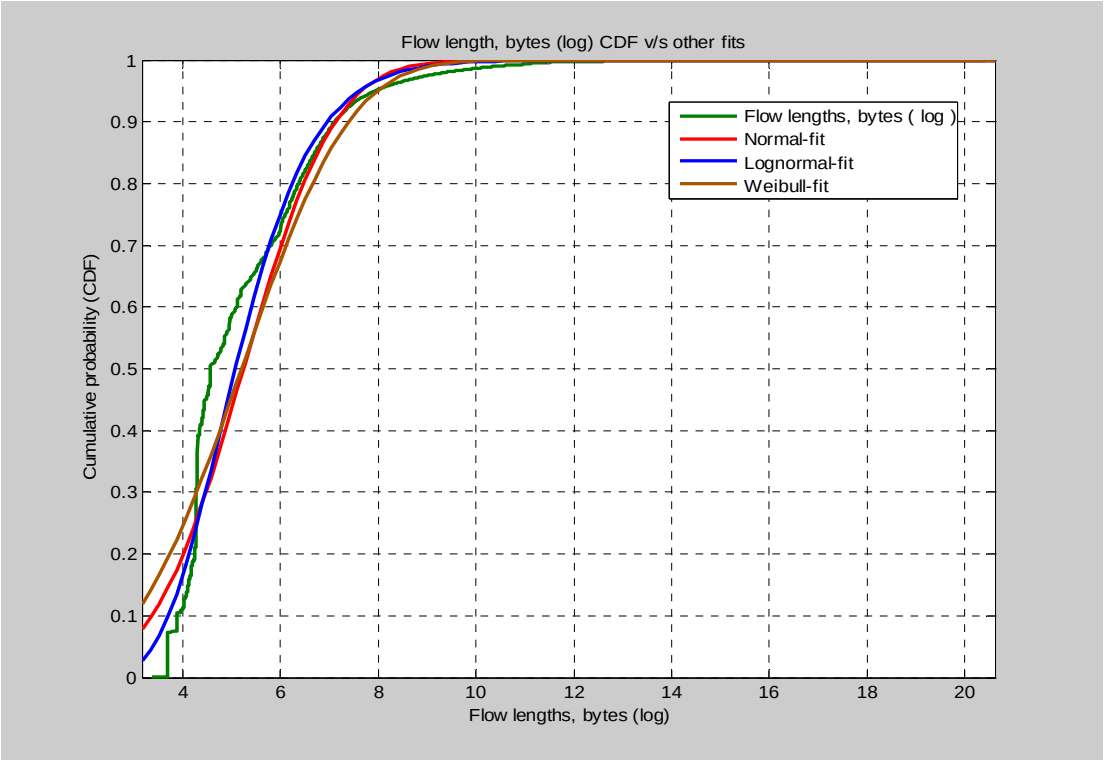

Figure 5.13: Cumulative probability of flow lengths v/s other fits

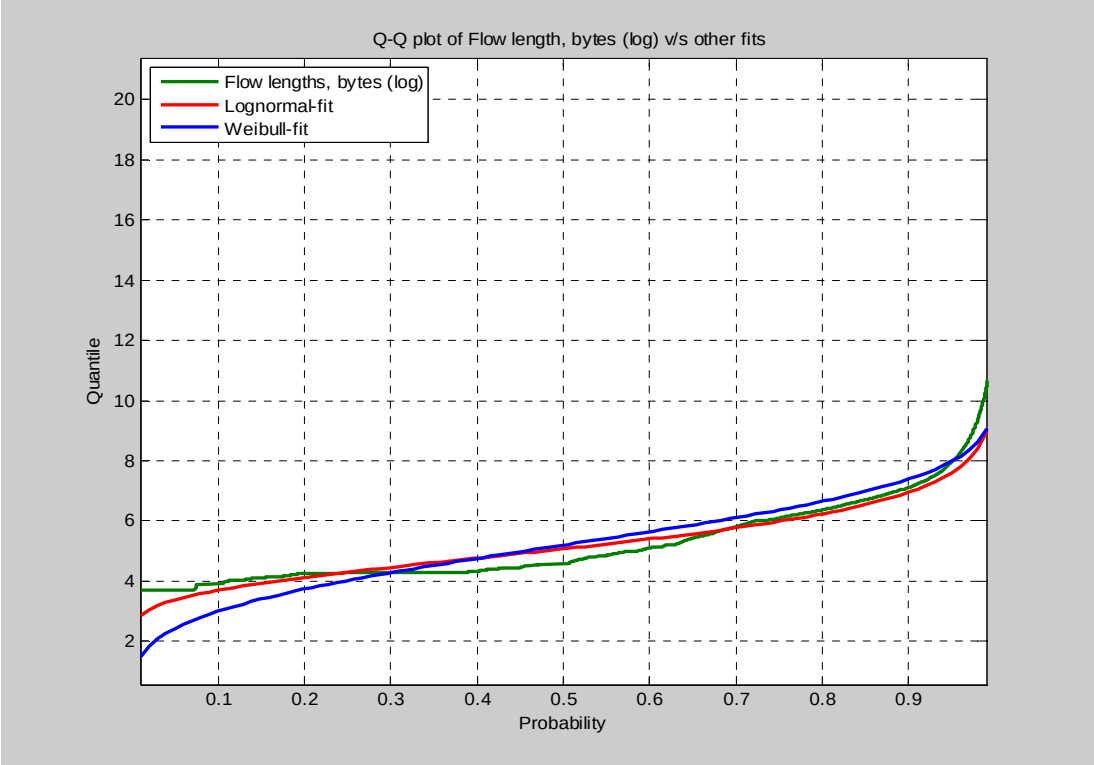

Figure 5.14: Q-Q plot of flow lengths v/s other fits

5.3.3 Flows for varying Timeouts

In this section, we analyze the dependancy of the number of flows for varying timeout values. We utilized the ***crl_flow*** tool with different time out values for four separate traces. The number of flows were aggregated for each file separately. The Figure 5.15 shows the plot of number of flows for different timeout values, where the *x-axis* is timeout value (in seconds) and *y-axis* corresponds to the number of flows. The timeouts were varied in steps of 1, 10, 60, 120 and 1800 secs. From the plot, it infers that higher the timeout value, lesser is the number of flows. Ths is fair since a larger timeout value means the flow can last for longer duration than before. In other words, the timeout values share an inverse relationship with the number of flows. We saw a strong