HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering

Timo Viipuri

# Traffic Analysis and Modeling of IP Core Networks

Master's thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology

Supervisor: Professor Jorma Jormakka
Instructor: Lic. Tech. Marko Luoma

Espoo, 3rd December 2004

| | |
|---|---|
| **Author:** | Timo Viipuri |
| **Department:** | Department of Electrical and Communications Engineering |
| **Major:** | Networking Technology |
| **Minor:** | Telecommunications Software |
| **Title:** | Traffic Analysis and Modeling of IP Core Networks |
| **Date:** | 3rd December 2004 **Number of pages:** 123 |
| **Professorship:** | S-38 Networking Technology |
| **Supervisor:** | Professor Jorma Jormakka |
| **Instructor:** | Lic. Tech. Marko Luoma |

There are many reasons for analyzing traffic in an operational IP network: studying user behavior, measuring the network performance, observing network usage levels and identifying fault situations. All communication networks in general are monitored at some level by the administrators to be aware of how the users behave and how the network meets their demands.

Simulations provide an efficient and economical way of estimating the performance of a network. They are widely utilized when studying communication networks and they have proven especially useful when designing new and novel networks. However, an unresolved issue has always been: how reliable are the results from the simulations?

This thesis presents a method for evaluating the accuracy of network traffic simulations. The method consists of three stages: analysing the traffic in a real network, performing simulations of the corresponding scenario, and finally, comparing the performance of the real and simulated networks. If the two networks function similarly, simulations can be said to accurately mimic the real world. In this thesis, only the first stage — analysing the real network — is completed and the simulations are left for further studies.

A passive measurement infrastructure for monitoring and analyzing a network was designed and implemented. Traffic was captured from multiple locations in the network and analyzed on a one-week time scale. One day of the week was selected where busy and slow hours were analyzed in more detail. The analysis had two objectives: gathering traffic generation statistics and measuring the network performance. Based on the former objective, a traffic model was built which can be used for replicating the network conditions into a simulation environment. The latter objective was achieved by measuring two parameters: one-way packet latency and packet loss ratio. In addition, a traffic property called self-similarity was measured which can be used in the simulations to evaluate the quality of the traffic model.

**Keywords:** IP traffic, analysis, modeling

| | |
|---|---|
| **Tekijä:** | Timo Viipuri |
| **Osasto:** | Sähkö- ja tietoliikennetekniikan osasto |
| **Pääaine:** | Tietoverkkotekniikka |
| **Sivuaine:** | Tietoliikenneohjelmistot |
| **Työn nimi:** | IP-runkoverkkojen liikenteen analysointi ja mallinnus |
| **Päivämäärä:** | 3.12.2004 **Sivumäärä:** 123 |
| **Professuuri:** | S-38 Tietoverkkotekniikka |
| **Valvoja:** | Professori Jorma Jormakka |
| **Ohjaaja:** | Lisensiaatti Marko Luoma |

Toiminnassa olevan IP-verkon liikenteen analysoimiseen on monta syytä: käyttäjien toiminnan seuraaminen, verkon suorituskyvyn mittaaminen, verkon käyttöasteen tarkkaileminen sekä vikatilanteiden tunnistaminen. Yleisesti ottaen kaikkia tietoverkkoja valvotaan jollain tasolla, jotta ylläpitäjät pysyisivät tietoisina käyttäjien toimista ja verkon toiminnallisuudesta.

Simulaatiot tarjoavat tehokkaan ja taloudellisen tavan arvioida verkon suorituskykyä. Niitä hyödynnetään paljon tietoverkkojen tutkimuksessa ja ne ovat osoittautuneet erityisen hyödyllisiksi uusien ja uudenlaisten verkkojen suunnittelussa. Ratkaisematon ongelma on kuitenkin aina ollut: kuinka luotettavia ovat simulaatioista saadut tulokset?

Tämä diplomityö esittelee menetelmän verkkoliikennesimulaatioiden tarkkuuden arvioimiseksi. Menetelmä on kolmivaiheinen: oikean verkon liikenteen analysoiminen, vastaavanlaisen tilanteen simuloiminen, ja lopuksi, oikean ja simuloidun verkon suorituskykyjen vertaaminen. Mikäli verkkojen toiminnallisuus on samankaltaista, simulaation voidaan arvioida jäljittelevän tarkasti oikeaa verkkoa. Tässä diplomityössä suoritetaan ainoastaan menetelmän ensimmäinen vaihe: oikean verkon analysointi. Simulointiosuus jätetään tulevan tutkimuksen varaan.

Työssä suunniteltiin ja toteutettiin verkon passiivinen liikennemittausjärjestelmä. Liikennettä tarkkailtiin useista eri mittauspisteistä ja analysoitiin viikon pituiselta ajanjaksolta. Tältä ajalta valittiin yksi päivä, jolta analysoitiin tarkemmin kiireisen ja hiljaisen tunnin liikenne. Analysoinnilla oli kaksi tavoitetta: liikenteen luomisen tilastollinen tarkastelu ja verkon suorituskyvyn mittaaminen. Edellisen tavoitteen perusteella kehitettiin liikennemalli, jota voidaan käyttää verkko-olosuhteiden jäljentämisessä simulaatioympäristöön. Jälkimmäinen tavoite saavutettiin mittaamalla kahta tunnuslukua: yksisuuntaista pakettiviivettä ja pakettihäviösuhdetta. Lisäksi mitattiin liikenteen ominaisuutta nimeltä itsesimilaarisuus, mitä voidaan hyödyntää simulaatiovaiheessa liikennemallin laadun arvioimisessa.

**Avainsanat:** IP-liikenne, analysointi, mallinnus

# Preface

This work has been done in the Networking laboratory of Helsinki University of Technology, Finland.

I would like to thank my supervisor, professor Jorma Jormakka and my instructor Lic. Tech. Marko Luoma for their contribution to this master's thesis. Also, Lic. Tech. Markus Peuhkuri deserves my gratitude for providing me with many useful hints and advices on the work. Special thanks go to my colleagues and good friends in the Networking laboratory with whom I have had lots of fun and laughs both during and outside the office hours.

Finally, I would like to express my gratidude to my parents who have helped and supported me throughout my life.

Espoo, 3rd December 2004

Timo Viipuri

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

The need to analyse traffic in an existing communication network is characteristic for many types of network research studies. Whether the goal is to improve the functionality of the network, to perform user level behavioral studies or simply to be aware of the operational aspects of the network, traffic analysis will prove to be an essential part of the process. The primary purpose of continuous traffic analysis is usually to find out the usage levels on different parts of the network. It includes monitoring user behavior and observing how the network copes under the traffic loads at different times. By identifying bottle-neck links and links with low utilization levels the network operator can equalize the traffic patterns, for example by creating alternative paths, making the network run overall more efficiently. Traffic analysis also offers a way of identifying fault situations occuring somewhere in the network and most importantly observing how the faults impact the communication between the users.

Network simulation is a quick and economical way to analyse the performance of a network. It offers many benefits compared to analysing an existing physical network. First of all is the availability: with a simulator the analyst always has full and unrestricted access to the target network even if the network does not physically exist — a useful aspect when designing future networks. Second benefit is modifiability: the analyst can make any alterations to the network and not have to worry about the effect they have on the users. Third issue is speed: analysing a real network often requires establishing one or more monitoring points and also a data collection and analysis points. Not only does it take time to build the analysis infrastructure but it is also time consuming to collect and process the measurement data. Fourth and typically the most important reason is the cost: with a simulator there is no need for additional hardware or proprietary equipment. Usually one desktop computer and a free simulator software are sufficient. However, the unresolved issue with simulators has always been reliability and accuracy of the results: can they be trusted and how well does a simulation reflect the situation in the real network?

## 1.2 Objective

The objective of the thesis is to analyze the performance of an IP (Internet Protocol) network and to create a traffic model for use in simulating the network. A core network with the traffic entering and exiting the network in specific locations called PoPs (Point of Presence) was to be analyzed and modeled.

The task required for a measurement infrastructure to be designed and implemented. There were two design criteria: modifiability and cost. The target network has several PoPs connected to it and measuring the network comprehensively demands basing at least one monitoring point in each of them. Building the measurement infrastructure was to be a time consuming and tedious task so the end-result needed to be useful for many tasks to come with minimal changes. Using proprietary hardware would have been relatively simple but the solution would have been too expensive and robust. This thesis presents a low cost and highly modifiable packet monitoring system using standard off-the-shelf hardware and free packet capture and analysis software.

The traffic statistics gathered by the monitoring infrastructure was to be used for creating a traffic model covering all communication in the whole network. Eventually, the network will be recreated in a simulation environment and the model will be used for reproducing the traffic patterns in the simulation as accurately as possible. However, the simulations are not part of this thesis.

The performance of the network under varying traffic conditions needed to be evaluated. The need to recreate the network and its traffic in a simulated environment was set by the demand to be able to compare the performance of the simulated and the real network. Ultimately, the question to answer is: how well does a simulation reflect the situation in a real network? This thesis covers the part involving the real network.

## 1.3 Structure of the Thesis

The thesis is organized as follows: chapter 2 covers various network monitoring methods in principle and in practice as well as awareness of time. Chapter 3 takes a view on the structure of IP network traffic and the relevant parameters needed in understanding it. Chapter 4 presents the network under study and explains the methodology used in the analysis. Chapter 5 shows the hardware and software included in the measurement infrastructure. Chapter 6 presents the results, including the traffic model and network performance parameters. And finally, chapter 7 presents the conclusions of the thesis.

# Chapter 2

# Network Monitoring

## 2.1 Methods for Monitoring a Network

Analyzing a network requires first connecting to it and then gathering appropriate information. There are two methods of monitoring a network: passive and active. The choice of which one to use depends on the goals of the analysis.

### 2.1.1 Passive Monitoring

Passive monitoring means that the traffic is only listened to — no packets are sent to the network by the monitor. Passive monitoring is usually performed in two alternative ways: by using existing network devices to gather statistics or by introducing a separate monitoring equipment to the network. The former method relies on switches and routers to gather statistics of the traffic. The statistics are fetched from the devices for analysis, for example by using SNMP (Simple Network Management Protocol) [CFD90]. The latter method — using a separate equipment — is a more effective and customizable way of passive analysis. It can be used to store all the traffic traversing through the monitoring point or just collecting the statistics required for the analysis.

The advantage of passive monitoring is that it gives a comprehensive and unbiased view of the network. The analyst needs no prior knowledge of the network or its traffic before beginning the data collection. Accordingly, passive monitoring is the preferred choice when studying an unknown network. No major decisions need to be made about what kind of data is required for the analysis when installing the measurement equipment — it is usually sufficient to record full traces of the traffic. Post-processing the traces offers a variety of different ways to study the data giving a detailed insight into the essence of the network, for example the protocol distribution and the daily variations in link utilization. Also, it is often used for detecting possible error situations occurring somewhere in the network by sniffing error packets and unusual routing messages. An example of using a passive monitoring system to help identify congested or misconfigured network segments is presented in [AGJT03].

Passive monitoring is passive only in the sense that it does not generate traffic when observing the network. However, the trace files it generates are often quite large and transferring them from the monitoring points to the point where analysis is performed can cause a relatively high load on the underlying network. The problem can be solved by transferring the data by using a secondary network but in case of one's absence other methods of transmission might have to be considered. Another disadvantage of passive monitoring is the performance requirement it sets on the measurement equipment. Especially in the case of multiple high speed links the monitoring hardware must be capable of storing large amounts of data and processing it very fast.

Issues of security and privacy are often concerns on passive analysis. Recording all the traffic on a link allows a chance for eavesdropping. The opportunity for misuse does exist and is up to the network provider to decide the cases it allows someone to monitor the network. Excluding the misbehaving analysts from the scope of view, security and privacy can be maintained with two ways of action. Firstly, in order to improve security only the headers of the packets are stored in the trace files. From the traffic analysis point of view, only the packet header is interesting — the data field of the packet offers no extra information. Secondly, to improve privacy no information which can identify a user are published in the analysis results. Usually it is sufficient to hide the real IP addresses by some kind of address translation. Various methods for address hiding and translation are presented in [Peu02].

### 2.1.2 Active Monitoring

Active monitoring relies on probing the network for discovering its characteristics. Probes of various types can be sent to test and sample the properties of the network or some elements in it. An example of probing is Unix's *ping* command which uses "ICMP Echo" packets (Internet Control Message Protocol [Pos81a]) to measure the round-trip time between the source and the destination. Active monitoring is generally used to study the underlying network whereas passive monitoring is most often used for examining the traffic in it.

Active monitoring tools generate traffic to the network. Depending on the number and type of tests performed for the analysis the quantity of the extra traffic allowed should be considered in advance. A problematic question is: how much traffic are the monitoring tools allowed to generate in relation to the amount of normal network traffic. From the research point of view the optimal solution would be to keep a constant ratio between the amount of existing and generated traffic. Otherwise the data may become biased because of the effect the generated traffic has on the network. However, from the network provider's point of view the optimal solution would be to make most of the measurements during the slow hours and save all the available bandwidth on the busy hours for the users. Another contrast is that the more probes are used to perform a test the more accurate and reliable results are achieved. However, at some point the probes will generate so much traffic that it becomes disruptive for the normal operation of the network.

Probing does not require heavy performance from the monitoring equipment. The tests used for gathering data often consist of quite simple procedures, for example sending a few ICMP packets or opening and closing some TCP connections. Meaningful analysis requires planning several types of individual tests and deciding where and when to launch them to get the most benefit of them. All this has to be done before starting the — often very time-consuming — data gathering. As a result, designing an active monitoring system generally requires more effort than setting up passive monitoring points. It is easy to miss something in the design phase and consequently lack important data which would be useful in the analysis. Launching new tests independently afterwards to gain supplementary data may prove fruitless because combining results from several tests usually means that the tests have to have been performed at the same time. On the other hand if the tests are well designed the measurement data is often quite compact and easy to interpret making the post-processing phase quick and efficient.

Security and privacy are easier to maintain than it is with passive sniffers. Normal traffic in the network is not recorded and the communication between hosts is not observed leaving the users outside the scope of interest. The tests are only concerned about the operation of the network support functionalities — such as DNS, DHCP and SIP — and possibly the most used services — such as SMTP and HTTP. Section 3.1 provides more information about the protocols and services mentioned here. Most network providers use active monitoring tools to check the operation of their systems and services. Also the fulfillment of QoS requirements (Quality of Service) stated in the customer's SLA (Service Level Agreement) can be checked by using end-to-end packet probing.

### 2.1.3 Hybrid Solution

A hybrid of passive and active monitoring methods is sometimes used to combine the benefits of the two. If a measurement can be performed passively or actively the passive method is usually the preferred choice. However, if not enough traffic streams are present the active method may alternatively be used. Available bandwidth is an example of a network property which can be determined actively or passively as described in [ZL03]. The paper presents and compares various techniques for either actively or passively discovering the available bandwidth on a communication path.

Another way to combine active and passive monitoring methods is to use them for studying different properties of the network. An excellent paper to demonstrate how it can be achieved successfully is [LB03]. It presents a measurement system for analyzing flows using a modified version of NeTraMet ([Bro01]) which sends probe packets every $N^{\mathrm{th}}$ packet or second. This effectively divides the flow into monitoring blocks consisting of $N$ packets or alternatively of a time period of $N$ seconds. The incoming traffic is analyzed at the receiver end. The number of received packets and bytes in a flow are counted for calculating the throughput. The probe packets are used for measuring one-way delay and delay variation by using NTP synchronized timestamps taken at the sender and receiver hosts. The monitoring blocks are used for calculating packet loss ratio by equation:

$$PLR_i = 1 - \frac{N_{\text{received,i}}}{N_{\text{sent,i}}}, \qquad 0 \leq N_{\text{received,i}} \leq N_{\text{sent,i}} \qquad \forall i \in \mathbf{N} \tag{2.1}$$

where $i$ is the sequence number of the monitoring block. To limit the effect of disordering of packets caused by routing changes more than one monitoring block can be included in the loss ratio calculations. Equation 2.1 is then modified as:

$$PLR_{\text{i,j}} = 1 - \frac{\sum_{k=i}^{i+j} N_{\text{received,k}}}{\sum_{k=i}^{i+j} N_{\text{sent,k}}}, \qquad \forall i, j \in \mathbf{N}, \tag{2.2}$$

where $i$ is the sequence number of the first monitoring block to be included and $j$ is the total number of monitoring blocks in the PLR calculation. It should be noted that the inequality in equation 2.1 is no longer valid in equation 2.2 because of possible packet reordering. The drawback of taking sums of the monitoring blocks is that the resolution is no longer 1 monitoring block but $j$ blocks.

## 2.2 Connecting to a Network

Performing efficient traffic analysis requires a physical measurement equipment to be introduced directly to the point of measurement. There are two common ways of connecting a measurement computer to the network: by using link taps to duplicate all traffic on a link or by mirroring a port on a switch or router. Both have their benefits and areas of applicability.

### 2.2.1 Network Tapping

A network tap is used to split a link so that the traffic is duplicated to a secondary destination as illustrated in figure 2.1. Tapping is an efficient method of passively monitoring a link. Nearly all kinds of networks can be monitored with taps and they can be used for splitting both optical and copper links. Optical splitting means dividing the light beam into two separate output paths by a given ratio, for example 50:50. The intensity of the light wave is diminished by the same ratio. The attenuation of the beam lowers the maximum length of the fiber and it has to be taken into consideration when implementing tapping. Copper taps, on the other hand, require an external power source to function. If for some reason the power to the splitter is lost it is unable to perform splitting. However, most copper taps have a safety mechanism which short circuits the splitted wire allowing the traffic to still flow through in case of a power failure. Also, using an external power source means that the maximum length of the wire is not shortened by the tap. On the contrary, they can even be used as repeaters. In case of a link failure, network taps should also be able to mediate the link state knowledge to the other side of the splitted link. This was found to be the case for the taps used in our measurements, however, manufacturer specific differences may exist. More information about copper and fiber taps can be found in [Fis02].

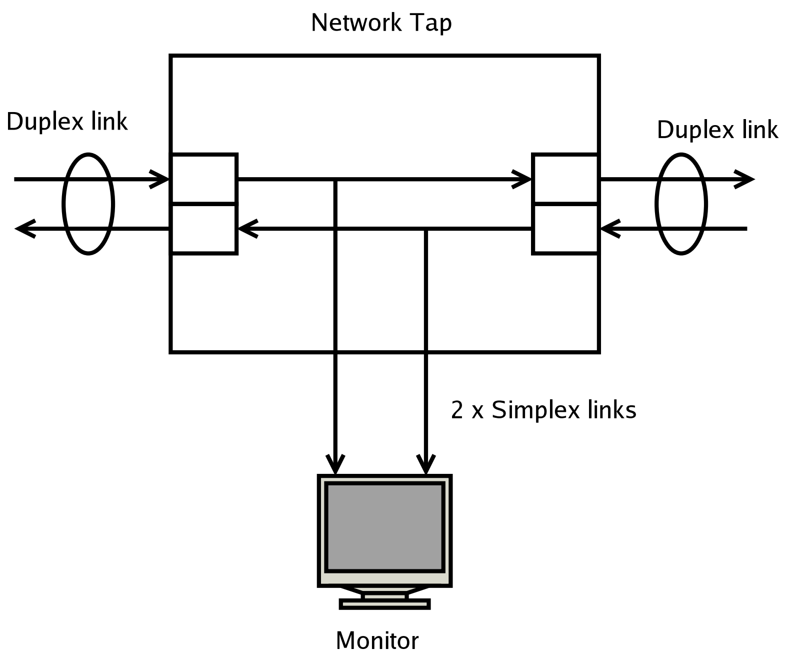

Figure 2.1: Network tapping on a duplex link

The simple design of a network tap brings many benefits. The tap forwards the carrier signal — the light beam or the electrical signal — to the analyzer. The forwarding does not require any signal processing or higher level modifications, meaning it can be done very quickly and reliably; all traffic traversing through the link is repeated perfectly and without any extra delay. Network taps prevent direct access to the monitoring system, thus securing it from potential hackers.

A negative aspect in using taps is that the traffic on the monitored link is momentarily cut off when inserting the tap. However, for most networks such a short break is not a critical issue. A bigger concern comes from the fact that one tap can split only one link at a time and both traffic directions in a link produce a separate output. The result is that the number of taps needed is equal to the number of monitored links and twice as many interfaces are needed in the monitoring system as there are links to be monitored. Since network taps are relatively expensive despite their simple design the price of the entire monitoring system becomes easily quite high, especially when monitoring optical gigagit Ethernet links.

### 2.2.2 Port Mirroring

Port mirroring is a quick and simple method of duplicating traffic to the monitor. One or more ports in a packet switch is mirrored to another port connected to the monitor as illustrated in figure 2.2. In addition to performing normal packet forwarding the switch duplicates all packets seen in the mirrored port or ports to the monitor port.

The main benefit of using port mirroring is that it is easy and cheap to use if the switch supports it. Mirroring can be configured without causing any disturbance to the ongoing traffic. It is also possible to monitor traffic aggregates, for example VLANs or port groups.

The biggest handicap of port mirroring is that its performance is entirely dependent on the quality of the switch. Many switches — especially the cheaper models — offer few mirroring options making efficient monitoring impossible. For example, monitoring a duplex link requires the mirrored port to be forwarded to two monitor ports — one for each direction. In many switches this kind of a configuration is not supported and it results in packet drops when link utilization is higher than 50 per cent. Packet drops can also occur when more than one ports are mirrored to a single monitor port. In a worst case scenario, buffer overflows in the monitor port can affect the traffic of the mirrored ports as well. Packet

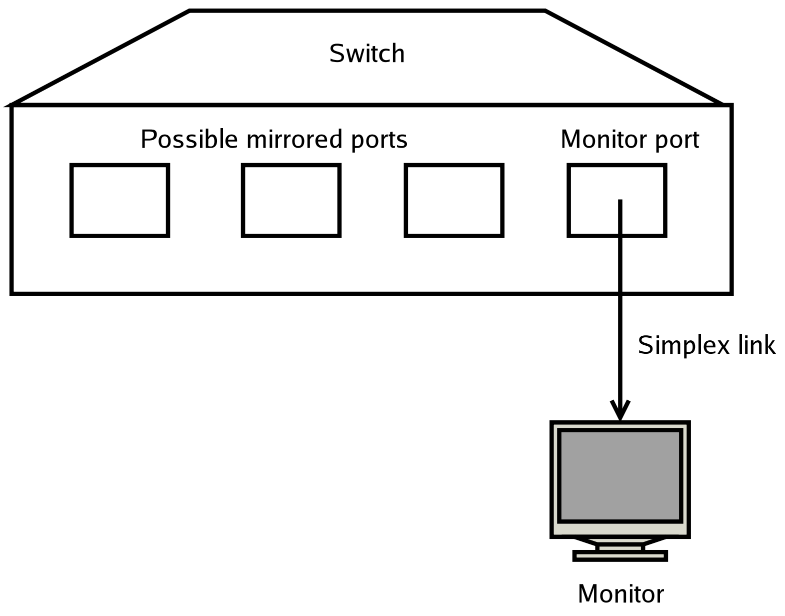

Figure 2.2: Port mirroring on a switch

buffering in the monitor port also adds delay in the timestamp of the packet and makes time-sensitive measurements inaccurate. Another defect of port mirroring is that — again depending on the switch — certain packets, for example some error packets, are not mirrored at all.

### 2.2.3 Areas of Applicability

Network tapping is the recommended choice for situations where high performance and accuracy are required. Measuring time-sensitive applications and protocols requires low latency from the monitor system and taps are well suited for that purpose. It is also the preferred choice when creating IDSs (Intrusion Detection System) because of good isolation from the monitored network, as described in [Ein02].

Port mirroring is the preferred choice when precision is not required. It is sufficient for cases where only a general idea of the operation of the network is needed. Accuracy diminishes when the utilization level of the monitor link increases since filling packet buffers introduce more latency to the system. Therefore port mirroring can be used with reasonably high accuracy when the bandwidth of the monitor link can be substantially overprovisioned in relation to the cumulative bandwidth of the monitored links, i.e. if the load in the monitor link remains low enough.

Additional information on tapping and mirroring can be found at [Fin02]

## 2.3 Network Time Protocol (NTP)

Having an accurate knowledge of time is essential in modern communication networks. It makes managing and controlling the network more efficient and reliable. Network analysis includes various distributed tasks where parties are scattered throughout the network and need to be launched independently on given times. To make post-processing of the data produced by these tasks meaningful clock offsets between the parties need to be withing seconds of each other. More strict offset requirements are introduced when studying traffic at a more detailed level, for example measuring one-way packet delays. Offsets between the clocks need to be well below a millisecond in these cases.

NTP (Network Time Protocol) is used to synchronize clocks over a network. First release of NTP was introduced in 1985 and the latest version — v3 — was published in 1992 [Mil92]. The accuracy of an NTP adjusted clock depends on the quality of the local oscillator and of the delay properties of the underlying network used to relay NTP messages. Standard PCs contain quite low quality oscillators but still accuracy within 10 millisecond level in a WAN network and 1 millisecond level in a LAN network is routinely achieved [Cis03].

### 2.3.1 Network Structure

An NTP network consists of a set of distributed servers and clients. The network is hierarchical and each host is given a place in the hierarchy by assigning a stratum number between 1 and 16 — the lower the stratum number of a host, the higher up in the hierarchy it resides. Stratum 1 hosts are primary servers which use atomic clocks as a reference signal. A client connects with a server and after clock synchronization receives a stratum number which is one lower than that of the server it is attached to. The client can also act as a server and let other clients synchronize to it, and so on. The result is a hierarchical structure presented in figure 2.3.

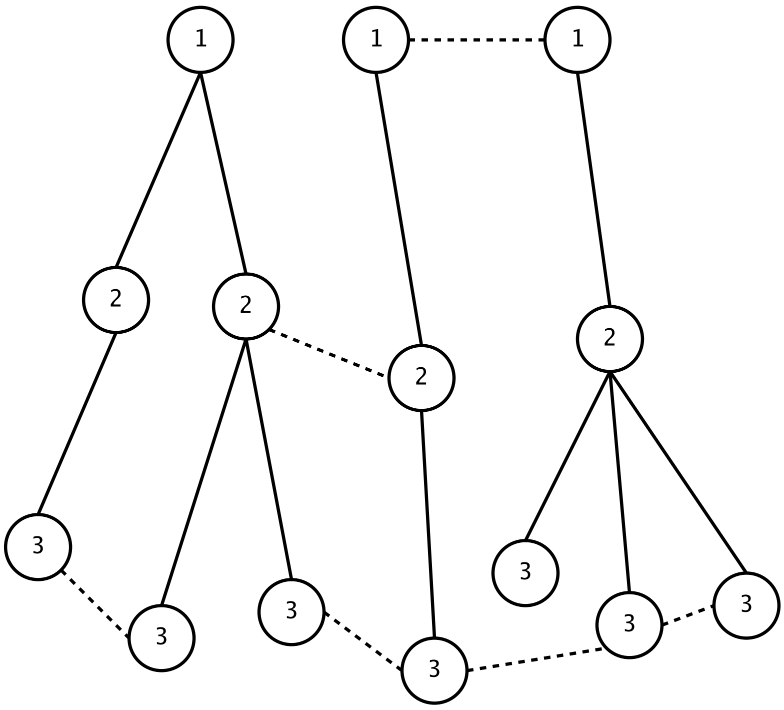

Figure 2.3: Structure of a hierarchical NTP network

The solid lines in figure 2.3 represent the traditional server/client relationship between NTP hosts. The dotted lines represent a peer-to-peer relationship between hosts with equal stratum. Peer relationships are used to create groups of equal hosts to provide redundancy and crosschecking. Because of its scalability hierarchical NTP network structure is best suited for large heterogeneous communication network such as the Internet. A survey of the status of NTP in Internet in 1999 can be found in [Min99]. Among other things it describes the problem of so called bad clocks in Internet. Bad clocks are hosts which configure their stratum value falsely. At worst, a host with a poor quality clock announces itself as a stratum 1 clock.

Besides a hierarchical structure, NTP can be configured as a flat peer structure or a star structure as illustrated in figure 2.4. Flat peer structure is not widely used because of its low scalability and accuracy. However, it is robust and not vulnerable to defects of key nodes. Star structure is a simplification of the hierarchical structure with only two stratum levels. The network only has one or a few servers which clients use to synchronize to. It is best suited for relatively small intranets with a requirement of self-sufficiency. The biggest problem is that the system is very vulnerable to attacks or system faults in the servers. The problem can be alleviated by assigning backup servers. Hierarchical network is autoconfigurable in the sense that clients can circumvent a faulty server by connecting directly to the server one step up in the hierarchy.

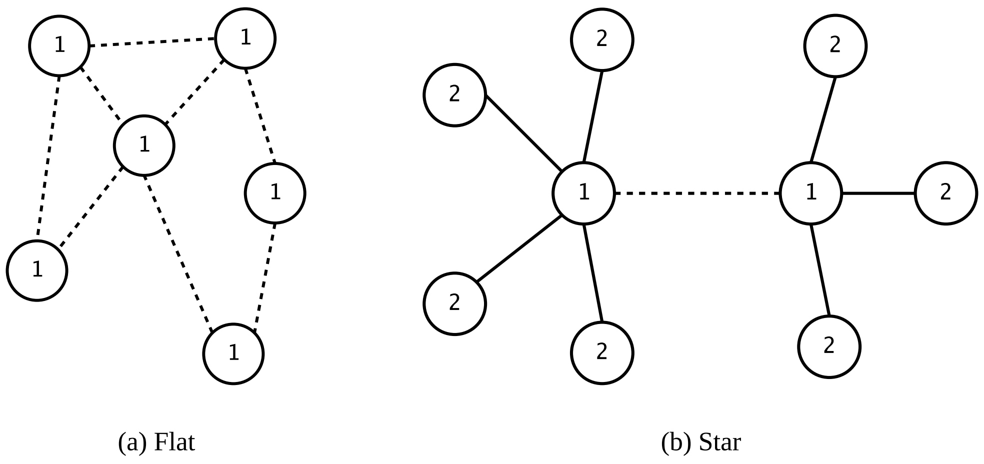

Figure 2.4: Alternative NTP network structures

### 2.3.2 NTP Internals

The accuracy of timekeeping on an NTP client depends mainly on the quality of the local clock oscillator and the packet transfer delay to the server. Accuracy is measured by the offset between local clock and the real time. Real time is a somewhat delicate concept and the notion of relative offset is sometimes used when referring to the difference between local clock and any given reference clock. In any common case the time offset between two clocks change in time because of a slight variation in frequency between them. Variation of frequency is the derivative of offset with respect to time and is commonly called clock skew. Also the frequencies of clocks tend to change in time. The result is a variation in the clock skew and is commonly called clock drift. In other words, it is the second derivative of offset with respect to time. Clock's ability to maintain constant frequency is called stability. The terms used here are adopted from [Mil92]. The terms used and recommended by ITU-T (in [Sec96]) are jitter and wander to descibe the short- and long-term “variations of the significant instants of a timing signal from their ideal positions in time” respectively. The terms skew and jitter as well as drift and wander are interchangeable.

NTP operates by discovering the offset and the skew between local and remote oscillators. The objective is to minimize offset to the reference clock by adjusting the rate at which local clock advances. The clock is either slowed down or sped up until offset is near zero.

Finding the same clock pace as that of the reference clocks is the next task. Initially the local clock advances at the rate dictated by the local oscillator:

$$t_1(t) = t_0 + \frac{f_{\mathrm{osc}}}{f(t)}t, \qquad t \in \mathbf{R}_+ \tag{2.3}$$

where $f_{\mathrm{osc}}$ is the set frequency and $f(t)$ is the actual frequency of the oscillator. The term $t$ refers to the absolute or atomic time and $t_0$ and $t_1$ are moments in time as seen by the local oscillator. Frequency of an ideal oscillator is $f(t) = f_{\mathrm{osc}}, \forall t \in \mathbf{R}$. However, real oscillators introduce instability as defined by $f(t) = f_{\mathrm{osc}} + f_{\mathrm{skew}}(t)$ which combined with equation 2.3 yields

$$t_1(t) = t_0 + \frac{f_{\mathrm{osc}}}{f_{\mathrm{osc}} + f_{\mathrm{skew}}(t_0)}t, \qquad t \in \mathbf{R}_+ \tag{2.4}$$

The objective of NTP is to evaluate skew ($f_{\mathrm{skew}}(t)$) and accordingly adjust the local clock. Skew has to be evaluated frequently since it is not a constant but a function of time. The evaluation interval can be adjusted but the default is to check the reference time every 64 seconds. If connection to the reference clock or server is lost the accuracy of the clock will gradually deteriorate because of inherent oscillator instability.

### 2.3.3 Improving Accuracy

Accuracy of the clock relative to the reference clock is in most cases the primary concern. A survey of the subject is presented in [Smo03]. It demonstrates that offset between two NTP synchronized clocks is most of the times only a fraction of a millisecond when the network delay is of the order of tens of milliseconds. Occasionally there are time deviances where offset goes as high as one millisecond. NTP can, however, detect the deviances accurately withing a fraction of a millisecond and this information can be used to compensate the offset error — instead of using the local time directly the offset reported by NTP is first subtracted from it. This is because NTP does not adjust the local clock immediately upon time discovery but by slowly sliding the time towards the right direction. Consequently, the clock error can occasionally be quite substantial, even though the NTP client would have a better knowledge of the time. With this procedure the total error of a timestamp remains a fraction of a millisecond.

### 2.3.4 Adding Redundancy

The timekeeping of the system is quite vulnerable because of only one existing server. In case of a defect or an attack against the server the clients would gradually lose their synchronization and the end-to-end delay measurements would be practically useless. This is not a major defect if the NTP network used is relatively small and the measurement period does not have to be continuous. In the unlikely situation where NTP fails, the data that is collected after the failure cannot be used. However, NTP and the synchronization between all the clocks in the network can presumably be restored in a matter of hours. Also, the benefits of using a lightweight NTP network structure will outweigh the risk of a key server failure. The benefits are mainly the ease of configuration and the accuracy of the error estimation in the clock offsets.

Reliability can be improved by adding more NTP servers. There are two options to add a second server to the network: either by adding a backup server on the same hierarchy level as the original server or by adding another hierarchy level and configuring the original server to synchronize itself with the new server. These methods are illustrated in figure 2.5.

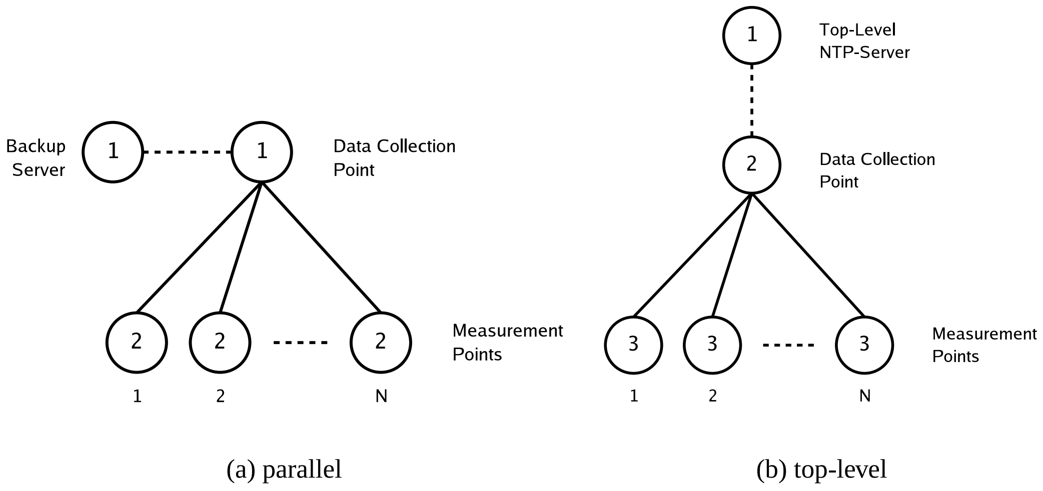

Figure 2.5: Creating redundancy into the NTP network. $N$ is the amount of monitoring points.

