Analysis of the Delay in the SURFnet Network

by

ALBERTO CASTRO HINOJOSA

Master Thesis

Supervisors:

Dr. ir. A. Pras (INF/DACS)
Dr. ir. P.T. de Boer (INF/DACS)
Dr. I. Soto Campos (Universidad Carlos III de Madrid)

Design and Analysis of Communication Systems
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente

September 2005, Enschede (The Netherlands)
“Let me tell you the secret that has led me to my goal. My strength lies solely in my tenacity”.

Louis Pasteur
French biologist & bacteriologist (1822 - 1895)
Abstract

SURFnet is a high-grade computer network specially reserved for higher education and research in The Netherlands. Some of the being used services are conferencing (Internet using a video, audio and/or data connection) and streaming technology (offers its users the possibility of watching or listening to a video or audio file while it is being downloaded). This kind of services has very concrete requirements of QoS that need to be guaranteed. One of them is the delay.

The goal of this M.Sc. project is to find the best delay figure (or groups of figures) for evaluating the “health” of a network. Our approach is to perform passive measurements at TCP/IP level, because we do not want to inject traffic in the network. We used the data from the M2C repository to extract the delay, since it was not possible to do the required measurements in real-time.

We focus on the round trip delay as our main metric to quantify latency. We investigate three groups of RTT figures; these figures have been proposed in literature and show RTT, its variability and its relationship with the number of hops. We compare these figures using the same data, to get an idea of the advantages and drawbacks of each of them.

Our results show that we are able to infer the performance of the network based on passive measurements of the delay and that all figures complement each other.

Keywords: Delay, passive measurements, round trip time, packets monitoring, TCP/IP, Internet, network’s measurements, SURFnet.
Preface

This report is the result of 7 months (March – September 2005) master assignment in the chair Design and Analysis of Communication Systems (DACS), Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) in the University of Twente (The Netherlands), under the supervision of Dr.ir. Aiko Pras (first supervisor), Dr.ir. Pieter-Tjerk de Boer and Dr. Ignacio Soto Campos.

Chapter 1 contains an introduction of the assignment and background information about the SURFnet network, delay and traffic measurements. Chapter 2 presents the state-of-the-art in passive delay measurements read from the books and papers. Chapter 3 includes the main work of the project with all the results and figures obtained, and Chapter 4 completes this thesis and it contains the conclusions and the future work about the developed research.
Acknowledgments

This project is the last step in my way before getting my degree in Telecommunications Engineering at the University Carlos III of Madrid. It has taken me many years working very hard and studying alone and sometimes without enough courage to keep going. That's why, I would like to dedicate this project to the people who always have been close to me, encouraging me during difficult moments, such as exams' months.

To you mum, thanks for giving me what I have always needed. I have no words to express what you signify for me. To Mónica, my sister, who was always visiting me in my room to encourage me. I would like you could also read this dad, I know that you would be proud of me. I love you all. To my grandmother Nati, for teaching me the necessity of always making a good use of the time, thanks. To María, the person who better understands the meaning of this project, because we have arrived side by side till the very end. I would not have achieved it without you. Thank you for helping me always. I love you. Of course I cannot forget to cite here the rest of my family, who were always interested in the progress of my studies (special thanks to my brother in law Luis, who listens to my university’s stories very often).

I would also like to thank to my university’s classmates for all their help, because we have shared many hours together and unforgettable moments. Thanks to Jose, Juan Carlos, Fran (thanks a lot for the English’s proof-reading!), Almudena, Kike, Rebeca, Carlos and the rest of the nice people who I have met at the University Carlos III of Madrid.

To my friends Tello (the answer to your question is 26), Julio, Jaime, my companions of the “mechanical orange” and rest of friends of Miraflores de la Sierra (Fernando, Julia, Irene, Tony...) thanks for being always there.

The saddest thanks to Miguel, one of my best friends who unfortunately I will never see him again. I hope you share with me this moment, wherever you are. I miss you.

To all the fantastic people that I met in Enschede and who helped me to spend very nice moments in this seven months far of my home: Marta, Nayeli, Tuomas, BRo, Fix, Antoine, Maher, Ruth, Asia, Ania, Kasia, Sylvie, Salvo, Chema, Pep, Hui, Kelvin, Kemal, Hasan, Johannes, Grace, Estela, Mariano, Federico... WBW 399 Forever!

I have had the opportunity to complete my studies, accomplishing my final project, at the University of Twente (Enschede, The Netherlands) as an Erasmus student and I want to acknowledge to my supervisor, Aiko Pras, for the manner that he offered me during my stay and for teaching me how to research in a very independent form. I also want to thank Pieter-Tjerk De Boer, Tiago Fioreze and Ignacio Soto Campos for the given help whenever I have needed it.
Analysis of the Delay in the SURFnet Network

Contents

ABSTRACT .................................................................................................................................................. 4
PREFACE .................................................................................................................................................... 6
ACKNOWLEDGMENTS ................................................................................................................................. 8
LIST OF FIGURES ...................................................................................................................................... 12
LIST OF TABLES ....................................................................................................................................... 14
ACRONYMS ................................................................................................................................................ 16

1. INTRODUCTION ..................................................................................................................................... 18
  1.1 Background .......................................................................................................................................... 19
    1.1.1 SURFnet Network ......................................................................................................................... 19
    1.1.2 Delay ............................................................................................................................................ 22
      1.1.2.1 Definition ................................................................................................................................. 22
      1.1.2.2 Motivation: VoIP .................................................................................................................... 24
    1.1.3 Active vs. Passive Traffic Measurements .................................................................................... 26
  1.2 Research Question .............................................................................................................................. 28
  1.3 Approach ............................................................................................................................................ 29
  1.4 Outline of the Report .......................................................................................................................... 29

2. STATE-OF-THE-ART ............................................................................................................................. 30
  2.1 Terminology ........................................................................................................................................ 30
    2.1.1 About General Measurements Issues .......................................................................................... 30
    2.1.2 One Way Delay (OWD) ................................................................................................................. 31
    2.1.3 Round Trip Time Delay (RTT) ....................................................................................................... 32
    2.1.4 Delay Variation, Jitter or IPDV (IP Packet Delay variation) ......................................................... 33
  2.2 About RTT Measurements ................................................................................................................... 34
    2.2.1 RTT Estimation Techniques ......................................................................................................... 34
    2.2.2 Some Figures which Use RTT Measurements ............................................................................. 37
    2.2.3 Other RTT Issues .......................................................................................................................... 40
    2.2.4 Network’s Health Candidates Figures .......................................................................................... 41
  2.3 The Data Repository ............................................................................................................................ 42
    2.3.1 Description ................................................................................................................................... 42
    2.3.2 Locations under Study ................................................................................................................... 43
  2.4 The RTT Measurement Tool: Tcptrace .................................................................................................. 43
    2.4.1 Why Tcptrace? ............................................................................................................................... 43
    2.4.2 Valid RTT Samples: Extraction Process ....................................................................................... 44
    2.4.3 Considerations ............................................................................................................................... 47

3. SEARCHING THE NETWORK'S HEALTH FIGURES ............................................................................. 50
  3.1 Introduction ........................................................................................................................................... 50
  3.2 RTT Figures .......................................................................................................................................... 50
    3.2.1 About RTT Figures ....................................................................................................................... 50
    3.2.2 CDF of the RTT in Terms of TCP Connections ......................................................................... 51
    3.2.3 CDF of the RTT at Different Time Scales .................................................................................. 55
    3.2.4 Frequency Distribution of the RTT ............................................................................................... 61
    3.2.5 Conclusions about RTT Figures ................................................................................................. 63
  3.3 RTT Variation Figures .......................................................................................................................... 63
    3.3.1 About RTT Variation Figures ....................................................................................................... 63
    3.3.2 RTT Ratios ................................................................................................................................... 63
    3.3.3 RTT Variability using the Standard Deviation ............................................................................. 69
    3.3.4 Jitter ............................................................................................................................................ 71
List of Figures

Figure 1.1.1, SURFnet Network ............................................................... 20
Figure 1.1.2, A new networking s-curve is developing ...................... 21
Figure 1.1.3, Voice compression impairment .................................... 25
Figure 1.2.1, Average RTT SURFnet backbone ................................. 28
Figure 2.1.1, Round Trip Time ............................................................. 33
Figure 2.2.1, SYN RTT ................................................................. 36
Figure 2.2.2, Example of RTT distribution in terms of connections .... 37
Figure 2.2.3, \{max, 90\%, med\} RTT / min RTT .................................. 38
Figure 2.2.4, Comparison of the minimum and median RTTs a connection observes ............................................................... 39
Figure 2.2.5, Minimum RTT against hops ......................................... 40
Figure 2.3.1, Flow chart of ack_in function ...................................... 46
Figure 2.3.2, Flow chart of rtt_ackin function ................................... 47
Figure 2.4.3, The measurement point problem .................................. 48
Figure 3.2.1 a), CDF of RTT in Location 1 ......................................... 52
Figure 3.2.1 b), CDF of RTT in Location 1 (Logarithmic) .................... 53
Figure 3.2.1 c), CDF of RTT in Location 2 ......................................... 53
Figure 3.2.1 d), CDF of RTT in Location 2 (Logarithmic) .................... 54
Figure 3.2.1 e), CDF of RTT in Location 3 ......................................... 54
Figure 3.2.1 f), CDF of RTT in Location 3 (Logarithmic) .................... 55
Figure 3.2.2, CDF comparison at different hours in the same day (Location 1) ............................................................... 56
Figure 3.2.3, CDF comparison of different days in a week in the same hour (Location 1) ............................................................... 57
Figure 3.2.4, CDF comparison of two Tuesdays at the same hour in different months (Location 1) ............................................................... 57
Figure 3.2.5, CDF comparison of different hours (Location 2) ............... 58
Figure 3.2.6, CDF comparison of different days in a week in the same hour (Location 2) ............................................................... 58
Figure 3.2.7, CDF comparison of average RTT in three months (Location 2) ............................................................... 59
Figure 3.2.8, CDF comparison at different hours in the same week (Location 3) ............................................................... 60
Figure 3.2.9, CDF comparison of different months (Location 3) ............ 60
Figure 3.2.10 a), Frequency of RTT samples in Location 1 .................. 61
Figure 3.2.10 b), Frequency of RTT samples in Location 2 .................. 62
Figure 3.2.10 c), Frequency of RTT samples in Location 3 .................. 62
Figure 3.3.1 a), Avg. RTT/min. RTT vs. min RTT (Location 1) ............ 64
Figure 3.3.1 b), Avg. RTT/min. RTT vs. min RTT (Location 2) ............ 64
Figure 3.3.1 c), Avg. RTT/min. RTT vs. min RTT (Location 3) ............ 65
Figure 3.3.2 a), Ratios avg. RTT/min. RTT and max. RTT/min RTT CDF (Location 1) ............................................................... 66
Figure 3.3.2 b), Ratios avg. RTT/min. RTT and max. RTT/min RTT CDF (Location 2) ............................................................... 66
Figure 3.3.2 c), Ratios avg. RTT/min. RTT and max. RTT/min RTT CDF (Location 3) ............................................................... 67
Figure 3.3.3 a), Ratio’s Frequencies (Location 1) ................................ 67
Figure 3.3.3 b), Ratio’s Frequencies (Location 2) ................................. 68
Figure 3.3.3 c). Ratio's Frequencies (Location 3)......................................... 68
Figure 3.3.4 a), Std. deviation vs. average RTT – minimum RTT in Location 1.................................................................................................................. 69
Figure 3.3.4 b), Std. deviation vs. average RTT – minimum RTT in Location 2.................................................................................................................. 70
Figure 3.3.4 c), Std. deviation vs. average RTT – minimum RTT in Location 3.................................................................................................................. 70
Figure 3.3.5, CDF of the standard deviation.................................................................................................................. 71
Figure 3.3.6, CDF of maximum RTT – minimum RTT.................................................................................................................. 72
Figure 3.3.7 a), Frequency of average RTT - minimum RTT (Location 1).... 72
Figure 3.3.7 b), Frequency of average RTT - minimum RTT (Location 2).... 73
Figure 3.3.7 c), Frequency of average RTT - minimum RTT (Location 3).... 73
Figure 3.4.1, Frequency distribution of the TTL values (Location 1)........ 78
Figure 3.4.2, Distribution of the initial TTL estimation (Location 1)......... 79
Figure 3.4.3 a), Hops' number distribution (Location 1).......................... 80
Figure 3.4.3 b), Hops' number distribution (Location 2).......................... 80
Figure 3.4.3 c), Hops' number distribution (Location 3).......................... 81
Figure 3.4.4 a), Min. RTT vs. hop’s number during two different days at different hours (Location 1).................................................. 82
Figure 3.4.4 b), Avg. RTT vs. hop’s number during two different days at different hours (Location 1).................................................. 82
Figure 3.4.5, Min. And Avg. RTT vs. hop’s number (Location 1).......... 83
Figure 3.4.6 a), Min. RTT vs. hop’s number during a week at different hours (Location 2).................................................................................................................. 83
Figure 3.4.6 b), Avg. RTT per hop during a week at different hours (Location 2).................................................................................................................. 84
Figure 3.4.7, Min. And Avg. RTT per hop (Location 2).......................... 84
Figure 3.4.8 a), Min. RTT vs. hop’s number during a week at different hours (Location 3).................................................. 85
Figure 3.4.8 b), Avg. RTT per hop during a week at different hours (Location 3).................................................................................................................. 85
Figure 3.4.9, Min. And Avg. RTT vs. hop’s number (Location 3).......... 86
Figure 3.4.10, Comparison of the Min. RTT vs. hop’s number for all the locations.......................................................... 87
Figure 3.4.11, Comparison of the Avg. RTT vs. hop’s number for all the locations.......................................................... 87
Figure 3.4.12, Comparison of the Avg. RTT less Min. RTT vs. hop’s number for all the locations.......................................................... 88
Figure 3.4.13, Comparison of the Min. RTT / hop’s number for all the locations.......................................................... 89
Figure AppB. 1, CDF of the Ratio Min. RTT / SY N RTT............................... 104
List of Tables