correlation between the number of flows detected and the length of the timeout value, as corroborated in [32]. The paper shows that the number of flows varies inversely with the length of the timeout value.

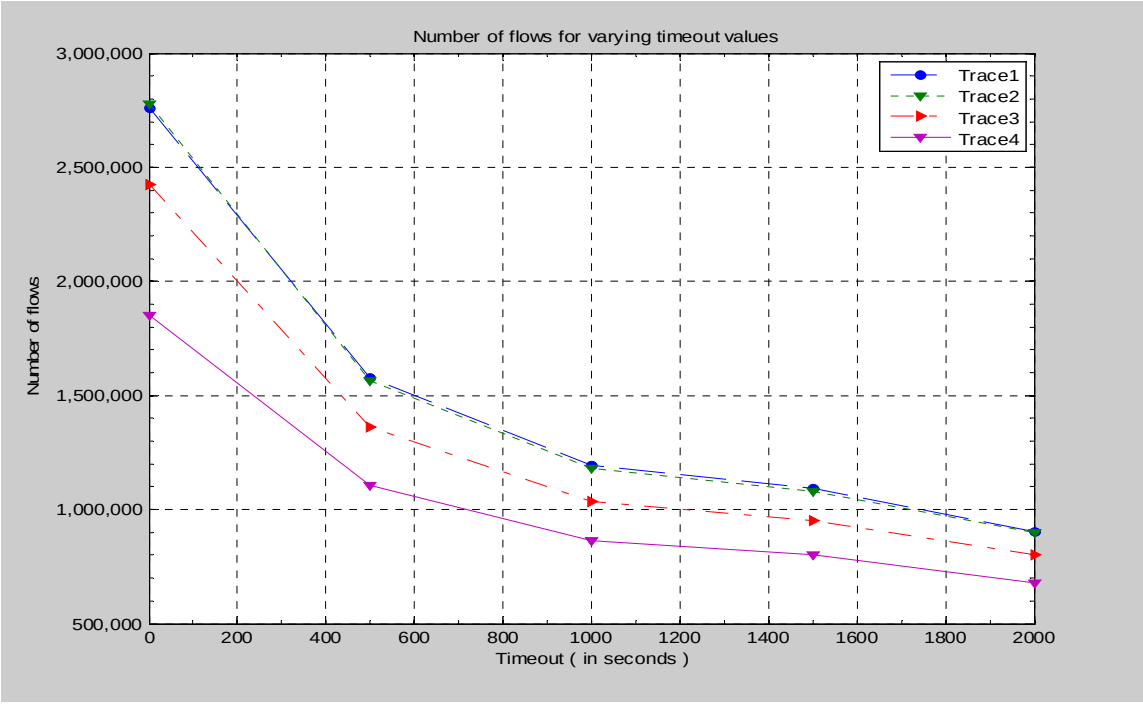

Figure 5.15: Impact of timeout values

5.3.4 Origin-Destination Pairs

In this section, we confine our research to the amount of data flowing between the pairs of IP address (Origin-Destination pairs). The preprocessing of the data is carried out as discussed in chapter 3 to obtain an aggregated IP matrix comprising of origin-destination pair, packets, bytes and flows. From the matrix obtained, we filtered it into origin-destination volume and origin-destination flows. The total number of unique origin-destination pairs found in the data set is 2734965. Figure 5.16 shows the histogram plot for the log of total number of bytes transferred between each set of IP. The plot infers that it is a heavy tail distribution which is common for network traffic [56, 57].