### 2.3.5 Measurement Errors

#### Clock Relative Measurement Error

Measurements concerning time always includes some degree of error resulting from non-ideal clocks. There are typically two scenarios where this problem becomes apparent: either we are using one clock to measure the time duration between two events or we are using two clocks to measure the same event. The time error in the former results from clock wander caused by a non-ideal oscillator and the latter is caused by the offset between the two clocks. There are also scenarios where a combination of the two errors becomes apparent, for example when measuring the time it takes for a packet to travel from host to host in a network where the time stamps need to be taken with two clocks at different times. However, in these cases the offset between the clocks is the predominant error and it alone should be included in the error analysis for the sake of simplicity.

When using two different clocks to measure a time duration it is important to note with whom the clocks are synchronized when estimating the error. If the clocks are synchronized with each other we are dealing with the relative clock offset. This means that the total time error estimate is the same as the offset error of a single clock. However, if the two clocks are synchronized with a third clock the total time error becomes more complicated. This is the situation in most measurement systems, for example in the star shaped NTP network in figure 2.4 where the monitoring points would be the client nodes in the NTP hierarchy. The third clock can be understood as the real time clock and both client clocks have a offset error relative to it. A combination of two error sources — one for each clock — have to be included in the total time error. In this case, the cumulative error is twice the single clock offset error.

### Percentual Measurement Error

Percentual time error means the size of the measurement error compared to the magnitude of the measurement values. With a single sample the percentual offset can be stated as:

$$\sigma_{\mathrm{rel}} = \frac{\sigma_{\mathrm{abs}}}{\delta_{\mathrm{sample}}}, \tag{2.5}$$

where $\sigma_{\mathrm{abs}}$ is the measurement time error and $\delta_{\mathrm{sample}}$ is the measured value of the sample. The percentual time error is often a more informative parameter than the absolute error and it dictates the level of accuracy needed by the clocks of the system. For example, if a sample measurement lasts for 100 seconds a clock accuracy of 1 second still yields an error of 1 per cent. If a measurement lasts for 10 milliseconds the clock accuracy must be better than 0.1 milliseconds to reach an error of 1 per cent.

# Chapter 3

# IP Network Traffic

## 3.1 The 5-layer TCP/IP Model

The traffic in modern IP networks is diverse. All kinds of applications use the shared networking environment to transmit their data. The traffic produced by these applications have highly varying characteristics and the need to examine the protocols separately becomes apparent. Various models have been developed to divide the communication protocols into a distinct hierarchical structure. The most used in IP network studies are the 7-layered OSI model ([Zim80]) and the 5-layered TCP/IP model ([WD01]). In this thesis, the 5-layered TCP/IP model — illustrated in figure 3.1 — is used. Furthermore, we will limit our scope of view to the three top layers: network, transport and application protocols. The first and second layer in the model contain protocols for maintaining physical and link-level connectivity and ensuring bit-level integrity in data transfer. They are not interesting from the operational point of view of the network when examining packet traffic.

### 3.1.1 Network Protocols

Network protocols reside in the third layer of the TCP/IP model and they are generally the lowest-level protocol set which is of interest when studying network traffic. The function of network protocols is to establish connectivity between all routers and hosts in the network and allow communication between them, even

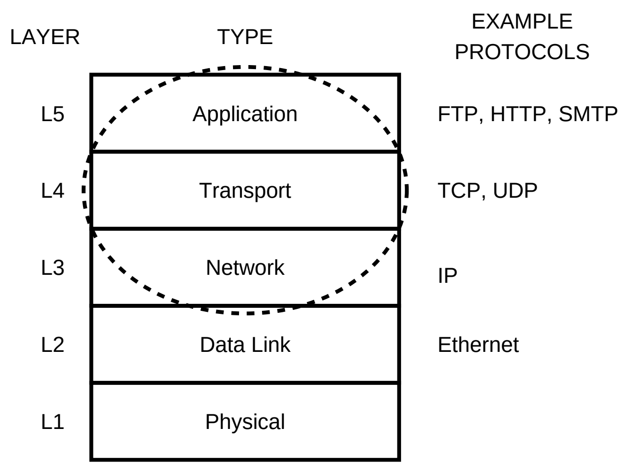

Figure 3.1: 5-layer TCP/IP model with some examples of protocols residing in each layer. The focus of our study is in the top three layers.

though not directly connected to each other. IP protocol resides in this layer in the TCP/IP model.

In Internet, IP protocol — currently mostly IPv4 [Pos81b] — is used to carry packets between end-hosts on a hop-by-hop basis. It offers a connectionless, packet switched transfer medium where reliability of the transfer is not measured or guaranteed in any way. The benefit is a highly scalable transmission environment where many of the functionalities are left up to the higher level protocols — which run on top of IP — to decide and implement. The connectivity is established with routing protocols, such as OSPF ([Moy98]), RIP ([Mal98]) and BGP ([RL95]). Managing the IP address allocation to the hosts can be automated by configuring a DHCP environment ([Dro97]) to the network. Error reporting and control message delivery is handled with ICMP ([Pos81a]) — nowadays an integral part of IP networks. ICMP runs on top of IP and can be considered to belong either to the network layer or to the transport layer.

### 3.1.2 Transport Protocols

Transport protocols run on top of IP and manage the higher level functionalities of the communication. Nowadays, the two absolutely most used transport protocols are TCP [Pos81c] and UDP [Pos80].

TCP is used on data transfers which require a reliable end-to-end packet delivery. It creates a logical connection between two end-hosts by allocating them TCP ports within the hosts. The quadruple of source and destination IPs and ports uniquely identifies a connection within the TCP protocol. The bidirectional connection is established with a three-way handshake which includes exchanging SYN, SYN+ACK and ACK packets respectively. The closing is done independently on both directions by both parties sending a FIN packet which is reciprocated with an ACK packet. An alternative way of closing the connection is to send an RST packet. Some operating systems also close the connection if it has been idle for a certain period of time. TCP has a dynamic rate control mechanism which is applied separately for each connection to take maximum advantage of the available bandwidth on the transmission path. It also uses acknowledgement packets and retransmissions to guarantee successful transmission of all the data.

UDP, on the other hand, is a much more simple protocol. It also specifies the quadruple of source and destination IPs and ports to make the same kind of definition of a connection as there is in TCP. However, there are no acknowledgements to guarantee end-to-end transmission integrity and the connection is not established or closed in any way as is the case with TCP protocol. Also, there is no rate control mechanism to adjust the transmission rate and the application using UDP must decide the transmission parameters to be used for sending a given set of data, namely datagram size and send rate.

### 3.1.3 Application Protocols

The set of network and transport layer protocols used in modern IP networks is relatively small and static. Network layer monitoring generally requires only the examination of IP, ARP and ICMP protocols to cover an overwhelming percentage of the traffic. On the transport layer, monitoring TCP and UDP protocols is sufficient. However, the set of application protocols which covers most of the network traffic is very large and dynamic. The traffic proportion between the application protocols is a rapidly changing attribute and is highly dependent on the customs and trends of the network users. For example, HTTP has been the predominant protocol in the Internet for many years but a quickly growing set of peer-to-peer applications are beginning to take more and more of the traffic share.

It is difficult to distinguish different application protocols when observing the packet traffic. Different applications produce various header types and it is not trivial to identify the application by simply observing the packet headers. For efficiency and simplicity, it is desirable to construct the analyzer so that only the packet header up to the transport layer are inspected. Port numbers of TCP and UDP protocols are generally used to identify the application. It is not a 100 % accurate method but will often be sufficient. The list provided by IANA (Internet Assigned Numbers Authority, [Aut04]) is used to map the port numbers to applications. However, each TCP and UDP packet has both source and destination ports and choice of which one to use is not trivial. The following logic is used here:

1. If one of the source and destination port numbers is found in the the list, use it to determine the application
2. If both port numbers are found in the list, use the smaller one to determine the application
3. If neither port number is found, set the application to be *None*

The mapping of application protocols to port number based on the logic described above is not guaranteed accurate. The protocols reserved or assigned are in the range of 0–1023. Above this range, ports may be reserved for known applications. However, they are not strictly maintained and another application may well assign dynamically a port number which coincides with a port of a known application. In this case, the traffic may be falsely interpreted of belonging it.

## 3.2 Self-Similarity

Self-similarity refers to an object or a phenomenon appearing roughly the same, looking at all scales. Fractals are a typical example of self-similar objects. They exhibit similar shapes and patterns when examined from close up or far away.

Same kind of behavior can be seen in certain time-related phenomena when observed at different timescales. Self-similarity has been discovered in many naturally occurring processes. It has also been proved to exist in network traffic ([LTWW93]) which has had a major impact in understanding how communication networks operate, and in particular, how they should be modeled.

### 3.2.1 Definition of Self-Similarity

The definitions and properties presented in this paragraph are for the most part taken from [LTWW93].

Let $\chi = (\chi_t : T = 0, 1, 2, \ldots)$ be a covariance stationary stochastic process with mean $\mu$, variance $\sigma^2$ and autocorrelation function $r(k), k \geq 0$. We assume that $\chi$ has an autocorrelation function of the form:

$$r(k) \sim k^{-\beta} L(t), \qquad k \to \infty, \tag{3.1}$$

where $0 < \beta < 1$ and $L$ is slowly varying at infinity and can be, for simplicity, assumed to be asymptotically constant.

Let us divide the original series $\chi$ to non-overlapping blocks of size $m$ and calculate the mean value of each block, so that block $k$ is given by:

$$\chi_k^{(m)} = \frac{1}{m}(\chi_{km-m+1} + \cdots + \chi_{km}), \qquad k \geq 1 \tag{3.2}$$

The process $\chi$ is called exactly second-order self-similar if the following equations apply

$$var(\chi^{(m)}) = \sigma^2 m^{-\beta}, \qquad \forall m = 1, 2, 3, \ldots \tag{3.3}$$

$$r^{(m)}(k) = r(k), \qquad k \geq 0, \quad \forall m = 1, 2, 3, \ldots \tag{3.4}$$

A more loose definition is often required in data analysis for practical reasons. Process $\chi$ is asymptotically second-order self-similar if the following equations apply:

$$\mathrm{var}(\chi^{(m)}) = cm^{-\beta}, \qquad m \to \infty, c = \text{constant} \tag{3.5}$$

$$r^{(m)}(k) = r(k), \qquad k \geq 0, \quad m \rightarrow \infty \tag{3.6}$$

The degree of self-similarity in a process is referred to as the *Hurst parameter* and is calculated by

$$H = 1 - \frac{\beta}{2} \tag{3.7}$$

**Heavy-tailed Distributions**

Distribution $X$ is said to be heavy-tailed if the following condition is met:

$$P[X > x] \sim x^{-\alpha}, \qquad \text{as } x \rightarrow \infty, \quad 0 < \alpha < 2 \tag{3.8}$$

A heavy-tailed distribution has infinite variance and if $\alpha < 1$ it also has infinite mean. When using heavy-tailed distributions the probability of generating very large values becomes non-negligible. It can have a negative effect in simulations and to counter it the distribution is often cut at some point $x_0$, meaning that values greater than $x_0$ are not produced. Consequently, the probability density function of the distribution has to be a scaled so that the the cumulative distribution reaches value $1$ at $x_0$.

A well known and simple heavy-tailed distribution is the *Pareto distribution* with probability density function given by equation:

$$\mathrm{PDF}_{\mathrm{Pareto}} = P(x) = ab^a x^{-a-1}, \qquad a, b > 0, \quad x > b \tag{3.9}$$

and cumulative distribution function given by equation:

$$\mathrm{CDF}_{\mathrm{Pareto}} = P(X \leq x) = 1 - \frac{b}{x}^{a}, \qquad a, b > 0, \quad x > b \tag{3.10}$$

Parameter $b$ indicates the lower limit for the random numbers produced by a Pareto random number generator.

### Other Distributions

In addition to heavy-tailed Pareto distribution, also other distributions are to be fitted to the data. It is difficult to say in advance which distributions can best be fitted to describe the data. Also, there are several parameters in the data which need to be modeled formally and different distributions may need to be used for different parameters. Accuracy of the estimation is evaluated by calculating root mean square error between the fitted distribution and the data.

Other distributions generally used for modeling statistical phenomena are exponential, Weibull and gamma distributions. Also, linear combination of two distributions can be used for getting the benefits of both. Absolute value of the distribution can be taken to guarantee only non-negative values. The following presents the probability density and cumulative distribution functions of the three distributions.

#### Exponential distribution

$$\mathrm{PDF}_{\mathrm{Exp}} = P_{(\lambda)}(x) = \lambda e^{-\lambda x}, \qquad \lambda > 0 \tag{3.11}$$

$$\mathrm{CDF}_{\mathrm{Exp}} = P_{(\lambda)}(X \leq x) = 1 - e^{-\lambda x}, \qquad \lambda > 0 \tag{3.12}$$

#### Gamma distribution

$$\mathrm{PDF}_{\mathrm{Gamma}} = P_{(\alpha,\beta)}(x) = \frac{1}{\Gamma(\alpha,0)\beta^{\alpha}} x^{\alpha-1} e^{-\frac{x}{\beta}}, \qquad \alpha, \beta > 0 \tag{3.13}$$

where $\Gamma(\alpha, 0)$ is a special case of Euler's gamma function:

$$\Gamma(\alpha, x) = \int_{x}^{\infty} z^{\alpha-1} e^{-z} \mathrm{dz}, \qquad \alpha > 0 \tag{3.14}$$

$$\mathrm{CDF}_{\mathrm{Gamma}} = P_{(\alpha,\beta)}(X \leq x) = 1 - \frac{\Gamma(\alpha, x)}{\Gamma(\alpha, 0)}, \qquad \alpha, \beta > 0 \tag{3.15}$$

**Weibull distribution**

$$\mathrm{PDF}_{\mathrm{Weibull}} = P_{(\alpha,\beta)}(x) = \alpha\beta^{-\alpha}x^{\alpha-1}e^{-(\frac{x}{\beta})^{\alpha}}, \qquad \alpha, \beta > 0 \tag{3.16}$$

$$\mathrm{CDF}_{\mathrm{Weibull}} = P_{(\alpha,\beta)}(X \leq x) = 1 - e^{-(\frac{x}{\beta})^{\alpha}}, \qquad \alpha, \beta > 0 \tag{3.17}$$

**Root Mean Square Error** Root mean square error for sample set $(x_i, y_i)$, where $i = 0 \ldots N-1$ is calculated with equation

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(y_i - P_s(x_i))^2}, \tag{3.18}$$

where $P_s$ is the scaled probability density function of the estimation distribution. The PDF is scaled by a factor so that the PDF fits the histogram of the data as closely as possible. Root mean square error can be used for estimating the goodness of functions fitted on a set of data.

### 3.2.2 Properties of Self-similar Processes

Self-similar processes occur commonly in nature but producing them artificially is not trivial. Superimposing heavy-tailed renewal processes produces an aggregate process which is self-similar, as shown in [WTSW97] and [CB96]. The property is highly applicable in network simulation models because traffic self-similarity can be achieved by setting one or more of the traffic generation distributions on individual sources to be heavy-tailed.

Leland, et al. showed in [LTWW93] that a self-similar process presents burstiness in the output set. If the self-similar process is generated by superimposing heavy-tailed renewal processes — as described above — the amount of burstiness increases with the number of aggregate processes involved. This behavior is completely different from superimposing flat-tailed processes — such as the Poisson — where increasing the number of aggregate processes smoothens the output and reduces burstiness. As a corollary, the aggregate output of many self-similar processes is nondegenerate, i.e. the structure and shape seen in the whole output set of the process is also seen when examining it in smaller and smaller time scales. This is in sharp contrast to Poisson process where the output of aggregates resembles pure noise at larger scales but present burstiness at smaller scales. The difference can be visibly seen in figure 3.2, which is taken from [LTWW93].

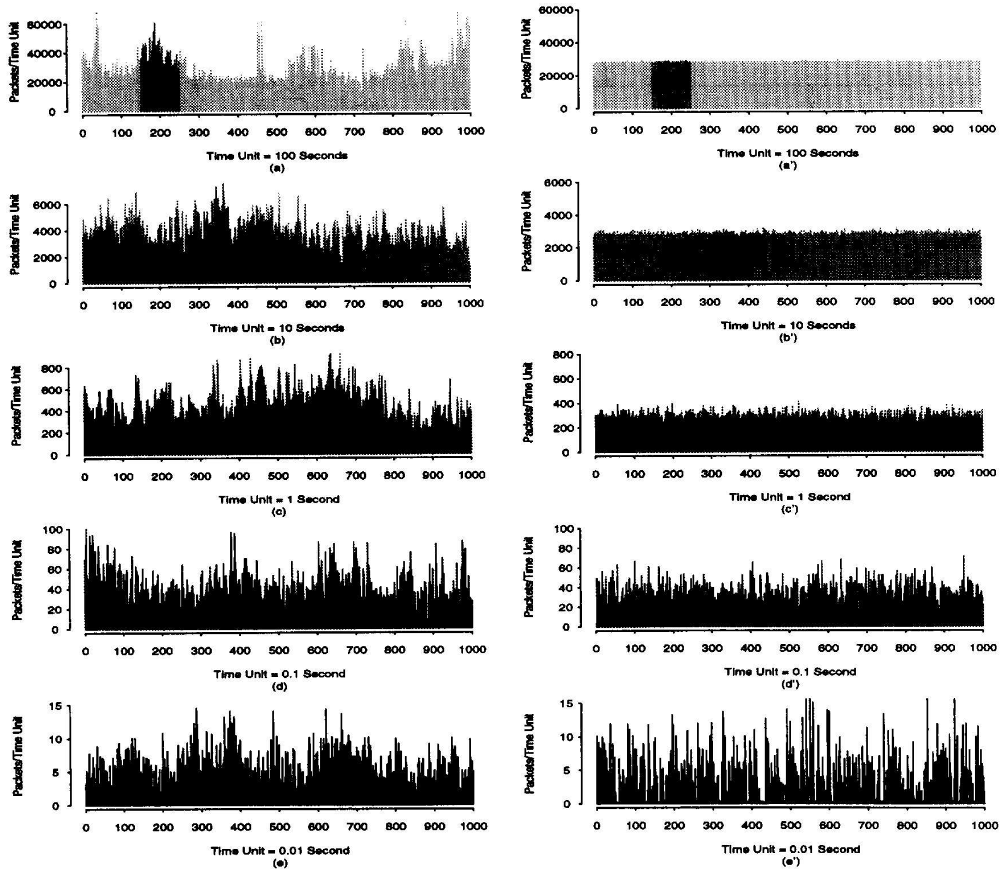

Figure 3.2: Self-similar (a)-(e) and Poisson traffic (a')-(e') seen in five different time scales. The figure is taken from [LTWW93]

### 3.2.3 Estimating Self-Similarity with Variance Time Plot

There are several different tests to measure self-similarity in a sample set: R/S analysis, periodogram based MLE (Maximum Likelihood Estimate), Whittle estimator and variance-time plot (VTP). All tests are presented and their accuracy measured with various traffic sample sets in [LTWW93]. The study shows that all tests are quite accurate; the figures from each test are relatively close to one another and they all fall inside the 95% confidence interval. We have chosen variance-time plot in our study because it is both intuitive and fast. In addition, its usefulness in further studies may be improved by the development of faster, real-time capable algorithms, as the one introduced in [HDTI00].

VTP is based on the fact that self-similar processes exhibit a slowly decaying variance i.e. the variance of the sample mean decreases more slowly than the reciprocal of the sample size $m$, as can be seen in equations 3.3 and 3.5. For each m-block $\chi_k^{(m)}$ (as defined in equation 3.2) variance $var(\chi_k^{(m)})$ is calculated and averaged over all $k$'s to get an estimate of the whole sample variance:

$$var'(\chi^{(m)}) = \frac{1}{k}\sum_{i=1}^{k} var(\chi_i^{(m)}), \tag{3.19}$$

where $k$ is large enough for the whole sample space $\chi$ to be covered. Variance in each m-block is calculated from the sample set using generic definition for variance (presented later in section 3.5.3).

A log-log plot of $m$ versus $var'\chi^{(m)}$ is crafted and a linear regression line is fitted to the sample points. The slope of the resulting straight line indicates $\beta$ as: $\beta = -$slope. The principle is illustrated in figure 3.3. Hurst parameter can be calculated by applying equation 3.7.

### 3.2.4 Self-Similarity of Web Traffic

Self-similarity of Ethernet traffic has been a subject of much research since its discovery in 1992 at the Bellcore network measurements by Leland et al., introduced in [LTWW93]. Their study had a major impact on how computer networks were to be modeled. Prior to their discovery, the arrival processes in computer networks were assumed to be similar to those seen in PSTN's (Public Switched Telephone Network). Their study showed that a traffic model using only Poisson process arrivals does not reflect the situation in real network accurately enough. It also had other major implications, for example, in how congestion control schemes should be designed. The paper focuses on examining LAN traffic (Local Area Network) but it also notes that LAN and WAN traffic (Wide Area Network) seem to be

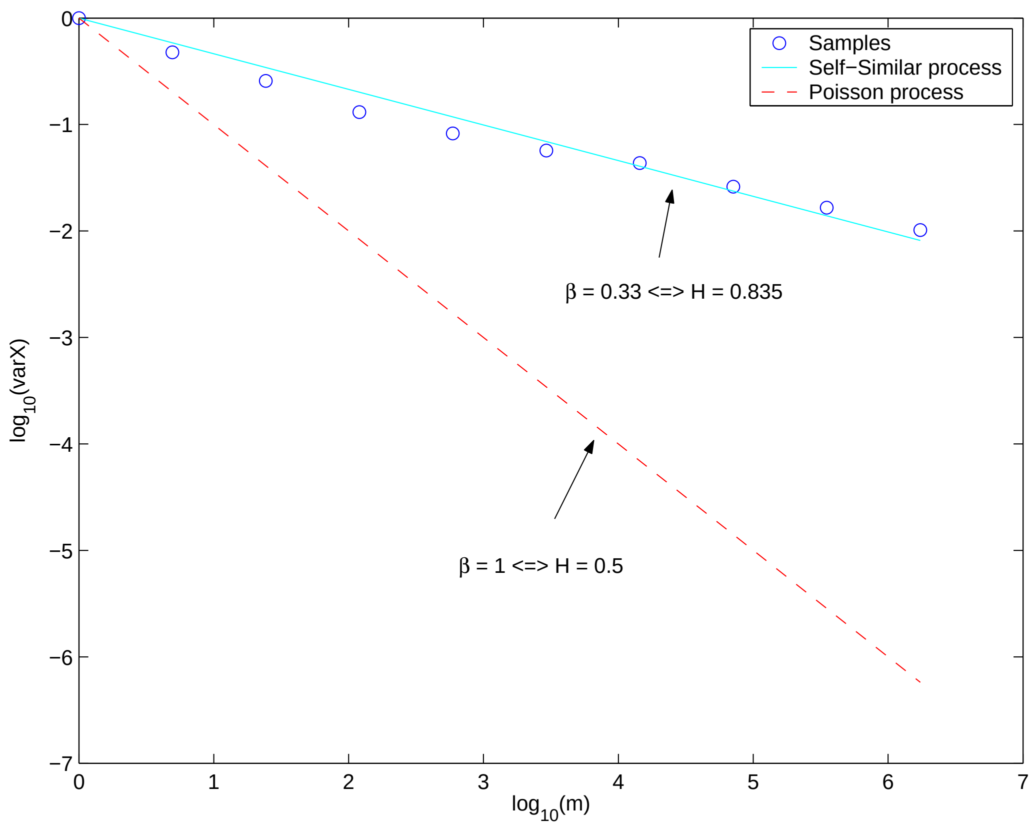

Figure 3.3: Variance time plot. Poisson process has $\beta = 1$ and Hurst parameter $H = 0.5$. A self-similar process has $0 < \beta < 1$ and Hurst parameter $0.5 < H < 1$.

equally self-similar.

Further studies have examined in more detail the reasons behind the self-similarity in web traffic and the scales at which it is most clearly visible. Mathematically speaking, a self-similar process should exhibit (statistical) similarity at all timescales but this is not the case in most physical systems. Instead, a process is self-similar only in some limited timescale. Web traffic can be said roughly to exhibit short-scale (ranging from milliseconds to tens of seconds) and long-scale (ranging from minutes to hours) self-similarity, as stated in [SV01]. The paper shows the short-scale self-similarity to be caused by TCP protocol's retransmission and congestion control mechanisms. Long-scale self-similarity has been studied more thoroughly in [CB96] and it is mainly the result of user activity and application level events. The paper shows the possible cause to be the heavy-tailed distribution of two attributes: available file size and user think time. However, it should be noted that the two studies examine only a part (TCP and HTTP) of the whole network traffic and in networks where only a small percentage of the traffic is composed of them the results may not be applicable. Also, a more recent study ([KMFB04]) has suggested that for sub-second time periods the degree of self-similarity is insignificantly small and can well be modeled by using only the Poisson process.

The degree of self-similarity is higher in the busy hours than at times of low network usage, as shown in [LTWW93]. This behavior is obvious since the number of connections in the busy hours is high and the amount of self-similarity rises with the number of aggregated connections, as described in section 3.2.2. Another characteristic property of network traffic is that the degree of self-similarity becomes higher as the packet loss ratio increases, as seen in [SV01] and [QC02]. The studies demonstrate that neglecting to use heavy-tailed distributions in the traffic generators may result in overly optimistic performance evaluation.

## 3.3 Traffic Flows

Analyzing traffic purely on per-packet basis hides an essential characteristic of today's IP networks: the communication tends to be connection oriented on user level. The actual packet forwarding is done independently on each packet — a fact that has been a major influence on making possible the rapid growth of the Internet. However, by inspecting the packet streams traversing through a router one can identify bursts of packets all flowing from the same source to the same destination. Also, the spacing between each packet is within a certain predefined time limit. In [JR86] these bursts were named packet trains. It was later realized that by using only the address fields of the packets it was not sufficient to model the most commonly used communication protocols and analyze the essence of a modern IP network.

From the end-host point of view a flow is easy to be defined as a transport layer (L4) transaction between two hosts: a TCP connection or a UDP burst. Hence, with TCP a flow would begin with a SYN packet and end with a FIN or RST packet. In the case of UDP, the transaction would be somewhat harder to define because UDP is purely a datagram delivery protocol which does not include a concept of a connection. A flow could still be defined as a transmission of a given data structure handed over to UDP by the controlling application. However, when observing the traffic in the core network or in an arbitrary router this definition is not valid. Routing changes in the network during the existence of a flow might cause some routers to see only a part of the flow; in the case of TCP, one router would see the SYN packet and another one the FIN packet. Also, the fact that a TCP connection may be ended in multiple ways — FIN, RST or timeout — might cause some confusion and inconsistency in the measurements. UDP flows were defined as a transaction that is assigned by the controlling application. Discovering the transaction in the middle of the communication path would be very difficult and would at least require inspecting the application protocol header in the packet. Also, each application has its own header structure and all of them would have to be implemented in the analyzer software. Dropped packets would also cause problems for the measurements of both TCP and UDP flows.

Instead of using the transport layer approach it is more meaningful to define flows with the principle used in defining packet trains — only the granularity for a connection needs to be rethinked. A flow can be defined using different granularities depending on the accuracy required for the analysis. The desired granularity determines which fields of the packet header are used to make a distinction between different flows. Various granularities are explained and their relevance to network analysis is discussed in [CBP95]. The granularity that is most often used is the 5-tuple of source and destination IP addresses and ports and the protocol, as illustrated in figure 3.4. This level of granularity is sufficient to identify a flow up to the transport layer, as explained in [Peu02]. In addition, a flow is determined by a timeout value indicating the maximum interval between consecutive packets in a flow. If the timeout value is exceeded between two packets with the same 5-tuple they are determined to be in different flows. Figure 3.5 illustrates this principle.

The directionality of the flow also has to be inspected. A burst of packets in one direction usually causes some kind of a response burst in the opposite direction. For example, a TCP connection consists of data packets flowing from a server to a client and acknowledgement packets from the client to the server. Accordingly, flows can be defined to be either unidirectional or bidirectional. Assigning the

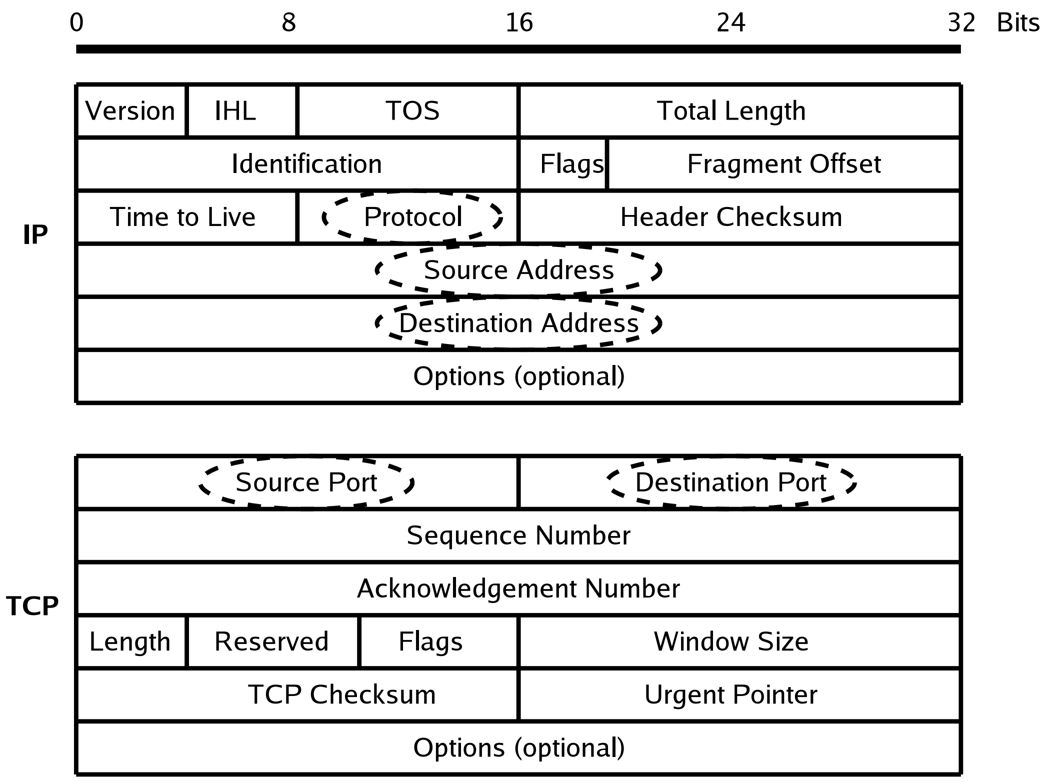

Figure 3.4: TCP/IP header and the 5-tuple flow to identify a TCP flow: source/destination IPs, source/destination ports and protocol

flows to be unidirectional is in most cases justified — the most prevalent reason being that unidirectional flows can always be combined afterwards to form bidirectional flows in a trace file. The motivation for using unidirectional flows is that asymmetric routing may cause discrepancies in bidirectional flow tracing because of packet streams using different paths on different directions. Also, the two unidirectional flows caused by a TCP connection have very different characteristics in terms of average packet size which makes the distinction of direction an important issue.

There is a strong correlation between the number of flows detected and the length of the timeout value, as can be seen in [CBP95]. The paper shows that the number of flows will rise very distinctly when the timeout value is decreased. It is concluded that a timeout value of 64 seconds is suitable for most purposes.

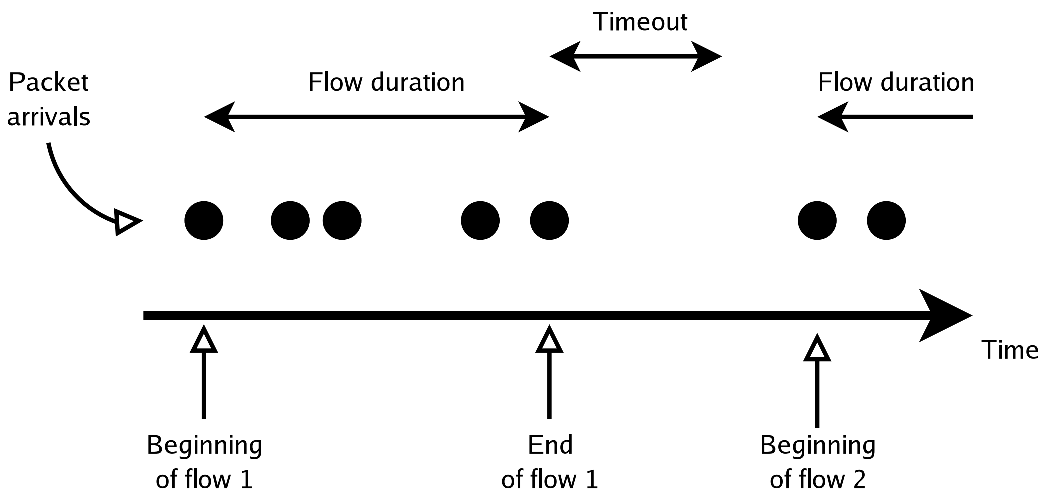

Figure 3.5: Traffic flow model. The duration of a flow is the difference between the timestamps of the last and first packet in a flow; the timeout value is not included there. This means that a flow consisting of a single packet has a duration of 0 seconds.

## 3.4 Application Models

An application model defines the hierarchy of various types of elements which compose the application and the processes which govern the way they are transmitted. The different element types are referred to as abstractions and their position in the hierarchy as abstraction level in this thesis. The hierarchy of elements forms a tree-like structure as illustrated in figure 3.6. The vertical axis tells the abstraction level of an element and the horizontal axis represents time.