Table 1,  
Delay Specifications........................................................................................................... 26
Table 2,  
Minimum RTT vs. Geographical Areas................................................................. 50
Table 3,  
Percentage of connections in each geographical zone............................... 55
Table 4,  
Inferred Operating System Packet Distribution........................................ 75
Table 5,  
Relation RTT vs. Hops Number for each POP.................................................. 77
Table 6,  
Relation RTT vs. Hops Number for some Universities all over the world.......................................................................................................................... 77
Acronyms

ACK, Acknowledgment
AS, Autonomous System
ATM, Asynchronous Transfer Mode
BDP, Bandwidth-delay product
BSD, Berkeley Software Distribution
CDF, Cumulative Distribution Function
CPU, Central Processing Unit
DF, Do not Fragment
DWDM, Dense Wavelength-Division Multiplexing
FEC, Forward Error Correction
GigaPort NG, GigaPort Next Generation Network
GPS, Global Positioning System
HFC, Hop-Count Filtering
ICMP, Internet Control Message Protocol
IP, Internet Protocol
IPPM, IP Performance Metrics
IPv4, Internet Protocol version 4
IPv6, Internet Protocol version 6
IP2HC, IP-to-Hop-Count
IQR, Interquartile Range
ITU, International Telecommunication Union
MSS, Maximum Segment Size
M2C, Measuring, Modelling and Cost Allocation
NACK, Negative Acknowledgment
NTP, Network Time Protocol
OS, Operating System
OWD, One Way Delay
PAM, Passive and Active Measurements Workshop
PCM, Pulse Code Modulation
PoPs, Points of Presence
QoS, Quality of Service
RFC, Request for Comments
RTT, Round Trip Time
RTT FNH, Round Trip Time as a Function of the Number of Hops
SA, SYN-ACK estimation
SONET, Synchronous Optical Network
SS, Slow-Start estimation
TCP, Transmission Control Protocol
TTL, Time To Live
UDP, User Datagram Protocol
UT, Universal Time or University of Twente
UTC, Coordinated Universal Time
VoIP, Voice over IP
WG, Working Group
WTCW, Wetenschap & Technologie Centrum Watergraafsmeer
Chapter 1

Introduction

If you are involved in the operation of an IP network, a question you may hear is: “How good is your network?” Or, in other words, “how can you measure and monitor the quality of the service that you are offering to your customers?” and “how can your customers monitor the quality of the service you provide them?”. Ultimately, we are interested in obtaining a method for evaluating the health of the network.

In the Internet, end hosts divide data into packets that flow through the network independently. In forwarding packets toward their destinations, the network routers usually do not retain information about ongoing transfers and do not provide fine-grain support for performance guarantees. As a result, packets may be corrupted, lost, delayed, or delivered out of order. This complicates the efforts of network operators to provide predictable communication performance for their customers. Rather than having complexity inside the network, the end hosts have the responsibility for the reliable, ordered delivery of data between applications. Implemented on end hosts, the Transmission Control Protocol (TCP) plays an crucial role in providing these services and adapting to network congestion. Inside the network, the routers implement routing protocols that adapt to equipment failures by computing new paths for forwarding IP packets. These automatic and distributed reactions to congestion and failures make it difficult for network operators to detect, diagnose, and fix potential problems (e.g. high delay links). The ability to detect, diagnose, and fix problems depends on the information available from the underlying network.

When outage or service degradation are likely to occur in a network, users begin to seek ways to characterize the quality of the service they get. The qualitative state of the Internet is currently difficult to estimate due to lack of such metrics and methods that provide objective information. Thus there is a high demand for both qualitative and quantitative metrics along with suitable measurement tools.

A functional description of network performance encompasses a description of speed, capacity, and distortion of transactions that are carried across the network. If it is known the latency, available bandwidth, loss, and jitter rates as a profile of network performance between two network end points, as well as the characteristics of the network transaction, it is possible to make a reasonable prediction relating to the performance of the transaction.

Given these performance indicators, the next step is to determine how these indicators may be measured, and how the resulting measurements can be meaningfully interpreted. There are two basic approaches to this task. One is to collect management information from the active elements of the network using a management protocol, and from this information make some inferences about network performance; or we can simply do this by monitoring the
packets coursing a link. This can be termed a passive approach to performance measurement, in that the approach attempts to measure the performance of the network without disturbing its operation. The second approach is to use an active approach and inject test traffic into the network and measure its performance in some fashion, and relate the performance of the test traffic to the performance of the network in carrying the normal payload.

In this M.Sc. assignment, we will focus in one of these performance indicators, the packet delay. We will use passive measurements as main method to obtain such delay, mainly from an available data repository ([8]) of the SURFnet network, our network under study. We will investigate the available information about the network’s performance with the resulting delay measurements.

Section 1.1 presents the background information about the SURFnet network, an introduction to the traffic measurements, the delay problem and its motivation. Section 1.2 describes the goal of this assignment. Section 1.3 shows how the first approach of the problem (the starting point) has been done. Finally, section 1.4 gives the structure of this thesis.

1.1 Background

1.1.1 SURFnet Network

We present in this section our network under study, though the research done in this project can be applied to whatever TCP/IP network. What is SURFnet?

SURFnet1 [1] is the advanced research broadband network infrastructure and organization in The Netherlands that is funded by member institutions and government grants. SURFnet is part of the GigaPort Project [2], an initiative of the Dutch government, universities, research organizations, and businesses that offers incentives for development of information and communications technologies to give The Netherlands a lead in the development and use of advanced and innovative Internet technology.

SURFnet5 is currently the production network built in the GigaPort Project and connects the networks of universities, polytechnics, research centers, academic hospitals and scientific libraries to one another and to other networks in Europe and the rest of the world. SURFnet is part of the world wide Internet. This network also offers companies and institutions a state-of-the-art test environment for new (network) services. Speed, reliability and security of the network are key issues.

The SURFnet5 network consists of a dark fiber core (the heart of the backbone) that is situated at two locations in Amsterdam, at SARA Reken and Netwerkdiensten in WTCW, the Wetenschap & Technologie Centrum Watergraafsmeer in Amsterdam-Oost, and at a BT site at the Hemptpoint

---

1 Most of these fragments of text have been copied directly from different parts of [1] and [2], as a resume way.
industrial estate in Amsterdam-West. Nineteen type 12416 Cisco routers have been placed within the SURFnet5 network; both core locations host two routers (the so-called Core Routers) and fifteen at the concentrator locations (the so-called Connection Routers). The four routers in the core are interconnected in a square. The two core locations are sufficiently distant for the entire SURFnet5 network to remain functioning on one location if the other should fail due to local calamities. Its dual realization on each location also serves to prevent failure of one location if a router fails there. Fifteen Points of Presence (PoPs) are connected to the core routers (see Figure 1.1.1). These PoPs are situated at SARA, the universities of Delft, Eindhoven, Enschede, Groningen, Leiden, Maastricht, Nijmegen, Tilburg, Utrecht and Wageningen, at the polytechnics of Den Haag, Rotterdam and Zwolle, and at the NOB in Hilversum. These PoPs have separate links to each of the backbone locations, which ensures resilience: one connection is always maintained in case of a single line disruption.

![Figure 1.1.1- SURFnet Network](Source: www.surfnet.nl)

SURFnet5 makes use of IP-over-DWDM and has connections of 10 Gbps. Transmission in a fibre-optic cable occurs via light pulses. The DWDM protocol (Dense Wavelength-Division Multiplexing) divides this light in a large number of colours, allowing the capacity of both the existing and the new fibre-optic cables to be increased considerably. The network also uses the latest Cisco software, which simultaneously supports IPv4 and IPv6.

SURFnet started increasing the number of PoPs in the SURFnet5 network at the end of 2001. With GigaPort funding the fifteen current PoPs are extended with ten additional PoPs. The aim is to increase the density of SURFnet5, reducing the physical distance from the institutions to the network. This makes the roll-out of fibre-optics over the last stretch from the institutions to SURFnet5 more cost-
efficient. The ten additional connection points are connected to the fifteen larger PoPs over two separate lines.

The volume of data transported on the successive SURFnet networks grows continuously in a steady pace (traffic growth is about 150% per year)\(^2\) [33]. To accommodate for this traffic growth and to provide new network functionality, it is essential that SURFnet introduces a new generation network every four years. Since its start in 1989 the network architecture has not changed fundamentally from that of the first generation Internet infrastructure. While the topology, the transmission speed and the framing protocols have all been changed, routers can still be found at every Point of Presence and transmission is directly coupled to these routers. It has become evident that a next generation Internet cannot be an extrapolation of this architecture. The main cause for this is that costs for routers continually increase while costs for bandwidth decrease. Routers will always play an essential part in the transport of data on the network and IP level; they form the basis of end-to-end connections. However, there is an imminent need for decreasing the amount of routers. This calls for a new architecture, with a more prominent role for switching and optical technologies, and new developments in routing, e.g. IPv6 and multicast. Since 2002 experiments with the concept of light paths and lambda switching have been carried out. Lambdas are the new technology pushing networking possibilities forwards (see Figure 1.1.2).

![Figure 1.1.2- A new networking s-curve is developing](Source: www.surfnet.nl)

Lambda-based networking [11] is ultimately about using different “colors” or wavelengths of (laser) light in fibers for separate connections. Each wavelength is called a “lambda”. Current coding schemes allow for typically 10 Gbps to be encoded by a laser on a high-speed network interface. In lambda networking, the goal is to achieve ultimate Quality of Service by giving applications and user communities their own sets of lambdas on a shared (dark) fiber infrastructure; thus, isolating the different communities from each other. The

\(^2\) Most of these fragments of text have been copied directly from different parts of [33] and [11], as a resume way.
implementation requires DWDM to accommodate many wavelengths on a fiber, optical switches, and other optical networking equipment. A LambdaGrid requires the interconnectivity of optical links, each carrying one or more lambdas, or wavelengths, of data, to form on-demand, end-to-end “light paths”, in order to meet the needs of very demanding e-science applications. Lambda-based networking is not constrained by traditional framing, routing, and transport protocols and provide excellent quality on point-to-point connections at very high speed (1-10Gbps).

The current SURFnet5 network is scheduled to be replaced by SURFnet6, a hybrid optical and packet switching infrastructure, in 2005. SURFnet6 (that is being developed in the GigaPort Next Generation Network [33]) will be a fully operational congestion-free world leading network infrastructure for higher education and research in The Netherlands, and will serve as a test bed for research on the scaling-up of new network technologies. It will include congestion-free and low latency connections with other research networks and the general purpose Internet. SURFnet6 will deliver unicast and multicast services both on IPv4 and IPv6 to all of its users, as well as lambda services for the demanding users. These services will be delivered over a single fiber transmission infrastructure. Transmission rates of up to 100Gbps are envisioned in the production SURFnet6 network. The use of lambdas within the network will ensure seamless communication to all parts of the Internet; hence the use of lambdas will not create islands disconnected from the Internet.