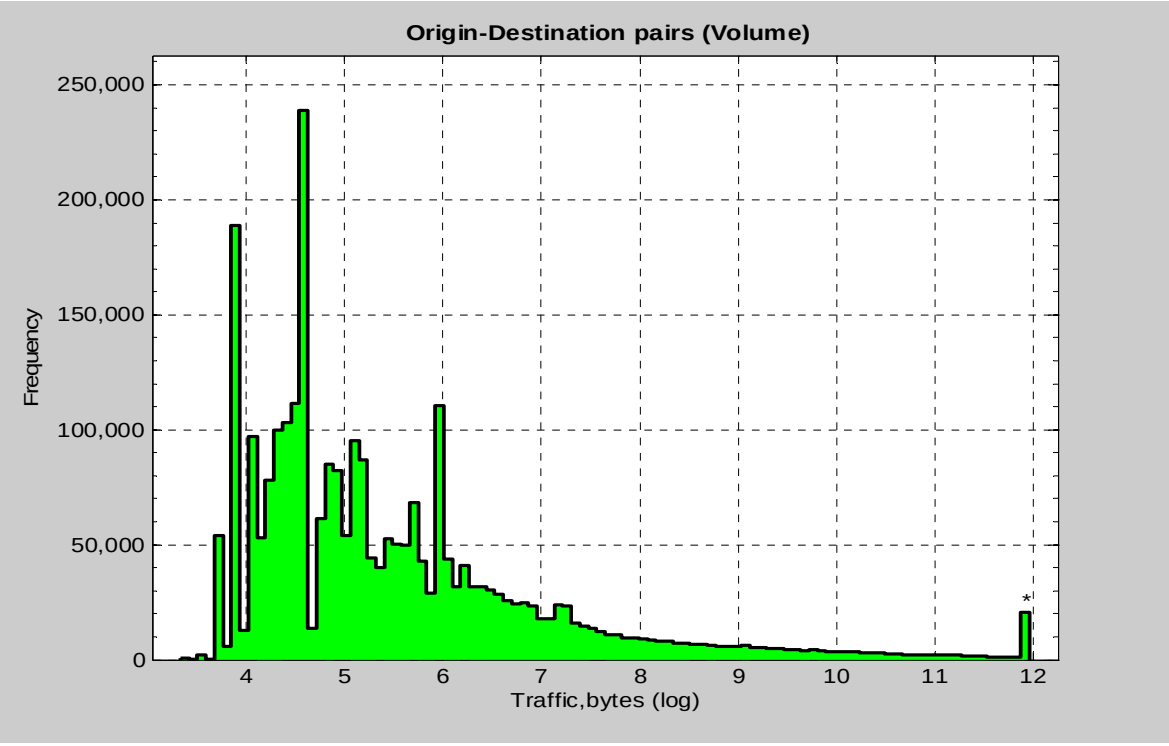

Figure 5.16: Origin-Destination Pairs

In order to visualize the number of flows per origin-destination pair, we tried to generate a *Zipf-type* plot. The reason behind generating *Zipf-type* plot is that it delineates the dependency

between the number of occurrences (x-axis: number of flows) of a variable and its rank (y-axis: frequency). The Zipf's law states that for each following rank, the number of occurrences is α times less than the previous rank. We observe that there are very few flows which are greater than 2000000. The distribution is so extreme that it exhibits a perfect *L-shaped* curve in Figure 5.17. Whereas the Figure 5.18, below shows the same plot, but on a *log-log* scale where the same distribution shows itself to be linear. This is the peculiar signature of a *power-law*. We could see from the plots that the data is highly variable, hence we employed Zipf-type plot with both axes in logarithmic scale. The Zipf's law is often incorporated in network measurements as described in [58, 59].