To create the model, i.e. the tree structure in figure 3.6, the number of abstractions must first be determined, usually by using some a priori knowledge of the structure of the application. There generally is not a single correct model to represent a given application and the choice of how many abstraction levels are used affects greatly on the accuracy and simplicity of the model. The two key attributes must also be determined *for each abstraction level* in the model:

1. The number of child elements, i.e. the vertical direction

2. The rate of creating new elements, i.e. the horizontal direction

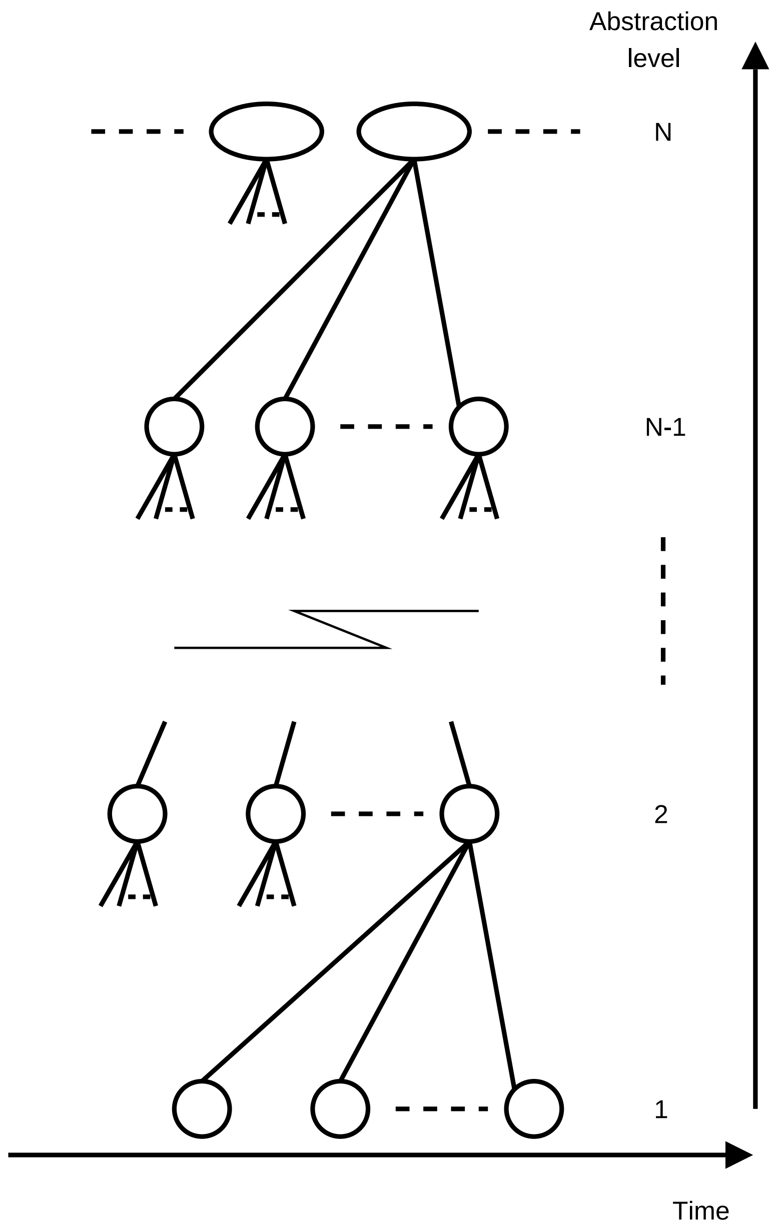

Figure 3.6: The tree-like hierarchy of application models

Both attributes are described by statistical distributions, such as exponential or Pareto. The idea is that an element in abstraction level $j$ determines the number of its child elements on level $j - 1$ by using a random number generator $X_j$ for each element separately. The subindex $j$ means that the same random number generator is used on the whole abstraction level. The interval between child element arrivals must also be drawn by a random number generator $Y_j$ for each child element separately. The fact that child elements of one parent are created independent of the child elements of another parent causes tree branches to be crossed with each other.

As an example, one can consider an FTP model using 3 abstraction levels, illustrated in figure 3.8. The first level represents sessions or user activity. Session arrival process follows exponential distribution. Each session has $N_1$ child elements representing files where $N_1$ is determined by using a geometrical distribution. Each file has one child element representing the number of bytes in the file. The number of bytes $M$ in each file is determined by a geometrical distribution. The crossing of branches — explained in the previous paragraph — means that the sending of files in different sessions will overlap in time.

The following sections present traffic models for three common applications: HTTP, FTP and VoIP. The model designs are motivated by the traffic generators in [Luo00]. However, it must be noted that they are only indicative models and for example do not include heavy-tailed arrival or hold time distributions required for creating self-similar traffic, as explained in section 3.2.4.

### 3.4.1 HTTP

HTTP protocol (Hyper Text Transfer Protocol) is used to transmit web pages in the Internet. Two versions of the protocol are commonly used: 1.0 [BLFN96] and 1.1 [FGM$^+$99]. There are several differences between the two but the one influencing most on our study is the way new TCP connections are created. In HTTP/1.0 a new TCP connection is created for each file that is transmitted. HTTP/1.1 uses persistent connections and request pipelining to transmit multiple files on a single TCP connection. This has a significant effect on the total throughput of the transmission because of the congestion control mechanisms and slow-start feature on most TCP implementations. Since most of the files transmitted using the HTTP protocol are small there is a huge difference in the number of connections created in HTTP/1.0 and HTTP/1.1. However, the effect on the total number of bytes transmitted is not as great because connection establishment and tear-down do not produce much data.

HTTP is designed to transfer HTML documents (Hyper Text Markup Language) [CM00] over the web. An HTML document consists of a text file which defines both the layout and the content of a web page. The content generally consists of pictures or other binary objects in addition to the text. In HTTP/1.0 each object is transmitted in its own TCP connection whereas in HTTP/1.1 the whole page is transmitted in a single connection.

The principle of traffic generation using HTTP protocol is presented in figure 3.7. The vertical axis shows the location of an abstracted object in the HTTP traffic hierarchy and the horizontal axis represents time. Session arrivals are modeled as a Poisson process with a mean inter-arrival time of $L1$ seconds. A session typically represents a a user surfing the web. Each session consists of page requests whose amount is geometrically distributed with a mean of $N1$. The read time between page requests is exponentially distributed with a mean value of $L2$ seconds. A page represents an HTML document and the number of objects in a page is modeled with a geometric distribution with a mean value of $N2$. The time between object retrievals in a page can be estimated as a constant with a sub-second value $L3$. An object represents a file and the size of the object is modeled with a geometric distribution with a mean value of $M$ bytes.

The difference between HTTP/1.0 and HTTP/1.1 when studying the schematic figure 3.7 is the way objects are transmitted. In HTTP/1.0 a separate TCP connection is created for each object whereas in HTTP/1.1 the same TCP connection is used to transmit all the objects. This is a significant difference and causes some concern when simulating certain aspects of the network, such as page completion times and also the quality of service, especially when using IntServ [Wro00]. However, for cases where the utilization level of the network and the individual links are the topic of study the number of connections is not as relevant. Connection establishment produces more time overhead than data overhead so the overall effect for workload simulations is not necessarily that considerable.

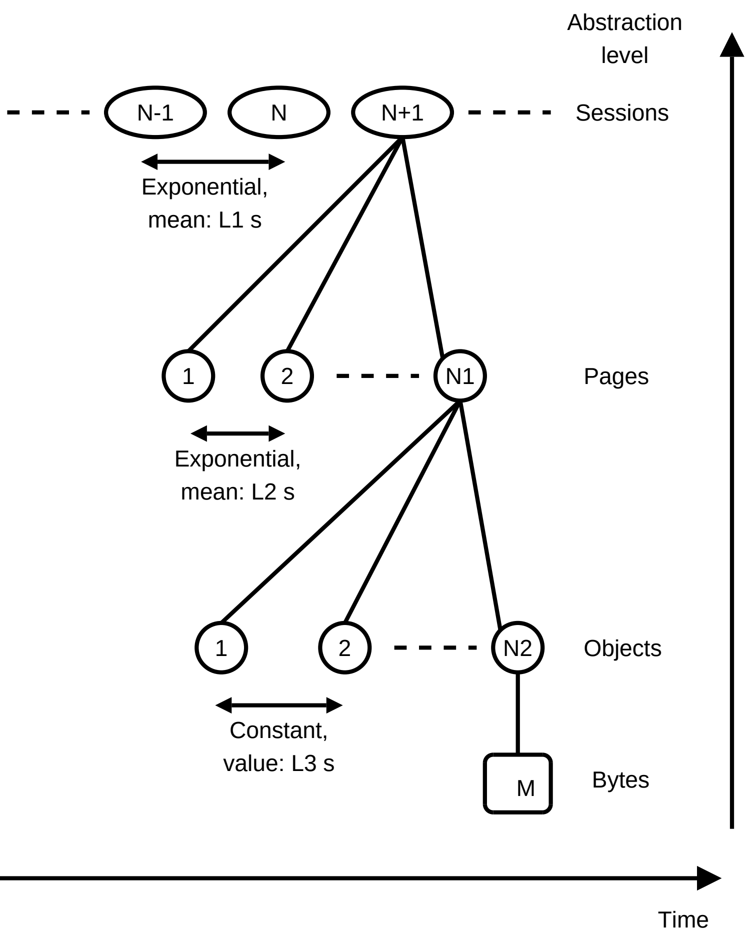

Figure 3.7: Application model of HTTP protocol

### 3.4.2 FTP

FTP protocol (File Transfer Protocol) [PR85] is designed to transfer files between a server and a client. Basically, all the functions of FTP can be achieved with HTTP but FTP is still widely used because of its reliability in transferring large files and advanced features implemented in many FTP applications.

FTP uses a single TCP connection as a control channel. Also, it creates a new data TCP connection for each file it transmits. This principle makes the TCP connection creation process quite similar to that of the HTTP with the difference of a missing object layer. See figure 3.8 for clarification. The arrival processes and the size distributions are the same as in HTTP except that the mean values are different. The main difference is that the average file size $M$ is much larger than in HTTP. The reference “page” is changed to “file” when comparing figures 3.7 and 3.8.

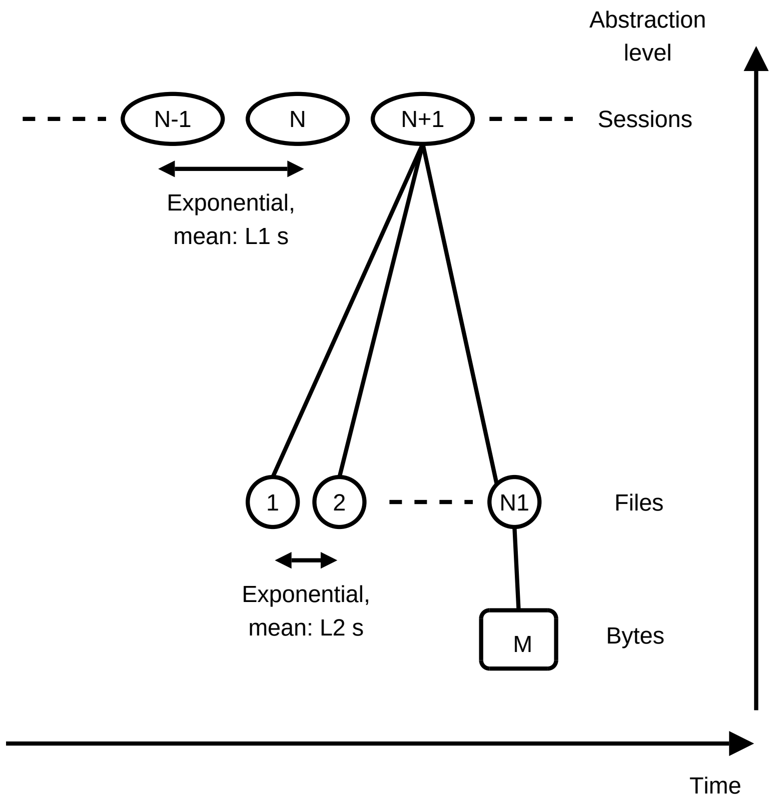

Figure 3.8: Application model of FTP protocol

## 3.4.3 VoIP

VoIP (Voice over IP) is not a protocol but a concept of transferring audio data over IP networks using UDP connections. There are many protocols for setting up and managing VoIP connections, the most common ones being IETF’s SIP [HSSR99] and ITU-T’s H.323 protocol suite [Tho96]. SIP handles only the setting up and managing of the connections but H.323 includes also the definitions on how to handle the voice channels. Both protocols are designed to be independent of the underlying transport protocol. In practice, in IP networks both SIP and H.323 use

RTP (Real Time Protocol) over UDP to transmit the data and RTCP (Real Time Control Protocol) over TCP to control and manage the timing requirements of audio communication. Both RTP and RTCP are defined in [SCFJ03].

The traffic creation processes of VoIP are illustrated in figure 3.9. The call arrivals are modeled as a Poisson process with a mean inter-arrival time of $L1$ seconds. If we assume a call hold time that corresponds to that seen in the PSTN environment (Public Switched Telephone Network) it is exponentially distributed with a mean holding time of $T$ seconds as suggested in [Rah91]. Both the data packet length and time between packet transmissions are constant values.

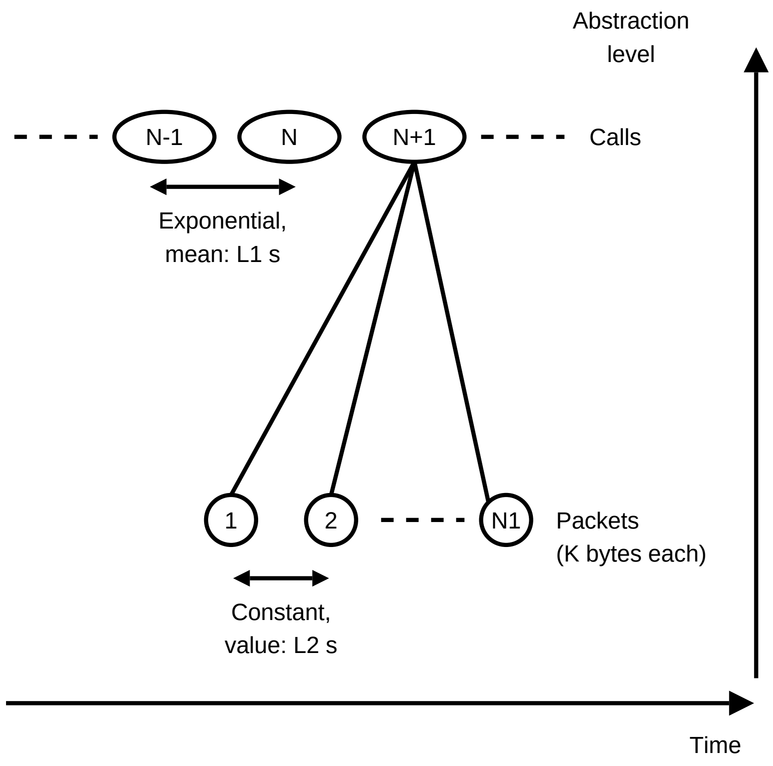

Figure 3.9: Application model of VoIP

## 3.5 Network Performance

The way packets and flows are handled in the network is the other point of analysis after the discovery of the traffic patterns. Traffic pattern discovery is motivated by the aim to understand the user behavior and the type of data that they want to transmit: applications and data amounts. On the other hand, measuring the effect of the network on the packets offers an insight into how it is suited for the users' requirements. For example, issues like: are there bottleneck links or hotspots in the network or is the network overall under-provisioned or chronically congested, can be studied.

There are three parameters which influence the quality of the users' transmissions the most: packet loss ratio, packet delay and delay variation.

## 3.5.1 Packet Loss Ratio

Packet loss ratio tells the fraction of packets that are lost during the communication according to equation:

$$PLR = 1 - \frac{N_{\text{received}}}{N_{\text{sent}}}, \qquad 0 \leq N_{\text{received}} \leq N_{\text{sent}}, \tag{3.20}$$

where $N_{sent}$ is the number of packets sent by the server and $N_{received}$ is the number of packets received by the client. Packet losses are usually the result of network congestion. When packet receive or send buffers in the routers overflow the excess packets are dropped without any notification to the sender or the receiver of the packet. Also, most modern routers use a RED queuing (Random Early Detection, [Bra98]) discipline which causes packets to be dropped at an early stage of congestion even before the buffers overflow. It is up to the transport layer — L4 in the OSI-model — to notice the packet losses and recover from them.

Packet loss ratio is defined to be a parameter on a single direction of communication. In the case of a TCP connection, for example, packets are transmitted on both directions: data packets from the server to the client and acknowledgement packets on the opposite direction. Hence, packet loss ratio is calculated separately on both directions which is important because the network may be congested only in one direction and more lightly loaded on the other.

### 3.5.2 End-to-End Packet Latency

End-to-end packet latency is a sum of four types of delay occurring in the communication path: processing, transmission, queueing and propagation delay. The total delay is cumulated on all links and routers on the communication path:

$$T_{\mathrm{e2e}}(t) = \sum_{i=0}^{N-1} T_{\mathrm{i,proc}}(t) + \sum_{\mathrm{j}=0}^{N} \left(T_{\mathrm{i,queue}}(t) + T_{\mathrm{j,tr}} + T_{\mathrm{j,prop}}\right), \tag{3.21}$$

where the index $i$ represents a router and $j$ a link in the communication path. $N$ is the number of routers in the path and consequently the number or links is $N+1$. Transmission and propagation delays are constant and don't vary as functions of time. Queueing delay and processing delay vary as functions of time because they both are dependent on the congestion level of the network.

Processing delay occurs in the end-hosts and at each IP router in the network. Packet forwarding in the routers is in most cases performed by a fast switching architecture which uses only the destination IP address in the packet and the routing table in the router to decide where the packet is forwarded. The handling of a packet takes a small and approximately constant delay which is also router dependent. If the packet contains information in the "IPv4 options" field of the header it has to be handed over to the router's CPU. Software processing by the CPU takes considerably longer time compared to the normal forwarding. However, the "IPv4 options" field is very rarely used and its effect on on the network performance is thus quite limited. In general, processing delay is hard to dissect or describe formally so we will simply state it to be dependent on time by an unknown factor.

Transmission delay is caused by the actual sending of the packet in one of the router's interfaces. The amount of the delay is given by equation:

$$T_{\mathrm{tr}} = 2 \cdot \frac{P_{\mathrm{pkt}}}{B}, \tag{3.22}$$

where $P_{\mathrm{pkt}}$ is the size of the packet and $B$ is the bandwidth of the sending network interface. The multiplier $2$ is needed because transmission delay occurs both at the sending and receiving end on each link.

Queueing delay is caused by buffering packets in the router and it is the predominant source of delay on a connection. On most routers, there are separate receive buffers for each incoming link and send buffers for each outgoing link which all cause a separate queueing delay. For simplicity, queueing delay can be modeled with just the send buffers. The amount of queueing delay an arriving packet experiences depends on the number of packets in the buffer at the time of the arrival and also on the other delays occurring in the router according to equation:

$$T_{\text{queue}}(t) = m(t) \cdot T_{\text{tr}} \approx m(t)\frac{P_{\text{avg}}}{B}, \tag{3.23}$$

where $m(t)$ is the number of packets in the queue at the time the packet was queued. The packet size $P_{\text{pkt}}$ in $T_{\text{tr}}$ was replaced by $P_{\text{avg}}$ to indicate the average packet size in the connection. Ultimately the total queueing delay is affected by the congestion level of the network. Equation 3.23 is the lowest limit estimate of the total queuing delay for an arriving packet because it does not take into account the remaining transmission time for the packet that is currently being sent. The equation does not include the processing delay for the packets in the queue because the processing of a packet is generally being performed parallel to the transmission of the previous packet and the processing delay is invariably smaller than the transmission delay.

Propagation delay means the time it takes for a packet to traverse through a link. The delay is given by equation:

$$T_{\text{prop}} = \frac{L}{C}, \tag{3.24}$$

where $L$ is the length of the link and $C$ is the propagation speed of the signal. On copper links the electrical signal travels approximately at the speed of light in a vacuum: $C_{\text{copper}} \approx c = 3.0 \cdot 10^8 \frac{m}{s}$. On optical wires the optical signal speed is limited by the speed of light in glass: $C_{\text{optical}} \approx 1.45^{-1} \cdot c \approx 2.1 \cdot 10^8 \frac{m}{s}$.

As a final result, by combining equations 3.21–3.24 we can deduct the total end-to-end packet delay on a communication path:

$$T_{\mathrm{e2e}}(t) \approx \sum_{i=0}^{N-1} T_{\mathrm{i,proc}}(t) + \sum_{j=0}^{N} \left( m_j(t) \frac{P_{\mathrm{avg}}}{B_j} + \frac{P_{\mathrm{pkt}}}{B_j} + \frac{L_j}{C_j} \right) \tag{3.25}$$

**Measuring One-Way Delay**

The concept of one-way delay is simple: it is the difference between the time packet was received and the time it was sent. In practice, things get more difficult because the clocks must be synchronized with each other, i.e. they have to show the same time. The problem of clock synchronization was discussed in section 2.3. There will always be some offset between the clock times and the measured one-way packet delay is given by equation:

$$\delta = T_{\mathbf{rec}} - T_{\mathbf{send}} + \sigma_{\mathbf{rec-send}}, \tag{3.26}$$

where $T$ is the timestamp recorded at either the sending or the receiving side of the communication path and $\sigma_{\mathbf{rec-send}}$ is the offset between the clock times in them. The offset will always remain unresolved beyond a certain degree of accuracy. However, the problem can be alleviated by trying to reduce the offset or attempting to figure out the size of it. NTP, for example, offers the tools to combine the two methods and generally one can reach reasonable levels of accuracy with it. Ultimately, there will always be some inaccuracy in the measurements and the task left for the measurer is to firstly minimize the offset and secondly to make a reliable error estimate of it.

### 3.5.3 Jitter

The variation in the packet delay — or jitter — is a factor that in many cases influences the quality of the connection more than the delay itself. A high but steady delay is more easily controlled and compensated than a lower but fluctuating delay. As seen in equation 3.25, the only time dependent delay in the communication path is caused by queueing. Jitter is caused mainly by the rapid variations in the occupation levels in the send and receive buffers of the routers.

Jitter is a problem mostly concerning applications with real-time requirements, such as applications delivering VoIP calls. A real-time application in the receiving end of the connection contains a packet play-out unit which is used to store packets prior to the play-out. The buffer causes an extra delay from the transmission to the presentation. The size of the buffer — consequently the amount of the extra delay — is determined by the packet jitter occurring in the connection: the more jitter, the greater the play-out buffer. In principle, if there was no jitter in the packet delay there would be no need for the play-out buffer. Non-real-time or elastic connections can tolerate very high jitter because of the high retransmission time-out (RTO) marginals in most TCP protocols as suggested in [PA00].

Jitter refers to the tendency of the packet delay to alternate over time and between packets. However, jitter is not a uniquely defined concept and there have been some disagreement of what is meant by it. Therefore, there are several ways to estimate jitter: packet delay variation, delay variance and max-min delay comparison.

### IP Packet Delay Variation

IP packet delay variation (IPDV) is the most common way to study jitter. It is presented by both IETF in [DC02] and ITU-T in [Y.199] and is sometimes referred to as inter-packet delay variation or successive delay variation. The method is to compare the difference in the one-way end-to-end delays of consecutive packets. IPDV is derived by equation:

$$\Delta_{IPDV} = \delta_i - \delta_{i+1}, \tag{3.27}$$

where $\delta_i$ is the one-way end-to-end delay of a packet with index number $i$. Index number is an incrementing integer which tells the order in which packets are sent. The main benefit of measuring IPDV is that it is insensitive to the offset error between the sending and receiving side clocks. Results are always accurate and no error marginal needs to be taken into consideration. This feature can be verified by combining equations 3.26 and 3.27:

$$\Delta_{IPDV} = T_{\mathbf{i,rec}} - T_{\mathbf{i,send}} + \sigma_{\mathbf{i,rec-send}} - (T_{\mathbf{i+1,rec}} - T_{\mathbf{i+1,send}} + \sigma_{\mathbf{i+1,rec-send}}), \tag{3.28}$$

where it is safe to presume that usually $\sigma_{\mathbf{i,rec-send}} \approx \sigma_{\mathbf{i+1,rec-send}}$ because of the small duration between consecutive packet. This combined with the above equation yields:

$$\Delta_{IPDV} = (T_{\mathbf{i,rec}} - T_{\mathbf{i,send}}) - (T_{\mathbf{i+1,rec}} - T_{\mathbf{i+1,send}}), \tag{3.29}$$

where we can see that the offset term between the two clocks is canceled out.

## Variance

Calculating the variance is the generic method of estimating the volatility in a measurement series. In [ML00] it is used in estimating the stability of bandwidth usage on a link. Variance is calculated according to equation:

$$\sigma^2 = \frac{1}{N} \cdot \sum_{i=1}^{N} (x_i - \overline{x})^2, \tag{3.30}$$

where $N$ is the number of measurement samples, $x_i$ is the $i$:th sample and $\overline{x}$ is the mean value of the sample set $x$.

## Max-Min Comparison

Max-min comparison means finding the highest and lowest samples in the measurement series and comparing their difference:

$$\sigma_{\max-\min} = x_{\max} - x_{\min}, \qquad x_{\min} \leq x_{\mathrm{i}} \leq x_{\max} \qquad \forall i \in \mathrm{S}, \tag{3.31}$$

where $\mathrm{S}$ is the index set used in the measurements. It is a very simple way to analyze a data set and the results can sometimes be misleading and non-informative. However, it does offer a convenient and fast way to see the region to be examined and the order of magnitude of the samples. It can also be calculated real-time as the measurement progresses. The real benefit of max-min method comes when the data set is inbuilt with a theoretical maximum or minimum value which one wants to discover. For example, when studying the end-to-end packet delay on a given communication path with a large sample set of packets, the minimum delay seen in the samples is presumably quite close to the theoretical minimum transmission delay on the path, i.e. the delay stated in equation 3.21 with the queueing factor removed.

# Chapter 4

# Methodology

## 4.1 Network Structure

The network to be analyzed is a core network with gigabit and 10/100 Ethernet links interconnecting the routers. However, possibly also 2 megabit links exists in the network. The topology of the network is unknown but it is not needed in the analysis because only the traffic at the edges needs to be inspected — routing and traffic behavior inside the core network is irrelevant for our study.

The point where the core network is connected to the periphery is called a PoP (Point of Presence). It can be used to attach one or more access networks and servers to the core. It usually acts as a media switch, for example to convert the gigabit Ethernet links to 10/100 megabit Ethernet links or to 2 megabit PDH (Plesiochronous Digital Hierarchy) links. Each PoP is connected to another one to create redundancy and fault tolerance to the communication between the core and local area networks. The structure of the PoPs existing in our target network is presented in figure 4.1.

## 4.2 Measurement Infrastructure

The measurement infrastructure consists of three types of platforms: monitoring, data collection and analysis. In this thesis, the platforms are most of the times referred to as points to emphasize their different location in the network under

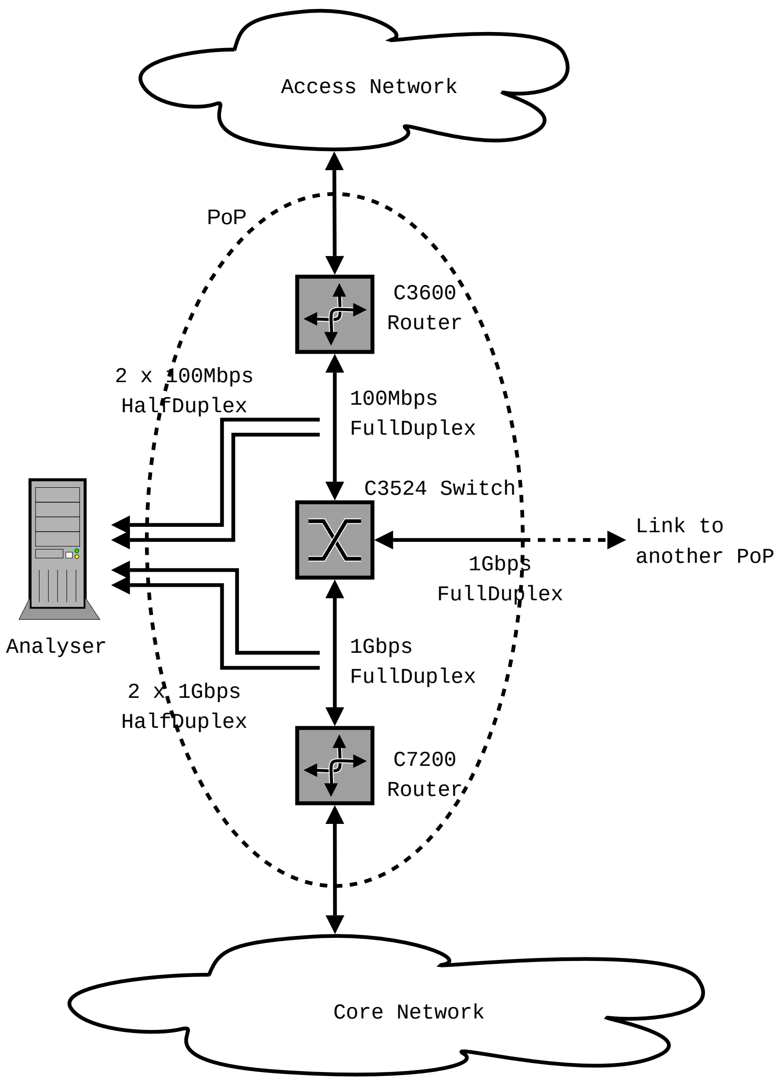

Figure 4.1: PoP (Point of Presence) and a traffic monitoring computer connected to capture traffic on core and access network links.

study. The monitoring point's task is to perform raw capturing of packet traffic on one or multiple links. Also, remote terminal connections by using SSH2 protocol must be supported to enable easy configuring of geographically distant monitoring points from a single point of control. The SSH2 protocol set is currently being developed by IETF's Secure Shell working group [Som03]. The data collection point's function is to fetch the raw trace files from the monitoring points using SSH2 and store the files for later analysis. The transfer is automated and is performed during night time — or the slow hours — to minimize the effect it has on the existing network traffic. It also preprocesses the raw trace files into a format that is better suited for analysis purposes. The analysis point is where the data is examined and the formal traffic model is created.

The order of performing network measurement tasks and handling trace files is illustrated in figure 4.2.

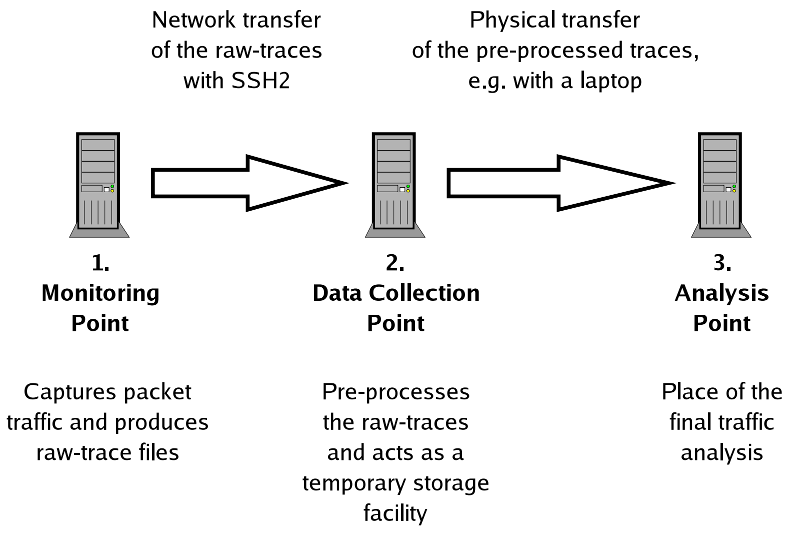
Network transfer of the raw-traces with SSH2
Physical transfer of the pre-processed traces, e.g. with a laptop
1. Monitoring Point
2. Data Collection Point
3. Analysis Point
Captures packet traffic and produces raw-trace files
Pre-processes the raw-traces and acts as a temporary storage facility
Place of the final traffic analysis

Figure 4.2: The principle of measuring the network and handling the trace files. The whole measurement infrastructure consists of multiple monitoring points but only one data collection point and one analysis point.

### 4.2.1 Monitoring Point

The monitoring point captures traffic on selected links of a PoP. A single computer can be used to monitor multiple links as long as they reside physically in the same place. High speed traffic capturing on multiple high speed links may cause performance problems when traffic loads are high. A lot of effort has been put in creating an efficient monitoring platform and the hardware and software used are presented in section 5.1.

### 4.2.2 Data Collection Point

The data collection point's primary function is to gather all the trace files taken in the monitoring points to a single platform. The files are transferred using secure SSH2 protocol with strong encryption algorithms. The transfer is automated and is performed during night time — or the slow hours — to minimize the effect it has on the existing network traffic.

The packet trace files are preprocessed in the data collection point with Coral-Reef tools, explained later in section 5.2.2. The processing is done in three stages and the following properties are inspected: traffic rate on a per minute basis, flow separation and per-packet timestamping. The preprocessed data is retrieved with a portable media — such as a laptop computer — and taken to the final analysis point.

### 4.2.3 Analysis Point

The analysis point is the final destination for the measurement data. The preprocessed data is passed there from the data collection point and is further processed using generic mathematical and data extraction tools like Matlab and AWK.

## 4.3 Maintaining Time with NTP

NTP is configured as a star shaped structure (presented in section 2.3.1) where the server has a set clients synchronized with it. However, there are multiple servers which are not synchronized with each other. They are estimated to be accurate and reliable enough without mutual synchronization. Each monitoring point is configured with an NTP client and synchronized with one server which is closest to it. Accordingly there will be as many client-server relations in the network as there are monitoring points.

### 4.3.1 Accuracy Requirements

The purpose of NTP is to keep the offset between the server and the client clocks as low as possible. The most stringent requirement for the clock offsets is set by the need to measure one-way packet delay. The key parameter is the relative time error of the measurement, explained in section 2.3.5. The estimate for the minimum end-to-end packet delay is 10 milliseconds. A reasonable maximum relative time error would be 5 per cent. Applying these figures on equation 2.5 yields an absolute time error of 0.5 milliseconds. As estimated in section 2.3.3 the timestamp error on a single clock synchronized with NTP should be a fraction of a millisecond. However, it should be noted that the delay measurements contain the offset error of two independent clocks, as explained in section 2.3.5. Hence, the maximum allowable offset error for a single clock is $0.5ms/2 = 0.25ms$.

### 4.3.2 Measuring and Controlling Offset

NTP is set up so that periodic control packet are sent every 64 seconds between the server and the clients. This is the default value for NTP and a suitable compromise between the speed of offset convergence and the amount of extra traffic generated. Also, NTP is quite well aware of the synchronization error, i.e. the offset between the local and the remote clock, as also explained in section 2.3.3. Periodic checks are performed every 60 seconds from the server to each client to maintain awareness of the clock offsets. This information is used later in the analysis phase for more accurate delay analysis.

## 4.4 Traffic Models

A traffic model resembles the application model, explained in section 3.4. The hierarchy structure of elements is similar but the model is constructed to better fit the results of the network traffic analysis, i.e. the traffic generation parameters measured in section 4.5.1. The traffic model created is to be used in the simulations.

Traffic models are assigned pair-wise between those core routers which are also connected to an access network. The benefit of this approach is its accuracy; if routing in the simulated and the real networks are configured identically the packet transmission paths will be chosen similarly. Consequently, the traffic loads and congestion hot spots should also coincide in the real and simulated networks. A drawback of a pair-wise approach is the large number of traffic generation models: $PN(N-1)$ where $N$ is the number of core routers and $P$ is the number of traffic models needed for replicating traffic accurately between two routers.

The difficulty of designing a traffic model is that the elements used should be easy to identify in the real network and to implement in the simulation. Application models in section 3.4 would model the traffic of these applications very precisely but the identification of the data structures — such as a page or session in HTTP — is not easy by examining only the ongoing packet traffic. Instead, a more simple model must be used.

### 4.4.1 Flow Based Model

In this thesis, a model with two abstraction levels is used: flows and packets. Flows are identified from the network traffic using the 5-tuple of protocol and source and destination IPs and ports, as explained in section 3.3. Their arrival times and sizes in packets or bytes are also easily recognized. Since flows can be thought of as virtual connections it is natural to replace flows with TCP and UDP connections in the traffic model. Thus, the arrival process of the packets need does not be defined by any statistical distribution. In TCP, only the size of the flow in bytes is measured which can lead to some inaccuracy because the transmission rate of TCP cannot be set. In UDP the length of the flow and the constant interval between consecutive packets is measured which makes UDP flows much more controllable compared to TCP.