Today, a small but increasing group of high-end users needs ultra high-bandwidth, point-to-point connectivity. For example, radio astronomers that want to interconnect radio telescopes around the globe, high-energy physics scientists using data replication to distribute the analysis burden and medical scientists researching data base correlations. Dedicated light paths can serve these Grid and e-Science applications better than traditional IP networks, as their performance characteristics are critical and much more controlled. From a network provider point of view, using light paths is desirable since large point-to-point data streams can be split off from the expensive routed IP layer in order to improve the economics. Transporting the large dedicated volume of traffic in the optical or switched layer is cost-effective, and reduces its impact on the performance of the routed IP layer.

1.1.2 Delay

1.1.2.1 Definition

As this thesis is called “Analysis of the Delay in the SURFnet Network”, and we have described in section 1.1.1 what such a network is like, the next step is to define the delay (it is called latency as well), although we probably have a previous idea of this topic.

A general definition of network delay, following [4], [5] and [6], is “the time between when the first part (e.g. the first bit) of an object (e.g. a packet) passes an observational position (e.g. where a host’s network interface card connects to the wire) and the time the last part (e.g. the last bit) of that object
or a related object (e.g. a response packet) passes a second (it may be the same point) observational point”.

The network delay can be further split up into several components:

- The **propagation delay** (of 5 μs per km) is the delay to transport information over the links of the networks.
- The **packet processing delay** consists of all delays needed to process the packet in the network nodes. This includes route look-up delay, delay due to the **Forward Error Correction** (FEC) process, etc.
- The **serialization delay** (also transmission delay) is the delay a node requires to put all bits associated with a packet on the link. This delay is proportional to the packet size (including all overhead bits) and is inversely proportional to the link rate.
- The **queuing delay** is due to the fact that in packet-based nodes a packet possibly has to wait for other packets before it can be put on the link. This delay may differ from packet to packet and is also the cause of jitter.

We can also consider the delay due to the server response, especially when we are measuring round trip time delays, but actually we are not going to discuss the different delay components, because we will obtain global delay measurements. So, basically we can simplify the delay components in two: the **minimum delay** (sum of propagation, serialization and packet processing delays) and the **queuing delay**.

We will present what kind of measurements are usually used to characterize the network delay in the **Chapter 2** (RTT, OWD and Jitter). We advance now that we will focus our work on **RTT measurements**, basically due to their easiness of measurement.

Why is it necessary to measure the delay? As we can also read in [5] and [6], delay of a packet from a source host to a destination host is useful for several reasons:

- “Some applications do not perform well (or at all) if end-to-end delay between hosts is large relative to some threshold value”. We can think, for example, in a voice call across the Internet, where an excessive value of delay between the end hosts can result annoying.
- “Erratic variation in delay makes it difficult (or impossible) to support many real-time applications”. Continuing with the previous example, it is desirable that such delay does not change too much, in order to maintain a normal conversation.

---

3 **Forward Error Correction** (FEC) is a type of error correction which improves on simple error detection schemes by enabling the receiver to correct errors once they are detected. This reduces the need for retransmissions. FEC works by adding check bits to the outgoing data stream. Adding more check bits reduces the amount of available bandwidth, but also enables the receiver to correct for more errors. **Forward Error Correction** is particularly well suited for satellite transmissions, where bandwidth is reasonable but latency is significant.
“The larger the value of delay, the more difficult it is for transport-layer protocols to sustain high bandwidths”. TCP cannot send a new segment until one of the previous acknowledgements has been received, when the window size is full. So, the larger the value of delay is, the more time TCP has to wait to send a new segment.

“The minimum value of this metric provides an indication of the delay due only to propagation and transmission delay”. Some packet should find the path to its destination with congestion free (without spending too much time in router's queues). We also have to add the packet processing delay in each node.

“The minimum value of this metric provides an indication of the delay that will likely be experienced when the path traversed is lightly loaded”.

“Values of this metric above the minimum provide an indication of the congestion present in the path”. That's why this metric is going to be very important for us, it can be used as a threshold value for the best network path performance.

Nowadays, new world applications, such as voice and video, are more susceptible to changes in the transmission characteristics of data networks. It is imperative to understand the traffic characteristics of the network before deployment of these applications to ensure successful implementations. We realize then the usefulness to find ways to characterize the network delay. For example, multimedia applications generate and consume nonstop data flows in real time. These contain important quantities of audio, video and more time's dependent data elements, and the processing and delivering in time for the individual elements of data (low latency) are essential.

1.1.2.2 Motivation: VoIP

As an example of the delay’s value importance in these new multimedia applications, we discuss in this section some topics about Voice over IP (VoIP).

One possible definition for VoIP can be: “Voice over IP (also called VoIP, IP Telephony, and Internet telephony) is the routing of voice conversations over the Internet or any other IP network. The voice data flows over a general-purpose packet-switched network, instead of the traditional dedicated, circuit-switched voice transmission lines. One advantage of VoIP is that the telephone calls over the Internet do not incur a surcharge beyond what the user is paying for Internet access, much in the same way that the user does not pay for sending individual e-mails over the Internet”.

As we can read in [34], we have here more components of delay: Coder or Processing Delay (to compress a block of PCM samples), Algorithmic Delay (compression algorithm to correctly process a sample block), Packetization Delay (time taken to fill a packet payload with encoded/compressed speech), Queuing/Buffering, Serialization Delay, Network Delay (Public Frame) and De-jitter Buffer Delay (de-jitter buffer transforms the variable delay into a fixed delay). Jitter is the variation in delay over time from point-to-point. If the delay of transmissions varies too widely in a VoIP call, the call quality is greatly

The amount of jitter tolerable on the network is affected by the depth of the jitter buffer on the network equipment in the voice path. The more jitter buffer available, the more the network can reduce the effects of jitter. The processing delay is caused by the process of encoding and collecting the encoded samples into a packet for transmission over the packet network.

VoIP is susceptible to network behaviors, referred to as delay and jitter, which can degrade the voice application to the point of being unacceptable to the average user. Delay causes two problems: echo and talker overlap. Echo is caused by the signal reflections of the speaker’s voice from the far-end telephone equipment back into the speaker’s ear. Echo becomes a significant problem when the round trip delay becomes greater than 50 milliseconds. Talker overlap (or the problem of one talker stepping on the other talker’s speech) becomes significant if the One Way Delay becomes greater than 150-200 milliseconds. The end-to-end delay budget is therefore the major constraint and driving requirement for reducing delay through a packet network.

What quality is considered acceptable in a VoIP call? As with most human factors, everyone has his or her own opinion on this issue. However, there is a definite limit of quality degradation that will be tolerated by users. The E-model [7] has been used as a computational tool to predict the subjective quality of a telephone call based on how it characterizes transmission parameters. The model combines the impairments caused by these transmission parameters into rating R, which ranges between 0 and 100. Figure 1.1.3 shows E-model rating R to categories of speech transmission quality and to user satisfaction. R below 50 indicates unacceptable quality. All connections below R=70 will suffer from some combination of distortion and long delay. The region between R=50 and R=70 encompasses the “Many users dissatisfied” and the “Nearly all users dissatisfied” (Exceptional limiting case) categories and therefore deserves the low quality. An acceptable quality category is then bounded by a lower limit of R=70. Figure 1.1.3 illustrates the point by comparing the best-case curves for three popular IP codecs, G.711, G.729A and G.723.1.

![Figure 1.1.3- Voice compression impairment](Source: [7])
“How much delay is too much? Delay does not affect speech quality directly, but instead affects the character of a conversation. Below 100ms, most users will not notice the delay. Between 100ms and 300ms, users will notice a slight hesitation in their partner’s response. Beyond 300ms, the delay is obvious to the users and they start to back off to prevent interruptions”, [7].

The International Telecommunication Union (ITU) considers network delay for voice applications in Recommendation G.114 (see [35]). This recommendation defines three bands of one way delay as shown in Table 1.

<table>
<thead>
<tr>
<th>Range in Milliseconds</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-150</td>
<td>Acceptable for most user applications.</td>
</tr>
<tr>
<td>150-400</td>
<td>Acceptable provided that administrators are aware of the transmission time and the impact it has on the transmission quality of user applications.</td>
</tr>
<tr>
<td>Above 400</td>
<td>Unacceptable for general network planning purposes. However, it is recognized that in some exceptional cases this limit is exceeded.</td>
</tr>
</tbody>
</table>

Table 1- Delay Specifications

We would be able to continue talking about different applications that need a moderate delay to work properly. This fact has motivated the interest in the measuring and analyzing of the networks’ latency. Instead of studying all kind of applications in top layers protocols, we will study the delay at TCP level, because is widely used and the end-to-end performance observed by TCP transfers is a much closer match to the service Internet users actually obtain from the network.

1.1.3 Active vs. Passive Traffic Measurements

Now that we know what we want to measure (delay) and the network where we want to perform the measurements (SURFnet), we need to know the existing possibilities to perform such measurements.

Network measurements fall into two broad categories:

- **Active measurements** create and inject artificial packets into the network under observation. Later, these packets are intercepted and metrics based on their behaviour are calculated. The idea behind this technique is to use a well-defined sample to draw conclusions about the overall behaviour of a certain part of the network.

- **Passive measurements** capture packets transmitted by applications running on network-attached devices over a network link. Usually, the arrival of each packet is earmarked with a timestamp. Storing all captured packets along with their timestamps in a trace file provides an accurate representation of network traffic. However, the achievable measurement accuracy strongly depends on the accuracy of the timestamps supplied by the measurement system.
Active and passive measurements both have their specific advantages and disadvantages making them suitable for different purposes. One of the major drawbacks of active measurements is the potential interference of injected packets with normal network traffic. Depending on the network load and the amount of data transmitted by an active measurement platform, this could not only lead to a distortion of the very effects to be measured but also actually create an overload situation. This can pose a serious limitation as network measurements are especially interesting during periods of high load. However, active measurements allow much more direct methods of analysis.

The passive approach does not have such a limitation. There is no interference of the measurement with network traffic. This is a very attractive prospect because any information we can obtain through passive techniques is “free” in the sense that we do not have to impose any extra load on the network under study. However, each and every packet needs to be captured to gain a complete picture of a link’s traffic behaviour. This imposes a serious scalability problem to passive measurements. With the Internet link capacities growing faster than other computer technologies such as CPU, memory, disk, and tape performance, it is just a matter of time until full network packet traces (even for short periods of time) become all but unfeasible. In this respect, active measurements scale much better because they often work with a data sample of negligible size in comparison to the overall traffic on a measured link. Also, passive measurements depend entirely on the presence of appropriate traffic on the network under study and it can be much more difficult or impossible to extract some of the desired information from the available data.

Safety and privacy are very important issues of any network measurement. Neither network operation nor user privacy should be adversely affected. The first aspect applies to active measurements whereas user privacy is more of a concern for passive measurements. Active measurements generate their own data. Only these data are used for analyses, and user data remain untouched. The situation is somewhat different for passive measurements. User data are intentionally captured and often stored for analysis purposes. This is one of the major sources of difficulties involved in conducting a passive measurement in an operational network. These privacy concerns have to be addressed by dropping any unnecessary data (e.g. any packet payload) and by anonymising IP addresses to prevent end user identification from the trace data.

We will work in this M.Sc. project with passive measurements. Passive measurements are a powerful tool for modeling Internet traffic. They produce a trace of the actual traffic on the measured link at a certain time. Such a trace can be seen as a snapshot of an Internet link. All the information that we could get is “real”, in the sense that is not coming from a probe traffic, so we would obtain the best approximation to the network performance perceived by users.

We will use an available data repository to do that, where all the passive measurements have been previously stored. We present it in Chapter 2.
1.2 Research Question

In order to make clear the motivation of our research question, we are going to briefly introduce the SURFnet’s current approach to delay measurements. If we take a look at the RTT SURFnet statistics web site [36], we will find the “Last minute IPv4 average RTT SURFnet backbone”, like in Figure 1.2.1.

![Figure 1.2.1- Average RTT SURFnet backbone](Source: [36])

The figure shows the average RTT (also the minimum, the maximum and the jitter are available) between the fifteen POPs of the SURFnet backbone. In order to know how the network is going, it classifies the values of the delay in three groups: green (good performance), yellow (moderated performance) and red (bad performance), as we can look at the top part of the Figure 1.2.1. These measurements are taken with the ping\(^5\) tool, and as a result, active measurements have been used. Could it be possible to build something like this with the use of passive measurements?

The goal of this M.Sc. project is to find the best delay figure (or groups of figures) for evaluating the “health” of a network. So basically our research question is the following: “Is it possible to determine ‘network health figures’\(^6\) with the use of passive measurements of delay?”