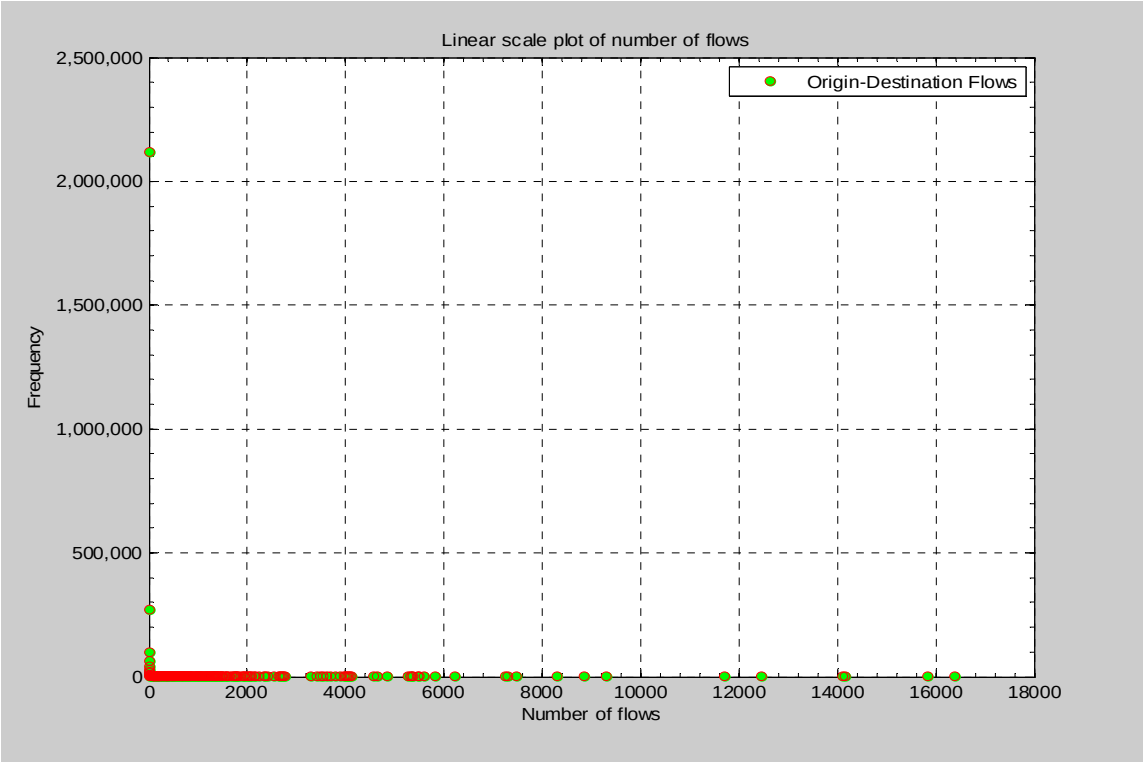

Figure 5.17: Linear scale plot of number of flows

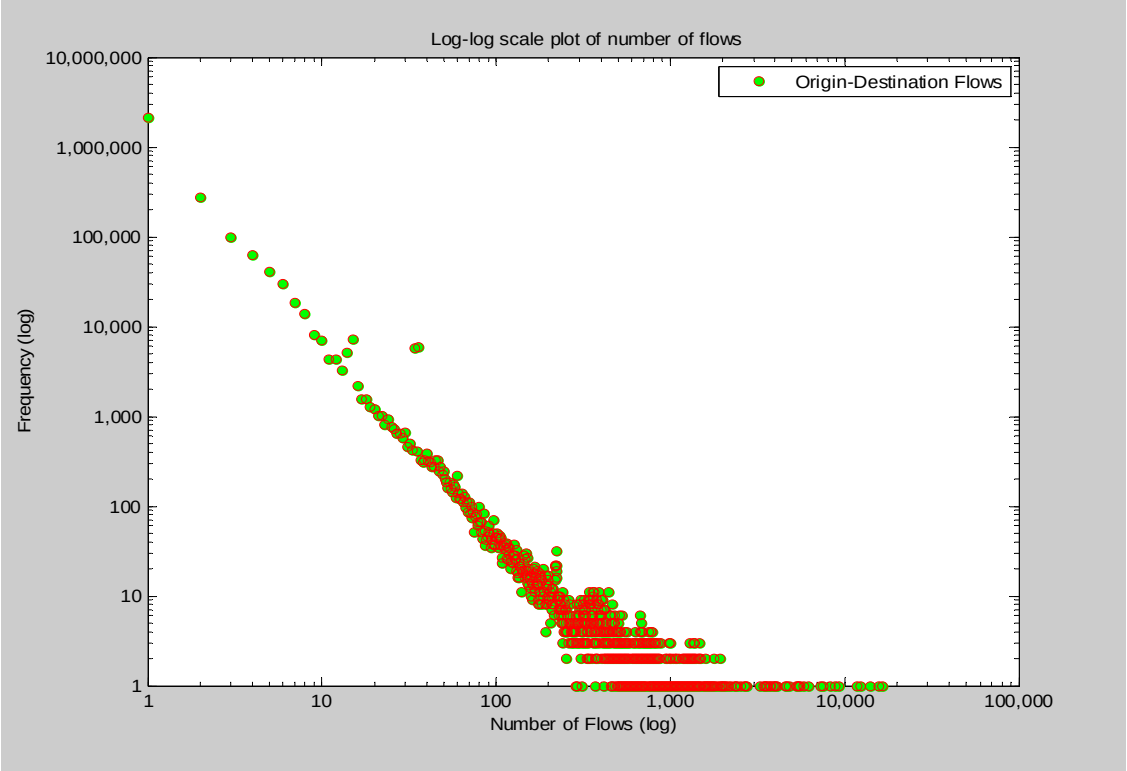

Figure 5.18: Log-log scale plot of the number of flows

5.4 TCP Statistics

In this section, we focus on TCP-specific aspects of the obtained traces. We have utilized custom scripts to extract necessary columns from the output generated by ***tcptrace***. In total, the tool identified 260899 TCP connections in one sample trace file. Our main aim is to relate retransmissions and packet round-trip times. As mentioned earlier in section 3.2.4, the ***tcptrace*** tool calculates the RTT values as difference between the sent data packets and received ACKS preventing dubious cases of retransmissions. While manually traversing through the obtained results, we found out that the RTTs observed in both directions differ hysterically. The tool generates two values for retransmissions as well. For simplicity, we aggregate the two fields representing the retransmissions for the each connection. Further, we try to correlate TCP retransmissions with RTTs within each connection. The Figure 5.19 shows the plot of RTT value

in milliseconds for each TCP connection and Figure 5.20 shows the respective number of retransmissions. From the results obtained, it infers that there is no compelling correlation between the magnitude of retransmission and the RTT value for each TCP connection. We could further deduce that for a group of TCP connections where the RTT value is very high, the corresponding retransmission value is very low. Most of the values for retransmissions are in the range of 200 milliseconds to 300 milliseconds corresponding to the RTT Values. Finally, estimating the correlation coefficients between the two vectors yielded a very low value corroborating the fact that there is no correlation between the RTT and retransmissions.