In TCP, only the data flows need to be fitted into the model; acknowledgement flows will be automatically generated by TCP. An ACK packet is 40 bytes long, i.e. only the IP and TCP headers are included. ACK flows are defined here to be flows with mean packet size less than 60 bytes and they are discarded from the traffic model.

### 4.4.2 Protocol Layers

Before creating multiple traffic models one must consider the basis on which flows are divided between them. A separate traffic model for each protocol is a natural starting point. The protocol layer to be modeled can be either transport or application layer in the 5-layer TCP/IP model, presented in section 3.1. The network layer does not introduce enough separation because in addition to IP it holds very few protocols. Traffic model for each transport protocol (TCP, UDP, etc.) is a feasible solution and can bring reasonably accurate results, despite the fact that most of the traffic comprises of TCP protocol. Application protocol offers a large variety of protocols and a large number of traffic models will replicate very accurately the traffic in the real network to the simulations. However, it also makes the analysis process very time consuming and the simulations may become too heavy. In order to work efficiently, application protocols must be grouped or aggregated so that one traffic model can cover multiple protocols.

### 4.4.3 Protocol Aggregates

Protocol aggregates will be used for reducing the number of traffic models. Similarly behaving protocols are grouped together to form an aggregate. A traffic model is created for each protocol aggregate. The beauty of using aggregates is that the accuracy and also the robustness of the simulation can be controlled by adjusting the number of aggregates; using few aggregates produces a light-weight but potentially inaccurate simulation and increasing the number of aggregates results in heavier but more accurate simulations. The aggregates are produced by using the data clustering tools, such as self-organizing maps (SOM). The SOM algorithm is published in [Koh00] and applied for clustering application protocols of network traffic in [Lam01].

### 4.4.4 Source and Destination Points

The traffic models are created based on the traffic traces which are directed into the network. Thus, the router from which the traffic was captured acts as the source point of the model. The destination point, i.e. another router in the core network, must also be determined. Two options are available: pair-wise inspection and heuristic destination deduction.

In this thesis, the traffic will be analyzed pair-wise between each core router. It enables separate traffic models to be individually created between all router pairs in the network. The result of each pair-wise analysis is a traffic model between the routers, i.e. the number of analysis models is the same as the number of resulting traffic models: $PN(N-1)$. However, certain routing information from the network must be known in order to use this method because the traffic traces must be split according to the destination router of the packets. This requires at least that the subnets residing in the access networks behind each core router are known.

Another option is to use a heuristic method of deducting a destination for the traffic models. In this approach, traffic passing through a router $i$ is analyzed without first splitting it according to the destinations of the packets — the router now has one traffic model which covers the total volume of traffic it passes on to the core network. The model is split into $N-1$ sub-models whose targets are the other core routers. Each sub-model $j$ is scaled with a weight $\lambda_j$ so that $\sum_{j=1}^{N} \lambda_j = 1, \; j \neq i$. The factor $\lambda_j$ depends on the amount of traffic $T_j$ seen exiting the router $j$ in the real network: $\lambda_j = \frac{T_j}{\sum_{k=1}^{N} T_k}$.

## 4.5 What Is Analyzed

There are two different sets of results: the traffic generation parameters and the performance parameters. They differ both in terms of the environment and the time they are taken in. The traffic generation parameters are derived only from the data received from the real network. They are used in creating the traffic model which will later be used in the simulation environment. The performance parameters indicate how the network treats the traffic injected into it. For this thesis, they are measured only in the real network but eventually they are measured also in simulated network and finally compared to each other. The comparison will tell how well the simulated environment reflects the situation in the real network. In addition, self-similarity from both the real and simulated networks are measured. The overall principle of data gathering and result analysis and how they are used in later studies with the simulations is illustrated in figure 4.3.

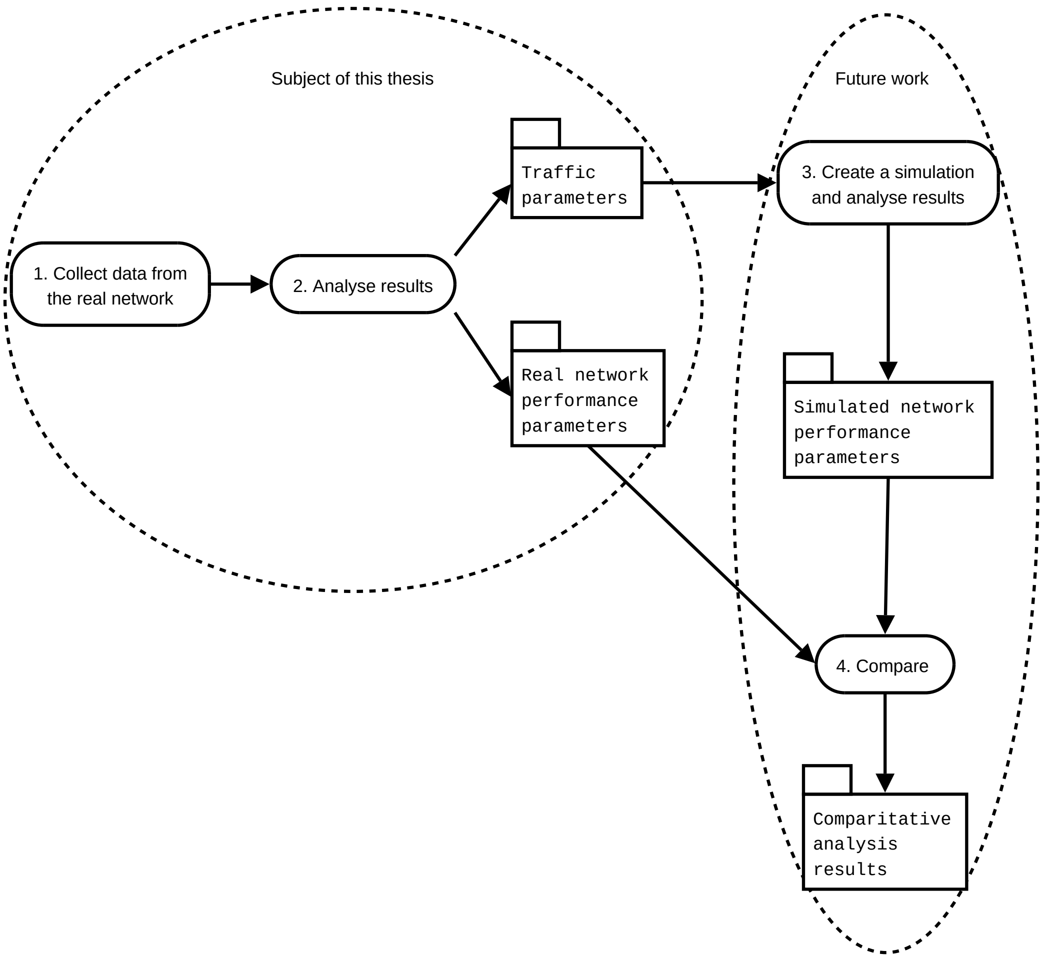
Subject of this thesis
Future work
1. Collect data from the real network
2. Analyse results
Traffic parameters
Real network performance parameters
3. Create a simulation and analyse results
Simulated network performance parameters
4. Compare
Comparitative analysis results

Figure 4.3: The work flow of gathering data and analyzing the results and how they are used with the later simulations. The rounded boxes represent tasks and the sharp edged boxes represent results.

### 4.5.1 Traffic Generation

Traffic generation parameters describe in a formal way the types and amounts of data the users are sending to the network. They are the starting point for creating the traffic models.

The traffic is dissected into hierarchical layers to reveal the characteristics of the traffic in varying levels of detail. The layers of hierarchy to be studied are: rate, traffic composition by protocol, flows and finally packets. The principle is illustrated in figure 4.4. This kind of layering provides a flexible and easy way to use various traffic models in the simulations — in the one end of the spectrum, the traffic generator can inject the network with only a single type of packet on a desired rate and on the other end of the spectrum many types of protocols with varying flow characteristics and packet distributions can be used.

Using the complete information on all four layers in figure 4.4 will yield the most accurate and detailed traffic models in our approach. However, for improving scalability and efficiency the results can be simplified to some extent at the expense of accuracy. There are two approaches: limiting the examination on only certain layers or aggregating the results within the protocol layer. The former can be easily accomplished by excluding the packet level outside of the scope of view. The latter means finding the protocols with similar characteristics and combining them to form a common traffic generating source.

#### Traffic Rate

The traffic rate inspection will show the total bandwidth usage as a function of time on the monitored link. It offers a convenient way to identify the busy and slow hours and the difference in the traffic volumes between them. It is important to identify them as well as any other semi-stable state between them because they typically have different traffic characteristics and have to be modeled separately. Traffic rate shows only the number of bytes sent per second and should not be confused with throughput of a connection or a link. The rate is monitored on a given link somewhere at the edge of the core network and does not take into account any packet drops that have occurred previously or might occur further down the communication path.

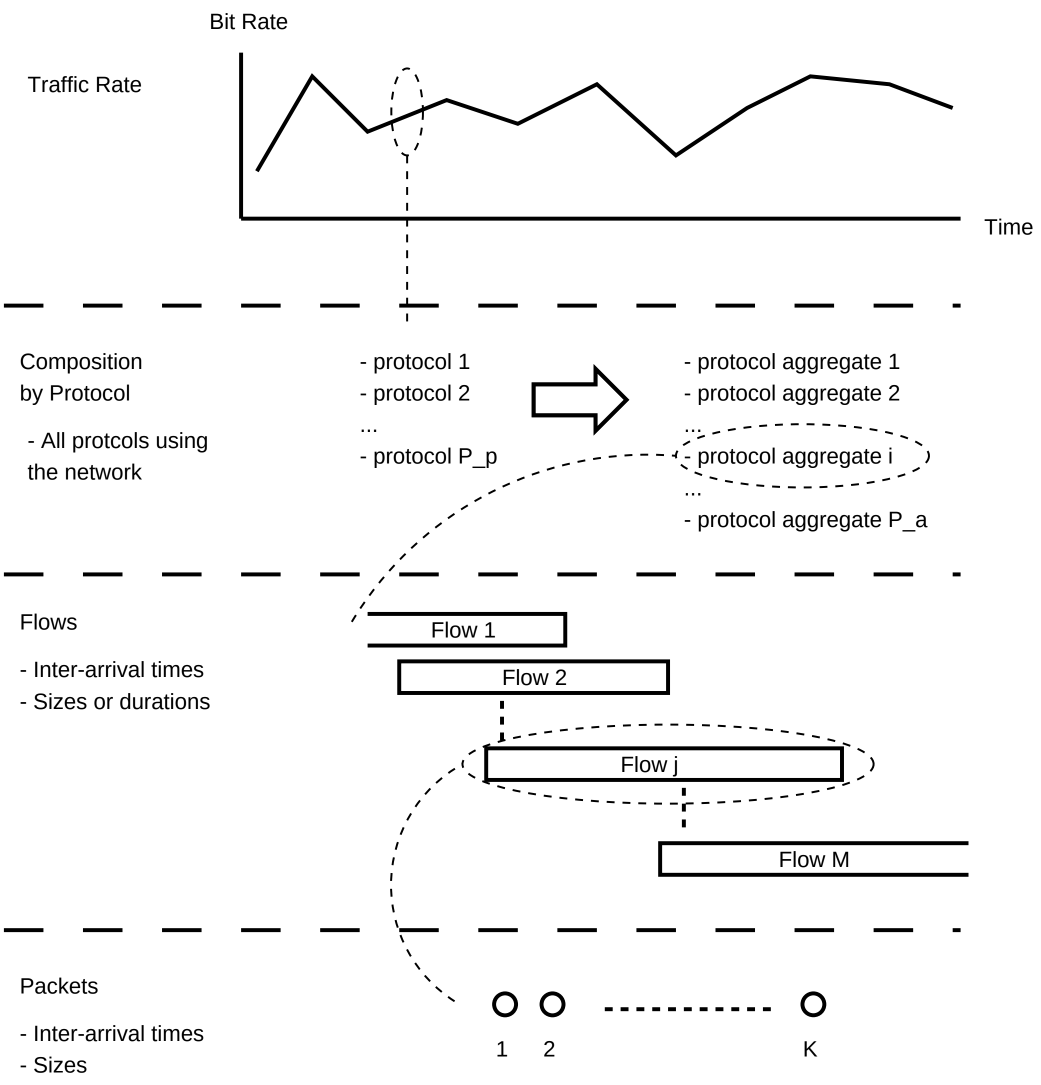

Figure 4.4: Network traffic dissected into four layers: rate, composition by protocol, flows and packets. A snapshot taken in a given time shows $P_p$ concurrent protocols which can be selectively combined to form $P_a$ protocol aggregates where $P_a \leq P_p$. Protocol aggregate $i$ is being used by $M$ different flows, of which for example flow $j$ consists of $K$ packets.

### Traffic Composition by Protocol

The traffic consists of various protocols which generally are dependent of each other on vertical layers, i.e. some protocols rely on the lower layer protocols to function. The detail level of the protocol analysis can be varied by selectively choosing protocols only on some OSI layers, for example only studying network or application layer protocols. The hypothesis is that the most meaningful results can be achieved by studying two set of protocols: firstly the transport and secondly the application layer protocols.

### Flows

Flows are used as building blocks in traffic generation. The ease of using flows is caused by the fact that they can be modeled as TCP or UDP connections to a certain degree of accuracy. To be able to use TCP and UDP connections to model flows accurately the following properties have to be discovered:

- Arrival time distribution

  Type (exponential, Pareto, etc.)

  Parameters

- Size distribution

  Type (exponential, Pareto, etc.)

  Parameters

The number of parameters depends on the distribution, for example exponential distribution has only one identifying parameter: mean value. The size distribution is set in bytes but generally speaking it could also be set in packets. Identifying the flow arrival process may prove a rather challenging task to be done accurately.

A UDP connection has no flow control mechanism and the way packets are generated within the connection has to be determined. At first, it is assumed that UDP connections are controlled by a CBR (Constant Bit Rate) generator where the size and inter-arrival times of the datagrams are constant. These two parameters also have to be discovered for the traffic model when using UDP flows. If the assumption that UDP connections can be modeled with as CBR generators is not valid the model will have to improved in the future.

Many applications create connections according to a hierarchical model, as is the case in HTTP (presented in section 3.4.1). The ideal would be to discover the model for our studies the way we know it works in real networks, i.e. as illustrated in figure 3.7. However, the traces offer no way of identifying which pages individual flows are a part of and only a crude method to identify sessions — by mapping them with source/destination IP pairs. Consequently, we will only study and model the flows on a per protocol basis and as a single arrival process within a protocol. Because of the inherent complexity in the way real applications produce data the arrival process seen can be quite complex and not necessarily coherent with any of the commonly used distributions depicting arrival events.

### Packets

Packet level studies offer more insight into the structure of individual flows. Some flows can be modeled reasonably well with TCP or UDP connections whereas other flows have very different packet level characteristics. Some application protocols produce periodic transmissions of single or multiple packets. NTP is an example of such a protocol and it performs periodic transmissions on a predefined interval to maintain clock synchronization between hosts. Depending on the length of the synchronization interval and the flow time out value the transmissions can be interpreted as either short individual flows or as one very long flow. The effect this has on both the number and properties of the flows is considerable.

The same list of parameters as was presented in the previous section (3.3) have to be discovered for packets as well. The arrival processes can be observed on each flow or protocol separately to find out any possible deviances or anomalies in them. Since not all protocols are widely known or even used anywhere else, we must be prepared to discover all uncommonly behaving protocols as well. However, in most cases, TCP and UDP connections depict the flows well enough and using the flow based model will be sufficient.

### 4.5.2 Network Performance

The network performance parameters indicate how the traffic is being treated by the network. The same performance parameters are determined for both the real and the simulated network. The conclusion of how well the simulation reflects the real network behavior are based upon the comparison of these parameters.

The parameters measured are: one-way packet delay, packet drop probability and throughput. Delay and packet drop probability are strongly correlated with the congestion level of the network and inform quite accurately how well the network is coping with the given traffic. Both of the two are also correlated with the third parameter, throughput.

#### One-Way Packet Delay

One-way packet delay is the time it takes for a packet to reach the destination after it has been sent. The difficulty in measuring it is that we can not know in one place both the time it has been sent and the time it has reached its destination. A two-way delay is the more commonly used parameter to measure network delay because of the ease of its use. However, the opposite directions in the communication path may have very different delay compared to each other and studying only the two-way delay hides this aspect.

We will measure the one-way delay by taking packet traces at each side of the core network; a trace file recorded in one point will show the sending of the packet and a trace file recorded in another point will show the receiving of the packet. The difficulty is to have accurate knowledge of time at both points and we are using NTP to alleviate this problem. There will be some error in the delay measurement, as discussed in section 4.3.1. The goal is to keep the delay measurement error below 5 per cent.

#### Packet Drop Probability

Packet drop probability is calculated by measuring the number of packets dropped in the network during the measurement period. Measuring it also has its difficulties; if some router in the core network decides to drop a packet it does not send any notification about the occurrence. Also, studying the end-host communication generally speaking yields no information about missing packets in the communication. Instead, we will again study the packet traces taken at the edges of the core network and determine from them the number of packets that were sent but never received. The simplest procedure to know the overall packet drop probability in the whole network would be to calculate and compare the number of all packets entered and all packets exited the network. However, this would be inaccurate, since we cannot assume that all packets that are sent to the core network will exit it at some other point — for example, some probe packets can be sent to a router inside the core network. Instead, we have to calculate the packet drop probability pair-wise between edge routers in the core network. For this we must be able to determine at which edge router a given packet will exit the core network by comparing the destination IP in the packet against the known subnetworks behind each edge router.

### Throughput

Throughput is the rate at which traffic is successfully transmitted in the network. Like packet drop probability, it can be calculated pair-wise between edge routers of the core network. In addition, it makes sense to calculate throughputs of individual flows for a more detailed knowledge of the way bandwidth on the communication path is being utilized.

Throughput is the only parameter of the three which is strongly attached to the specified performance parameters of the network, in this case: link bandwidths. In theory, if we know the traffic generation parameters accurately we can calculate the expected throughputs in the network. However, in reality the actual packet forwarding capability of individual routers fall far back from the specified link bandwidths. The actual forwarding speed depends on the type of traffic being forwarded: packet lengths, packet types, IPv4 options, etc. Implementing all these characteristic in the simulation model is a difficult task and not to be done to the fullest degree of accuracy withing this thesis. As a result, the simulation model will expectedly show more throughput on the traffic compared to the real network and some artificial handicap coefficient to compensate this situation might have to be implemented into the simulations.

### 4.5.3 Self-Similarity

The degree of self-similarity will be measured in various timescales. Studying short and long scale separately offers insight on the sources of the self-similarity, as explained in section 3.2.4. It also helps in deciding which distributions to use in the traffic model.

Self-similarity is an interesting property because it is one aspect to be taken when the traffic of the real and the simulated network are compared. It is a property which is not explicitly introduced to the simulation but rather it will emerge if the simulation environment operates as it should, i.e. the same as its counterpart in the real network. If the degree of self-similarity is the same in both environments it is one indicator of the validity of the simulated network and the traffic in it.

# Chapter 5

# Analysis Tools

## 5.1 Traffic Monitor Hardware

One of the driving design concepts of the analysis project has been to keep the cost of the equipment at an economic level. No proprietary measurement hardware is used for capturing the traffic, only standard PC's. This is especially important for a measurement infrastructure used in our project because of the number of monitoring points intended to be used in the target network. The cost of using proprieatary hardware would be too great for our budget. The hardware of a single monitoring point is as follows:

- Single processor rack-mountable PC
- 2.0 GHz, 512 MB RAM, 32 bit / 33 MHz PCI-bus
- RedHat Linux 9.0, version 2.4.20 kernel
- Ethernet interfaces:

  3 or 5 * 10/100 Mbps Intel Pro NICs

  2 * 1 Gbps fiber-optical SysKonnekt NICs
- Copper and fiber-optical Netgear splitters for the monitored duplex links

Customizability is also an important benefit of using standard PC's. The monitoring points can be designed to fit the goals of the project at hand and if the goals change the equipment can also be easily changed with them. In addition, the work load of traffic analysis can also be divided from the data collection and analysis points to the monitoring points. Proprietary infrastucture is often designed to perform one task as efficiently as possible, for example doing simple packet capturing. However, PC's can be assigned also to do analysis tasks with the raw traces before sending them off to the data collection point.

### 5.1.1 Network Connections

Network connections serve two purposes: they capture traffic on the monitored links and connect the monitoring point to the control network. One dedicated interface is used for accessing the control network through which communication and data transfer to the data collection point is performed. Monitored links are duplex links which means that two network interfaces are needed to monitor one link. In addition a duplex splitter (or a link tap) is needed for each link as described in section 2.2.1.

### 5.1.2 Optimization

Packet tracing on high speed links requires a great deal of processing power from the measurement computer. The biggest bottleneck is in the interrupt handling mechanism and the strain high rate interrupt handling causes on a standard PC. The maximum handling rate varies depending on the task the computer has to perform on an interrupt. Network interrupts where data has to be read by the processor and written to a hard drive by DMA transfer can typically be handled at a rate of 50–100 k/s. This is the case where interrupts arrive from a single network interface; in case of more interfaces the figure decreases.

Interrupts are created depending of the network drivers. Many simply create one interrupt for each incoming and outgoing packet. Others use interrupt mitigation, or interrupt moderation, to limit the rate of interrupts to a desired level. The allowed number of interrupts per second is a compromise between timing accuracy and the number of unseen packets at high data rates. The more interrupts are created the more packets are dropped by the kernel causing the packets to be unseen. However, the resolution of the timestamps is the inverse of the number of interrupts per second so to maintain accuracy interrupts must ge generated at a high enough rate. Quick measurements showed that a suitable limit for interrupt creation for our measurement platform is 2000 interrupts per second per interface. Any figure above this will cause packets to be dropped by Linux's kernel even at a low link utilization level. However, timestamps taken on a packet now have an inaccuracy of $1/2000s = 500\mu s$.

Another figure that needs to be adjusted is the size of network receive buffers in Linux's kernel. Initially the buffer size is 65535 bytes. The ability to handle short bursts of traffic is improved if the buffer is extended and for these measurements it is set to 262142 bytes. However, increasing buffer size also increases the queueing delay of the packets and causes extra latency in the measurements.

### 5.1.3 Security Considerations

Traffic monitoring is quite secure when using network taps. A link from a tap to the monitoring computer is unidirectional so no packets can be sent to the target network by the monitoring computer. This increases the security by the network administrator's point of view. Also the network interfaces in the monitoring computer are not given any IP addresses, so hacking it by using the monitoring interfaces is impossible.

The vulnerable point in the monitoring computer is its control interface through which it is administrated and the data transmitted to the data collection point. If there is a seperate private control network the problem is minor but if the control interface is connected to the user network, as is the case here, then security is an issue. The following is done to improve the security of the monitoring computer:

- All incoming TCP connection requests are refused except the ones coming from trusted IPs
- All unnecessary services are disabled to minimize the risk of software bugs
- All network services are disabled except SSH on port 22
- Access by using SSH port is tightened:

Access only by using RSA2 authentication — no password login allowed

Only SSH2 protocol is allowed - SSH1 is denied

No root login allowed

Every host has a preset host-RSA key

No X11 forwarding allowed

- Linux kernel is recompiled with all unnecessary functionality disabled
- BIOS and GRUB boot loader password are set

## 5.2 Software

The software used in our study is open source and free with the exception that we are using Matlab which is a commercial product. The most important programs used are: Tcpdump (for capturing packet traffic), CoralReef (for pre-processing the traffic traces) and Matlab (for statistical analysis and visualization of the results).

### 5.2.1 Tcpdump

Tcpdump is argueably the most popular network sniffing tool and it is proven to be very efficient. In addition, the choice of using it was favoured by the fact we can further analyse the traces with a wide range of tools since Tcpdump trace format is very well supported in most of them. Tcpdump offers a way of capturing all packets arriving at the monitored interface by assigning it to promiscuous mode. Typically only the headers of the packets are interesting and Tcpdump can be set to capture only the first N bytes of a packet. By default N is 68 which is enough to cover most used protocol headers, for example Ethernet/IP/TCP and Ethernet/IP/ICMP.

Tcpdump uses PCAP packet capture library to retrieve packets from the network interface. After this the packet passes through BPF (Berkley Packet Filter) which can be used to include or exclude packets by various criteria.

### 5.2.2 CoralReef

CoralReef is a comprehensive open source packet/traffic analysis software developed by CAIDA. It is designed to be a highly configurable and extendible analysis tool. The fact that it is used and tested extensively by researchers in the open source community brings reliablity and credibility to the analysis results.

The flexibility and efficiency of CoralReef is achieved by a 3-level layered design:

1. **RAW API** contains code for physical devices, such as ATM and POS cards and pcap-accessable Ethernet devices. Also a low-level API for these devices is included to offer an interface to the traffic analysis applications.

2. **Traffic applications** include various tools for accessing the raw API and for performing the most frequently used operations, such as tracing individual packets and packet flows as well as giving traffic rate information.

3. **Traffic flow applications** are used to analyse the packet and flow traces. Output can a slightly modified printout of the traces or a user friendly and intuitive HTML report.

Our analysis applies tools on the second layer in the previous listing. The most important feature is the ability to separate and inspect the traffic on a per flow basis. Also, the toolset includes a traffic rate analyser. The first layer features of CoralReef are not needed in this thesis since we use the more commonly appreciated Tcpdump for capturing the packet traffic. The third layer is also not needed because we decided to use Matlab for higher level statistical processing since it is more customizable to meet our requirements. Also, some tool was needed for visualization of the results and since Matlab is well suited for this task it made more sense to leave all the final stages of the analysis entirely to Matlab.

### 5.2.3 Matlab

Matlab is a multipurpose mathematical tool which can be used for performing most tasks requiring high speed numerical calculations and visualization. It also has its own scripting language for automating complicated operations and reducing the need for pre-processing of the input data. The versatility of Matlab is improved by the ability to include special purpose toolboxes, such as the statistical and neural network toolboxes. In principle, they offer unlimited number of ways of usage and also in practice there are many useful toolboxes available for highly specialized purposes.

Matlab is the only commercial tool used in our study. However, it can be replaced by a number of alternative tools which are also free. The decision to use Matlab was mostly influenced by reasons of convenience since we already have a licence and it is still the best tool for our purposes. But basically, any tool which is capable of plotting graphs and performing basic statistical operations and data manipulation can be used.

# Chapter 6

# Results

This chapter presents a snapshot of the measurements needed for the creation of the traffic models and the performance analysis of the network. The total amount of results is too vast to be included here but enough examples are included for understanding the relevant properties of the traffic and the underlying network.

A core network with measurement points set up on each PoP was studied. The internal structure and topology of the network is unknown. Only the traffic entering the network via the PoPs is studied and included in the results; the exiting traffic is only examined when calculating the network performance paremeters: one-way packet latency and packet loss.

## 6.1 Traffic Rate

Traffic rate tells how much traffic is sent into the core network.

### 6.1.1 Week

Network traffic rate on a period of one week (39/2004) is shown in figure 6.1. It tells how much traffic is sent to the core network on the 10 PoPs monitored. It can be seen that the weekdays exhibit similar traffic behavior to each other with high variability in slow and busy times. On Saturday and Sunday the network usage is minor throughout the day. The network is never fully silent; instead a constant traffic rate between 0.5–1 Mbps is maintained throughout the week. At busy times the rate reaches values of 14 Mbps. The graph is plotted average rate in 20 minute interval.

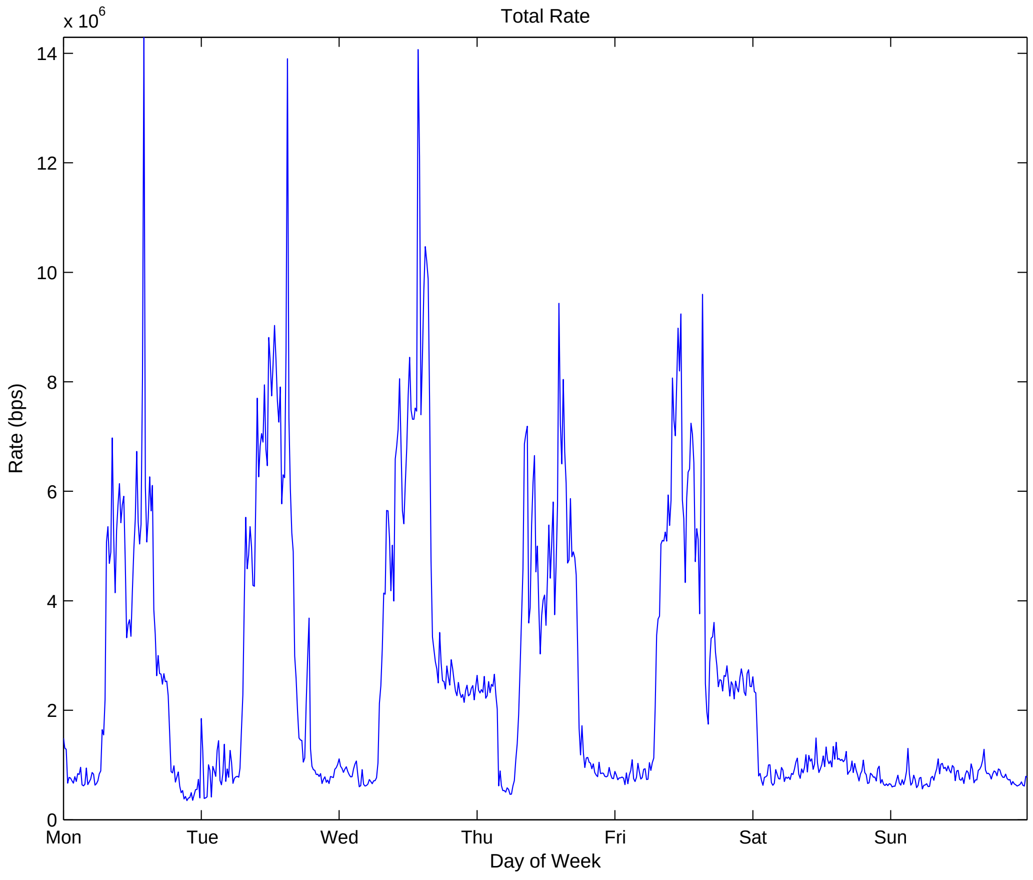

Figure 6.1: Network traffic rate on a one week period.

Network usage is unusually high on Wednesday and Friday evenings, as seen in figure 6.1. The rate remains high well to the next day and then drops suddenly. By examining the traffic rates at each PoP in figure 6.2 we can see that the cause is a locally high network usage between PoPs $A$ and $B$. It is also evident, that the traffic differs noticeably between PoPs. The last three PoPs — $H$, $I$ and $J$ — exhibit distinctly different traffic characteristics compared to the the other PoPs. The traffic rate in PoP $H$ is relatively stable and low at all times except for a high peak at a certain time in the working days.

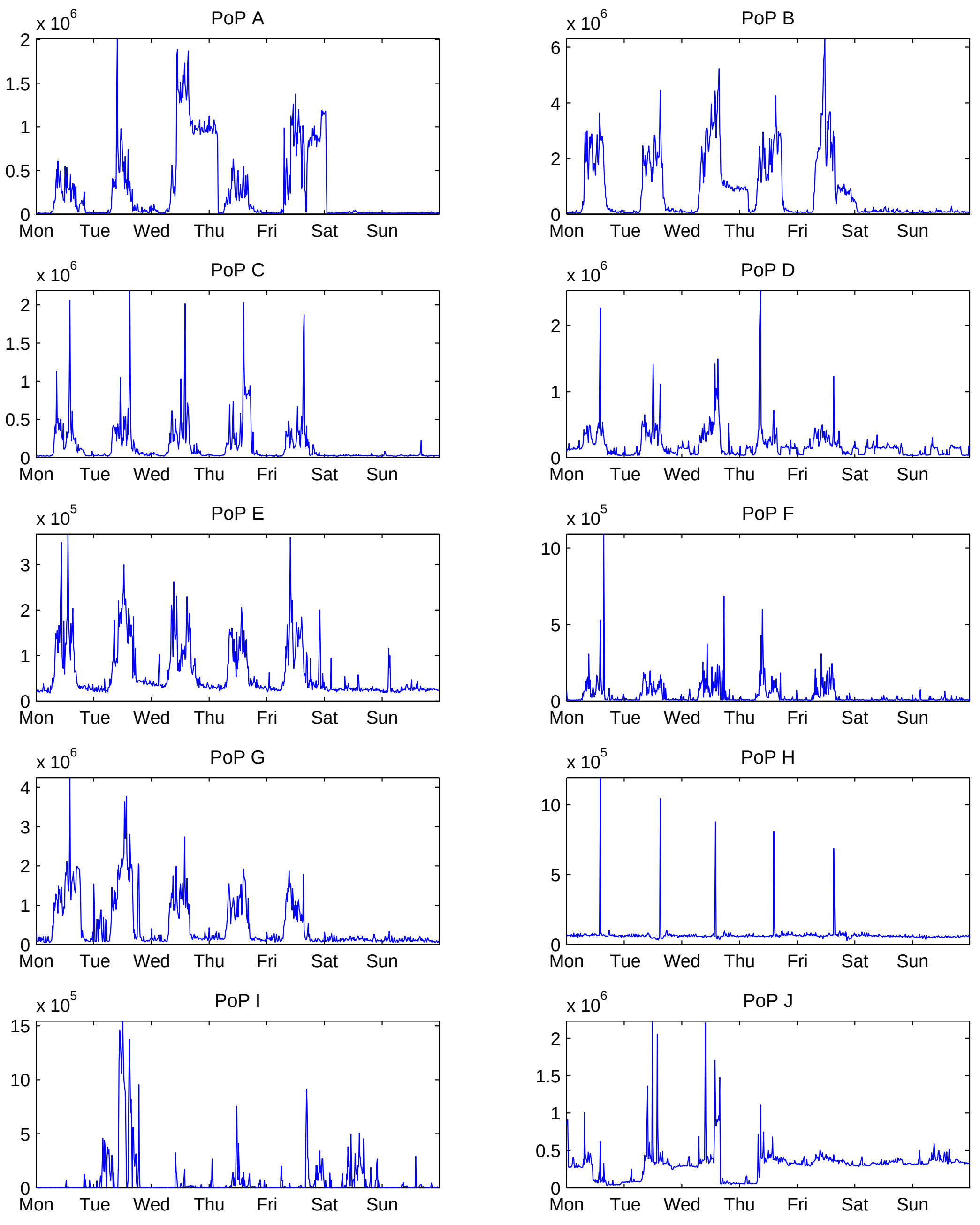

Figure 6.2: Weekly traffic rates on the links directed to the network in each PoP.

### 6.1.2 Day

Network traffic rate on a period of one day (Sep 21st, 2004) is shown in figure 6.3. The day selected for closer inspection was Tuesday since it demonstrates smooth overall traffic rate without major anomalies, as can be seen in figure 6.1. The choice to focus on working days was obvious since they contain the most traffic to be analyzed. The network is most stressed under the busy hour on a working day so it is most informational to analyze the network under these circumstances.