\(^5\) With Ping, A small ICMP packet is sent through the network to a particular IP address, so it belongs to the active measurements group. See [http://www.ping127001.com/pingpage.htm](http://www.ping127001.com/pingpage.htm).

\(^6\) The meaning of ‘Figure’ is ‘graph’ within this thesis, and it is not ‘number’.
1.3 Approach

We started the work with literature study. After doing a lot of research on the related topics, we decided to use the M2C Measurement Data Repository [8], with four different available locations, to develop similar works with the delay, to compare these locations between them (we will use only three) and to put all the information obtained together.

Our approach is to perform passive measurements at TCP/IP level, because we do not want to inject traffic in the network. We used the data from the M2C repository to extract the delay, since it was not possible to do the required measurements in real-time.

We focus on the round trip delay as our main metric to quantify latency. We investigate three groups of RTT figures; these figures have been proposed in literature and show RTT, its variability and its relationship with the number of hops. We compare these figures using the same data, to get an idea of the advantages and drawbacks of each of them. These figures/graphs are:

- **RTT Figures**: we will investigate the RTT in the same way as in Figure 1.2.1, but using passive measurements and not for a fixed set of destinations, but for all destinations (basically CDF of the RTT in terms of TCP connections figures).
- **RTT Variation Figures**: we will investigate the RTT variability within the TCP connections (this is comparable to SURFnet’s jitter figures that we can find in [36], with the same comments that in the previous point).
- **RTT Figures, as a Function of the Number of Hops**: we will infer the number of hops between two endpoints from the TTL field of the IP packets stored in the data repository. Thereby, we will measure the RTT and its variability for all the TCP connections depending on the hop’s number.

The tool that has been used in the data repository on the measurement PC to capture packets is the standard `tcpdump` [9] utility. From these TCP dump files, `tcptrace` [10] tool has been used for analysis of the traffic and as a method to obtain the delays (RTTs) within a connection. `Ethereal` [23] has also been used to analyze the packets in detail, when necessary. Graphs have been generated with `Matlab` [14]. Finally, some C programs were implemented during this project to manage the data obtained with `tcptrace` or divide the TCP connections in accordance with the hop’s number that the packets had jumped.

1.4 Outline of the Report

Chapter 2 presents the state-of-the-art in passive delay measurements read from the books and papers. Chapter 3 includes the main work of the project with all the results and figures obtained, and Chapter 4 completes this thesis and it contains the conclusions about the developed research and the future work.
Chapter 2

State-of-the-Art

2.1 Terminology

2.1.1 About General Measurements Issues

As a starting point and if we take a look at most of the papers about traffic measurements, we will find that the RFC 2330 “Framework for IP Performance Metrics” [4] is quite cited. It is because it begins by laying out several criteria for the metrics that it adopts, which are designed to promote an IP Performance Metrics (IPPM) effort that “will maximize an accurate common understanding by Internet users and Internet providers of the performance and reliability both of end-to-end paths through the Internet and of specific ‘IP clouds’ that comprise portions of those paths”. It also defines some Internet vocabulary about its components such as routers, paths, and clouds and the fundamental concepts of “metric” and “measurement methodology”, which allow us to speak clearly about measurement issues. Measurement uncertainties and errors are discussed as well. For example, when developing a method for measuring delay, you have to understand how any error in your clocks introduces imprecisions into your delay measurement, and you should quantify this effect as well as you can. Thereby, [4], [5] and [6] define some clock’s issues as accuracy (“measures the extent to which a given clock agrees with UTC”), synchronization (“measures the extent to which two clocks agree on what time it is”), skew (“measures the change of accuracy, or of synchronization, with time”) and resolution (“the smallest unit by which the clock’s time is updated. It gives a lower bound on the clock’s uncertainty”). Due to reasons which we will discuss later, only the clock’s resolution will concern us.

Internet measurement is often complicated by the use of Internet hosts themselves to perform the measurement. These hosts can introduce delays, bottlenecks, and the like that are due to hardware or operating system effects and have nothing to do with the network behavior we would like to measure. In order to provide a general way of talking about these effects, [4] introduces two notions of “wire time”. These notions are only defined in terms of an Internet host H observing an Internet link L at a particular location: “For a given packet P, the "wire arrival (exit) time" of P at H on L is the first time T at which any bit (all the bits) of P has appeared at H's observational position on L".

7 “The IPPM WG will develop a set of standard metrics that can be applied to the quality, performance, and reliability of Internet data delivery services. These metrics will be designed such that they can be performed by network operators, end users, or independent testing groups. It is important that the metrics do not represent a value judgment (i.e. define “good” and “bad”), but rather provide unbiased quantitative measures of performance”, [12].

8 Coordinated Universal Time or UTC, also sometimes referred to as "Zulu time", is an atomic realization of Universal Time (UT) or Greenwich Mean Time, the astronomical basis for civil time [see [37]].
Note that intrinsic to the definition is the notion of where on the link we are observing. This distinction is important because for large-latency links, we may obtain very different times depending on exactly where we are observing the link. When appropriate, metrics should be defined in terms of wire times rather than host endpoint times, so that the metric’s definition highlights the issue of separating delays due to the host from those due to the network. In this thesis we cannot apply this fact, because we will work with the available data repository which includes host endpoints times.

Built on notions introduced and discussed in [4], there are similar documents which define specific metrics and procedures for accurately measuring and documenting the One Way Delay (OWD), Round Trip Time Delay (RTT) and delay variation (jitter), as [5], [6] and [13] respectively. We will present them in the following sections.

2.1.2 One Way Delay (OWD)

The definition for OWD given in [5] is: “For a real number \(dT\), the Type-P-One-way-Delay\(^9\) from Source to Destination at \(T\) is \(dT\) means that Source sent the first bit of a Type-P packet to Destination at wire-time \(T\) and that Destination received the last bit of that packet at wire-time \(T+dT\)”.

One Way Delay is usually measured by timestamping a packet as it enters the network and comparing that timestamp with the time the packet is received at the destination. This assumes the clocks at both ends are closely synchronized. For accurate synchronization (tens of microseconds) the clocks are often synchronized with GPS\(^10\).

The measurement of OWD instead of RTT (defined in section 2.1.3) delay is motivated by the following factors [5]:

- “In today’s Internet, the path from a source to a destination may be different than the path from the destination back to the source (‘asymmetric paths’), such that different sequences of routers are used for the forward and reverse paths. Therefore round-trip measurements actually measure the performance of two distinct paths together. Measuring each path independently highlights the performance difference between the two paths which may traverse different Internet service providers, and even radically different types of networks (for example, research versus commodity networks, or ATM versus packet-over-SONET)”.
- “Even when the two paths are symmetric, they may have radically different performance characteristics due to asymmetric queueing”.
- “Performance of an application may depend mostly on the performance in one direction. For example, a file transfer using TCP may depend more on the performance in the direction that data flows.

\(^9\) A fundamental property of many Internet metrics is that the value of the metric depends on the type of IP packet(s) used to make the measurement (see [4]).

\(^10\) The Global Positioning System, is a satellite navigation system used for determining one’s precise location and providing a highly accurate time reference almost anywhere on Earth or in Earth orbit (see [37]).
rather than the direction in which acknowledgements travel". This assertion is disputable, since TCP has to wait to receive the ACKs for previous segments to transmit a new one, so when all is said and done RTT seems to be the magnitude of interest here.

- “In quality-of-service (QoS) enabled networks, provisioning in one direction may be radically different than provisioning in the reverse direction, and thus the QoS guarantees differ. Measuring the paths independently allows the verification of both guarantees”.

For these reasons, the OWD is a fantastic measurement to characterize the network’s delay, as we would have the latency for each path (from a source to a destination and vice versa) and we would not include other not desired effects, like the server response time, which is not a “pure” network delay.

On the other hand, we have to pay a high price for these advantages: the complex process of measuring. To measure the OWD, we need two clocks: one on the source and one on the destination. As we described in section 2.1.1, we need to consider the clock’s uncertainties. The accuracy of a clock is only important to identify the time at which a given delay was measured. Accuracy, in itself, has no importance to the accuracy of the measurement of delay. As we have said at the beginning of this section, there is a big problem with the synchronization between both clocks, and we need to use other resources like GPS or NTP\textsuperscript{11} to get an accurate synchronization, which involves adding complexity to the system and/or an increment of the price. The skew of a clock is not so much an additional issue as it is a realization of the fact that the synchronization error is itself a function of time. The resolution of a clock adds to uncertainty about any time measured with it, so we have to evaluate this issue in both clocks.

2.1.3 Round Trip Time Delay (RTT)

The definition for RTT given in [6] is: “For a real number $dT$, the Type-P-Round-trip-Delay from Source to Destination at $T$ is $dT$ means that Source sent the first bit of a Type-P packet to Destination at wire-time $T$, that Destination received that packet, then immediately sent a Type-P packet back to Source, and that Source received the last bit of that packet at wire-time $T+dT$”.

Round trip delays are usually easier to measure than one way delays, and RTTs are usually measured directly. Round trip delay is usually measured by noting the time when the packet is sent (often this time is recorded in the packet itself), and comparing this with the time when the response packet is received back from the destination (Figure 2.1.1).

While in OWD there is an issue of the synchronization of the source clock and the destination clock, in RTT there is an (easier) issue of self-synchronization, as it were, between the source clock at the time the test packet is sent and the

\textsuperscript{11} The Network Time Protocol (NTP) ([37]) is a protocol for synchronising the clocks of computer systems over packet-switched, variable-latency data networks. NTP uses UDP port 123 as its transport layer. It is designed particularly to resist the effects of variable latency. For more information about OWD measurements with NTP, read [38].
(same) source clock at the time the response packet is received. However, we must not forget the clock’s resolution.

![Figure 2.1.1 - Round Trip Time](image)

The measurement of round trip delay has two specific advantages [6]:

- “Ease of deployment: unlike in one-way measurement, it is often possible to perform some form of round-trip delay measurement without installing measurement-specific software at the intended destination. A variety of approaches are well-known, including use of ICMP Echo or of TCP-based methodologies. However, some approaches may introduce greater uncertainty in the time for the destination to produce a response”. Perhaps this server response time which is added to the RTT is the major drawback of this measurement. The fact that we cannot differentiate the path from a source to a destination from the inverse path, could be also a problem when we are trying to identify where the network’s failure is.

- “Ease of interpretation: in some circumstances, the round-trip time is in fact the quantity of interest. Deducing the round-trip time from matching one-way measurements and an assumption of the destination processing time is less direct and potentially less accurate”.

Due to simplicity for RTT measurement, we will use it instead of OWD to analyze the network delays.

2.1.4 Delay Variation, Jitter or IPDV (IP Packet Delay Variation)

The third way to characterize the network latency is to measure the delay variation. "For a real number \( ddT \) 'The type-P-one-way-ipdv from Source to Destination at \( T_1 \), \( T_2 \) is \( ddT \)' means that Source sent two packets, the first at wire-time \( T_1 \) (first bit), and the second at wire-time \( T_2 \) (first bit) and the packets were received by Destination at wire-time \( dT_1+T_1 \) (last bit of the first packet), and at wire-time \( dT_2+T_2 \) (last bit of the second packet), and that \( dT_2-dT_1=ddT \)" (see [13]).
“One important use of delay variation is the sizing of play-out buffers for applications requiring the regular delivery of packets (for example, voice or video play-out). What is normally important in this case is the maximum delay variation, which is used to size play-out buffers for such applications. Other uses of a delay variation metric are, for example, to determine the dynamics of queues within a network (or router) where the changes in delay variation can be linked to changes in the queue length process at a given link or a combination of links” (read [13]).

“In addition, this type of metric is particularly robust with respect to differences and variations of the clocks of the two hosts (if, as a first approximation, the error that affects the first measurement of One Way Delay was the same as the one affecting the second measurement, they will cancel each other when calculating ipdv). This allows the use of the metric even if the two hosts that support the measurement points are not synchronized” (read [13]).

Although this measurement is related to the OWD, we will define in Chapter 3 a jitter measurement using RTT samples (maximum RTT minus minimum RTT, that is to say, the maximum variability of RTT which has been seen in a TCP connection), trying to get knowledge about the network performance and its latency variability.

2.2 About RTT Measurements

2.2.1 RTT Estimation Techniques

The basic idea for extracting RTTs from packet traces collected near TCP sources is fairly simple: measure the time difference between the observed transmission of a data segment from the source and the observed receipt of an ACK containing an acknowledgment number that exactly corresponds to (it is one greater than) the highest sequence number contained in an observed data segment. This simple notion, however, is complicated by several factors. To choose how to deal with this, the guiding principle is to be conservative and include in the data only those RTT values where there is an unambiguous correspondence between an acknowledgment and the data segment that triggered its generation.