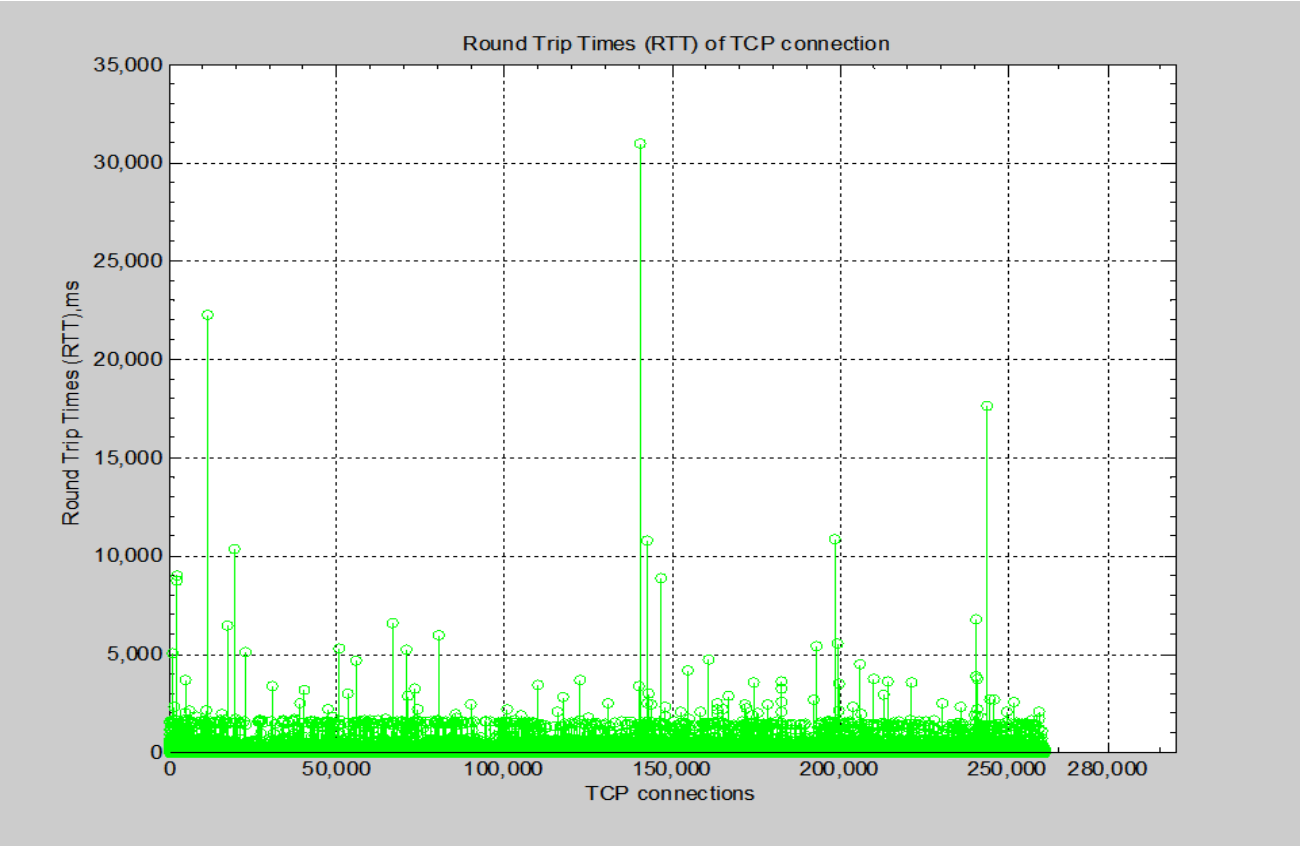

Figure 5.19: Round Trip Times (RTT) of each TCP connection

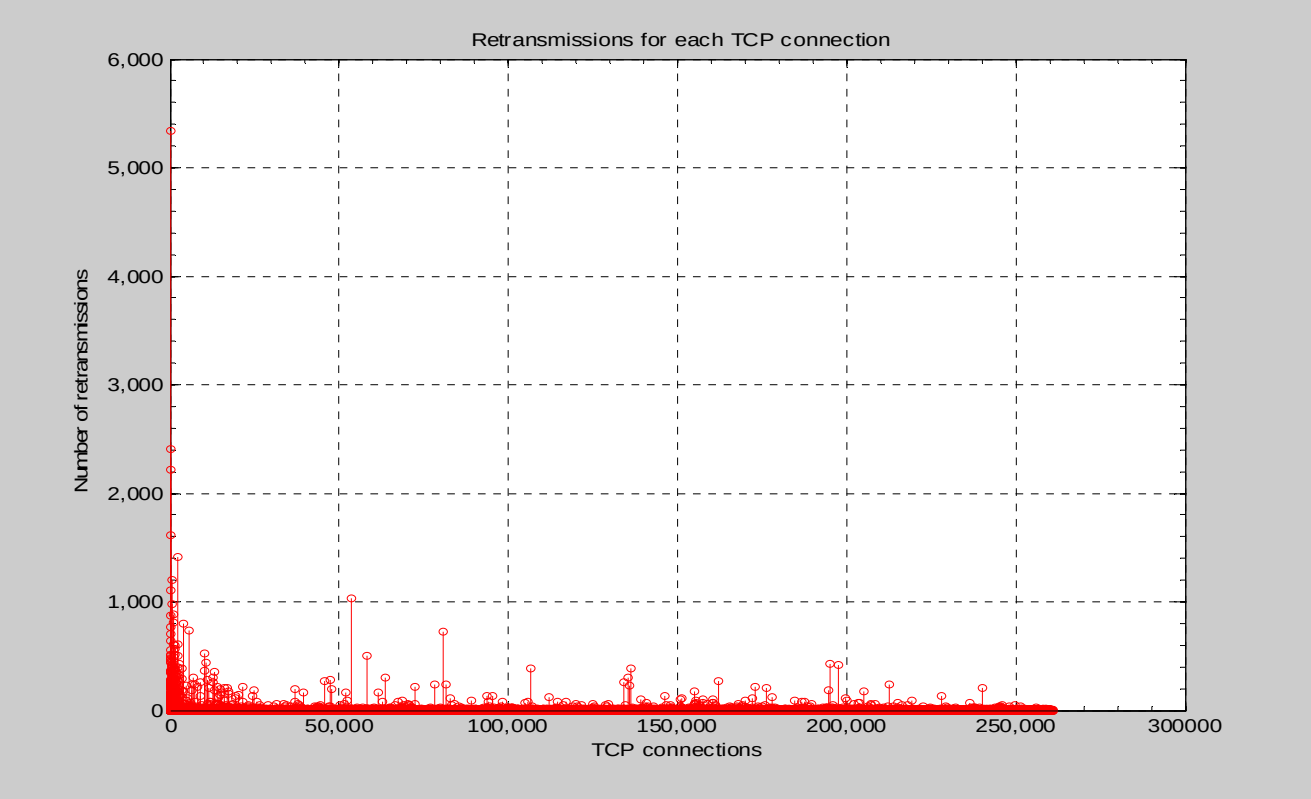

Figure 5.20: Retransmission for each TCP connection

5.5 Conclusion

In this chapter, various aspects of traffic have been analyzed. At packet level, we could not determine sufficient amount of well-known ports to know its contribution to traffic. We realized the distribution of packet lengths and tried different analytical fits to it, to understand the variation in packet lengths. At flow level, same principle was applied for distribution of ports and protocols as seen at packet level. We discovered that the distribution of flow length depicted a heavy tail distribution and of all the fits on different scales, log-normal was the closest fit to the distribution. Apart from the analytical fit to the flow lengths data, the five number summaries, box plot was studied to understand the variability in the data. We observed the inverse effect of timeouts on flows. The flow data was then analyzed as origin-destination pairs. The distribution for the number of flows was generated on both linear and log-log scale. It revealed Zipf type

nature, for the number of flows against its rank. Finally, TCP specific aspects were studied, in particular the effect of retransmissions and RTT's. Unfortunately, we could not find any correlation between them. In the next chapter, we conclude and provide the future scope of work based on this thesis.

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Summary

In this thesis, real traffic traces collected over Trans-Pacific backbone links (the MAWI repository, providing publicly available anonymized traces) was analyzed to study the underlying traffic patterns. One has to have a deep understanding of the packet header structure for various protocols to study the traffic pattern. For the given dataset, we investigated traffic volume temporal patterns and most common protocols in use.

During our research, we learnt the functionality of various tools and how they can be incorporated together to analyze network traffic effectively. The biggest challenge faced throughout this thesis was non-availability of the practical data and source of data. For instance, we tried our best to gather data locally from a campus network but it did not materialize. After one year of our failed efforts, we eventually managed to gather traces from MAWI. The results obtained would have been more compelling if the data utilized was not anonymized. However, the anonymity helps preserve the privacy of the clients or network users.