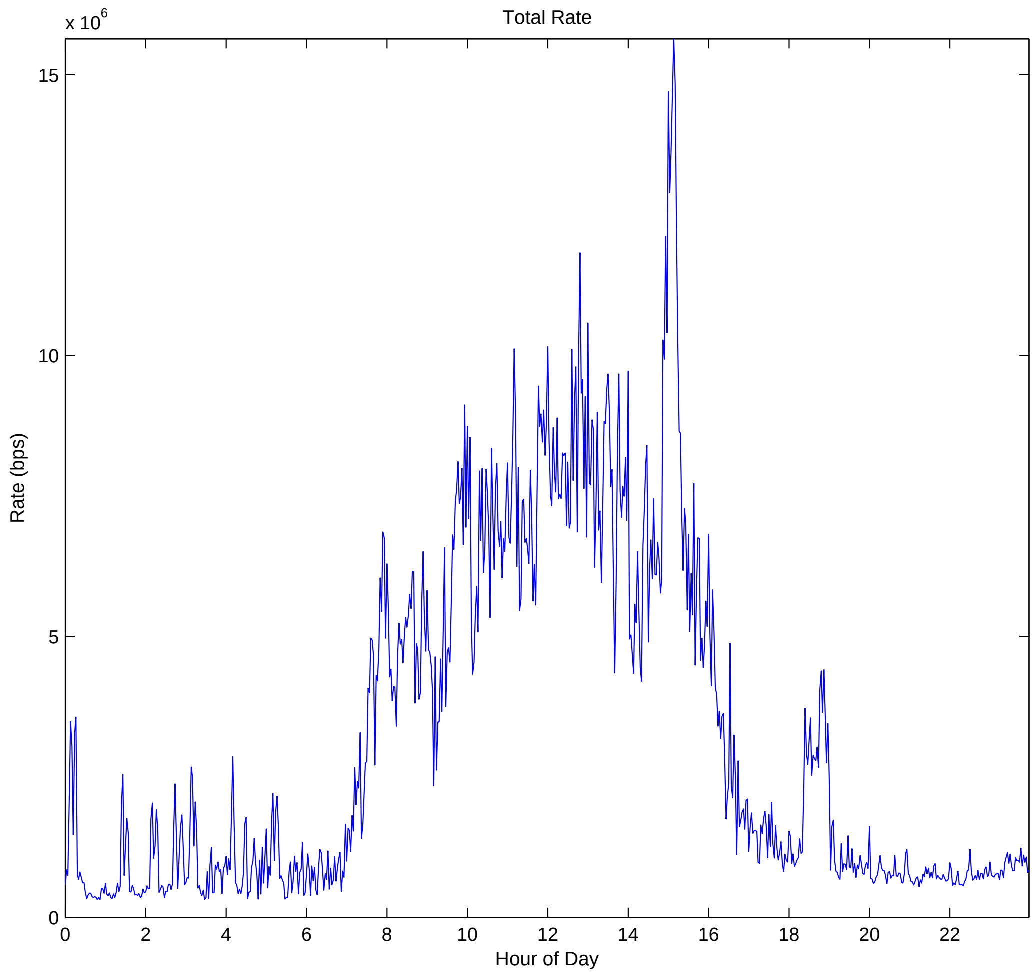

Figure 6.3: Network traffic rate on a one day period (Tuesday)

The traffic rate graph in figure 6.3 shows high rise in network usage between 7:00 and 8:00. The rate falls equally fast after office hours between 16:00 and

17:00. There are two noticeably high peaks: one at 15:00 and the other between 18:00 and 19:00. The source of the peaks can be inspected in figure 6.4. Almost every PoP has some increased activity at 15:00. However, PoP $H$ has approximately 20 times as much network usage then as on any other time of the day. Most likely there is a wide triggered event effecting most of the network. The second peak — occurring between 18:00 and 19:00 — is more local and occurs only in two PoPs: $G$ and $I$.

The network traffic rate graph separated by PoPs and averaged for each hour is presented in figure 6.5. Busy and slow hours are easily discovered in the graph:

- Busy hour: 12:00 – 13:00
- Slow hour: 1:00 – 2:00

Each PoP's share of the total traffic volume at each hour of the day is illustrated in figure 6.6.

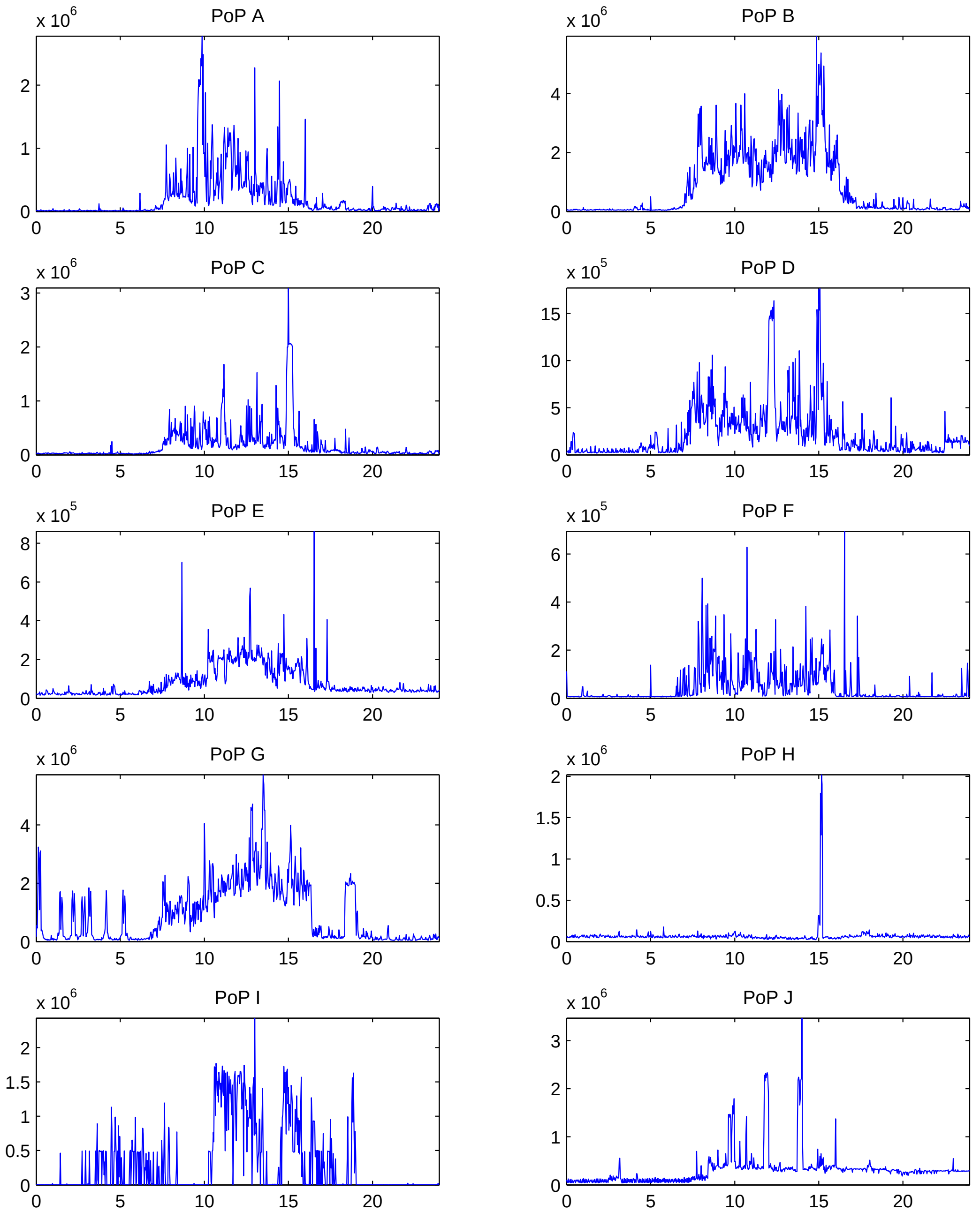

Figure 6.4: Daily traffic rates on the links directed to the network in each PoP.

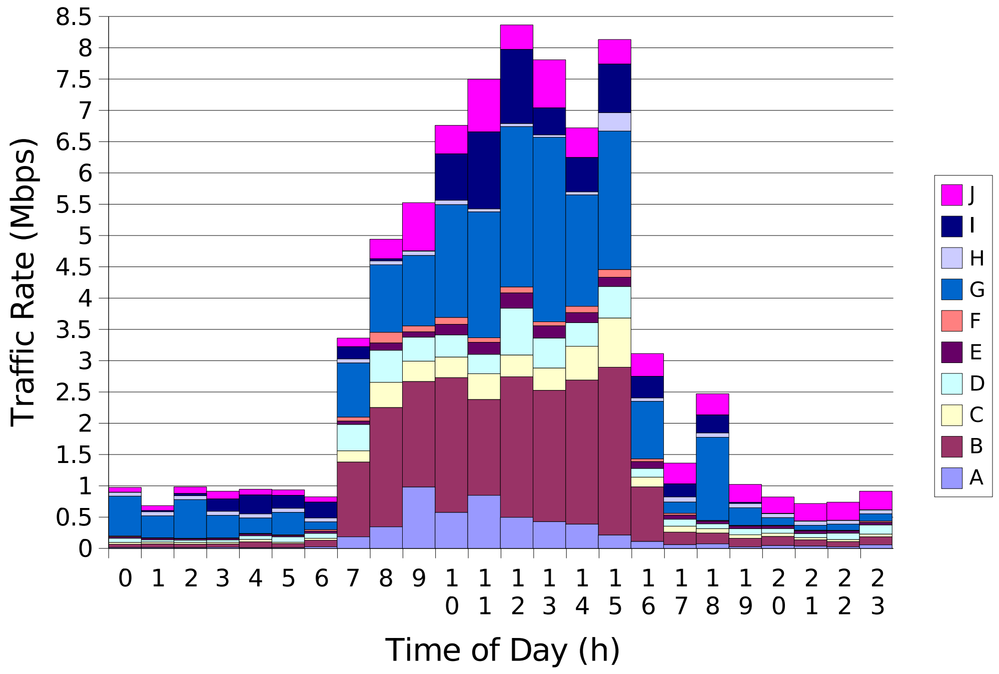

Figure 6.5: Cumulated network traffic rate on each hour of day.

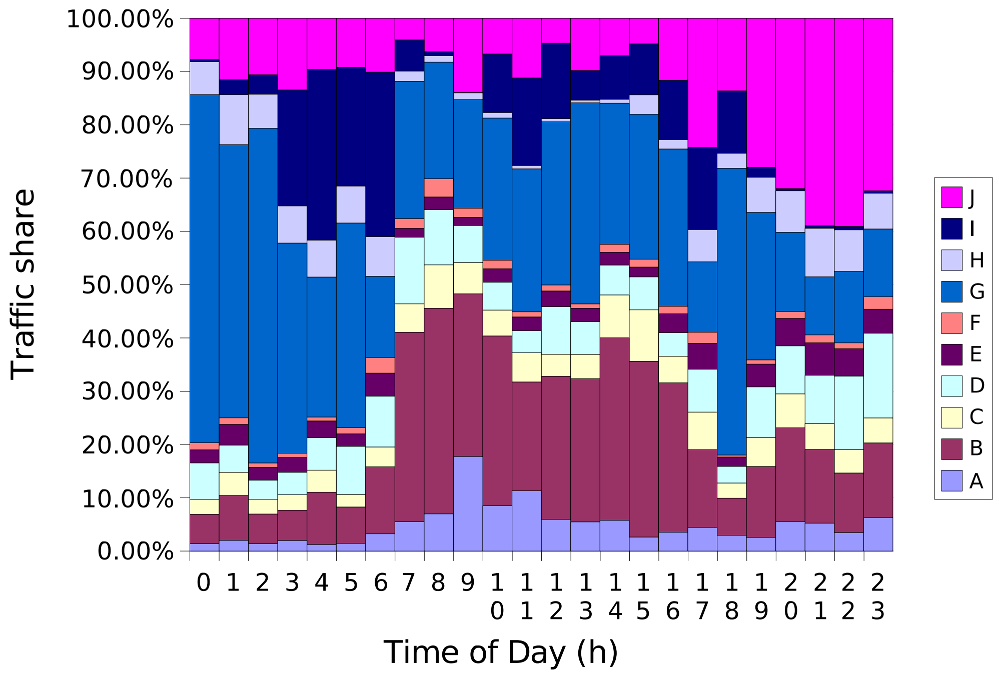

Figure 6.6: Each PoP’s share of total network traffic volume during the day.

## 6.2 Traffic Composition by Protocol

Traffic composition by protocol is examined on two layers: transport and application.

### 6.2.1 Transport Protocols

The share of traffic per transport protocol is seen in figures 6.7 (busy hour) and 6.8 (slow hour). A total of 6 transport protocols were found.

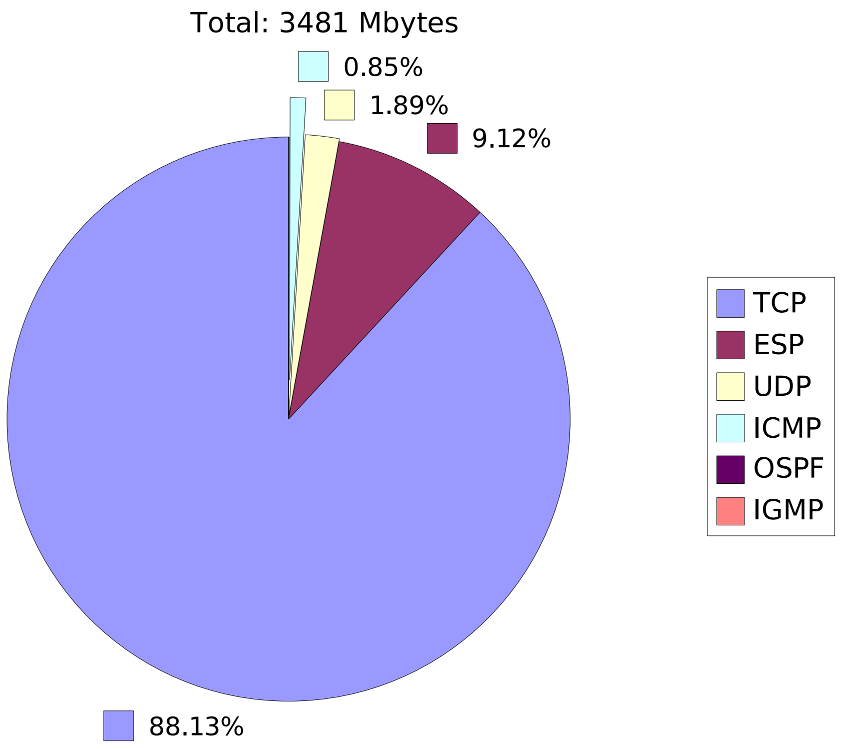

Figure 6.7: Composition of traffic by transport protocol in the busy hour

The volume of busy hour traffic (seen in figure 6.7) is comprised mostly ( $>$ 99 %) of TCP, ESP and UDP traffic. TCP is the dominant protocol constituting 88 per cent of the total load. ESP takes approximately 9 and UDP 2 per cent of the total share. Protocols ICMP, OSPF and IGMP are irrelevant when examining the network traffic purely quantitatively. The total volume of data in the busy hour was 3481 megabytes.

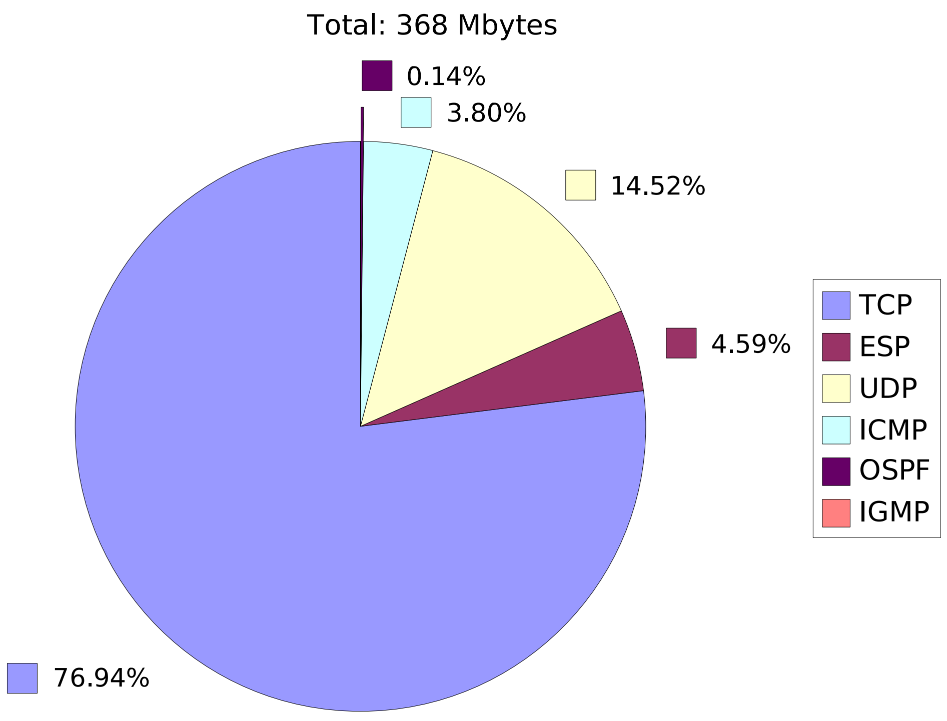

Figure 6.8: Composition of traffic by transport protocol in the slow hour

The volume of slow hour traffic (seen in figure 6.8) shows a little more diversity compared to the busy hour. TCP is still clearly the dominant protocol with a share of 77 per cent of the total traffic. However, ESP's share has halved compared to the busy hour while ICMP's share has risen to nearly 4 per cent of the traffic. The change in UDP traffic is the most apparent: the 15 per cent share in the slow hour is almost 8-fold compared to the busy hour. The volume of UDP traffic in the busy hour is $0.0189 \cdot 3481\text{MB} \approx$ 65.8 MB while in the slow hour the corresponding figure is $0.1452 \cdot 368\text{MB} \approx$ 53.4 MB. The difference is relatively small ($65.8\text{MB}/53.4\text{MB} \approx$ 1.2) compared to the difference between the total busy and slow hour traffic volumes ($3481\text{MB}/368\text{MB} \approx$ 9.5). This suggests that most of the UDP data is background traffic and independent of user activity.

### 6.2.2 Application Protocols

The shares of 10 most used application protocols are presented in figures 6.9 (busy hour) and 6.10 (slow hour). The application protocol information is derived based on the port numbers so only TCP and UDP applications are discovered. The application is deduced according to the logic presented in section 3.1.3.

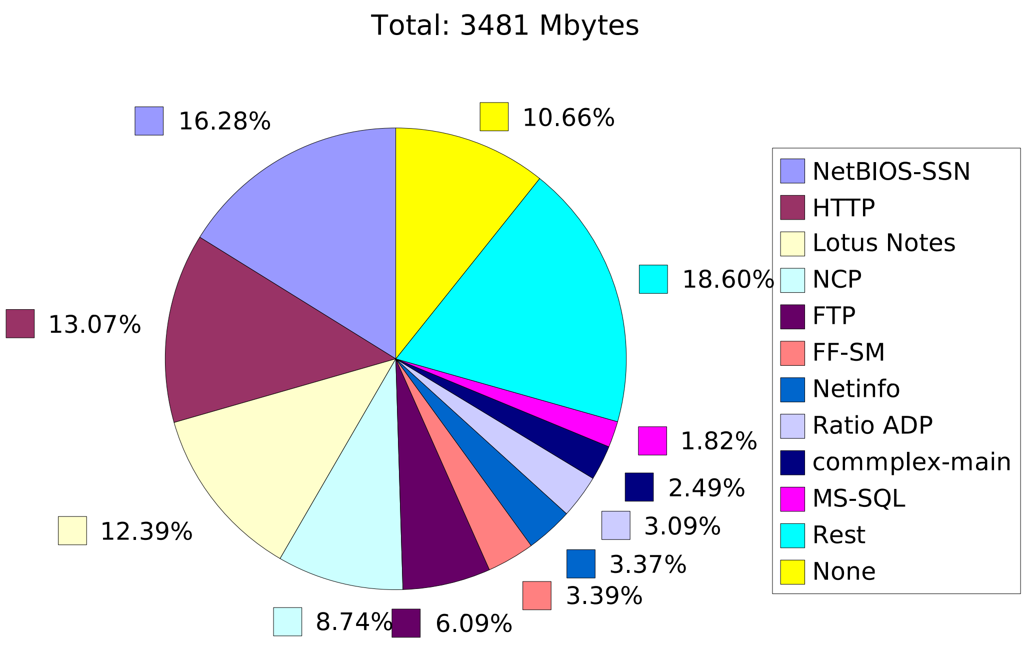

Figure 6.9: Composition of traffic by application protocol in the busy hour

The five most used application protocols in the busy hour (seen in figure 6.9) are NetBIOS (Network Basic Input/Output System), HTTP (HyperText Transfer Protocol), Lotus Notes, NCP (NetWare Core Protocols) and FTP (File Transfer Protocol). Together they form well over half of the total network traffic. The five remaining applications in the graph form about one fifth of the traffic. Approximately the same amount falls outside of the 10 most used applications (in class '*Rest*'). The class *None* represents mostly non-TCP and non-UDP traffic.

Application protocol statistics in the slow hour are shown in figure 6.10. Surprisingly, HTTP is most common protocol even though one would not expect so

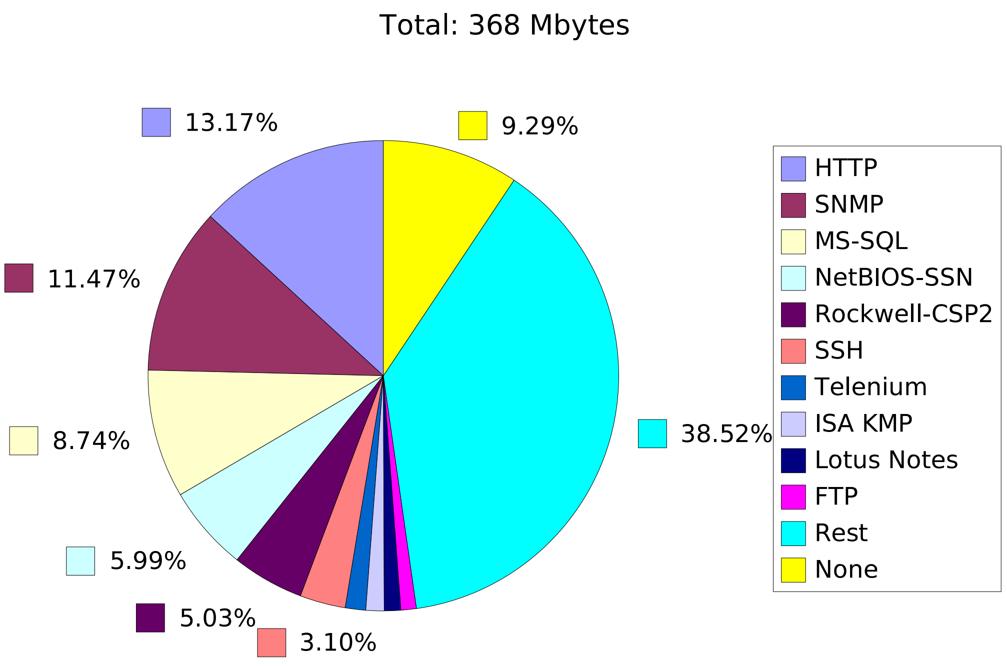

Figure 6.10: Composition of traffic by application protocol in the slow hour

much user activity in the slow hours. The next two protocols — SNMP (Simple Network Management Protocol) for configuring network components, and MS-SQL (Microsoft Standard Query Language) for updating and synchronizing databases — are more typical slow hour applications. Together, the three top applications form approximately one third of the total traffic. Most noticeably, compared to the busy hour, the amount of traffic in the class *Rest* has risen considerably. This means that the network usage is distributed more evenly in the slow hour compared to the busy hour.

### 6.2.3 Suitability for Traffic Modeling

The conclusion from the transport and application protocol inspection is the way the traffic should be divided into smaller components for implementing the traffic generators (explained in section 4.4).

Application protocols divide the traffic into smaller components, compared to transport protocols. By using them, the traffic model is potentially more precise and accurate at the cost of simplicity. Without aggregation, the number of traffic generators required is equal to the number of protocols included in the model. Thus, by using application protocols, a little less than 75 per cent of the traffic can be produced with 10 traffic generators. In the slow hour, the 10 traffic generators will produce only a little more than 50 per cent of the total traffic.

On the other hand, there are only 6 transport protocols discovered in the network. By using only three traffic generators for creating traffic of the three most used transport protocols — TCP, ESP and UDP — approximately 99 per cent of the total network traffic can be generated in the busy hour and 96 per cent in the slow hour. The cost is the accuracy of the resulting traffic model.

Based upon these findings, three traffic generators are created and protocols TCP, UDP and ESP are used for the basis of the traffic model.

## 6.3 Traffic Model

The traffic model is created either on the flow level or on the packet level — which ever better suits the characteristics of each protocol. The data is inspected here for each PoP separately and traffic parameters are discovered for each transport protocol within a PoP. However, in the final traffic model traffic should be inspected for each PoP pair separately for gaining full traffic generation information between each sending and receiving PoP in the network.

### 6.3.1 Choosing Between Flow and Packet Based Model

Two distributions are measured for the three protocols (TCP, UDP and ESP): interarrival time and size. However, the choice of using a flow or packet based approach should first be investigated. If the mean flow size in packets is high it is reasonable to use flows; if there are generally only a few packets per flow the aggregation by using flows does not bring any benefits and it is more sensible to use the packet level to gain better accuracy in the traffic generation.

The busy hour CDF of flow size in packets is presented in figure 6.11. It is seen that TCP protocol has only few flows with size of only one packet. The CDF function rises quite smoothly and the percentage of flows with more than 100 packets is higher than 5 per cent. For TCP, the flow based approach will be used.

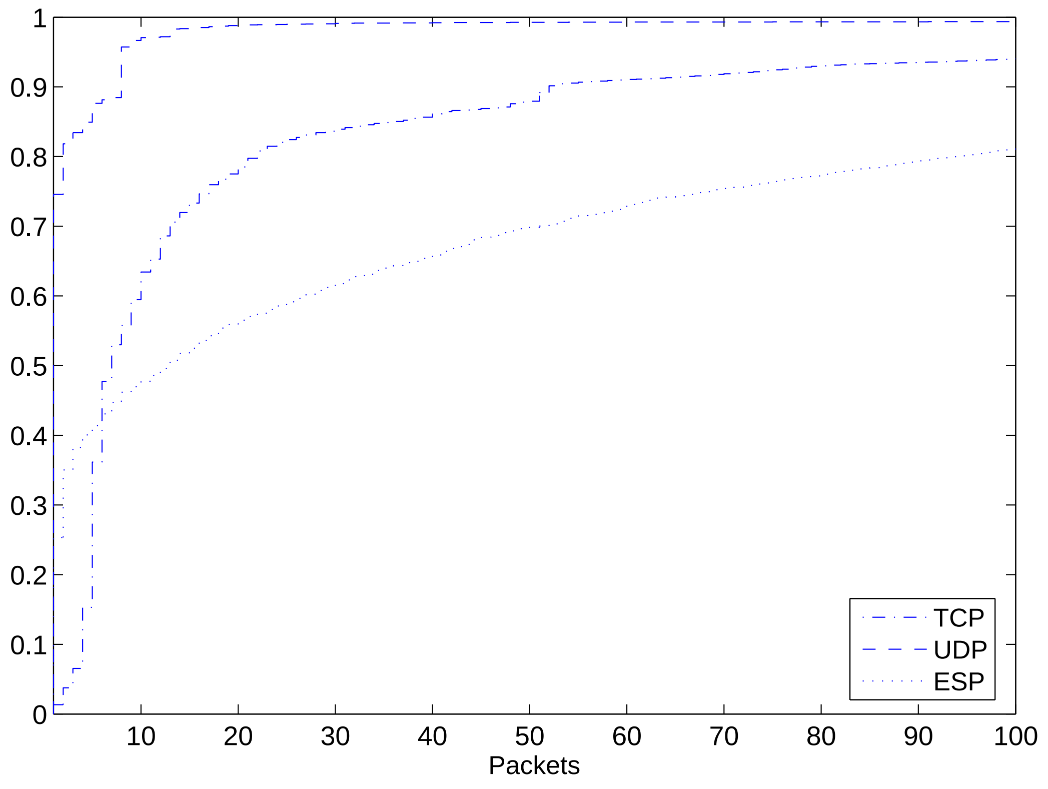

Figure 6.11: CDF of the number of packets per flow for the three protocols: TCP, UDP and ESP.

With UDP, on the other hand, 75 per cent of the flows contain only one packet. Less than 5 per cent of UDP flows contain more than 8 packets and 1 per cent of the flows more than 25 packets. There are no benefits in using the flow model for UDP so the packet level approach will be used.

The choice of approach on ESP is more complex. When examining the number of small flows (less than 5 packets) ESP falls between TCP and UDP, as seen in figure 6.11. More than 25 per cent of the flows contain only 1 packet, suggesting the use of packet level approach. On the other hand, the CDF of ESP has even a flatter tail than TCP showing that it has many large flows as well: almost 20 per cent of the flows contain more than 100 packets. The difficulty is that ESP is an encryption protocol which does not contain a flow control mechanism of its own.

Typically it is used to setup encrypted VPN tunnels (Virtual Private Network) which may carry any kinds of flows, including TCP and UDP. The flow control is hidden from the intermediate observer. Because of this ambiguity a safe packet level approach is chosen for ESP.

### 6.3.2 Flow Based Model

#### Flow Inter-arrival Time

The inter-arrival times of the sample data in a single PoP and different distributions fitted to the data are presented in figure 6.12. The data in the rest of the PoPs is similar to the one presented here. The ACK flows have been excluded from the data, as explained in section 4.4.1.

Exponential distribution has the worst fit to the data. Pareto distribution is more accurate for the small inter-arrival time values but fails to follow the steeper descent of the data curve after approximately 2 seconds. Gamma and Weibull distributions follow the data quite accurately but the combination of Pareto and Exponential distributions fits the data most precisely.

The root mean square errors (RMSE) between the distributions and the data for all PoPs are shown in table 6.1. The mean RMSE describes how suitable the distribution is to model the data. Exponential distribution clearly has the highest mean RMSE. Pareto, Gamma and Weibull distributions form an evenly good choice of candidates with the slight advantage on the latter. However, the combination of Pareto and Exponential distributions presents clearly the most accurate fit on the data. In all cases but one (PoP $H$) the Pareto-Exponential distribution has the lowest RMSE.

#### Flow Size

Flow size in bytes measured in a single PoP and the distributions fitted to the data is presented in figure 6.13. The graph is plotted using logarithmic x-axis and linear y-axis. The ACK flows have been excluded from the data.

The shape of the data is distinctly different compared to the inter-arrival time data. There is a low concentration of values in the small flow sizes. A peak con-

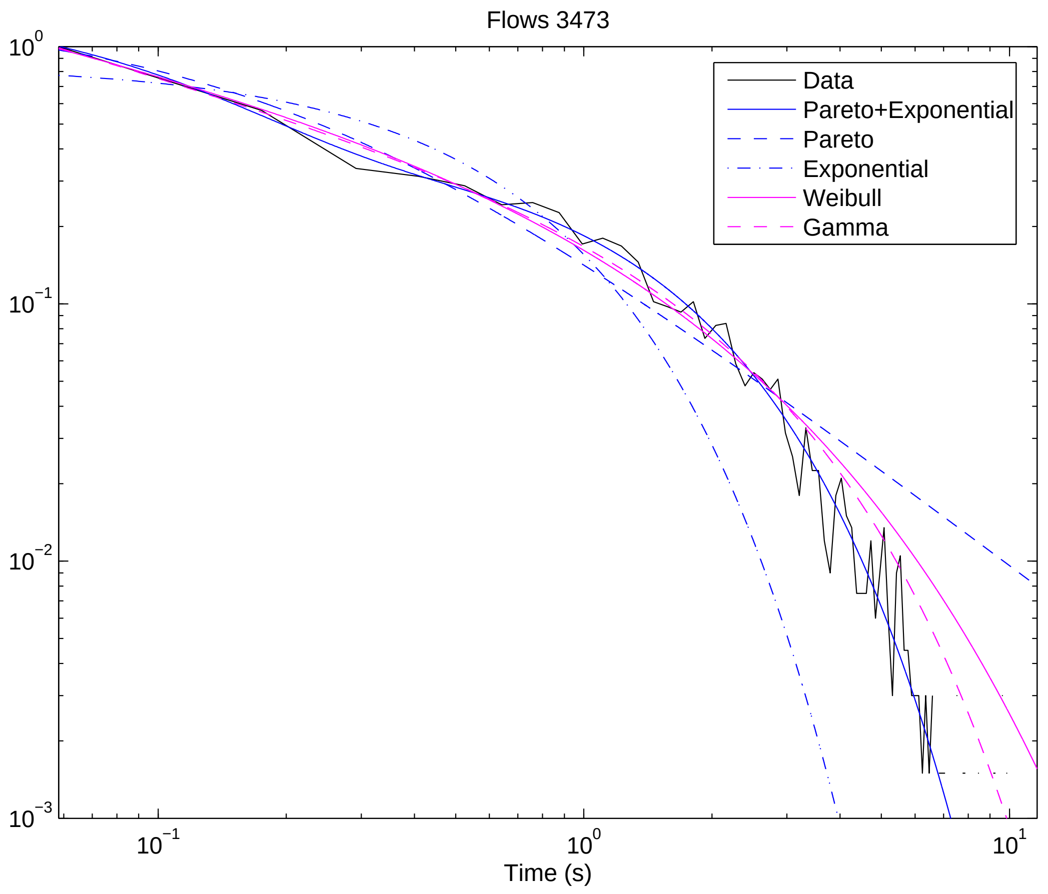

Figure 6.12: Inter-arrival time distribution of TCP flows during one hour in a single measurement point and different distributions fitted to the data (log-log scale).

centration occurs approximately at 1000 bytes after which the tail gradually diminishes with the exception of another peak at around 8000 bytes.

Modeling the flow size with the given distributions seems more inaccurate than modeling inter-arrival time. Exponential and Pareto distributions applied separately can only increase or decrease monotonically. Both find the shape of the data quite well with the exception of the lower values before the first peak. However, the linear combination of the two distributions can contain a local maximum and the result fits the data most accurately of all the five distributions tested here. The first peak at about 100 bytes in the x-axis is explained by the fact that absolute value of the combination distribution is taken to exclude negative values. Pareto and gamma distributions find the peak value of the data quite well but the tail is

| PoP | Exponential | Pareto | Weibull | Gamma | Pareto+Exponential |
|---|---|---|---|---|---|
| A | 21.7 | 8.3 | 10.9 | 14.7 | 1.2 |
| B | 21.1 | 4.6 | 2.8 | 1.6 | 1.6 |
| C | 33.3 | 15.4 | 11.7 | 10.0 | 5.6 |
| D | 38.9 | 20.8 | 13.9 | 12.0 | 8.2 |
| E | 12.3 | 4.2 | 9.3 | 11.6 | 1.4 |
| F | 36.1 | 16.3 | 16.0 | 15.2 | 8.5 |
| G | 26.1 | 12.7 | 10.0 | 9.2 | 6.3 |
| H | 73.1 | 73.1 | 60.7 | 64.6 | 72.9 |
| I | 20.1 | 17.4 | 18.0 | 19.8 | 13.3 |
| J | 32.2 | 12.1 | 11.9 | 11.2 | 5.9 |
| Mean | 31.5 | 18.5 | 16.5 | 17.0 | 12.5 |

Table 6.1: Root mean square errors (ms) of different distributions fitted to flow inter-arrival times (s)

lost in both cases.