The most serious complications arise from lost and reordered segments. If a SYN or data segment is retransmitted and an ACK matching is received, it is ambiguous whether the RTT should be calculated from the transmission time of the initial segment or from the retransmitted segment (see [30], [31]). Further, in a flight of data segments, the last segment may have a matching ACK but it could have been only generated after the retransmission and receipt of a lost segment earlier in the flight. To eliminate the possibility of invalid (and large) RTT measures in such cases, we should ignore all RTT estimates yielded by retransmitted data segments and by those transmitted between an original segment and its retransmitted copy. Another subtle complication arises because segments may occasionally be lost in the network between the sender and the tracing monitor. In this case, the retransmission of the segment will be detected as an out-of-order transmission of a sequence number, not as
a duplicate transmission. We should also tackle such cases by ignoring all RTT estimates for data segments that were in-flight (not yet acknowledged) when an out-of-order segment was seen. Another issue to consider in analyzing RTT values is that a TCP endpoint may delay sending the ACK for an incoming segment for up to 500ms in order to piggyback the ACK on the next outgoing data segment (common implementations delay the ACK only up to 200ms). This means that some RTT values may have additional time added because the ACK is delayed.

The objective in [15] is to estimate the Round Trip Times (RTTs) of the TCP connections that go through a network link, using passive measurements at that link, which adapts perfectly to our problem. In other words, it starts with a traffic trace from a link, and then attempts to measure the RTT of every TCP connection by only investigating the connection's unidirectional flow recorded in that trace. The proposed methodology is based on two techniques:

- The first technique (SYN-ACK (SA) estimation) is applicable to TCP caller-to callee[12] flows, and it is based on the 3-way handshake messages.
- The second technique (Slow-Start (SS) estimation) is applicable to callee-to-caller flows, when the callee transfers a number of MSS segments to the caller, and it is based on the slow-start phase of TCP.

It examines the accuracy of these RTT estimation techniques following two verification approaches. The first one is to compare the SA and SS estimates with active RTT measurements (ping) between that connection’s end-hosts. The second verification approach is indirect, and it is based on the relation between the SA and SS estimates. With a defined error tolerance, it shows that the fraction of inaccurate measurements is roughly 5-10% for SA estimates, and only slightly higher (10-15%) for SS estimates. Besides, it can be inferred that the two RTT estimates have an absolute difference that is less than 25ms in about 70%-80% of the processed TCP connections.

In relation with the SA estimation, [16] affirms that for almost 72% of connections, the minimum RTT is equal to the SYN RTT[13]. This suggests that the SYN RTT may be used as a reasonable approximation of the minimum RTT. However, for 14% of the connections, the SYN RTT exceeds the minimum RTT by more than 10% (see Figure 2.2.1). We also created this figure using our data repository (see Appendix B). Other considerations about the minimum RTT estimation are explained in [18] (using active probes).

Other two methods to obtain RTT measurements are cited in [39]:

- “The first method used packet loss to measure the round trip delay – each successfully recovered packet provided a sample of the RTT (i.e., the RTT was the duration between sending a NACK and receiving the corresponding retransmission). In order to avoid the ambiguity of which retransmission of the same packet actually returned to the client, the header of each NACK request and each retransmitted packet

12 If a TCP connection between hosts X and Y was actively opened by X, i.e., X sent the first SYN message, it defines that X is the caller and Y is the callee.
13 SYN RTT is the RTT sample yielded by the SYN/SYN+ACK pair.
contained an extra field specifying the retransmission attempt for that particular packet. Thus, the client was able to pair retransmitted packets with the exact times when the corresponding NACKs were sent to the server”.

- “The second method of measuring the RTT was used by the client to obtain additional samples of the round trip delay in cases when network packet loss was too low. The method involved periodically sending simulated retransmission requests to the server if packet loss was below a certain threshold“.

![Figure 2.2.1 - SYN RTT](Source: [16])

We need to remember that we can only use passive measurements in this project, we cannot add extra fields to the headers or to send simulated retransmissions, so these last two methods would not be suitable for us.

Finally, we can also find two new systems for passive estimation of round trip times for bulk TCP transfers in a new paper presented in PAM 2005\(^\text{14}\) [40]. “One method uses TCP timestamps to locate segments from a bulk data sender that arrive one RTT apart, while the other detects patterns caused by self-clocking that repeat every RTT. Both methods can be used throughout the lifetime of a TCP session. The timestamp based method can be used for symmetric routes, while the self-clocking based method works for both symmetric and asymmetric routes“.

Actually, our tool to extract RTT samples from the data repository will be tcptrace, which is presented in section 2.3. In this manner, we do not have to worry too much about the RTT extraction process, which will make our work easier.

\(^{14}\) PAM: Passive and Active Measurement Workshop (http://www.pam2005.org)
2.2.2 Some Figures which use RTT Measurements

Trying to answer our research question, we looked for previous works, which could serve us to identify network’s health figures with the use of RTT measurements.

The first figure that we found was the CDF\textsuperscript{15} of the RTT samples in terms of TCP connections, which is used in [15] and [16], for example.

One interesting objective in [15] is to study RTT distributions at different locations and the variation in different time scales. In general, the RTT distribution at a link depends on the geographical location of each connection’s end-points. Therefore, it is expected that different links can have significantly different RTT distributions. The effect of the geographical location is prominent in the case of the Figure 2.2.2, for example. The RTT distribution makes a significant ‘step’ between about 50ms and 200ms. About 35% of the connections have a RTT lesser than 50ms, while the rest of the connections have a RTT larger than 200ms. In this example, the former group is connections within Israel, or between Israel and Europe, while the latter is connections mainly to North America.

\textbf{Figure 2.2.2} - Example of RTT distribution in terms of connections (Source: [15])

In terms of a lower RTT bound, there is a significant fraction of TCP connections in all traces with a RTT of just a few milliseconds. These are connections within the local geographical area of the monitored link. It is noted that the RTTs at a monitored link cannot be lower than the round trip propagation delay of that link.

On the other hand, [15] affirms that the RTT distributions do not change significantly in the time scales of tens of seconds for the traces it examined. In the hour scales, we are mostly interested in differences between daytime and

\textsuperscript{15} CDF: Cumulative Distribution Function.
nighttime. In the month scales, variations in the RTT distribution can be due to technology changes (e.g., addition of new links or routers), or due to long-term Internet evolution trends (e.g., gradually lower queueing delays).

The measurement and analysis of the variability in round trip times within TCP connections using passive measurement techniques is studied in [16]. In order to analyze the RTT, it also plots the cumulative distribution (CDF) of all the RTT samples collected from all traces and the distributions of the minimum, maximum, mean, median and 90% percentile RTTs observed for each connection. These observations indicate that the range of RTTs experienced by TCP segments is extremely large and the connections exhibit great diversity in their fixed end-to-end delays.

Its measurements of variability are the standard deviation in RTTs, the interquartile range (IQR) measured for each connection and some combination of this measurements. Its results show that connections with higher median RTTs also exhibit a larger disparity in the distribution of RTTs. Besides, connections with smaller minimum RTT see a greater variability in RTTs. We will get from this some ideas to build figures, such as the CDF of the standard deviation.

To further assess the extent of variable delays in RTT samples within a connection, [16] shows a figure which normalizes the median, 90th percentile, and maximum RTTs observed for each connection by its minimum RTT (see Figure 2.2.3). With this information we can guess that around 25% of connections see a median RTT that is 2-10 times the minimum RTT and that around 7% of connections see a median RTT that is more than 5 times the minimum. The main conclusion of the study in this paper is the presence of significant variability in the per-segment RTTs of TCP connections.

![Figure 2.2.3 - \{max, 90%, med\} RTT/ min RTT](Source: [16])

A similar work has been developed in [17]. They find that connections do not generally experience large RTT variations in their lifetime. For example, for approximately 80-85% of the connections, the ratio between the 95th
percentile RTT value and the 5th percentile RTT value is less than 3; in absolute terms, the RTT variation during a connection’s lifetime is less than 1 second for 75-80% of the connections. The main conclusion between [16] and [17] seems to be different, but the results are approximate (the variability in TCP RTT is ‘significant’ but not ‘large’).

The last papers offer us some good ideas to start our work. This is also the case of the next one. Mark Allman in [27] examines the distribution of round trip times between a server and the clients. He also used tcptrace (as we will do) to produce the average and median RTT for each connection in a dataset. Figure 2.2.4 provides a comparison of the minimum RTT observed and the median RTT for each connection. The x-axis is the minimum RTT in milliseconds, while the y-axis is the median RTT for the same connection as a multiple of the minimum RTT. The median RTT was within a factor of 2 of the minimum RTT in slightly over 90% of the connections. However, the plot illustrates that for shorter RTTs the variability within connections is sometimes quite large (this result complements the same ones obtained in [16] and [17]). “One explanation for this decrease in variability as the RTT grows is the use of a network link with a high delay (e.g., a satellite channel) that has the effect of drowning out the variability in the rest of the network path. However, this cannot be further investigated without additional data. Another note about this data is that the minimum RTT may come from a short segment (e.g., a SYN). On slow links the transmission time of a short packet can be significantly shorter than that of a full-sized data segment, which could explain some of the variability shown in the figure” ([27]).

In a different way, in [26] some cases of study about RTT are examined, and different paths are analyzed. Although this paper deals with active measurements, we can see some changes in graphs (RTT vs. Different time scales) due to network failures, route changes and so on.
Finally, the last type of graph that we will examine is represented in Figure 2.2.5. It represents the minimum RTT against the hops number. It can be found in [41], which examines the ability to perform accurate topology-aware operations solely based on passive data. In order to study this problem, it explores the use of multi-variable linear regression techniques for RTT estimation using multiple metrics such as geographic distance, hop count, and AS (Autonomous System) count.

Using our data repository, we will build some of the figures that we have presented in this section. We will try to find the best graph which allows us to infer a lot of information about the network performance. All these issues are discussed in Chapter 3.

![Figure 2.2.5 – Minimum RTT against hops (Source: [41])](image)

### 2.2.3 Other RTT Issues

In this section we briefly introduce other interesting works and readings about networks delay, which give us more knowledge in this field.

Vern Paxson, a very famous researcher in the Internet measurements field, gives us a complete introduction of the end-to-end Internet dynamics [19]. It is a very wide thesis which dedicates a chapter to the packet delay. In that chapter he discusses the different roles of the RTT in the connection’s behavior. “First, a reliable transport protocol such as TCP needs to decide how long to wait for an acknowledgement of data it has sent before retransmitting the data. There is a basic tension between wanting to wait long enough to assure that the protocol does not retransmit unnecessarily, versus not wanting to wait too long so as to unduly delay the connection when in fact retransmission is needed. The second way in which a connection’s RTT influences the connection’s behavior concerns the important notion of bandwidth-delay product (BDP). A connection’s BDP is the product of \( \rho \), the available bandwidth, measured in bytes/sec, with \( \tau \), the RTT, measured in seconds. The result is a number \( B = \rho \cdot \tau \) of bytes indicating how much data the connection must have in flight to fully utilize the available bandwidth.”
After some RTT measurement considerations he analyses the RTT extremes. We would expect RTT extremes to be governed for the most part by geography. This is especially the case for network paths that include satellite links, as these can add hundreds of milliseconds due to the propagation delays up to and back down from the satellite. However, while geography certainly dominates upper RTT extremes, it is not the only factor. He shows that assumptions concerning network behavior can be violated in unexpected ways. RTT variation during a connection is also examined in [19] and he uses similar methods and graphs that we have seen in previous papers.

[24] describes how the shortage of bandwidth is a major reason for increased delays. Insufficient supply of bandwidth causes queuing delays at network devices, and limited peak data rates add to the per hop delay due to packet deserialisation times. The arrival of a packet at a network link is not an atomic event, but due to bit deserialisation, it is a function of the packet’s size. At several points within this paper, typical packet sizes and their distributions are identified as an important factor for the delay patterns observed. However, the traffic patterns by themselves are insufficient to fully describe the observed packet delay and loss figures and the conclusion is that there is a router specific component which cannot be accurately predicted. Relevant to this, in [25] one series of experiments was designed to determine the network delays with respect to packet length and the data clearly show a strong correlation between delay and length, with the longest packets showing delays two to three times the shortest.

Finally, some interesting websites related to the Internet performance monitoring, that offer tools, documents, real time measurements and a lot of information about current projects are [20], [21], [22].