The size of each trace file is about 1 GB lasting for 15 minutes. Most of the tools available online were capable of reading the trace file only up to few megabytes. The tool *tcpdump* came to rescue the limitations imposed by other software applications. It proved to be a very powerful command line tool in our analysis. Hence, during the course of our research, we learned about several tools like ***tcptrace***, ***tcpstat*** and software packages such as ***perl***, ***CoralReef*** and Matlab. We not only learned the software, but also tried to understand the nature of output generated by them.

The analysis of the data revealed IP, as the most frequently used protocol. At packet level, we studied the contribution of protocols TCP/UDP to the network traffic. Further, we tried to analyze the distribution of packet lengths to check the amount of variability in the distribution.

The packet data was converted to flow data, to study the flow based characteristics. Next, we drilled down to the ports utilized at the flow level. In order to study the amounts of data flowing between pairs of IP addresses, we computed the number of flows and the sum of bytes transferred for each origin-destination pair. We utilized a script to account for both directions of flows between the pairs. To confirm our hypothesis for Zipf type distribution, we generated a Zipf type plot for the number of flows between origin-destination pairs, which corroborated our hypothesis. From the Zipf's distribution, it inferred that there were very few flows that occur frequently. The distribution for flow lengths was also analyzed and confirmed a heavy tailed distribution. We tried to fit various analytical models as discussed in Chapter 2, to the distribution of flow lengths. Log-normal distribution was the only distribution, closest to the fit, comparatively. We further analyzed the effect of timeout on the number of flows. The number of flows varies inversely with timeouts.

Finally, we analyzed TCP related statistics such as RTT and retransmission time. Unfortunately, we could not find any correlation between them. The data was analyzed from three different perspectives: on the *per-packet*, *per-flow* and *per-connection* levels. Further, we have confirmed several well-known facts about the general nature of the traffic such as, *Zipf-type* nature of distribution of number of flows and heavy-tailness of flow sizes. Further, we tried to derive and evaluate analytical models that would fit the data best.

6.2 Future Scope of Work

The first thing to do is to create the simulation of the network analyzed. Another, interesting topic to extend our work is analyzing the multi-fractal nature of the traffic. For instance, network measurements done at various time scales (milliseconds, day or month) provide fractal or multi-fractal behavior of the traffic. This information is vital in the simulation of traffic patterns or generating simulated traffic for network analysis.