Root mean square errors between the data and the distributions is presented in table 6.2. The linear combination of Pareto and Exponential distributions is overall the best choice when comparing the mean RMSEs. However, there are more cases when the choice here is not as obvious as it was with flow inter-arrival time fitting; in PoPs $D$, $E$, $F$ and $I$ the optimal choice is not Pareto-Exponential distribution. Weibull and Gamma distributions offer a better fit if measured with RMSE. Possibly, more than one distribution should be used in the traffic model to gain most accurate results.

### 6.3.3 Packet Based Model

#### Packet Inter-arrival Time

Packet inter-arrival times and the distributions fitted to the data are presented in figures 6.14 (UDP) and 6.15 (ESP). The behavior of data and distributions is similar compared to the flow inter-arrival times. However, the data for packet inter-arrival times is noisier. Also, there are higher peaks in the packet inter-arrival times making the distribution fitting harder. Peaks are to be expected because UDP protocol is often used for creating VoIP channels and the inter-arrival time of packets in such a channel is fixed.

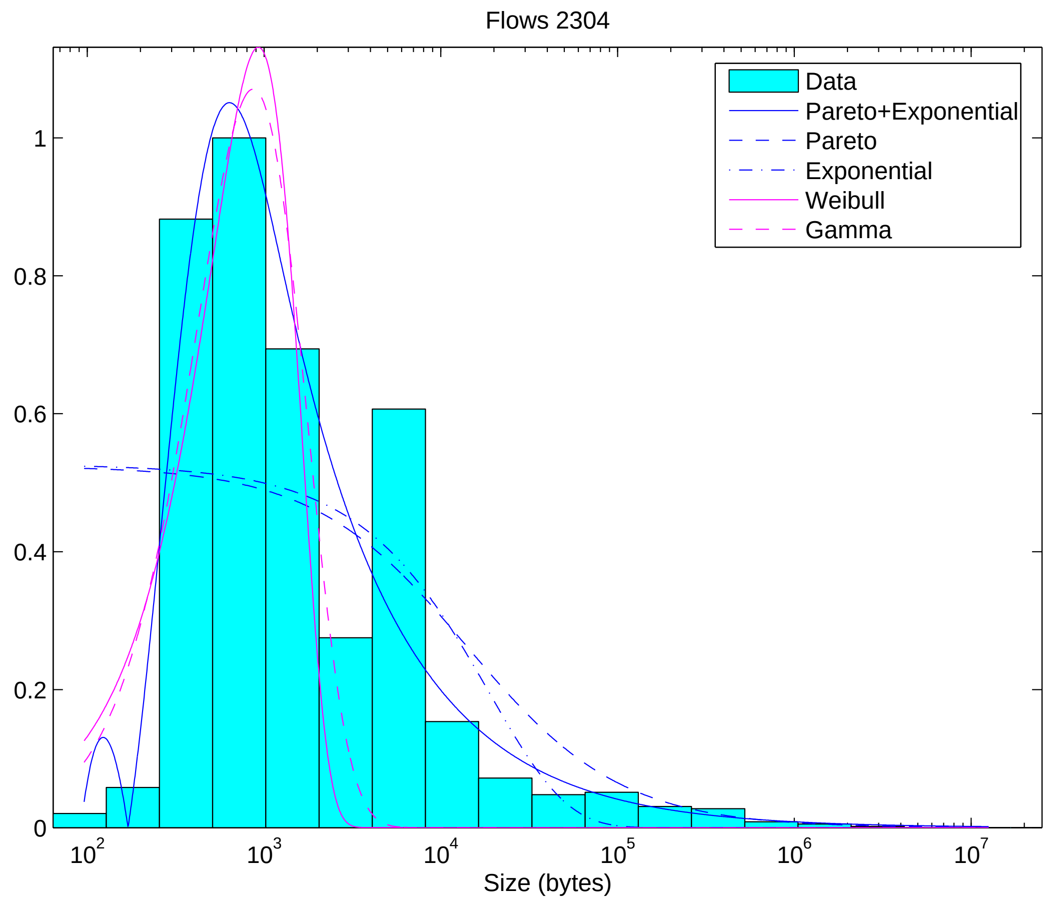

Figure 6.13: Size distribution of TCP flows during one hour in a single measurement point and different distributions fitted to the data (log-linear scale).

Root mean square errors of the different distributions fitted to the UDP packets is presented in table 6.3. Once again the linear combination of Pareto and Exponential distributions proves superior to the others. It has the lowest mean RMSE as well as the lowest RMSEs for each PoP individually. Gamma distribution performs surprisingly badly considering how well it worked with the flow data — for both inter-arrival times and sizes. However, the situation is better than what could be determined by the mean RMSE because the value for Gamma distribution is crippled by two extremely bad fits compared to the other distributions: in PoPs $D$ and $F$.

The corresponding root mean square error table for ESP packets is shown in

| PoP | Exponential | Pareto | Weibull | Gamma | Pareto+Exponential |
|---|---|---|---|---|---|
| A | 204.8 | 204.8 | 148.7 | 134.7 | 63.9 |
| B | 217.3 | 216.8 | 188.7 | 183.8 | 114.9 |
| C | 225.4 | 225.1 | 196.1 | 195.8 | 178.1 |
| D | 176 | 176 | 156.6 | 152.6 | 174.6 |
| E | 233.5 | 198.5 | 48.6 | 49.6 | 102.1 |
| F | 165.3 | 165.3 | 160.4 | 159.5 | 164.7 |
| G | 254.7 | 253.4 | 213.5 | 209.4 | 134.5 |
| H | 212.8 | 212.8 | 236.9 | 236.9 | 164.7 |
| I | 274.4 | 255.6 | 213.9 | 216.9 | 246.4 |
| J | 235.0 | 237.9 | 184.8 | 173.1 | 90.0 |
| Mean | 219.9 | 214.6 | 174.8 | 171.2 | 143.4 |

Table 6.2: Root mean square errors (ms) of different distributions fitted to flow sizes (B)

| PoP | Exponential | Pareto | Weibull | Gamma | Pareto+Exponential |
|---|---|---|---|---|---|
| A | 4.6 | 3.5 | 4.0 | 4.4 | 1.2 |
| B | 10.0 | 11.9 | 8.4 | 8.9 | 5.6 |
| C | 22.4 | 14.5 | 14.7 | 17.5 | 12.6 |
| D | 2.9 | 2.5 | 2.6 | 13.6 | 2.4 |
| E | 10.0 | 5.8 | 6.5 | 9.0 | 2.8 |
| F | 5.1 | 4.8 | 5.1 | 130.2 | 1.5 |
| G | 12.3 | 5.9 | 5.6 | 8.3 | 5.3 |
| H | 8.1 | 7.3 | 7.7 | 8.0 | 5.2 |
| I | 19.6 | 17.8 | 18.8 | 19.5 | 3.6 |
| J | 6.9 | 5.7 | 6.3 | 6.8 | 1.2 |
| Mean | 10.8 | 8.5 | 8.4 | 24.6 | 4.5 |

Table 6.3: Root mean square errors (ms) of distributions fitted to the inter-arrival times of UDP packets.

figure 6.4. The comparison of mean RMSEs is more even than with the corresponding UDP packets. The Pareto-Exponential distribution still outperforms all others with both its mean and individual RMSEs. Only in one case — PoP $C$ — have three other distributions performed better. Exponential distribution is once again the worst candidate.

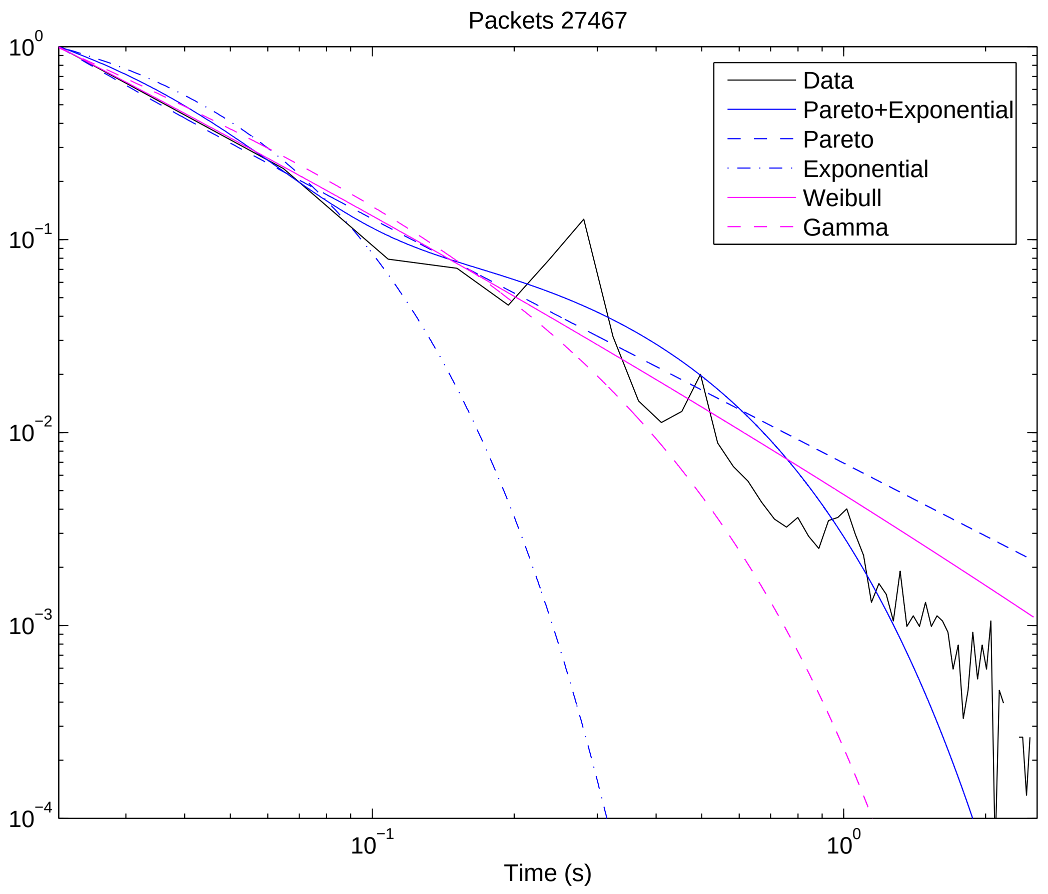

Figure 6.14: Packet inter-arrival time distribution of UDP traffic during one hour in a single measurement point and the Pareto-Exponential -distribution fitted to the data.

### Packet Size

The distributions fitted in the previous data sets can not be applied with packet size because the packet size data does not behave smoothly. Protocols have few discrete packet sizes of which most of the traffic consists of. Hence, only discrete packet sizes are chosen for both UDP and ESP and the traffic model will deploy only packets of these sizes. The following remarks are made on the packet size CDF of the whole network — better accuracy would be acquired by inspecting each PoP separately.

The packet size CDF for each transport protocol in the network during the busy

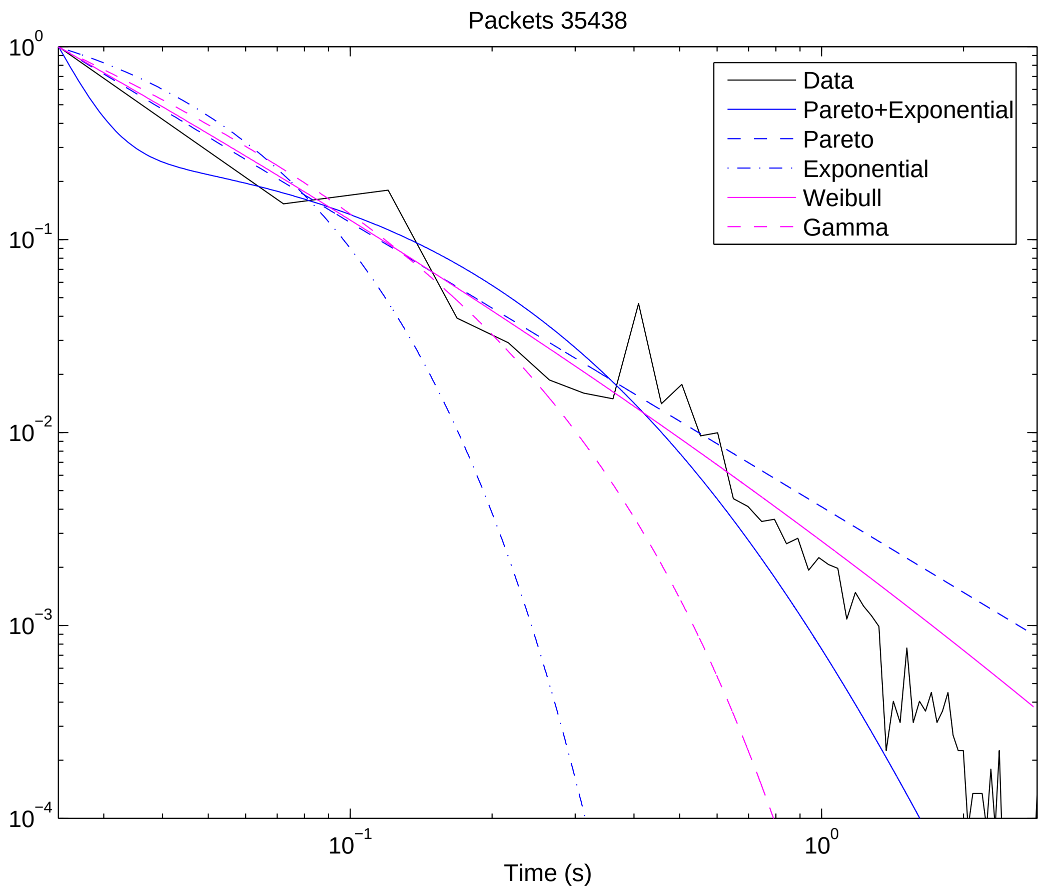

Figure 6.15: Packet inter-arrival time distribution of ESP traffic during one hour in a single measurement point and the Pareto-Exponential -distribution fitted to the data.

hour is presented in figure 6.16. UDP has two small regions where the packet sizes are concentrated. The first one is between 60–110 bytes: more than 30 per cent of the UDP packets are of this size. The second one is between 130–175 bytes: approximately 55 per cent of the packets are of that size. About 5 per cent of the packets are 1500 bytes in size and the remaining 10 per cent is mostly between 175–500 bytes. By generating packets of size 85, 150, 340 and 1500 in the ratio of 30:55:10:5 respectively reasonable accuracy should be acquired.

Packet size estimation for ESP is more inaccurate since the values are not as discretely located as with UDP. Instead, a smoother curve is seen for ESP in figure 6.16, resembling more the curve of TCP. It also resembles TCP curve in

| PoP | Exponential | Pareto | Weibull | Gamma | Pareto+Exponential |
|---|---|---|---|---|---|
| A | 36.2 | 29.5 | 28.9 | 28.7 | 26.1 |
| B | 11.5 | 3.3 | 3.1 | 6.6 | 0.8 |
| C | 7.2 | 29.9 | 6.1 | 6.4 | 9.6 |
| D | 21.4 | 13.8 | 14.1 | 16.4 | 11.9 |
| E | 7.3 | 2.9 | 3.5 | 5.8 | 1.8 |
| F | 6.0 | 6.0 | 5.9 | 5.8 | 4.4 |
| G | 5.2 | 0.5 | 0.5 | 2.8 | 0.5 |
| H | 19.1 | 19.1 | 19.1 | 19.2 | 16.8 |
| I | - | - | - | - | - |
| J | 7.7 | 6.5 | 7.2 | 7.4 | 4.4 |
| Mean | 13.5 | 12.4 | 9.8 | 11.0 | 8.5 |

Table 6.4: Root mean square errors (ms) of distributions fitted to the inter-arrival times of ESP packets.

the sense that it has more full size packets than UDP. Still, about 45 per cent of ESP packets are concentrated in the region between 75–125 bytes. The length of an ESP header is at least 10 bytes; deducting the header from the total packet size we see a close resemblance with the UDP packets where a packet concentration was found between 60 -110 bytes. Most likely, ESP is used to carry both UDP and TCP packets which makes the packet size CDF for ESP somewhat harder to generalize. However, after the first concentration of packets between 75–125 bytes, approximately 35 per cents of the packets are distributed quite evenly in the region between 125–1500 bytes. The remaining 20 per cents of the packets are of size 1500 bytes. Thus, ESP packets of size 100, 700 and 1500 bytes in the ratio of 45:35:20 should be generated in the traffic model.

The packet size CDF for the slow hour is presented in figure 6.17. Similarly, representative packet sizes for the protocols are chosen for the slow hour traffic model.

### 6.3.4 Observations on Distribution Fitting

The parameters for the distributions are calculated with Matlab’s non-linear least squares data fitting function: *nlinfit*. The function needs an initial guess of the parameters, an initializer vector, to be entered upon launch. The better the initializer

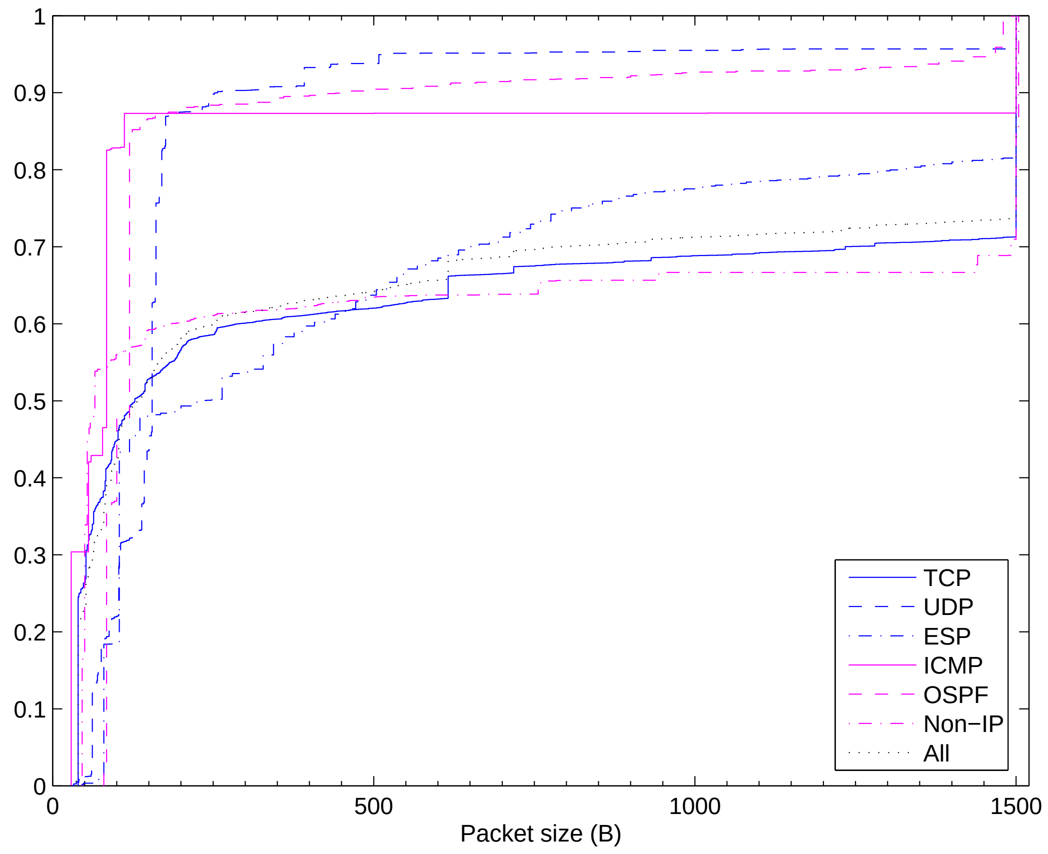

Figure 6.16: Busy hour packet size CDF for each transport layer protocol in the whole network.

vector the more precisely the optimal parameters are discovered. If the number of input samples is small the data is usually quite noisy. A better initializer vector will be required in such cases. The final traffic model which connects each PoP to other PoPs in the network may have a low number of samples between some PoPs. It is therefore suggested, that distribution parameters should first be estimated for the traffic of the whole PoP and these parameters be used as the initializer vector for the traffic sent from the given PoP to other PoPs separately. This procedure provides better chance of convergence of the fitting algorithm.

From the distributions tested here the linear combination of Pareto and Exponential distributions proved most accurate for fitting against the data for all three parameters: flow inter-arrival time and size distributions and packet inter-arrival time distribution. However, the combination distribution is more complex com-

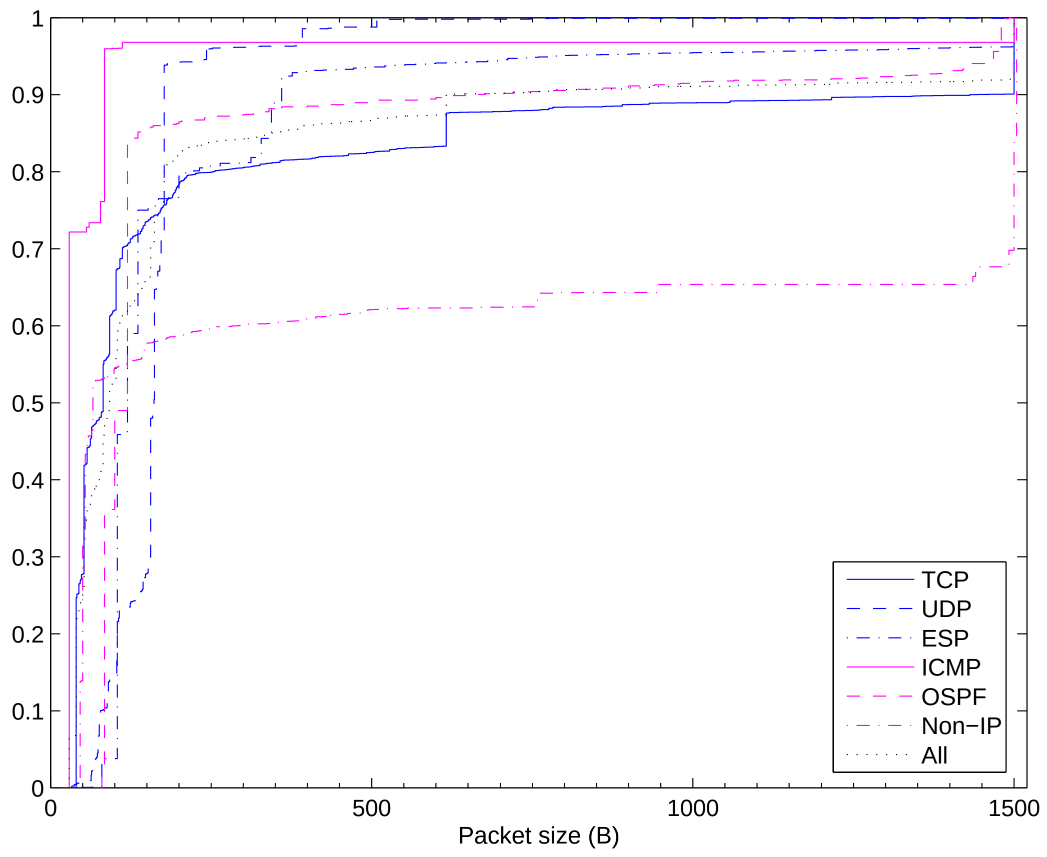

Figure 6.17: Slow hour packet size CDF for each transport layer protocol in a the whole network.

pared to the others and it may be simpler to use the single distributions in some cases. Weibull, Gamma and Pareto are all good candidates for this purpose but the result should be tested more carefully. Exponential distribution should not be used for modeling any of the parameters here.

In any case the goodness of fit should be monitored closely for each data set. A too small data set or high variance in its values may cause the distribution not to converge to the optimal values. The resulting distribution may give totally invalid values or values which are not so obviously faulty. Root mean square error and preferably also the visual fit of the distribution to the data should be observed for detecting faulty cases. In most cases, altering the initializer vector should solve the problem.

### 6.3.5 Cutting the Tail of the Distribution

Flow modeling requires some knowledge of the quantity of traffic sent in flows of different sizes. The distributions implemented in the flow model to represent flow size needs to be cut at some point, at least when using heavy-tailed distributions, as explained in section 3.2.1. An uncut heavy-tailed distribution could quite possibly select randomly an exceedingly large value which would hinder the comparison between consecutive simulations.

Choosing the cut point is not trivial. The flow size in bytes is represented in figure 6.18. ACK flows, i.e. TCP flows with mean packet size less than 60 bytes, are not included in the graphs. Subfigure (a) shows the frequency of flows with size given in the x-axis. Below it — in subfigure (c) — is the same frequency of flows multiplied by the size of the flow. It shows how much bytes (y-axis) do flows of the given size (x-axis) carry in the network. At the right-hand side are the corresponding CDF plots, in subfigure (b) and (d).

The figures show that 96 per cent of the flows are less than 100 kB in size but they carry less than 10 per cent of the total traffic. This demonstrates the concept of mice and elephants, i.e. the network traffic consists mostly of small flows but few very large flows create a large portion of the traffic. Because of this phenomenon it may be necessary to manually implement one or two long lasting flows and use the flow generator only for creating the small flows.

## 6.4 Performance Parameters

Performance parameters describe how the network treats the packet traffic. The parameters chosen for inspection in this thesis are: one way packet latency and packet loss ratio. The loss ratio is simpler to study in the sense that the sample material is binary in nature: packet is either dropped or successfully transmitted. Packet latency can be inspected more thoroughly and different ways to observe it gives valuable insight into the behavior of the network.

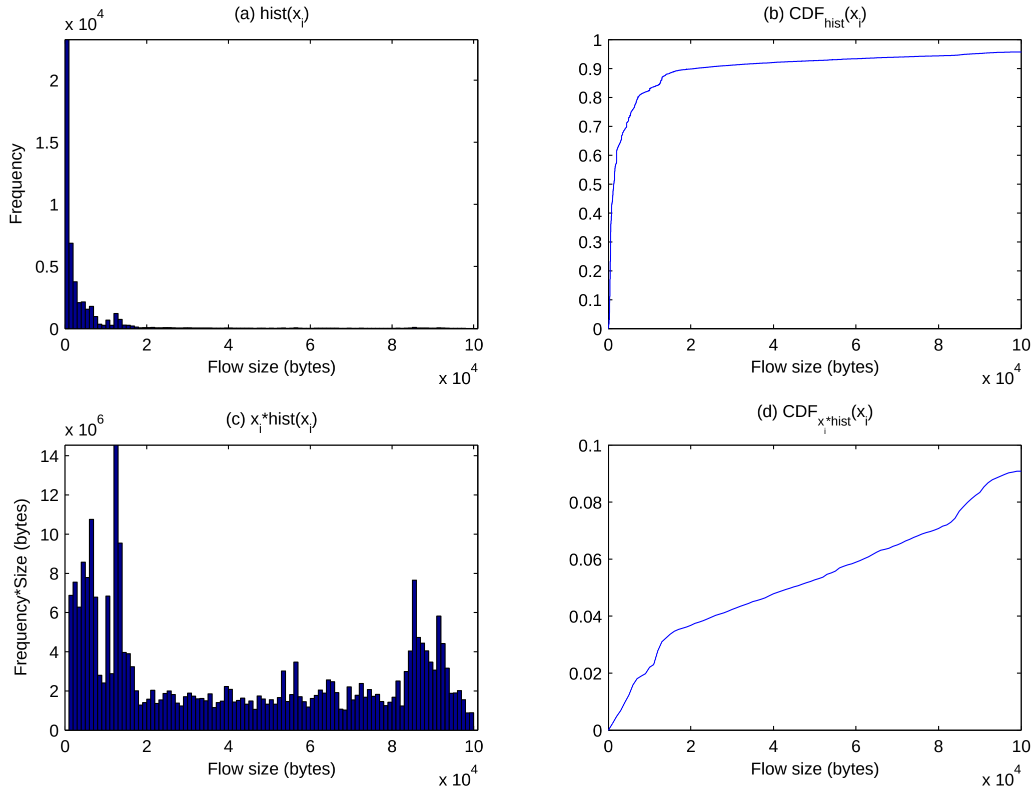

Figure 6.18: TCP flow size (in bytes) in the whole network with ACK flows not included. Figure (a) is the histogram which is multiplied by $x_i$ in figure (c). Figures in the right-hand side ((b),(d)) are the corresponding CDF plots.

### 6.4.1 Splitting Traffic Between PoPs

The performance parameters are inspected pair-wise between each PoP in the network. The method of splitting the traffic is explained in section 4.5.2. However, the problem in our measurements was that the subnets behind each PoP were not known. Instead, we had to try to discover the subnets by listening in on the traffic originating from each PoP and studying the IP addresses seen. Another problem was, that the measurement infrastructure did not cover all entry and exit points in the network. Much of the traffic was seen entering the network but were not seen exiting it. It clearly showed that there were many leak points in the network which were not monitored by our system. However, enough of the traffic for was successfully split for the analysis and the resulting traffic matrix of transmitted packets between PoPs is presented in table 6.5.

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| A | - | 147751 | 72607 | 3241 | 1962 | 20 | 31321 | 2289 | - | 104 |
| | (-) | (17935) | (8435) | (12) | (669) | (-) | (408) | (419) | (-) | (26) |
| B | 291181 | 459873 | 137677 | 108404 | 179774 | - | 11820 | - | - | - |
| | (29199) | (67704) | (5687) | (12449) | (31661) | (-) | (1528) | (-) | (-) | (-) |
| C | 70932 | 123902 | - | 9149 | 56097 | - | - | - | - | - |
| | (13135) | (12763) | (-) | (9440) | (22593) | (-) | (-) | (-) | (-) | (-) |
| D | 2247 | 66230 | 9228 | 46 | 55953 | 10104 | 12799 | 19072 | - | - |
| | (12) | (3355) | (8857) | (42) | (38356) | (2455) | (43) | (18192) | (-) | (-) |
| E | 1969 | 84917 | 49839 | 127587 | - | - | 12914 | - | - | - |
| | (664) | (19860) | (6350) | (57001) | (-) | (-) | (2169) | (-) | (-) | (-) |
| F | 21 | - | - | 6999 | - | - | 2001 | 1073 | - | - |
| | (-) | (-) | (-) | (2194) | (-) | (-) | (11) | (1746) | (-) | (-) |
| G | 24098 | 11098 | - | 16010 | 13289 | 13286 | 4 | 1907 | 6 | 46960 |
| | (407) | (1459) | (-) | (32) | (1790) | (1616) | (3) | (153) | (-) | (13342) |
| H | 2870 | - | - | 22975 | - | 1572 | 230 | - | 76 | 23972 |
| | (347) | (-) | (-) | (19242) | (-) | (1779) | (989) | (-) | (-) | (22462) |
| I | - | - | - | - | - | - | - | 124 | - | 1127 |
| | (-) | (-) | (-) | (-) | (-) | (-) | (-) | (124) | (-) | (1083) |
| J | 154 | - | - | - | - | - | 38356 | 22350 | - | - |
| | (28) | (-) | (-) | (-) | (-) | (-) | (19618) | (22497) | (-) | (-) |

Table 6.5: Number of packets transmitted between PoPs in the busy hour (and slow hour). Each row tells the number of packets sent from the given source PoP to the other PoPs.

Because the method of producing the traffic matrix is not perfect, one should not try to conclude much of the total amounts of data actually transmitted between PoPs. The traffic matrix in table 6.5 is mainly intended for showing how many samples were used in measuring the packet latency and drop per cent in the next sections. The more samples there are in the latency and drop percentage calculations the more reliable the results.

The number of IP hops between PoPs is shown in table 6.6. The values were derived by monitoring the decrease in TTL value of IP packets sent between PoPs. The resulting matrix should be symmetrical, i.e. element $a(i,j) = a(j,i)$ for all $i,j$ in the range, if the routing in the network is symmetrical and only one path is used between each PoP. However, there are at least two paths used for traffic from PoP $A$ to $H$, for example. Also, the traffic in the direction of $F \rightarrow H$ is sent along a path with 11 hops while the opposite direction has a path with 8 hops. Some paths have a hop count of 0 indicating that the PoPs are interconnected with a layer-2 device, such as a switch.

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| A | X | 0 | 2 | 3 | 2 | 8 | 2 | 5-8 | - | 6-9 |
| B | 0 | X | 2 | 3 | 2 | - | 2 | - | - | - |
| D | 2 | 2 | X | 5 | 4 | - | - | - | - | - |
| D | 3 | 3 | 5 | X | 0 | 8 | 5 | 5-8 | - | - |
| E | 2 | 2-3 | 4 | 0 | X | - | 4 | - | - | - |
| F | 8 | - | - | 8 | - | X | 7 | 11 | - | - |
| G | 2 | 2 | - | 5 | 4 | 6-7 | X | 3-6 | 3 | 4-8 |
| H | 5 | - | - | 5 | - | 8 | 3-4 | X | 6 | 4 |
| I | - | - | - | - | - | - | - | 6 | X | 7 |
| J | 6 | - | - | - | - | - | 5 | 4 | - | X |

Table 6.6: Number of hops between PoPs. If more than one path is in use, the hop range is printed. Each row tells the number of hops between the given source PoP and the other PoPs.

From now on, the analysis focuses on the traffic between PoPs. The path used inside the core network is irrelevant and unknown. Only the number of hops between the PoPs is known (shown in figure 6.6). The notation used when discussing for example the traffic sent from PoP $A$ to $B$ is: $A \rightarrow B$.

## 6.4.2 Latency

The mean one-way mean packet latency between PoPs is presented in table 6.7. The overall view is that the busy hour packet latencies are noticeably larger than the corresponding values in the slow hour.

As explained in section 2.3 the clocks used in measuring the delays are not at exactly the same time. Delays contain some measurement error which is clearly manifested as the negative values in the latency matrix, for example between PoPs $B \rightarrow A$.

An interesting anomaly is found in the latency between PoPs $J \rightarrow G$. The latency in the slow hour (38.5 ms) is significantly higher than the corresponding in the busy hour (10.1 ms) even though the number of packets sent is lower in the

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| A | - | 1.282 | 9.381 | 1.957 | 1.987 | 89.536 | 64.377 | 8.452 | - | 11.404 |
| | (-) | (0.480) | (1.261) | (1.683) | (1.511) | (-) | (5.029) | (8.717) | (-) | (14.670) |
| B | -0.847 | 0.053 | 15.227 | 1.332 | 0.945 | - | 68.803 | - | - | - |
| | (-0.243) | (0.032) | (0.933) | (1.301) | (0.922) | (-) | (7.828) | (-) | (-) | (-) |
| C | 4.962 | 11.394 | - | 7.047 | 10.141 | - | - | - | - | - |
| | (2.760) | (3.001) | (-) | (5.112) | (3.853) | (-) | (-) | (-) | (-) | (-) |
| D | 1.689 | 2.810 | 7.393 | -0.260 | 0.469 | 206.628 | 30.127 | 8.156 | - | - |
| | (1.795) | (2.118) | (2.594) | (-0.240) | (0.058) | (9.373) | (6.415) | (6.908) | (-) | (-) |
| E | 1.141 | 2.207 | 8.162 | 0.010 | - | - | 15.810 | - | - | - |
| | (1.628) | (1.956) | (2.739) | (0.373) | (-) | (-) | (5.977) | (-) | (-) | (-) |
| F | 17.016 | - | - | 17.844 | - | - | 10.678 | 15.650 | - | - |
| | (-) | (-) | (-) | (8.600) | (-) | (-) | (3.306) | (8.203) | (-) | (-) |
| G | 9.259 | 23.326 | - | 36.424 | 26.540 | 125.415 | 0.326 | 11.284 | 5.509 | 39.382 |
| | (4.649) | (8.916) | (-) | (5.826) | (6.477) | (4.094) | (1.522) | (5.418) | (-) | (16.864) |
| H | 9.041 | - | - | 10.111 | - | 17.240 | 3.078 | - | 1.121 | 1.738 |
| | (8.640) | (-) | (-) | (8.859) | (-) | (7.510) | (3.975) | (-) | (-) | (1.564) |
| I | - | - | - | - | - | - | - | 1.663 | - | 3.223 |
| | (-) | (-) | (-) | (-) | (-) | (-) | (-) | (1.051) | (-) | (2.458) |
| J | 22.509 | - | - | - | - | - | 10.095 | 1.683 | - | - |
| | (10.044) | (-) | (-) | (-) | (-) | (-) | (38.503) | (1.864) | (-) | (-) |

Table 6.7: Mean packet latency matrix in the busy hour (and slow hour) in milliseconds. A hyphen means that no samples were available. Each row tells the mean latency for the packets sent from the given source PoP to the other PoPs.

slow hour, as seen in table 6.5. The number of packets sent is still large enough ( $> 10000$ in both cases) to make the explanation of measurement inaccuracy due to too few samples to be valid. The increased latency may be because of local network congestion in parts of the network not monitored by our equipment. Another explanation is a malfunctioning device. And one possibility is that the traffic sent between these PoPs in the slow hour is burstier creating congestion in the transmission path. A similar situation can be found in the traffic between PoPs $A \rightarrow J$. The rest of the cases, where latency is higher in the slow hour than in busy hour, the difference is small enough to be explained with inaccurate clock synchronization.