2.2.4 Network’s Health Candidates Figures

Within the section 1.3, we said that we would pick out three groups of figures to represent the network’s health. Well, after reading the literature about passive measurements of the delay, here we are going to briefly describe them.

These three possible figures (or three subsets of figures) to evaluate the performance of the network are called RTT, RTT Variation and RTT as a Function of the Number of Hops\(^{16}\) Figures respectively:

- The first group, the RTT Figures, will be the CDF of the RTT in terms of TCP connections (linear and logarithmic scales) and other graphs related to this figure (frequency distribution), namely it should be similar to Figure 2.2.2. We use the minimum, average and maximum RTT to build such figures and some comparisons at different time scales will be done.
- The RTT Variation Figures group the graphs related to the RTT variability within a TCP connection. Figures 2.2.3 (RTT ratios) and 2.2.4 and others which use the standard deviation of the RTT and jitter, are examples of figures that belong to this class.

\(^{16}\)To simplify, we will use the term RTT FNH Figures.
Finally, the RTT FNH Figures will analyze the minimum and average RTT of the TCP connections with the different hops in the network that they have needed to reach their destinations. Figure 2.2.5 illustrates the case.

Of course, we should not forget the fact that we will use passive measurements of the RTT to perform these figures, using a data repository that we will describe in the next section.

2.3 The Data Repository

2.3.1 Description

The M2C\textsuperscript{17} (Measuring, Modelling and Cost Allocation) traffic repository \cite{8} currently contains several hundred (fifteen minutes) traces, measured at four different locations, various times a day, seven days per week.

The measurements are performed by capturing the headers of all packets that are transmitted over the (Ethernet) “uplink” of an access network to the Internet, as outlined in Figure 2.3.1. The switch (can also be a router) copies all traffic flowing in to and out of the access network to the measurement PC. The tool that has been used on the measurement PC to capture packets is the standard \texttt{tcpdump} \cite{9} utility.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{measurement_setup.png}
\caption{Measurement setup (Source: \cite{27})}
\end{figure}

\texttt{tcpdump} is run for fifteen minutes, generating a binary file that is stored on disk, containing a packet trace: a dump of the headers of all packets that have been transmitted over the uplink in that period. Only the first 64 octets of each Ethernet frame have been captured. The resulting packet trace is a file of possibly several gigabytes, depending on the load of uplink. In order to save resources, the traces are compressed.

\textsuperscript{17}This section is a resume taken from \cite{28}.
The headers in the packet trace include source and destination IP addresses and port numbers. Although the payload of the IP packets is discarded, careful analysis of the packet trace still may reveal possibly sensitive information, such as which websites are visited by who, which threatens users' privacy as we saw in section 1.1.3. On the other hand, removal of addresses etc. from the packet traces severely reduces their usefulness. Thus there is a trade-off to be made between protecting privacy and usability of the traces. Hence, to protect users' privacy, the packet traces are made anonymous, by scrambling the source and destination IP addresses, using the tcpdpriv [29] utility. This process is called anonymization. Other information, such as transport port numbers and the timestamps at which packets arrive are left unchanged.

All the details about the data repository can be found in [28].

2.3.2 Locations under Study

In this section we present the three different locations that we have used to get the data and generate all the graphs. Although the data repository has one more location, we decided not to analyze it, because we did not have enough time to process its data and because actually the study of three locations is enough. The next three short descriptions are taken from [8]:

“On location number 1, the 300 Mbit/s (a trunk of 3 x 100 Mbit/s) Ethernet link has been measured, which connects a residential network of a university to the core network of this university. On the residential network, about 2000 students are connected, each having a 100 Mbit/s Ethernet access link. The residential network itself consists of 100 and 300 Mbit/s links to the various switches, depending on the aggregation level. The measured link has an average load of about 60%. Measurements have taken place in July 2002”.

“On location number 2, the 1 Gbit/s Ethernet link connecting a research institute to the Dutch academic and research network has been measured. There are about 200 researchers and support staff working at this institute. They all have a 100 Mbit/s access link, and the core network of the institute consists of 1 Gbit/s links. The measured link is only mildly loaded, usually around 1%. The measurements are from May - August 2003”.

“Location number 3 is a large college. Its 1 Gbit/s link (i.e., the link that has been measured) to the Dutch academic and research network carries traffic for over 1000 students and staff concurrently, during busy hours. The access link speed on this network is, in general, 100 Mbit/s. The average load on the 1 Gbit/s link is usually around 10-15%. These measurements have been done from September - December 2003”.

2.4 The RTT Measurement Tool: Tcptrace

2.4.1 Why Tcptrace?

We can try to build a C/C++ program to obtain the valid RTT samples from the data repository files. It is perfectly possible using for example WinPcap [32], a
free, public system for direct network access under Windows that allows us to handle offline dump files among other things. But reading papers about RTT measurements (for example [27]), we finally decided to use the tcptrace [10] program to extract the RTT samples, because it works pretty good and because it is already done. Tcptrace is a tool that can take TCP dump files from several popular packet-capture programs and generate detailed reports about individual TCP connections. It can also generate several graphs for further analysis.

Tcptrace is pretty smart about choosing only valid RTT samples. An RTT sample is found only if an ACK packet is received from the other endpoint for a previously transmitted packet such that the acknowledgment value is one greater than the last sequence number of the packet. Further, it is required that the packet being acknowledged was not retransmitted, and that no packets that came before it in the sequence space were retransmitted after the packet was transmitted. The former condition invalidates RTT samples due to the retransmission ambiguity problem, and the latter condition invalidates RTT samples since it could be the case that the ACK packet could be cumulatively acknowledging the retransmitted packet, and not necessarily ACK-ing the packet in question. But we will learn how tcptrace does that exactly in the following section.

2.4.2 Valid RTT Samples: Extraction Process

In order to know how tcptrace works to obtain the RTT samples, we can analyze the file rexmit.c from its source files and examine the functions ack_in() and rtt_ackin(), which calculates the RTT values, is called from ack_in() only if new data (a segment which has not been acknowledged before) is getting acknowledged. Obeying Karn’s algorithm (not calculating an RTT sample if retransmission of unacknowledged data is found to occur), tcptrace uses the difference between timestamps of the data segment and its corresponding ACK. Both functions return a value that corresponds with a type of ACK:

```c
/* ACK types */
enum t_ack { NORMAL = 1,  /* no retransmits, just advance */
            AMBIG = 2,  /* segment ACKed was rexmitted */
            CUMUL = 3,  /* doesn't advance */
            TRIPLE = 4,  /* triple dupack */
            NOSAMP = 5}; /* covers retransmitted segs, no rtt sample */
```

Figure 2.4.1 shows the flow chart of the ack_in function. This function is called from trace.c when the ACK field of the TCP header of the new packet is set to 1, and it receives the sequence number of the ACK (among other arguments). Tcptrace saves the TCP segments in a list of segment structures. This structure is as follows:

```c
typedef struct segment {
    seqnum    seq_firstbyte;  /* seqlnumber of first byte */
```

18 The current stable version of tcptrace (v6.6.7), was used during this project.
The program divides the sequence numbers in four quadrants (each quadrant with $2^{30}$ numbers), depending of the ACK sequence number (there are $2^{32}$ possible values due to the TCP header's length). Each quadrant has a pointer to a segments list and to the previous and the next quadrants.

Once we know which is our current quadrant, we check first the previous one (segments with smaller sequence number than the actual ACK) in order to acknowledge (increment the field `acked`) the segments without previous ACK. We also increment a counter for cumulatively ACKs (`rtt_cumack`) to count the segments that were cumulatively acknowledged and not directly acknowledged.

After looking over the previous quadrant, we examine the current one. If the segment was already acknowledged, the current ACK can be a duplicate. For an acknowledgement to be considered as duplicate ACK in BSD version, following rules must be followed [10]:

1. “The received segment should contain the biggest ACK TCP has seen,
2. the length of the segment containing duplicate ACK should be 0,
3. advertising window in this segment should not change, and
4. there must be some outstanding data”.

If these conditions occur, then the variable `ret` is set to `CUMUL`, and it is set to `TRIPLE` if three duplicate acknowledgments acknowledge the same segment, a condition commonly used to trigger the fast-retransmit/fast-recovery phase of TCP.

If the segment still was not acknowledged, we do it and ask if the acknowledgment value is one greater than the last sequence number of the packet. If it is not the case, we consider it as a cumulative ACK. Otherwise, we check if packets that came before it in the sequence space were retransmitted after the packet was transmitted, the situation in which the segment being ACK-ed was sent a while ago and we have been piddling around retransmitting lost segments that came before it. We indicate this conditions with the values `TRUE` or `FALSE` in one of the arguments of the `rtt_ackin()` function.

The flow chart of the `rtt_ackin()` function is displayed in Figure 2.4.2. We can observe that a valid RTT sample is obtained when the packet being acknowledged was not retransmitted, and that no packets that came before it in the sequence space were retransmitted after the packet was transmitted (`ret = NORMAL`). Otherwise, the ACK can be considered as ambiguous (due to the retransmission ambiguity problem, the segment being ACK-ed was retransmitted and it is impossible to determine if the ack is for the original or the
retransmitted packet), or as no valid sample (ret = NOSAMP) when the
\texttt{rtt\_ackin()} function is called with the TRUE value in the last argument from
\texttt{ack\_in()}. 

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{flowchart.png}
\caption{Flow chart of \texttt{ack\_in} function}
\end{figure}
2.4.3 Considerations

One of the problems of the passive monitoring using only one measurement point is the location of such point. In order to obtain the RTT, tcptrace calculates the time between when a segment was sent and when the acknowledgement for it was received. Therefore, technically, it is the RTT between the measurement host and the data receiver. Figure 2.4.3 shows the problem of the location of the measurement point. If the measurement point is too close to one of the end hosts, then only one direction of the data measurement is valid. So, as we can observe in the figure, if we send a packet from host A to the host B, the measured RTT is RTT’ 1, which is almost equal to the real RTT\(^\text{19}\) (RTT 1). Though, if we send a packet from host B to the host A, the

\(^{19}\)The best approximation to the real RTT is got when we put the measurement point on the sender.
measured RTT ($RTT'$ 2) is not valid, because it is quite smaller than RTT 2. If we want to measure the RTT in both directions, the best thing we can do is to capture the packets “on both sides” and analyze them separately. If that is not possible, then tcptrace will not be able to find such RTT for us.

![Diagram showing measurement point problem](image)

**Figure 2.4.3 - The measurement point problem**

Inside the data repository we can detect this problem, because tcptrace provides RTT statistics for both directions inside a TCP connection and the times for the minimum RTT should be similar for each direction, however, one of the directions always presents a senseless minimum RTT measurement (almost 0 ms). That’s why we decided to analyze only the RTT in one of the directions of the TCP connection, filtering the data with the criteria of maximum minimum RTT between the two directions of the same end hosts.

In practice this method works, but it does not work right if by some weird coincidence, the minimum RTT to the local host is longer than the RTT to the remote host. This is of course rather unlikely, but on a flow with only a few packets it might happen, if those few packets are just sent, by any chance, at a moment when there is some local congestion.

These two assumptions have been done during this report:

- Although tcpdump [9] timestamps have a precision of one microsecond, they may not accurately represent the time at which the packet arrived on the link. In particular, interrupt scheduling and driver executions may introduce variable time-stamping delays. We reduce the precision of RTT values by rounding them to the nearest millisecond (RTTs < 1ms are set to 1ms).
• Connections that see a larger number of samples are likely to yield better estimates of variability in what follows, therefore, we only consider connections with at least 10 valid RTT samples. Thus, we will do more unlikely that the minimum RTT due to the local host happens to be longer than the RTT to the remote host.

An example of tcptrace RTT stats and its explanation is shown in [42]. As tcptrace accepts compressed input files (as the ones in our data repository), we can process our files directly. We obtained a new text file for each dump file and from these ones, we extracted the RTT stats of interest by using a simple C program which deals with text files. Finally, we processed the obtained data with Matlab.

\footnote{The tcptrace command we used for this aim was tcptrace -lnrc -f' ((c_rtt_count>10) AND (s_rtt_count>10) ) filename, which besides provides only RTT stats for complete TCP connections.}
Chapter 3

Searching the Network’s Health Figures

3.1 Introduction

This is the main chapter of this master thesis. Hitherto, we learnt the existing and necessary knowledge to come near to the solution of the problem. At this point, it should be clear what our aim is and the assumptions that we have done: *Is it possible to determine ‘network health figures’ with the use of passive measurements of round trip delay?*

It should be also clear, as we could see in section 2.2.4, that we will work with three groups of figures (based on literature’s studies): RTT Figures, RTT Variation Figures and RTT as a Function of the Number of Hops Figures.