The number of traffic traces in the data set was quite large and the work demanded the automation of handling and processing the raw data. The distributions incorporated here, were primarily selected to be fitted to flow and packet inter-arrival time curves. Applications in other areas such as Quality of Service (QoS) and network management need to be investigated. By using the analyzed traffic data, a rough guess of the topology can be created in the absence of network topology. One could create a Matlab toolkit for carrying out the analysis at various levels of granularity. We could extend the same approach for detecting denial of service attacks by utilizing the tools from this research. Traffic profiling is another important aspect of network traffic analysis. Further, visualizing the network traffic data enhances its understanding. CAIDA has developed several tools for visualizing the network traffic data.

BIBLIOGRAPHY

[1] K. Cho, K. Mitsuya, and A. Kato, “Traffic data repository at the WIDE project,” in Proceedings of the annual conference on USENIX Annual Technical Conference, San Diego, California, 2000, pp. 51-51.

[2] B. R. Chang, and H. F. Tsai, “Improving network traffic analysis by foreseeing data-packet-flow with hybrid fuzzy-based model prediction,” *Expert Systems with Applications,* vol. 36, no. 3, Part 2, pp. 6960-6965, 2009.

[3] www.stanza-consulting.com. "MATADOR Product Description and Factsheet," 28, 2011.

[4] F. Sally, and P. Vern, “Difficulties in simulating the internet,” *IEEE/ACM Trans. Netw. %@ 1063-6692,* vol. 9, no. 4, pp. 392-403, 2001.

[5] V. Paxson, “Strategies for sound internet measurement,” in Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, Taormina, Sicily, Italy, 2004, pp. 263-271.

[6] C. Kenjiro, M. Koushirou, and K. Akira, "Traffic data repository at the WIDE project," *Proceedings of the annual conference on USENIX Annual Technical Conference*, USENIX Association, 2000, pp. 51-51.

[7] P. Borgnat, G. Dewaele, K. Fukuda *et al.*, "Seven Years and One Day: Sketching the Evolution of Internet Traffic." pp. 711-719.

[8] J. Schormans, and T. Timotijevic, "Evaluating the Accuracy of Active Measurement of Delay and Loss in Packet Networks Management of Multimedia Networks and Services," Lecture Notes in Computer Science A. Marshall and N. Agoulmine, eds., pp. 409-421: Springer Berlin / Heidelberg, 2003.

[9] S. Chevul, L. Isaksson, M. Fiedler *et al.*, "Measurement of Application-Perceived Throughput of an E2E VPN Connection Using a GPRS Network Wireless Systems and Network Architectures in Next Generation Internet," Lecture Notes in Computer Science M. Cesana and L. Fratta, eds., pp. 255-268: Springer Berlin / Heidelberg, 2006.

[10] K. Kivinen, *Milestones in telecommunications in some selected countries*, vol. Report 3/94, Helsinkin University of Technology, 1994.

[11] A. Lakhina, K. Papagiannaki, M. Crovella *et al.*, “Structural analysis of network traffic flows,” *SIGMETRICS Perform. Eval. Rev.,* vol. 32, no. 1, pp. 61-72, 2004.

[12] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemometrics and Intelligent Laboratory Systems,* vol. 2, no. 1-3, pp. 37-52, 1987.

[13] M. Roughan, “Fundamental bounds on the accuracy of network performance measurements,” *SIGMETRICS Perform. Eval. Rev.,* vol. 33, no. 1, pp. 253-264, 2005.

[14] S. Deering, "Internet Protocol, Version 6 (IPv6)," RFC2460, December 1998.

[15] C. Partridge, "Using the Flow Label Field in IPv6," RFC 1809, June 14 1995.

[16] H. Zimmermann, “OSI Reference Model--The ISO Model of Architecture for Open Systems Interconnection,” *Communications, IEEE Transactions on,* vol. 28, no. 4, pp. 425-432, 1980.

[17] A. S.Tanenbaum, *Computer Networks*: Prentice Hall, 2002.

[18] J. Postel. "Internet protocol (darpa internet program protocol specification). RFC791."

[19] J. Postel, "User Datagram Protocol," *RFC768*, August 1980.

[20] J. Postel, "Transmission Control Protocol," *RFC793*, September 1981.

[21] J. Postel, "Internet Protocol (Darpa internet program protocol specification).", September 1981.

[22] D. L. Mills, "Exterior Gateway Protocol Formal Specification," *RFC 904*, April 1984.

[23] J. Moy, "Ospf version 2," *RFC2328*, April 1998.

[24] G. S. Malkin, "Rip version 2," *RFC2453*, November 1998.

[25] T. L. Yakov Rekhter, "A border gateway protocol 4 (bgp-4) RFC1771," March 1995.

[26] D. C. Plummer, " An Ethernet Address Resolution Protocol," *RFC 826*, November 1982.

[27] T. M. Ross Finlayson, Jeffery Mogul, Marvin Theimer, "A Reverse Address Resolution Protocol," *RFC 903*, 1984.

[28] R. Droms, "Dynamic Host Configuration Protocol," *RFC 2131*.

[29] J. Postel, "Internet Control Message Protocol," *RFC792*, September 1981.

[30] "IANA (Internet Assignment Number Authority)," 2011; http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt.

[31] C. S. Lee, "Network Flow: Unidirectional VS Bi-Directional," June 2008.