For a more visual look at the packet latencies, the busy hour packet delay is illustrated in figure 6.19 and the corresponding slow hour situation in figure 6.20.

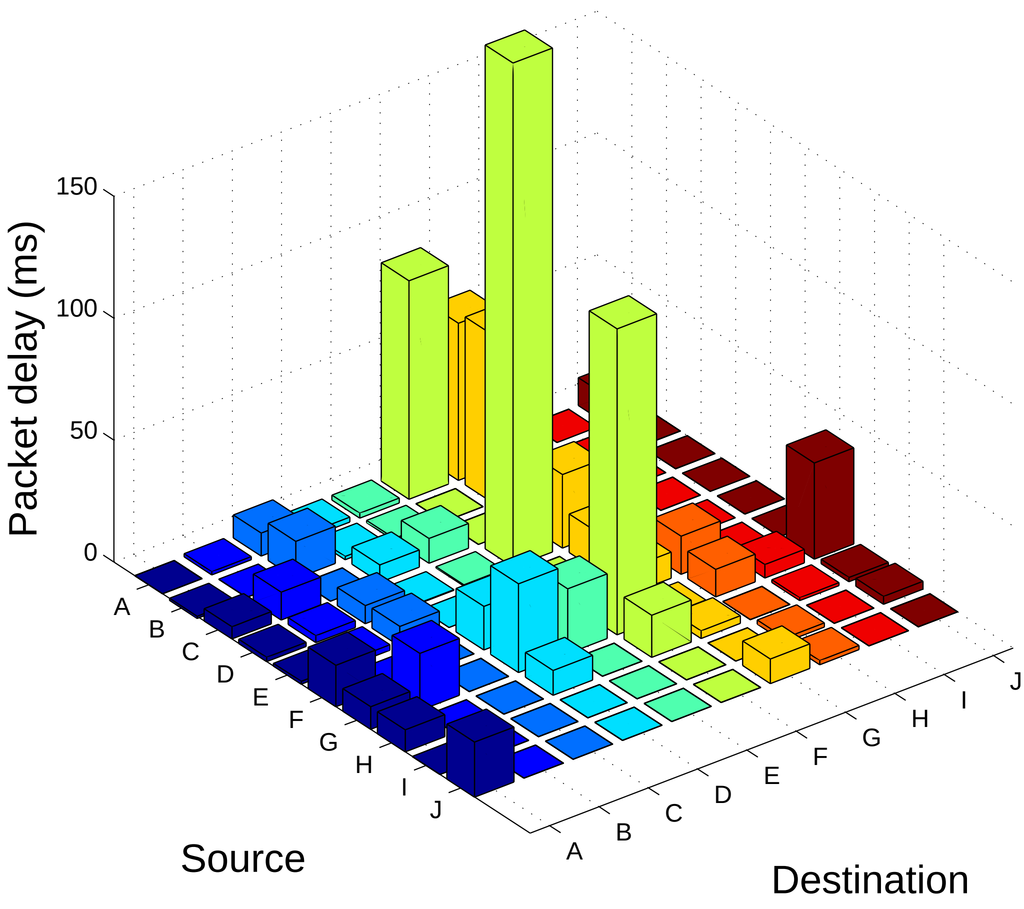

Figure 6.19: Packet latency between all PoPs in the network in the busy hour.

### Histogram

Packet latency histograms for the traffic from PoP $A$ to other PoPs are presented in figure 6.21 (busy hour) and figure 6.22 (slow hour). The highest 1 per cent of the samples has been excluded from the histogram to cut down the tail and reduce noise caused by few individual packets. The number of samples or packets included is shown above each subfigure.

Generally, a typical packet latency histogram seen in any communication path has zero values until at some point in time where the latency has a high peak. The peak value is an estimate for the latency of a packet which is sent to an empty network. After the peak, there is a gradually decaying tail and the rate of decay is the slower the more congestion there is in the network.

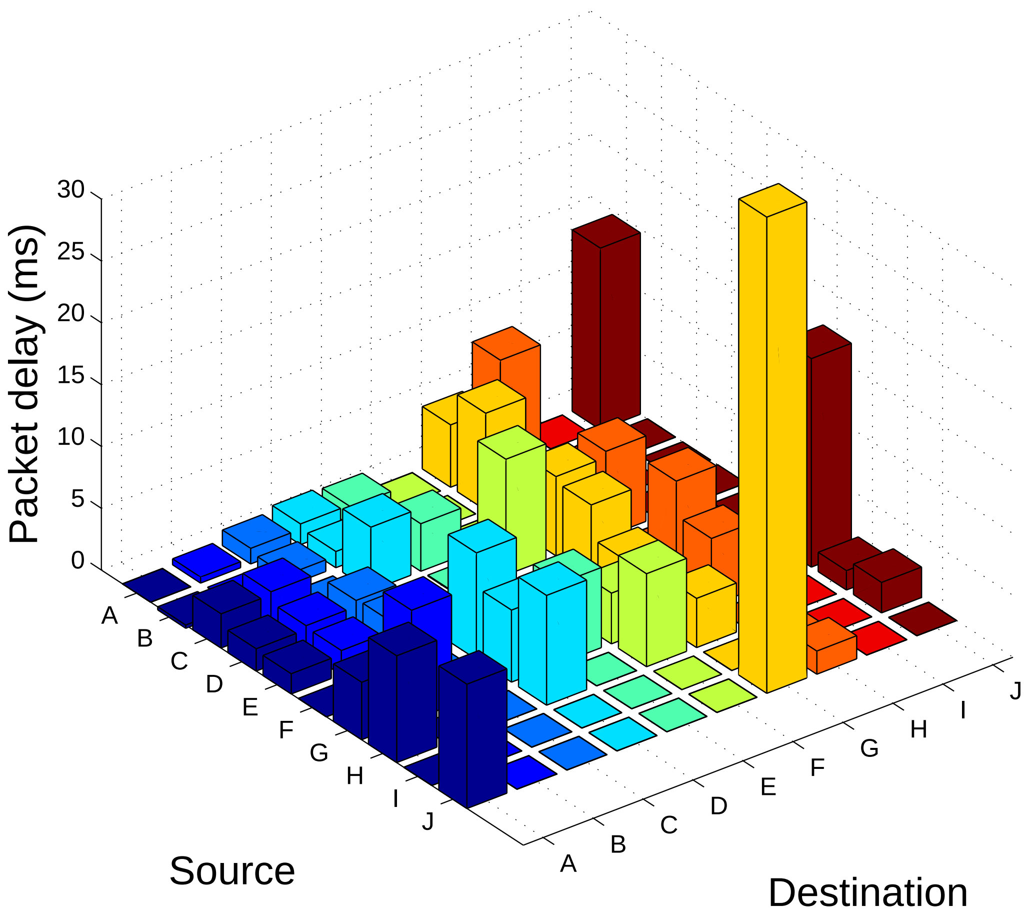

Figure 6.20: Packet latency between all PoPs in the network in the slow hour.

Many of the graphs in figures 6.21 and 6.22 have a distinct peak. The tails of the busy hour graphs seem to be heavier than the ones in the corresponding slow hour graphs. However, the difference is quite small, indicating that the network is probably quite over-provisioned and is well capable of handling the traffic also in the busy hour.

An interesting feature of the network is the double peaked latency histograms between some of the PoPs. The phenomenon is clearest in the busy hour communication between $A \rightarrow B$ (in figure 6.23) and the slow hour communication between PoPs $A \rightarrow E$ and $A \rightarrow H$ (in figure 6.24). This kind of behavior suggests that more than one path is used between the PoPs. The conclusion becomes more evident when looking at the hop count matrix in table 6.6. The traffic from

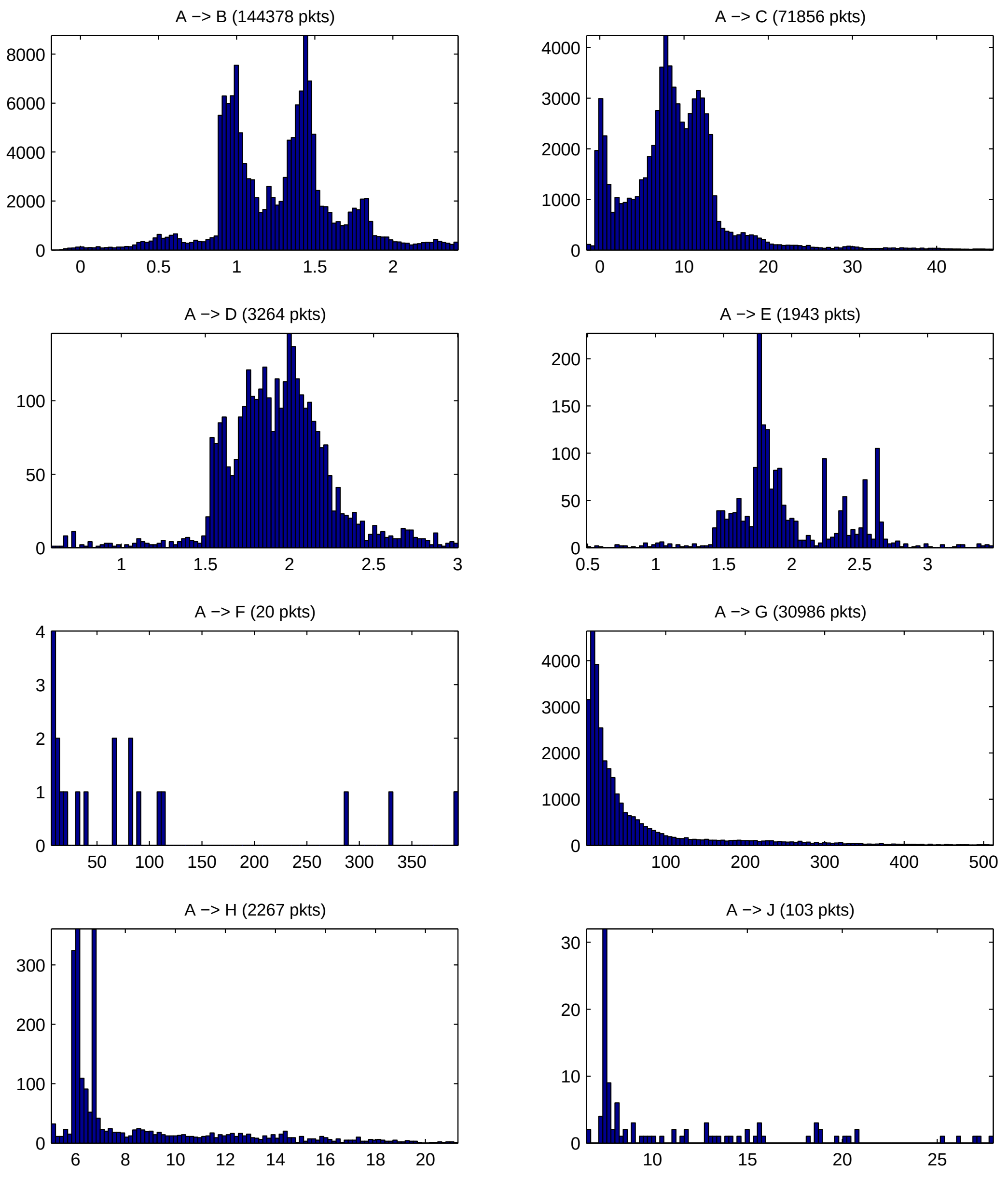

Figure 6.21: Histogram of one way packet latency from PoP $A$ to other PoPs in the network during the busy hour. All subfigures show latency (ms) in the x-axis.

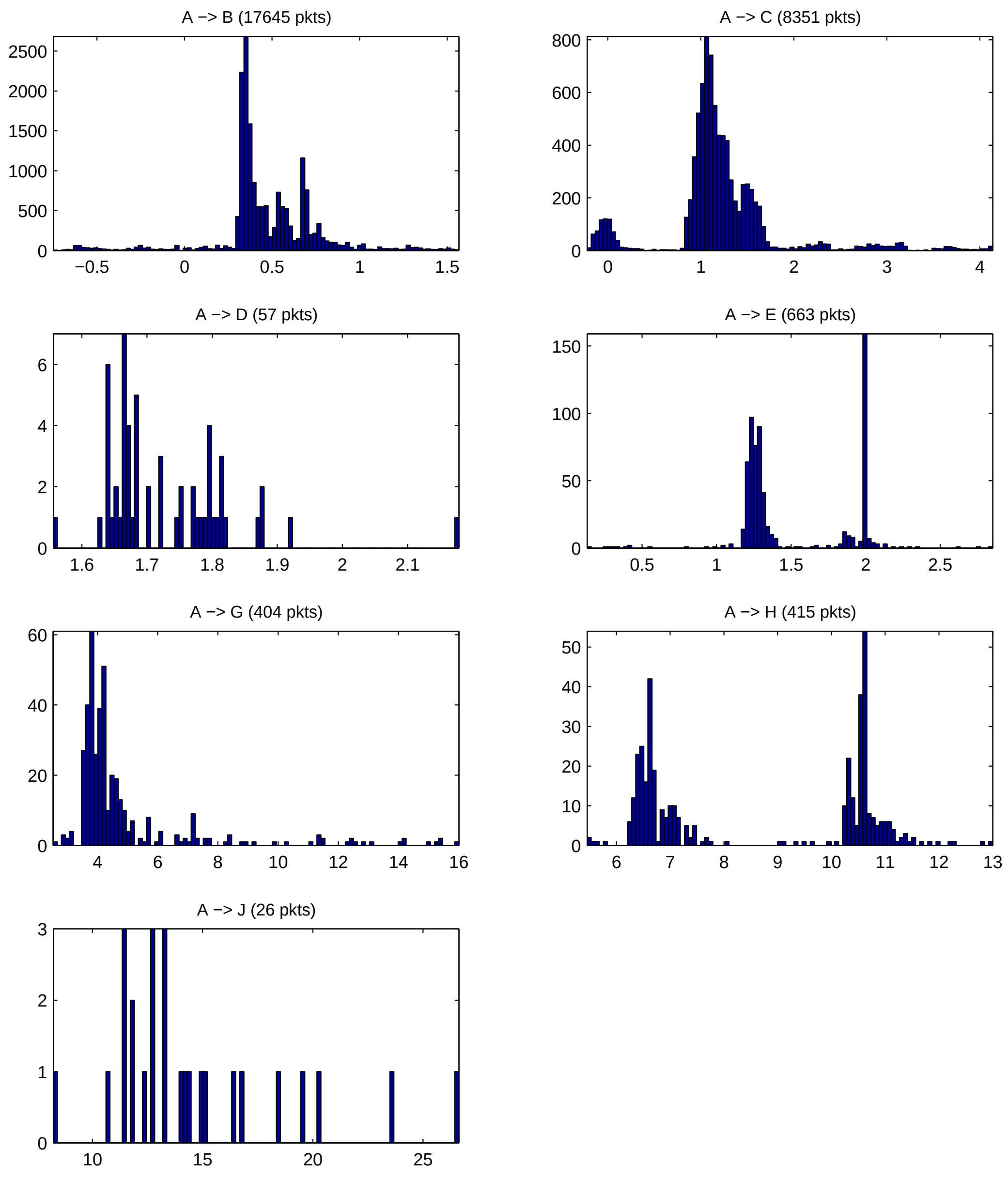

Figure 6.22: Histogram of one way packet latency from PoP $A$ to other PoPs in the network during the slow hour. All subfigures show latency (ms) in the x-axis.

PoP $A$ to $H$ utilizes more than one paths and is the probable cause of the double peaks. On the other hand, all the packets from PoP $A$ to $E$ use only 2 hops, so the existence of multiple paths can not be positively concluded. However, it also can not be completely ruled out because multiple paths may contain the same amount of hop counts but still introduce different amounts of latency on the packets. The two peaks in the busy hour latency between $A \rightarrow B$ might be because of inaccurate clock synchronization, or clock wander. The peaks in slow hour graphs from PoP $A$ to $E$ and $H$ seem too sharp and distinct for this explanation to be probable. In addition, the difference between the two peaks in the latency of $A \rightarrow H$ is 4 ms which seems too high to be caused by clock inaccuracy.

An interesting double peak is also visible in the slow hour latency of $A \rightarrow C$. There is a main concentration of values around 1 ms. A smaller but distinct peak is also located around 0 ms. Again, the reason for this behavior is not certain. There may be two paths between the PoPs. On the other hand, the latency difference between the peaks is only about 1 ms, so not much should be concluded from the existence of double peaks.

Most of the histograms have one or two distinct peaks with very few values preceeding the first peak. The rise of the first peak gives an estimate of the packet latency in an empty network. A difference of 1 ms between the first peak and the first values is seen in most of the graphs. There are two possible causes for the existance of values before the first peak: variance in packet size or measurement error in timestamps. Packet size in Ethernet networks varies between 64 and 1518 bytes. Assuming that there are mostly 100 Mbps links in the network, the transmission delay of the packets in a single link varies between 5.1–120 $\mu$s (refer to equation 3.22). The resulting end-to-end delay variation is derived by multiplying the transmission delay of a single link by the number of hops in the transmission path. For example, there 8 hops between PoPs $A \rightarrow F$, so the end-to-end delay variation caused by the existance of different packet sizes is approximately 0.04–1 ms. This explains some of the total delay variance seen in the histograms. The peaks may be caused by the packets of the most common size and the surrounding noice is produced by other packet sizes. However, there are no IP level hops between PoPs $A \rightarrow B$. Thus, the delay variance in the resulting histogram is caused only by the timing inaccuracy of the measurement system. Thus, a rough estimate of $\pm 1$ ms for the latency measurement error can be said to exist.

### Time-Latency Plot

Time plots of the packet latency are presented in figures 6.23 (for busy hour) and 6.24 (for slow hour). The graphs show the 1 hour period of latency measurements between PoP $A$ and other PoPs. If there are more than 4000 samples (packets) a mean value is calculated and plotted every $n/4000$ samples (where $n$ is the number of samples) for preserving readability in the graph.

The evolving of latency is clearly seen in graphs where the samples are evenly distributed along the time line. The time-plot graphs can be used in trying to find a reason for the two peaks in the latency histograms. The peaks for busy hour communication between $A \rightarrow B$ is not clearly seen because the curve is too noisy to see two distinct latency levels. The range at which the curve rapidly oscillates is about 0.5 ms which is about the same as the distance of the peaks in the corresponding histogram in figure 6.21. However, the averaging done every $n/4000$ samples may hide some of the information in the graph.

The traffic in the direction of $A \rightarrow H$ during the slow hour seems to have two latency levels, as seen in figure 6.22: one at approximately 6.5 ms and the other at 10.5 ms. This is probably caused by the existance of multiple paths between PoPs, as explained in the analysis of the latency histograms. The time-plot shows a clear shift from the lower latency level to the higher one in the middle of the slow hour. The reason is most likely a change in packet paths used by the traffic. However, the paths seem to coexist since latency values in both levels are seen throughout the hour. But for some reason more packets choose the slower path after the transition in the middle of the hour.

The traffic in the direction of $A \rightarrow E$ had two peaks in the histogram distanced at about 0.7 ms of each other — one at 1.3 ms and the other one at 2.0 ms — as was seen in figure 6.22. The corresponding time-plot in figure 6.24 sheds more light into the scenario. The latency seems to be at a base level of 1.3 ms with occasional higher latency values at approximately 0.7 ms higher level. One explanation might be the use of two separate transmission paths between the PoPs. Alternatively, it may be the result of transmitting packet bursts; if two packets arrive at the same

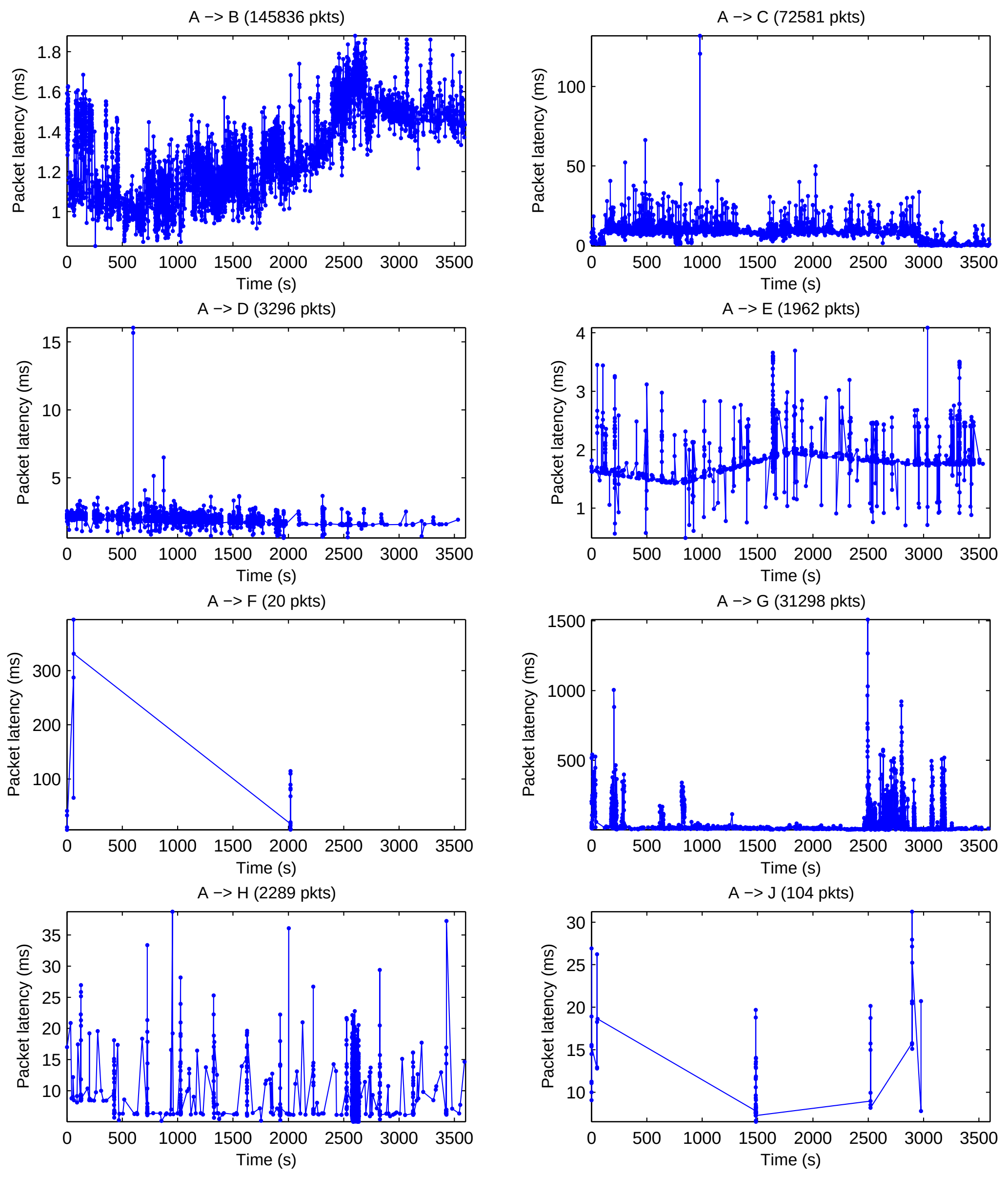

Figure 6.23: Time-plot of one way packet latency from PoP $A$ to other PoPs in the network during the busy hour. All subfigures show time (s) in the x-axis and latency (ms) in the y-axis.

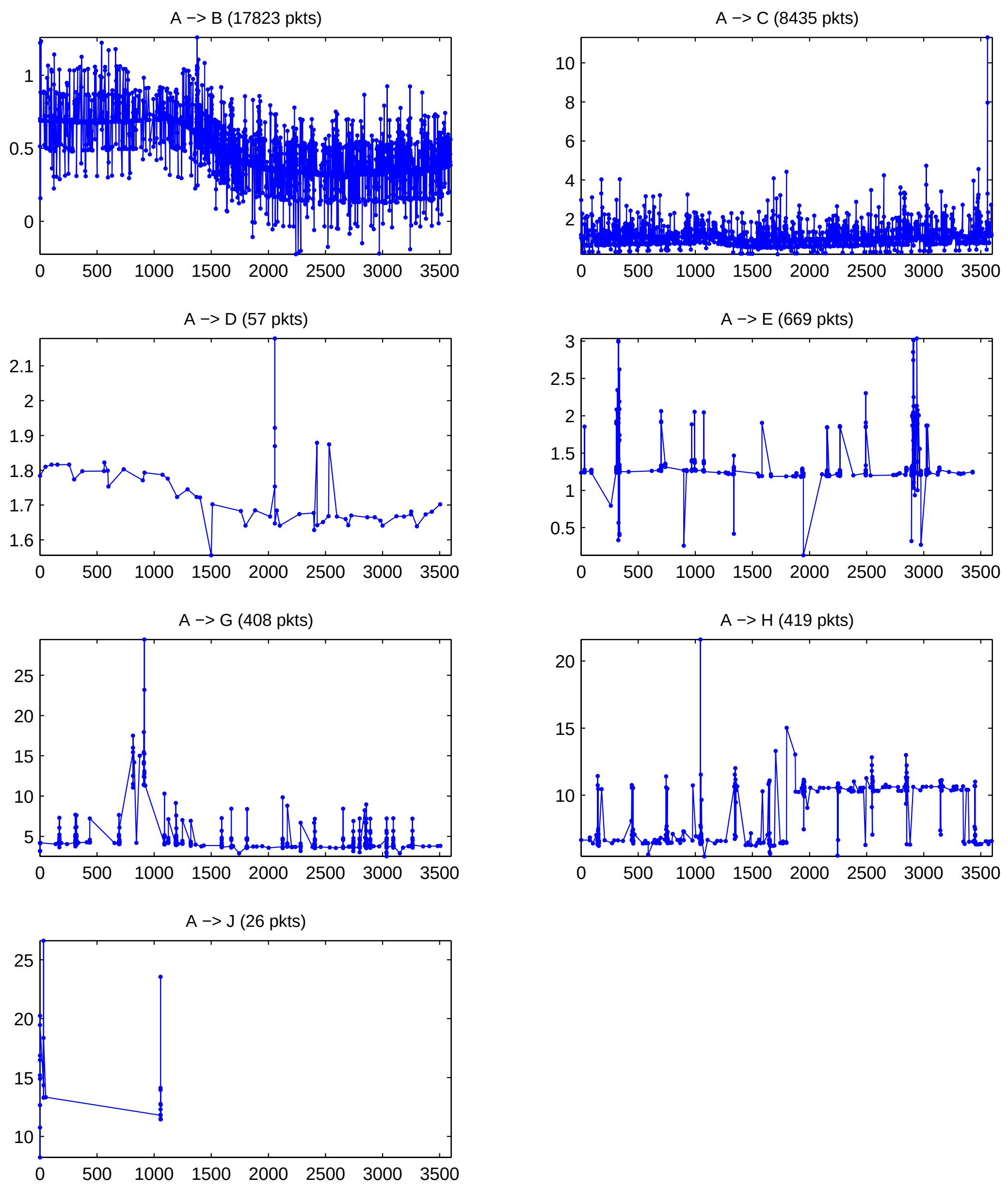

Figure 6.24: Time-plot of one way packet latency from PoP $A$ to other PoPs in the network during the slow hour. All subfigures show time (s) in the x-axis and latency (ms) in the y-axis.

time on the link, the other one will be queued in the send buffer until the other packet is fully transmitted. The waiting time is half of the transmission time on the link, formalized in equation 3.22. If we hypothesize that transmission path has a bottleneck link with a bandwidth of 2 Mbps the extra 0.7 ms of waiting time would be caused by a packet of size: $2\text{Mbps} \cdot 0.7\text{ms} =$ 175 B. Packet size CDF of the network (seen in figure 6.17) shows that the majority of UDP packets are approximately of that size.

There seems to be long term oscillation in latency with a period of about an hour in the many of the graphs. The phenomenon is clearly visible in the latencies measured between PoPs $A \rightarrow B$ and $A \rightarrow E$ during the busy hour (in figure 6.23) and between $A \rightarrow B$ and $A \rightarrow D$ during the slow hour (in figure 6.24). The peak-to-peak amplitude of the oscillation varies to some extent but remains below 0.6 ms in all cases. The phase of oscillation seems to be roughly the same for latency graphs measured during the same hour. The cause of the oscillation may simply be changes in the network traffic loads. However, more probably, the reason is the inaccuracy in clock synchronization and clock wander, as explained in section 2.3.5. The oscillation is not always pure sine form — as seen in the busy hour graph between $A \rightarrow E$ — because there are in fact two clocks timestamping the packet for measuring latency. Both typically have a sine form clock wander resulting in an error in latency which is a superposition of two sine curves.

### 6.4.3 Loss Ratio

Packet loss ratio matrix measured during the busy and slow hours is presented in table 6.8. The situation is illustrated in figures 6.25 (for busy hour) and 6.26 (for slow hour). The bars representing the traffic between $A \rightarrow B$, $H \rightarrow G$ and $H \rightarrow I$ have been cut to show the small values more clearly. The z-scale is the same on both figures for easier comparison.

The loss ratios have high variability when comparing the traffic between different PoPs. Mostly, there is no packet loss: more than half of the non-empty elements in the packet loss matrix are 0. The packet loss ratios above zero are mostly very low in both busy and slow hours. In all directions, the busy hour exhibits more or about the same amount of packet loss compared to slow hour except for

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| A | - | 2.204 | 0.036 | 0 | 0 | 0 | 0.070 | 0 | - | 0 |
| | (-) | (0.992) | (0) | (0) | (0) | (-) | (0) | (0) | (-) | (0) |
| B | 0 | 0.034 | 0.014 | 0 | 0.067 | - | 0.025 | - | - | - |
| | (0) | (0.019) | (0) | (0) | (0.032) | (-) | (0) | (-) | (-) | (-) |
| C | 0 | 0.088 | - | 0 | 0.052 | - | - | - | - | - |
| | (0) | (0) | (-) | (0) | (0.027) | (-) | (-) | (-) | (-) | (-) |
| D | 0 | 0 | 0 | 0 | 0.041 | 0.574 | 0.031 | 0 | - | - |
| | (0) | (0) | (0.011) | (0) | (0.031) | (0) | (0) | (0) | (-) | (-) |
| E | 0 | 0 | 0 | 0.003 | - | - | 0.008 | - | - | - |
| | (0) | (0) | (0) | (1.100) | (-) | (-) | (0.046) | (-) | (-) | (-) |
| F | 0 | - | - | 0 | - | - | 0 | 0 | - | - |
| | (-) | (-) | (-) | (0) | (-) | (-) | (0) | (0) | (-) | (-) |
| G | 0.008 | 0.207 | - | 0 | 0.060 | 0.602 | 0 | 0 | 0 | 0.002 |
| | (0) | (0) | (-) | (0) | (0.056) | (0.371) | (0) | (0) | (-) | (0) |
| H | 0 | - | - | 0 | - | 0.382 | 11.304 | - | 15.789 | 0.017 |
| | (0) | (-) | (-) | (0) | (-) | (0.225) | (0.404) | (-) | (-) | (0.018) |
| I | - | - | - | - | - | - | - | 0 | - | 0 |
| | (-) | (-) | (-) | (-) | (-) | (-) | (-) | (0) | (-) | (0) |
| J | 0 | - | - | - | - | - | 0 | 0 | - | - |
| | (0) | (-) | (-) | (-) | (-) | (-) | (0) | (0) | (-) | (-) |

Table 6.8: Packet loss ratio matrix in the busy hour (and slow hour) in per cents. A hyphen means that no samples were available. Each row tells the packet loss ratio between the given source PoP and the other PoPs.

the sender $E$.

The loss ratio for direction $E \rightarrow D$ is unusually high during the slow hour, as seen in figure 6.26. There is quite a lot of packet traffic between the two PoPs (as shown in table 6.5) but the number is smaller for the slow hour than for the busy hour so increased activity should not be the cause of higher packet loss. The packet latency is roughly the same or lower during the slow hour, as can be seen in table 6.7, also indicating no unusual behavior. However, only inspecting the total number of sent packets or the mean latency during the hour hides the effect of burstiness in traffic. If the traffic between the PoPs in the slow hour is much burstier compared to the busy hour it would explain the increased packet loss even though mean latency and the number of sent packets during the hour have diminished. Also, the cause could simply be the inaccuracy in packet loss calculation.