During next sections, we expand all the work done during this project and we show all the obtained results (working with our data repository). When necessary, we will deepen more in the developing of the figures, to make clear how we got such figures, mainly with the third group or RTT FNH.

3.2 RTT Figures

3.2.1 About RTT Figures

We use two basic approaches within this group of figures:

- CDF Figures of the RTT in terms of TCP connections (both, linear and logarithmic scales). We will also compare the linear CDF figures at different time scales inside the locations.
- Frequency distribution of RTT samples.

In order to help us out with the analysis of the data repository, some test with ping tool were performed from one of our computers to the rest of the world to get the approximate delay according to the geographical location of the end hosts. The results are shown in Table 2.

<table>
<thead>
<tr>
<th>Minimum RTT interval (ms)</th>
<th>Zone</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 20</td>
<td>I - Local</td>
<td>Netherlands</td>
</tr>
<tr>
<td>20 - 80</td>
<td>II - Europe</td>
<td>Spain, UK</td>
</tr>
<tr>
<td>80 - 160</td>
<td>III - North America</td>
<td>USA, Canada</td>
</tr>
<tr>
<td>&gt; 160</td>
<td>IV - Rest of the World</td>
<td>China, Japan, Australia</td>
</tr>
</tbody>
</table>

Table 2 - Minimum RTT vs. Geographical Areas

These results have been added to the RTT Figures in vertical lines form, in order to separate all the zones within the graphs. Of course, the values presented in
this table should not be considered as a general rule which is always valid, it is just an approximation to help us with the geographical location issues.

### 3.2.2 CDF of the RTT in Terms of TCP Connections

Figure 3.2.1\(^{21}\) plots the distributions of the minimum, maximum and average RTTs observed for each connection within location 1, 2 and 3. As we have seen in section 2.2.2, the RTT distribution at a link depends on the geographical location of each connection’s end-points. We recall again that we have added three vertical lines to the figures following the criteria showed in Table 2 to separate the different geographical zones. These figures contain all the data that we processed for each location\(^{22}\), without any pertinent distinction to the time when the samples were taken. So they represent a “general” behaviour of the corresponding locations.

We start our dissertation looking at Figure 3.2.1 a). In location 1 almost 60% of minimum RTT samples are under 20ms and belong to a traffic inside The Netherlands. This result is not surprising because in this location the users are students in a residential network and the staff working in the UT, and that most of their traffic was local is something expected (sharing files, webmail, etc). Besides, inside the local zone, we can see that 16% of connections are lower than 1ms, which could indicate that the end hosts would be in the same Ethernet link, and that 50% of connections are under 7ms (probably the connections between an end host in the residential network and another one crossing the core network of the university or a little bit farther away). About 21% of connections are inside the European zone and 12% inside the zone III. The rest of the connections are within the zone IV (7%).

Looking at the average RTT curve, it is apparently closer to the minimum RTT curve than to the maximum RTT one. We said in section 1.1.2.1 that “the minimum value of delay provides an indication of the delay that will likely be experienced when the path traversed is lightly loaded and that values of delay above the minimum provide an indication of the congestion present in the path”, so the feeling is that the network has less congestion when the “red line” is closer to the “blue line”. In this case, the network is not apparently very congested.

To appreciate in a better way that “the range of RTTs experienced by TCP segments is extremely large and the connections exhibit great diversity in their fixed end-to-end delays” ([16]), we notice in Figure 3.2.1 b) (with logarithmic scale) that the observed RTTs range is from 1ms to more than 10s. The minimum and maximum observed RTTs differ by more than 4 orders of magnitude.

---

\(^{21}\) Figures 3.2.1 a) and b) correspond to location 1 (the second one has logarithmic RTT scale). In the same way, Figures 3.2.1 c) and d) correspond to location 2 and Figures 3.2.1 e) and f), to location 3. To obtain percentages in the Y axis, we have to multiply the value per 100.


Data for location 3: from 03-09-2003 to 09-09-2003 at 04:10h, 10:05h and 17:00h and from 03-10-2003 to 09-10-2003 at 04:10h, 12:05h and 17:00h.
Figure 3.2.1 c) plots the distributions of the minimum, maximum and average RTTs observed for each connection in the location 2. In this case, almost 33% of minimum RTT samples are under 20ms and belong to a traffic inside The Netherlands. As a research institute, the fact that most of its traffic is external (to the rest of the world), is something we could expect. About 19% of connections are inside the European zone and 31% of them inside the zone III. Rest of the connections are in the zone IV (17%). Seemingly, most of the realized research by this institute is done inside The Netherlands and USA.

As in location 1, the observed RTTs range is from 1ms to more than 10s so the minimum and maximum observed RTTs differ by more than 4 orders of magnitude (see Figure 3.2.1 d)). Similar analysis can be done for location 3 and Figure 3.2.1 f). Looking at the average RTT curve, it is in the middle between the minimum RTT curve and the maximum RTT curve. It can indicate that the paths are only moderately congested.

We can observe quite well the effect of the geographical distribution in the delay for location 3 in Figure 3.2.1 e). There are small jumps in the graph of the minimum RTT just in the points of area’s changes. The minimum RTT identifies the geographical distribution of the connections. Almost 64% of minimum RTT samples are 20ms or less and belong to a traffic inside The Netherlands. About 9% of connections are inside the European zone and 22% of them inside the zone III. The rest of the connections are in the zone IV (5%). Again, as in location 1, most of the traffic is local and the average RTT is close to the minimum RTT.
Analysis of the Delay in the SURFnet Network

Figure 3.2.1 b) - CDF of RTT in Location 1 (Logarithmic)

Figure 3.2.1 c) - CDF of RTT in Location 2
Figure 3.2.1 d) – CDF of RTT in Location 2 (Logarithmic)

Figure 3.2.1 e) – CDF of RTT in Location 3
If we try to compare these figures (with the criteria "the more above the curve is, the lower the delay is"), we could think that delay in location 2 is much higher than in location 1 or location 3. Is this assertion true? Well, this difference is due to the user’s habits (in terms of habitual endpoints connections) more than the network features. We saw in section 2.2.2 that it is expected that different links can have significantly different RTT distributions. As we can read from the Table 3, location 1 and 3 have more similar distribution of the TCP endpoints, that’s why their delay figures are parallel. We could have guessed this previously if we have read the description of each location, because the users in location 1 and 3 are students, who have the same traffic habits.

<table>
<thead>
<tr>
<th>Zone</th>
<th>Location 1 (% connections)</th>
<th>Location 2 (% connections)</th>
<th>Location 3 (% connections)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>60</td>
<td>33</td>
<td>64</td>
</tr>
<tr>
<td>II</td>
<td>21</td>
<td>19</td>
<td>9</td>
</tr>
<tr>
<td>III</td>
<td>12</td>
<td>31</td>
<td>22</td>
</tr>
<tr>
<td>IV</td>
<td>7</td>
<td>17</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 3 - Percentage of connections in each geographical zone

3.2.3 CDF of the RTT at Different Time Scales

In order to know what the network’s health within each location is like, we need to separate the measurements in different time scales to compare them and to extract conclusions (as it is done in [15]).

We start this process with the location 1. Figure 3.2.2 shows the minimum, maximum and average RTT distribution for two different hours in the same day (Friday). We observe that the delay at 11:15h is bigger that at 14:00h in most...
part of the curves. This behaviour could be due to a break for lunch in a working day, when the level of traffic is supposed to be lower. However, in the local zone the delays are similar, which indicates that at this time on that Friday, the congestion inside the university and the SURFnet network is almost the same.

![Empirical CDF (Friday, 24-05-2002)](image)

**Figure 3.2.2 - CDF comparison at different hours in the same day (Location 1)**

We can also take a look at the Figure 3.2.3, which gives us the comparison between average RTTs at the same hour during a week. It is interesting to realize that the delay is quite high on weekends. One possible explanation is that in this period the students do not have to attend classes, so they expend more time in their rooms browsing Internet. Again we cannot appreciate too much differences in most of the part of the local zone. During that week, Tuesday was the day with less delay.

We use the monthly time scale in Figure 3.2.4. We compare two Tuesdays (one in May and the other one in June) at the same hour. We observe quite less level of congestion in May than in June. We know that in June the students have already finished their courses and they can spend more time in their rooms than in May, when they are usually at classroom. But we also know that in the time scales of months, variations in the RTT distribution can be due to technology changes, so we cannot be sure of the real cause of the difference between the two curves. At any rate, it seems to be at least strange, that they do some changes to deteriorate the network performance, so it could probably be a temporal change of route (inside the local zone and looking at the minimum RTT we appreciate a substantial difference between the two days).

---

23 Universities are connected to the SURFnet network. In the local zone (communications inside The Netherlands), this network is used during the first hops.
For the time being, it seems that these figures allow us to start knowing about when the network is working better or to identify some problems which cause bigger delays.

We continue examining in a similar way RTT distributions in different time scales, but now within location 2. Figure 3.2.5 shows the minimum, maximum and average RTT distribution for two different hours from various weeks. We clearly observe that the delay at 03:00h is bigger that at 15:30h. This behaviour could
be due to the hour’s difference between The Netherlands and USA for example, because when in The Netherlands is by night, in USA is by morning and all the servers are more congested because more people are working.

Figure 3.2.6 gives us the comparison between average RTTs during a week in location 2. The day with less congestion seems to be Sunday (discontinuous blue line), day of week when nobody works. Curiously on Wednesday the delay is also quite low. On the other hand, on Monday the delay in the network is maximum. The rest of days have more or less the same shape of the average RTT curve.
We use the monthly time scale in Figure 3.2.7. We compare one week of three different months (May, June and July) at the same hours. We clearly observe quite less level of congestion in July than in June and in May (these two months have the same delay). It is possible that people working in the research institute had holidays in July or that some links or routers were replaced by faster ones. We can say that the health of the network in July is better than during the two previous months (at least in the examined weeks), so these figures are really quite useful for our aims.

We conclude with this kind of analysis with similar graphs for location 3, specifically with Figures 3.2.8 and 3.2.9. In the first one, we have represented the minimum RTT at three different hours (04:10h, 10:15h and 17:00h) during a week in October. Whereas the minimum RTT at 10:15h and at 17:00h have similar distributions, at 04:10h presents quite more level of congestion. At that time the activity in the network increases considerably, maybe due to a kind of periodic process that takes place at that time or because the problem of the hour’s difference between the endpoints.

In the second one (Figure 3.2.9), we compare again the RTT distribution in two different months (September and October). With similar curve’s shapes, we see that the delay is lower in September than in October, when some people are on holidays.
Figure 3.2.8 - CDF comparison at different hours in the same week (Location 3)

Figure 3.2.9 - CDF comparison of different months (Location 3)
3.2.4 Frequency Distribution of the RTT

One way to complement the Figure 3.2.1 is to represent the appearance frequency of the RTT samples for each location. We did this in Figure 3.2.10.

This frequency distribution of RTT samples for location 1 is shown in Figure 3.2.10 a). The most likely values for the minimum RTT are 1ms and 6ms (it indicates the large number of local connections). If we compare with Figure 3.2.1 a), these peaks correspond to the abrupt changes of the minimum RTT curve. The most repeated value is 9ms for the average RTT, which allows us to imprecisely deduce the average delay due to the queueing in the university (between 3ms and 8ms). We will study this issue a little bit more in RTT Variation Figures section.

Within location 2, the most likely values for the minimum RTT are 1ms, 3ms and 15ms inside the local zone (see Figure 3.2.10 b)), which can be Ethernet connections, connections inside the core network of the research institute and connections with the rest of The Netherlands, respectively. There are also some peaks in the minimum RTT between 110ms and 120ms which show that there are a lot of connections within the zone III.
Finally, we do the same reasoning for the location 3 in Figure 3.2.10 c). The most likely values for the minimum RTT are 1ms, 5ms and 9ms inside the local zone. There are important peaks for the minimum RTT near the location's change points (84ms and 159ms), so again the effects of the geographical distribution of the RTT are more evident here. The average RTT curve seems to follow closer the minimum RTT curve (as we can also appreciate in Figure 3.2.1 e)) than in location 1 or 2, which could indicate a better network health.
3.2.5 Conclusions about RTT Figures

If we had to choose a figure to represent the health of the network within the section 3.2, then we would choose the CDF of the RTT in terms of TCP connections and linear scale. The logarithmic scale was used to see more clearly the range of the RTT values, but we appreciate better the shape of the curves using the linear scale.