[32] K. C. Claffy, H. W. Braun, and G. C. Polyzos, “A parameterizable methodology for Internet traffic flow profiling,” *Selected Areas in Communications, IEEE Journal on,* vol. 13, no. 8, pp. 1481-1494, 1995.

[33] "CAIDA Measurement and Analysis Tools," *CoralReef suite*, 1999.

[34] N. W. Group, "Traffic Flow Measurement Architure," *CRC Technical Note No. CRC-TN-2005-003*, 1999.

[35] M. E. Crovella, and A. Bestavros, “Self-similarity in World Wide Web traffic: evidence and possible causes,” *Networking, IEEE/ACM Transactions on,* vol. 5, no. 6, pp. 835-846, 1997.

[36] B. C. Arnold, "Pareto Distribution," *Encyclopedia of Statistical Sciences*: John Wiley & Sons, Inc., 2004.

[37] *NIST/SEMATECH e-Handbook of Statistical Methods*, 2011.

[38] "Electronic Statistics Textbook," http://www.statsoft.com/textbook/distribution-fitting/.

[39] B. A. H. Lada A. Adamic, “Zipf's law and the Internet,” *Glottometrics 3*, 2002.

[40] J. Chambers, William Cleveland, Beat Kleiner, and Paul Tukey, *Graphical Methods for Data Analysis*, Wadsworth, 1983.

[41] A. F. S. a. C. J.Morgan, *Statistics and Data Analysis: An Introduction*, 2nd edition ed.: John Wiley & Sons, Inc., 1996.

[42] C. L. Van Jacobson, and Steven McCanne "tcpdump," June 1989.

[43] S. Ostermann, "tcptrace."

[44] P. Herman, "tcpstat," 2001.

[45] "tcpdump man page," http://www.tcpdump.org/tcpdump_man.html.

[46] T. Viipuri, “Traffic Analysis and Modeling of IP Core Networks,” Department of Electrical and Communications Engineering, Helsinki University Of Technology, 2004.

[47] R. Fontugne, P. Borgnat, P. Abry *et al.*, “MAWILab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking,” in Proceedings of the 6th International COnference, Philadelphia, Pennsylvania, 2010, pp. 1-12.

[48] G. Minshall, "tcpdpriv," Ipsilon Networks, Inc., 31 Oct' 05.

[49] M. Jain, and C. Dovrolis, “End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput,” *SIGCOMM Comput. Commun. Rev.,* vol. 32, no. 4, pp. 295-308, 2002.

[50] Y. Zhang, L. Breslau, V. Paxson *et al.*, “On the characteristics and origins of internet flow rates,” *SIGCOMM Comput. Commun. Rev.,* vol. 32, no. 4, pp. 309-322, 2002.

[51] S. Ratnasamy, M. Handley, R. Karp *et al.*, "Topologically-aware overlay construction and server selection." pp. 1190-1199 vol.3.

[52] M. Siekkinen, G. Urvoy-Keller, E. W. Biersack *et al.*, “A root cause analysis toolkit for TCP,” *Computer Networks,* vol. 52, no. 9, pp. 1846-1858, 2008.

[53] B. Veal, K. Li, and D. Lowenthal, "New Methods for Passive Estimation of TCP Round-Trip Times," *Passive and Active Network Measurement*, Lecture Notes in Computer Science C. Dovrolis, ed., pp. 121-134: Springer Berlin / Heidelberg, 2005.

[54] S. Floyd, and V. Paxson, “Difficulties in simulating the internet,” *IEEE/ACM Trans. Netw.,* vol. 9, no. 4, pp. 392-403, 2001.

[55] R. Pang, V. Yegneswaran, P. Barford *et al.*, “Characteristics of internet background radiation,” in Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, Taormina, Sicily, Italy, 2004, pp. 27-40.

[56] V. Paxson, and S. Floyd, “Wide area traffic: the failure of Poisson modeling,” *IEEE/ACM Trans. Netw.,* vol. 3, no. 3, pp. 226-244, 1995.

[57] W. Willinger, M. S. Taqqu, R. Sherman *et al.*, “Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level,” *IEEE/ACM Trans. Netw.,* vol. 5, no. 1, pp. 71-86, 1997.

[58] V. Almeida, A. Bestavros, M. Crovella *et al.*, "Characterizing reference locality in the WWW." pp. 92-103.

[59] L. Breslau, C. Pei, F. Li *et al.*, "Web caching and Zipf-like distributions: evidence and implications." pp. 126-134 vol.1.

VITA

Harish Kapri was born in Andhra Pradesh, in December 1984, India. He earned his primary and secondary education from Army School in Secunderabad, Andhra Pradesh. After finishing his high school, he got admission in the Department of Electronics and Telecommunication Engineering, Anjuman College of Engineering and Technology in Nagpur. He received his Electronics and Telecommunication Engineering from Rashtrasant Tukadoji Maharaj Nagpur University (formerly known as Nagpur University), Nagpur, India, in May 2006. After his graduation, he worked with Dell International Services pvt. Ltd., India, where he served as a senior resolution expert in resolving computer hardware and software related issues. After working with Dell for two years, he left and joined with HSBC, India, and worked as an IT Executive, providing IT support for the bank's employees. After working for six months with HSBC, he came to the United States of America to pursue Master's Degree. He then joined the graduate program at Louisiana State University, January 2009. He is a candidate for the degree of Master of Science in Electrical Engineering to be awarded at the commencement of fall 2011.