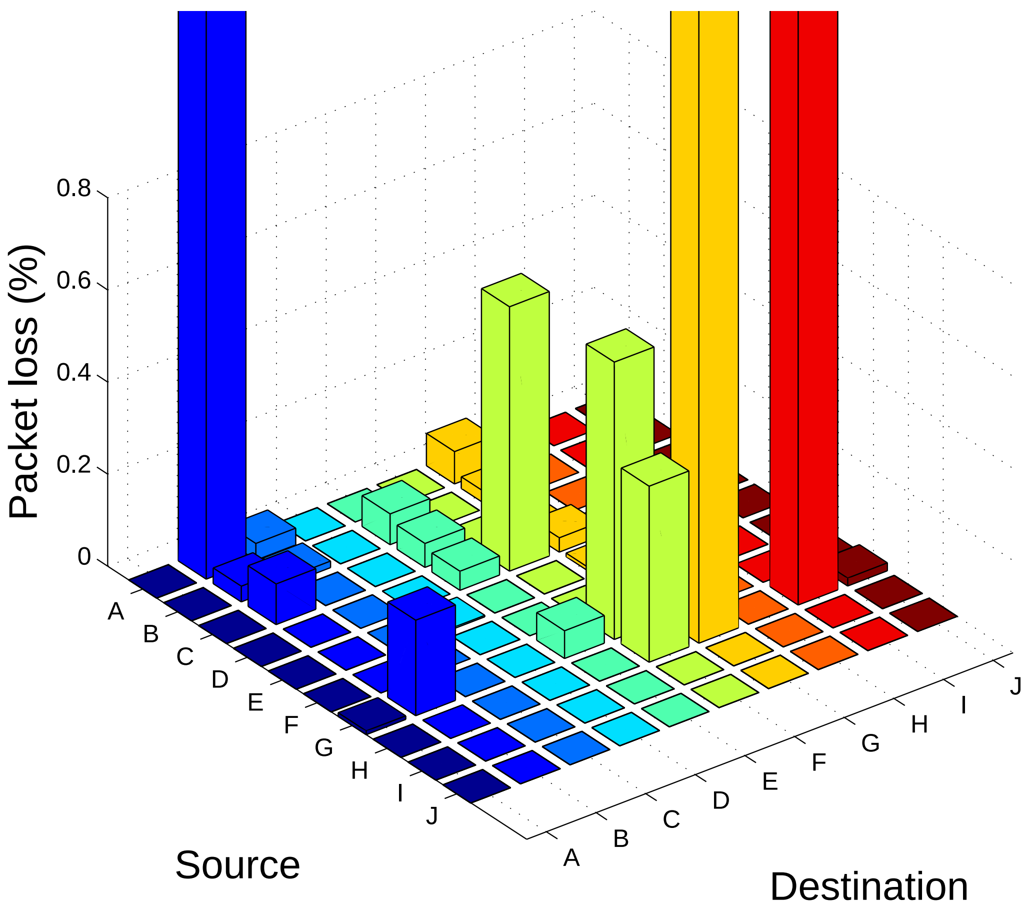

Figure 6.25: Packet loss ratio between all PoPs in the network in the busy hour

The calculation of packet loss ratio matrix in table 6.8 has some inaccuracy which may lead to overestimated values. The calculation method was described in section 4.5.2. However, there was no sure way of knowing where the packet entering the network should exit. A heuristic was devised where the destination PoP of each packet could be guessed at the sending side PoP based on the destination IP address; or more precisely, based on the C-class subnetwork of the destination IP address. If the traffic was seen exiting the correct PoP the guess was valid and the packet loss probability was calculated from the number of packets entering and exiting the network between each PoP pair. However, an upper limit for the maximum packet loss ratio was defined as 30 per cent for each subnet between each PoP pair. If the limit was crossed the initial guess of the correct destination PoP

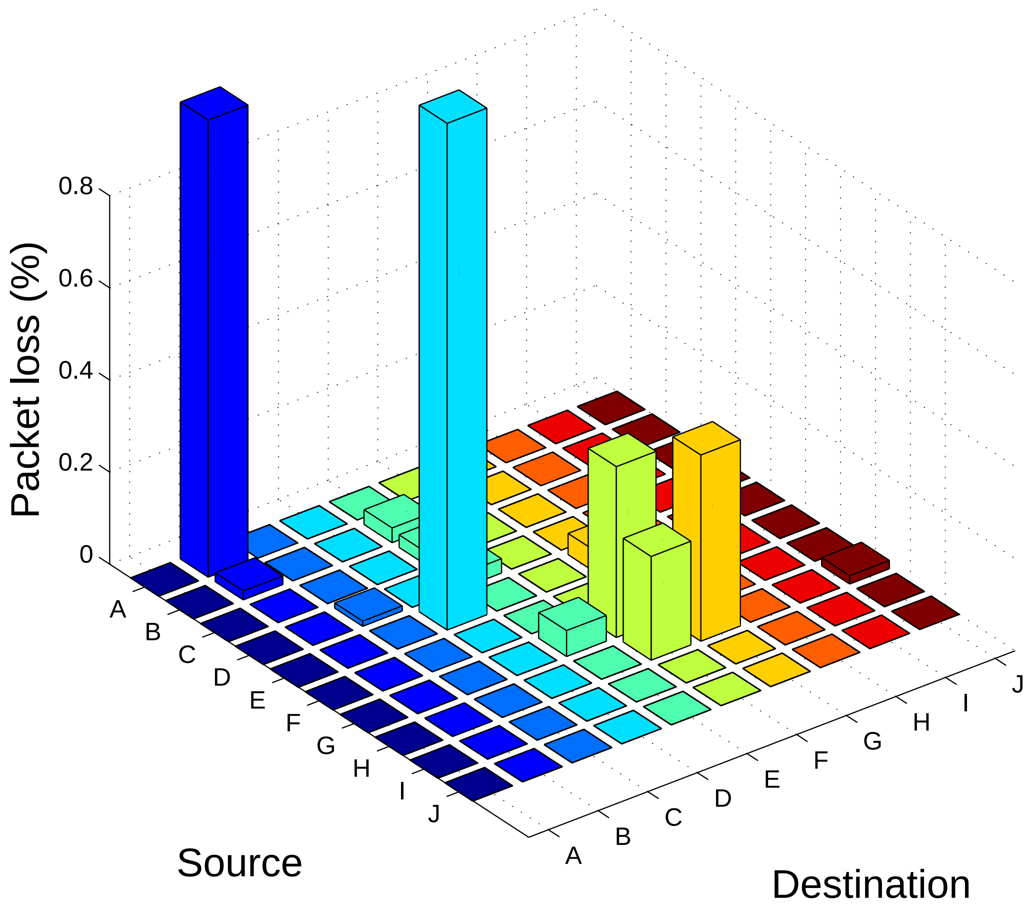

Figure 6.26: Packet loss ratio between all PoPs in the network in the slow hour

was abandoned and the subnetwork between the given PoP pair was discarded of the packet loss calculation. This set a total maximum packet loss ratio of 30 per cent between each PoP pair. Because of this heuristical packet loss calculation the very high values during the busy hour ($A \rightarrow B$, $H \rightarrow G$ and $H \rightarrow I$, seen clearly in figure 6.25) should be viewed with some skepticism.

## 6.5 Self-Similarity

The degree of self-similarity was estimated by producing a VTP (Variance Time Plot) on two time scales:

- Short scale (60 seconds)

Busy hour (figure 6.27)

Slow hour (figure 6.28)

- Long scale (1 week) (figure 6.29)

Each figure represents the VTP (Variance-Time Plot, refer to section 3.2.3) of traffic directed into the core network in a single PoP. For the calculations, mean traffic rate is calculated at an interval of 100 ms for the short scale and 60 seconds for the long scale measurements. The length of an m-block is the m-multiple of the interval in the VTP calculations.

Comparing figures 6.27 and 6.28 it seems that the busy hour has overall more self-similarity in the traffic than the slow hour. The two exceptions are PoPs $H$ and $J$ where the slow hour traffic notably more self-similar. The phenomenon is in accordance with the theory that more aggregate processes involved in producing the traffic cause more self-similarity, as explained in section 3.2.2. Only PoP $J$ is somewhat different since the sample set contains more packets but less self-similarity in the busy hour measurement. Also, the traffic in PoP $E$ is unusual compared to the other PoPs in the sense that it exhibits practically no self-similarity during both hours.

The week long self-similarity measurement is presented in figure 6.29. The degree of self-similarity is overall higher than in the busy hour, with the slight exception of PoPs $H$ and $I$. Also PoP $E$ exhibits a high degree of self-similarity even though none was present in the short-term measurements in the busy and slow hours.

The figures show that self-similarity is clearly an existing feature of the network traffic and the use of heavy-tailed distributions in the traffic model is justified. Especially, when planning long-lasting simulations the degree of self-similarity should be noted and compared in the results.

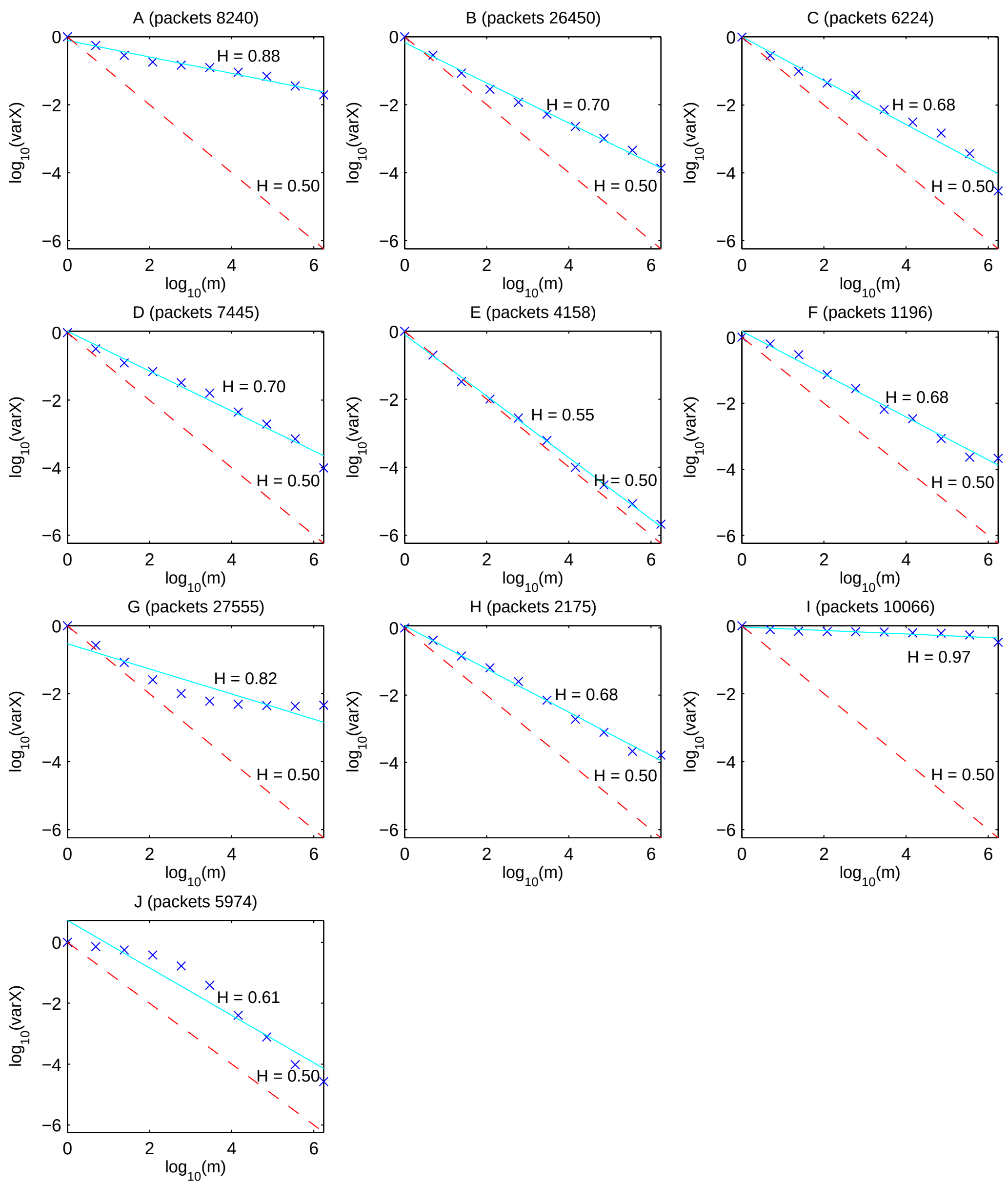

Figure 6.27: Self-similarity (Hurst parameter) of traffic on a time period of 60 seconds during the busy hour.

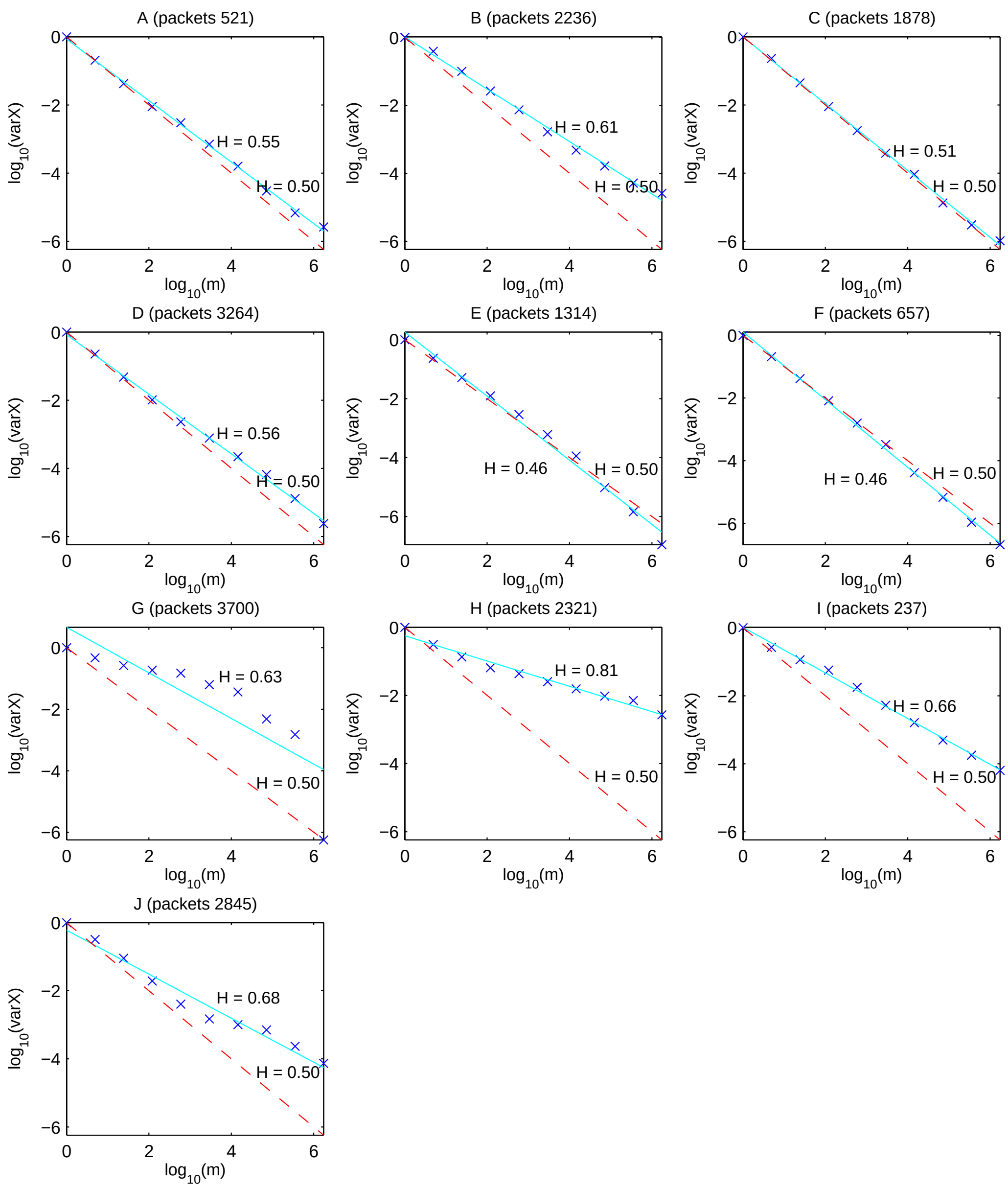

Figure 6.28: Self-similarity (Hurst parameter) of traffic on a time period of 60 seconds during the slow hour.

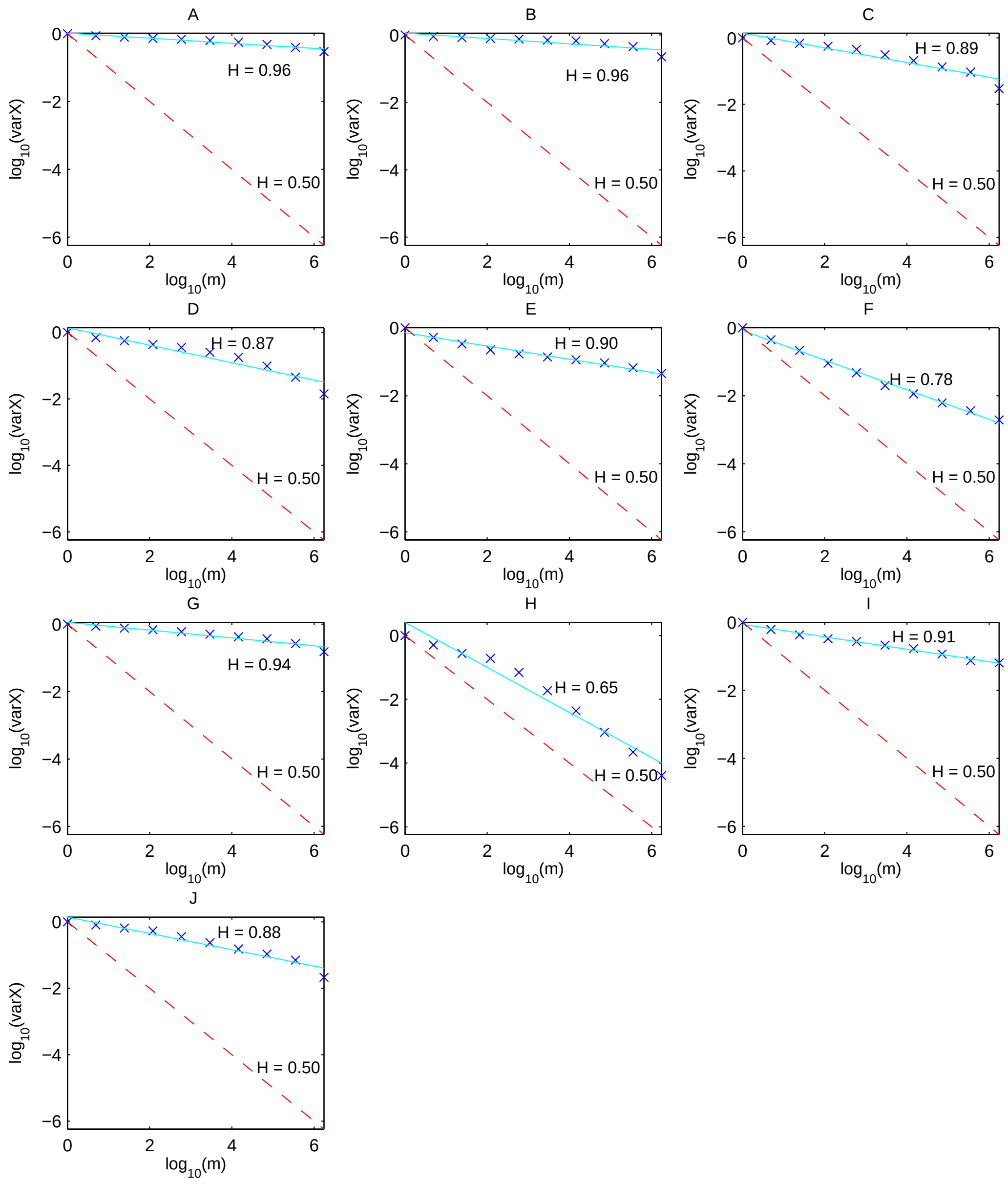

Figure 6.29: Self-similarity (Hurst parameter) of traffic on a time period of one week.

# Chapter 7

# Conclusions

## 7.1 Summary

In this thesis, network traffic was analyzed in a core network consisting of multiple entry and exit points, or so called points of presence (PoP). A measurement infrastructure consisting of inexpensive off-the-shelf computers was designed and implemented for the network. Each PoP was fitted with a traffic monitoring device listening in on all incoming and outgoing packets to and from the network. It made it possible to form a fully connected traffic matrix between the PoPs and the traffic to be inspected separately between each PoP. Based on this information, a traffic model which describes the network traffic as accurately as possible was devised. The measurement infrastructure also made it possible to analyze network performance parameters which are typically difficult to measure: one-way packet latency and packet drop probability. The goal of the thesis was to analyze the network traffic to the extent that it could be reproduced in a simulated environment and also that the performance capabilities between the simulated and real networks could be compared.

## 7.2 Network Traffic Analysis

Network traffic was analyzed on two perspectives: the traffic characteristics and network performance. The measurement infrastructure worked without problems.

The packet capture devices worked as they were supposed to and no major problems came up with the hardware or software.

The number of traffic traces was quite large and the work included a lot of automating the handling and processing of the raw data. It would have been interesting to study various phenomena on different time scales but the processing time or hardware resources were the limiting factors in designing some of the analysis tasks. Overall, all analysis tasks which where initially thought of were successfully completed.

The biggest difficulty in the analysis was splitting incoming traffic based on the destination PoP. No information about the topology of the network and the location of the IP addresses and subnetworks were supplied by the network administrators. The heuristics to determine the destination PoP proved good enough for getting a reasonable amount of samples for the latency and packet loss calculations. However, creating the complete matrix connecting all the PoPs together may prove to be too inaccurate with the limited sample set. Also, the number of leak points — where traffic would enter or exit the network unnoticed — seemed higher than anticipated. No accurate estimates of the amount of traffic leakage were formulated. How they will ultimately affect the accuracy of the traffic model will remain a question mark for the time being. However, it is known now that the packet loss ratio calculations can not be fully trusted and may give overly pessimistic values.

One-way packet latency was quite successfully measured. However, better methods of estimating the latency measurement error would have been useful to formulate. The maximum error estimate deduced from the measurements was $\pm 1$ ms. Compared to the total one-way latency between many PoPs the error is relatively high. The minimum packet latency in the network was notably lower than originally estimated making the percentual accuracy of the measurements high. Also, ruling out measurement inaccuracy as the cause of the double peaks in some of the latency histograms would have been worthwhile. Taking time-latency plots spanning more than 1 hour would have given more information about the oscillation in latency which was most likely caused by clock wander. The method of improving NTP accuracy, as described in section 2.3.3, did not give promising results and was not used in the measurements. The method resulted in unrealistically sharp changes in clock offset during periods of variable congestion.

## 7.3 The Success of the Traffic Model

The traffic model succeeded better than it seemed somewhere in the middle of the project. Using the three protocols (TCP, UDP, ESP) for separate modeling offered relatively simple and yet accurate method of formalizing the traffic. Modeling the inter-arrival times succeeded the way it was originally thought of. However, the size parameters presented more of a challenge. The flow size could ultimately be modeled as planned but the accuracy was less than perfect. It remains to be seen, when applying the model in a simulation, how closely the simulated flow size distribution resembles the corresponding real network case. The packet size CDF was distributed much more discretely and at the last minute was decided to be modeled as of individual values instead of a distribution. In the case of UDP, this turned out excellent but for ESP the approximation may be too rough.

It would have been interesting to use clustering methods to form aggregates of the application layer protocols and use them instead of the three transport layer protocols as the basis of the traffic model. It would probably have resulted in much more than three aggregates but the behavior of the flows or packets inside the aggregates would probably have been more similar, making the distribution fitting more easy and precise. However, time and space ran out and the aggregates could not be created and tested here.

The goodness of fit was evaluated visually and by studying the root mean square errors of the distributions. The linear combination of Pareto and Exponential distributions seemed to offer the most accurate results. However, the ability of a distribution to generate traffic which resembles the traffic seen in the real network can not be evaluated for certain by examining only the individual arrival and size processes. Multiple traffic generation processes located at different parts of the network contribute in an unpredictable total traffic profile and should be tested in a simulation. The other three distributions — Weibull, Pareto and Gamma — should also not be excluded from further studies. Exponential distribution seemed to fit too poorly on the data to be applied to the the traffic model meaningfully.

## 7.4 Future Work

The first thing to do is to create the simulation of the network analyzed and fit it with traffic models created in this thesis. The goal of the thesis was to analyze the real network to the extent that the results could be used for comparison with the corresponding simulated network. Also, the final conclusion on the quality of the traffic model can be estimated only after the analysis of the simulated traffic.

Better knowledge of the network topology should be acquired. This would benefit both the analysis and the forthcoming simulations. The traffic leak points would be discovered and traffic matrix could be created to contain a more complete data set. It would make the creation of the traffic model more accurate and packet drop percentage could be estimated more reliably. Simulations are not even possible without any knowledge of the network topology. If the topology information is not supplied by the network administrators, a rough guess of the topology can be created by using the analyzed traffic data. It would make the results more unreliable and further off from the corresponding real scenario. Nonetheless, it would be interesting to compare the results of the real and simulated environment.

Improving the traffic model could be done in many ways. The packet and flow size caused difficulties and they could probably be modeled better than what is done here. The distributions tested here were originally chosen to be fitted to the flow inter-arrival time curves. They could be used reasonably well with flow size and packet inter-arrival curves as well but better ways to model them should be studied. Clustering methods — such as self-organizing maps — should be used for application protocols. The resulting aggregates would possible behave more smoothly and could be better used for modeling than the transport layer protocols. However, creating the simulation and analyzing the results should be done first. It would help to identify the problems and biggest sources of inaccuracy in the model and shows which defects should be concentrated on first.

The accuracy of the latency measurements depends entirely of the quality of the timestamps. Clock synchronization should by studied more: more accurate timestamps and better error estimates should be striven for. Packet latency is the most important parameter in estimating the ability of the network to cope under high network load. Since many of the total packet latency values measured were quite low (in the order of 1–2 ms) the accuracy of $\pm 1$ ms estimated here should definitely be improved.

## 7.5 Closing Words

The subject and objectives of the thesis could probably have been less ambitious. The total length of the thesis grew to its upper limits and still I felt there was a lot that remained to be unsaid because of the lack of space and time. Some of the results could have been examined more carefully so that fewer guesses and more conclusions could have been made. However, the view over field of network analysis has been broad and has braught many new ideas on further studies.

Making of this thesis has been very interesting and highly educational in many respects. The project has had lots of twists and turns which have forced me to inspect the task from many perspectives. The work presented challenging problems involving both hardware and software. Overall, the process has provided me valuable insight into analysing network traffic and given me tools in understanding the way IP networks operate.

# Bibliography

[AGJT03] Deb Agarwal, Jose Maria Gonzalez, Goujun Jin, and Brian Tiernay. An infrastructure for passive network monitoring of application data streams. http://www-didc.lbl.gov/papers/SCNM-PAM03.pdf, 2003. Data Intensive Distributed Computing Group (DIDC), Proceedings of Passive and Active Monitoring Workshop 2003.

[Aut04] IANA (Internet Number Assignment Authority). Port numbers, February 2004. http://www.iana.org/assignments/port-numbers.

[BLFN96] Tim Berners-Lee, Roy T. Fielding, and Henrik Frystyk Nielsen. Hypertext transfer protocol – http/1.0. RFC1945, May 1996. http://www.ietf.org/rfc/rfc1945.txt?number=1945.

[Bra98] et al. Braden. Recommendations on queue management and congestion avoidance in the internet. RFC2309, April 1998. informational, http://www.ietf.org/rfc/rfc2309.txt?number=2309.

[Bro01] Nevil Brownlee. Netramet. Software, February 2001. An implementation of the Internet Accounting Architecture (RFC 2063 and RFC 2064), Home page: http://www2.auckland.ac.nz/net/NeTraMet/.

[CB96] Mark Crovella and Azer Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. In *Proceedings of SIGMETRICS'96: The ACM International Conference on Measurement and Modeling of Computer Systems.*, Philadelphia, Pennsylvania, May 1996. Also, in Performance evaluation review, May 1996, 24(1):160-169.

[CBP95] Kimberly C. Claffy, Hans-Werner Braun, and George C. Polyzos. A parametrizable methodology for internet traffic flow profiling. *IEEE JSAC*, Special Issue on the Global Internet, March 1995. http://www.nlanr.net/Papers/pmi.html.

[CFD90] Jeffrey D. Case, Mark Fedor, and James R. Davin. A simple network management protocol (snmp). RFC1157, May 1990. http://www.ietf.org/rfc/rfc1157.txt?number=1157.

[Cis03] Cisco Systems, Inc. *Network Time Protocol: Best Practices White Paper*, updated 2003. http://www.cisco.com/warp/public/-126/ntpm.html.

[CM00] Dan Connolly and Larry Masinter. The 'text/html' media type. RFC2854, June 2000. http://www.ietf.org/rfc/rfc2854.txt-?number=2854.

[DC02] Carlo Demichelis and Philip Chimento. Ip packet delay variation metric for ip performance metrics (ippm). RFC3393, November 2002. http://www.ietf.org/rfc/rfc3393.txt?number=3393.

[Dro97] Ralph Droms. Dynamic host configuration protocol. RFC2131, March 1997. http://www.ietf.org/rfc/rfc2131.txt?number=2131.

[Ein02] Nathan Einwechter. Implementing network taps with network intrusion detection systems. Web, June 2002. http://www.securityfocus.com/infocus/1594.

[FGM$^+$99] Roy T. Fielding, James Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, and Larry Masinter. Hypertext transfer protocol – http/1.1. RFC2616, June 1999. http://www.ietf.org/rfc/rfc2616.txt-?number=2616.

[Fin02] Finisar Co. *White Paper: The Pros and Cons of Tapping and Mirroring*, 2002. http://www.finisar.com/media/product_document_detail/-site2_ 296884688_site2_1684740469_ProsandConsofTappingand-Mirroring.pdf.

[Fis02] Amy Fisher. Network taps enable passive monitoring. *Network World*, November 2002. http://www.nwfusion.com/news/tech/2002/-1028techupdate.html.

[HDTI00] Tatsuya Hagiwara, Hiroki Doi, Hideki Tode, and Hiromasa Ikeda. High-speed calculation method of the hurst parameter based on real traffic. In *LCN*, pages 662–669, Tampa, Florida, November 2000. IEEE.

[HSSR99] Mark Handley, Henning Schulzrinne, Eve Schooler, and Jonathan Rosenberg. Sip: Session initiation protocol. RFC2543, March 1999. http://www.ietf.org/rfc/rfc2543.txt?number=2543.

[JR86] Raj Jain and Shawn A. Routhier. Packet trains – measurements and a new model for computer network traffic. *IEEE Journal on Selected Areas in Communications*, SAC-4. No. 6, September 1986. http://www.cis.ohio-state.edu/ jain/papers/ftp/train.pdf.

[KMFB04] Thomas Karagiannis, Mart Molle, Michalis Faloutsos, and Andre Broido. A nonstationary poisson view of internet traffic. In *IEEE INFOCOM*, Hong Kong, March 2004.

[Koh00] Teuvo Kohonen. *Self-Organizing Maps*. Springer-Verlag, 2000.

[Lam01] Timo Lampinen. Neurosumeiden analyysimenetelmien soveltaminen netflow-verkonhallintadatan klusterointiin (finnish). Master's thesis, Tampere University of Technology, June 2001.

[LB03] Thomas Lindh and Nevil Brownlee. Integrating active methods and flow meters - an implementation using netramet. Passive and Active Measurement Workshop (PAM), April 2003. http://moat.nlanr.net/-PAM2003/PAM2003papers/3726.pdf.

[LTWW93] Will E. Leland, Murad S. Taqq, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of Ethernet traffic. In Deepinder P. Sidhu, editor, *ACM SIGCOMM*, pages 183–193, San Francisco, California, 1993.

[Luo00] Marko Luoma. *Simulation studies of Differentiated Services Networks*. Lic. thesis, Department of Electrical and Communications Engineering, Helsinki University of Technology, October 2000. http://www.netlab.hut.fi/u/mluoma/publications/#Licentiate.

[Mal98] Gary Scott Malkin. Rip version 2. RFC2453, November 1998. http://www.ietf.org/rfc/rfc2453.txt?number=2453.

[Mil92] David L. Mills. Network time protocol (version 3). RFC1305, 1992. http://www.ietf.org/rfc/rfc1305.txt?number=1305.

[Min99] Nelson Minar. A survey of the ntp network, December 1999. http://www.media.mit.edu/ nelson/research/ntp-survey99/.

[ML00] Robert Morris and Dong Lin. Variance of aggregated web traffic. In *INFOCOM (1)*, pages 360–366, 2000.

[Moy98] John Moy. Ospf version 2. RFC2328, April 1998. http://www.ietf.org/rfc/rfc2328.txt?number=2328.

[PA00] Vern Paxson and Mark Allman. Computing tcp's retransmission timer. RFC2988, November 2000. http://www.ietf.org/-rfc/rfc2988.txt?number=2988.

[Peu02] Markus Peuhkuri. *Internet traffic measurements – aims, methodology, and discoveries*. Lic. thesis, Department of Electrical and Communications Engineering, Helsinki University of Technology, May 2002. http://keskus.hut.fi/u/puhuri/publications/li.shtml.

[Pos80] Jon Postel. User datagram protocol. RFC768, August 1980. http://www.ietf.org/rfc/rfc0768.txt?number=768.

[Pos81a] Jon Postel. Internet control message protocol. RFC792, September 1981. http://www.ietf.org/rfc/rfc0792.txt?number=792.

[Pos81b] Jon Postel. Internet protocol (darpa internet program protocol specification. RFC791, September 1981. http://www.ietf.org/-rfc/rfc0791.txt?number=791.

[Pos81c] Jon Postel. Transmission control protocol. RFC793, September 1981. http://www.ietf.org/rfc/rfc0793.txt?number=793.

[PR85] Jon Postel and Joyce Reynolds. File transfer protocol (ftp). RFC959, October 1985. http://www.ietf.org/rfc/rfc0959.txt?number=959.

[QC02] Zhi Quan and Jong-Moon Chung. Impact of self-similarity on performance evaluation in differential service networks. In *Proceedings of the 45th IEEE International Midwest Symposium on Circuits and Systems (IEEE MWSCAS'02)*, volume 2, pages 326–329, Tulsa, Oklahoma, 2002 2002.

[Rah91] Kauko Rahko. Measurements for control and modelling of teletraffic. In A. Jensen and V. B. Iversen, editors, *Teletraffic and Datatraffic in a Period of Change*, pages 609–614. ITC-13 (International Teletraffic Congress), June 1991.

[RL95] Yakov Rekhter and Tony Li. A border gateway protocol 4 (bgp-4). RFC1771, March 1995. http://www.ietf.org/rfc/rfc1771.txt-?number=1771.

[SCFJ03] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, and Van Jacobson. Rtp: A transport protocol for real-time applications. RFC3550, July 2003. http://www.ietf.org/rfc/rfc3550.txt-?number=3550.

[Sec96] ITU-T (International Telecommunication Union Telecommunication Standardization Sector). Definitions and terminology for syncronization networks, August 1996.

[Smo03] Vladimir Smotlacha. One-way delay measurement using ntp synchronization. Technical report, CESNET, 2003. http://www.ces.net/-project/qosip/publications/2003/tnc2003-NTP.ps, Presented in TERENA Networking Conference 2003.

[Som03] Bill Sommerfeld. Secure shell (secsh), November (latest update) 2003. http://www.ietf.org/html.charters/secsh-charter.html.

[SV01] B. Sikdar and K. S. Vastola. On the contribution of TCP to the self-similarity of network traffic. *Lecture Notes in Computer Science*, 2170:596–??, 2001.

[Tho96] Gary A. Thom. H.323: the multimedia communications standard for local area networks. In *IEEE Communications Magazine*, volume 34(12), pages 52–56. December 1996.

[WD01] Ronald Wyllys and Philip Doty. Notes on the 5-layer and 7-layer models of interconnection, February 2001. http://www.gslis.-utexas.edu/Ĩ38613dw/readings/NotesOnInterconnection.html.

[Wro00] John Wroclawski. Integrated services (intserv). IETF Charter, September 2000. http://www.ietf.org/html.charters/intserv-charter.html.

[WTSW97] Walter Willinger, Murad S. Taqqu, Robert Sherman, and Daniel V. Wilson. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, 1997.

[Y.199] ITU-T Recommendation Y.1540. Internet protocol data communication service - ip packet transfer and availability performance parameters, February 1999. http://www.itu.int/itudoc/itu-t/aap/sg13aap/-history/y1540/.

[Zim80] Hubert Zimmermann. Osi reference model — the iso model of architecture for open systems interconnection. In *IEEE Transactions on Communications*, pages 425–432. COM-28, April 1980.

[ZL03] Marcia Zangrilli and Bruce B. Lowekamp. Comparing passive network monitoring of grid application traffic with active probes. In *Proceedings of the 4th International Workshop on Grid Computing (GRID2003)*, November 2003.