The frequency distribution of RTT would probably be the first figure that we would choose at first moment, but if we compare graphs at different time scales (in order to decide when the network has better health), we will see more clearly the differences using the CDF than the frequency distribution.

We should not forget that these CDF graphs are not valid to compare different locations, because the behaviour of the users (in terms of endpoints destinations) can be quite different between them and hence, the shape of the figures is completely different.

3.3 RTT Variation Figures

3.3.1 About RTT Variation Figures

As we saw in section 3.1.2, the RTT Variation Figures try to quantify in some way the variability within TCP connections. To achieve this goal, we will represent some relations (like ratios or subtractions) among the measurements that we know (like the minimum, maximum and average RTT or Standard Deviation of the RTT). Concretely, we distinguish:

- Figures that use ratios (e.g. average RTT / minimum RTT). We will utilize CDF and frequency graphs.
- Figures in relation with the standard deviation of the RTT within TCP connections.
- Figures that characterize the jitter (e.g. CDF of maximum RTT minus minimum RTT).

For the rest, these measurements have been obtained as in the RTT Figures, and it is merely another way to represent the data.

3.3.2 RTT Ratios

Figure 3.3.1 (a), b) and c) for locations 1, 2 and 3 respectively) provides a comparison of the minimum RTT observed and the average RTT for each connection. The x-axis is the minimum RTT in milliseconds, while the y-axis is the average RTT for the same connection as a multiple of the minimum RTT. As we saw in Figure 2.2.4, the plot illustrates that for shorter RTTs the variability within connections is sometimes quite large (we found a sample with an average RTT that was 4,000 times the minimum RTT, which had a value of 2ms). We also saw that one explanation for this decrease in variability as the RTT grows is the use of a network link with a high delay (e.g., a satellite channel) that has the effect of drowning out the variability in the rest of the network path. The minimum RTT
may come from a short segment (e.g., a SYN) as well. On slow links the transmission time of a short packet can be significantly shorter than a full-sized data segment, which could explain some of the variability shown in the figure 3.3.1. This indicates that RTTs can change significantly on short time scales over some network paths. From this figure, we follow that this effect is more evident in the 1-15ms range of the minimum RTT, so we could say that all local connections have lower RTT delays but suffer more variability.

Figure 3.3.1 a) - Avg. RTT/min. RTT vs. min RTT (Location 1)

Figure 3.3.1 b) - Avg. RTT/min. RTT vs. min RTT (Location 2)
The results for the three different locations are practically the same, so this is an issue that we can label as “general” but does not let us say too much about the network performance.

Another way to characterize RTT extremes is in terms of the variation we observe in RTT over the course of a connection. Our interest lies in whether we can develop a “rule of thumb” such as “it is rare to observe a maximum or average RTT more than n times the minimum RTT”. This sort of empirical finding would aid us to figure out how transport protocols can best adapt to network conditions. In Figure 3.3.2 a) we can see the CDF of the ratios maximum RTT/minimum RTT and average RTT/minimum RTT for each connection within location 1. The 93% of connections have an average RTT that is less than 10 times the minimum RTT and 69% of them have also a maximum RTT less than 10 times the minimum RTT.

For the rest of locations, this measurement of variability is again very similar. From Figures 3.3.2 b) and 3.3.2 c), the 94% and 90% of connections have an average RTT that is less than 10 times the minimum RTT, and 71% and 66% of them have also a maximum RTT less than 10 times the minimum RTT, for location 2 and 3 respectively.

Hence, our ‘rule of thumb’ could be that “it is rare to observe an average RTT more than ten times the minimum RTT”. In order to make the same assertion for the maximum RTT with respect to the minimum RTT with the same level of confidence (90%), we should increase that quantity to 25. But, what are the most common values?
Figure 3.3.2 a) - Ratios avg. RTT/min RTT and max. RTT/min RTT CDF (Location 1)

Figure 3.3.2 b) - Ratios avg. RTT/min RTT and max. RTT/min RTT CDF (Location 2)
To observe this issue in a better way for location 1, we can take a look at the Figure 3.3.3 a). Here, the frequencies of the ratios are represented and we observe that it is very likely that the average RTT is between 1-4 times the minimum RTT, and the maximum RTT is between 6-8 times the minimum RTT.

For location 2, it is very likely that the average RTT is also between 1-4 times the minimum RTT (see Figure 3.3.3 b)), but the maximum RTT is quite dispersed between 1-15 times the minimum RTT (we cannot appreciate it very well in the figure), and it has a curious peak near 34 times the minimum RTT. In location 2,
the endpoints are usually farther than in location 1 or 3, so it would not be a surprise to find higher values of the maximum RTT.

**Figure 3.3.3 b) - Ratio's Frequencies (Location 2)**

Figure 3.3.3 c) shows the results for location 3 and here the average RTT is between 1-4 times the minimum RTT with more probability and the maximum RTT is almost uniform distributed between 1-40 times the minimum RTT.

**Figure 3.3.3 c) - Ratio's Frequencies (Location 3)**

From all of this we learn that the average RTT is, normally, between 1 and 4 times the minimum RTT, but the maximum RTT is a little bit more unpredictable.
However, our aim is to get knowledge about the network’s health and these figures, despite their interest, they are always quite alike and we cannot guess too much more about the performance of the network.

3.3.3 RTT Variability Using the Standard Deviation

Trying to find more information about the variability in TCP RTT, we linearly translated the average RTT from a connection by subtracting the minimum RTT to remove the fixed delay component, as in [16]. We also binned all connections by their (average - minimum) RTT value and computed the standard deviation of the individual connections in each bin. These results are plotted in Figure 3.3.4 a), b) and c) for the three locations. We found the same effect in all the locations: the standard deviation shows a linearly increasing trend as the translated average RTT increases. This means that connections with higher average RTTs also exhibit a larger disparity in the distribution of RTTs. The line with red colour represents the least-squares approximation of the data.

Are these last figures useful? Both of the axis in the figures represent a measurement of variability, so the linearly increasing trend seems to say “the more is the variability...the more is the variability”, which is obvious. At least for our aims, this figure is not useful, so we need to continue with our search of the network’s health figure.

Figure 3.3.5 shows the CDF of the standard deviation for all the locations. As it was expected, location 1 and location 3 have more similar distribution than location 3, because they have the same kind of users, and accordingly the same kind of traffic. From the figure, we note that 60% of connections present a standard deviation under 26ms within location 1, under 48ms within location 2 and under 9ms within location 3.
If we represented the frequency distribution of the standard deviation, we would find that the most likely values are within the range 1-5ms for location 1, within the range 1-15ms for location 2 and within the range 1-7ms for location 3.

We can say that if our measurement is the standard deviation, location 3 exhibits quite better “health” than location 2 in terms of variability. This figure could be representative of the network performance.

![Figure 3.3.4 b) - Std. deviation vs. average RTT - minimum RTT in Location 2](image)

![Figure 3.3.4 c) - Std. deviation vs. average RTT - minimum RTT in Location 3](image)
3.3.4 Jitter

Related Figure 3.3.5, it is the representation of the maximum jitter or absolute variability. As we presented in section 2.1.4, as a threshold value of the maximum jitter during a connection, we can use the difference between the maximum and minimum RTT observed in that connection (see Figure 3.3.6). Of course, this delay is important between two consecutive packets and that difference uses packets from all the connections (probably with very different packet sizes), so this figure represents only the worst case of jitter.

In like manner, the Figure 3.3.5, Figure 3.3.6 confirms that location 3 presents the best network performance in terms of variability. This fact could serve, for example, to choose the most adapted network for the use of VoIP, because jitter is a critical factor in the voice transmission. Of course, we have to consider that in this case the three locations do not have the same traffic (to the same endpoints), but could be an approximation between location 1 and location 3, which approximately present the same kind of traffic.

Trying to identify how much the delay due to congestion is (and not the delay due to propagation time, for example), we plot the frequency of the average RTT less minimum RTT, which removes the fixed part of the delay (Figure 3.3.7).

For location 1, we can observe that the delay due to congestion is wont to be between 1ms and 4ms, and for locations 2 and 3 between 1ms and 15ms (see Figure 3.3.7 a), b) and c) respectively). These results are almost the same for all the locations, because as we saw in Figure 3.3.2, it is very likely that the average RTT is between 1-4 times the minimum RTT (frequently between 1 or 2 times) and the subtraction is wont to be in the 1-20ms range.
Analysis of the Delay in the SURFnet Network

Figure 3.3.6 - CDF of maximum RTT - minimum RTT

Figure 3.3.7 a) - Frequency of average RTT - minimum RTT (Location 1)
Figure 3.3.7 b) – Frequency of average RTT - minimum RTT (Location 2)

Figure 3.3.7 c) – Frequency of average RTT - minimum RTT (Location 3)
3.3.5 Conclusions about RTT Variation Figures

From these groups of figures, we choose our approximation to the jitter (or absolute variability) displayed in Figure 3.3.6 as the best graph to represent the health of the network. We have seen how the figures in section 3.3.2 (RTT ratios) show general behaviours of an IP network, but we cannot appreciate important differences at different instants. Similar comments are valid with the standard deviation figures, but not with Figure 3.3.5 (similar to our chosen figure); we rule out this figure because it represents worse the absolute variability (useful to characterize the size of the buffers to control the jitter). The frequency figures shown in the last part of section 3.3.4 do not change too much at different time scales.

3.4 RTT as a Function of the Number of Hops Figures

3.4.1 About RTT as a Function of the Number of Hops Figures

As we briefly introduced in section 2.2.4, we also represent the delay with the RTT as a Function of the Number of Hops. The interest question here is “how can we inquire the hops number between two endpoints with passive monitoring?”. The answer seems to be at first not very difficult: using the Time To Live (TTL) field of the IP packets.

One paper that perfectly fits to our problem is [43]. There we can read: “Since hop-count information is not directly stored in the IP header, one has to compute it based on the TTL field. TTL is an 8-bit field in the IP header, originally introduced to specify the maximum lifetime of each packet in the Internet. Each intermediate router decrements the TTL value of an in-transit IP packet by one before forwarding it to the next-hop. The final TTL value when a packet reaches its destination is therefore the initial TTL subtracted by the number of intermediate hops (or simply hop-count). The challenge in hop-count computation is that a destination only sees the final TTL value. It would have been simple had all operating systems (OSs) used the same initial TTL value, but in practice, there is no consensus on the initial TTL value. Furthermore, since the OS for a given IP address may change with time, we cannot assume a single static initial TTL value for each IP address”.

We see that the hop count computation problem is not so simple. A list with the TCP TTL values for the main OSs is given in [45]. From there, we can verify that “...most modern OSs use only a few selected initial TTL values, 30, 32, 60, 64, 128, and 255. This set of initial TTL values cover most of the popular OSs, such as Microsoft Windows, Linux, variants of BSD, and many commercial Unix systems. We observe that most of these initial TTL values are far apart, except between 30 and 32, 60 and 64, and between 32 and 60” ([43]).

We know that very few hosts within Internet are reached with more than 30 hops, so continuing with this paper, “...one can determine the initial TTL value of a packet by selecting the smallest initial value in the set that is larger than its final TTL. For example, if the final TTL value is 112, the initial TTL value is 128, the smaller of the two possible initial values, 128 and 255”.
What happens with the TTL values that are not far apart? First of all, we have to explain that the aim of this paper is to build a defense against IP spoofing and it is based on the use of Hop-Count Filtering (HCF), which builds an accurate IP-to-Hop-Count (IP2HC) mapping table. Since they know how far away each received IP is (hops number stored in the IP2HC), they compute the hop estimation from the received packet and then they decide if it is valid or not. Then, “To resolve ambiguities in the cases of {30, 32}, {60, 64}, and {32, 60}, we will compute a hop-count value for each of the possible initial TTL values, and accept the packet if there is a match with one of the possible hop-counts” ([43]).

But we do not have an IP2HC mapping table (which can need quite amount of storage), so how can we solve the ambiguities? We noticed that [44] and [46] try passively to infer a host’s operating system from packet headers24. For example, [44] uses the TTL field, the presence of IP “do not fragment” (DF) bit, the initial TCP window size and the SYN packet size information, which are collectively distinct, and while using probabilistic learning, it develops a Bayesian classifier25 to passively infer a host’s operating system from packet headers. Some tested OSs can be found in [46] and a completed list of fingerprints for passive fingerprint monitoring, in [47].

The goal of this project is not to implement the most sophisticated method to inquire the initial TTL value, so we are going to exploit the results of [44] in order to simplify. The number of packets attributable to each operating system obtained in this paper is shown in Table 4. As we can check, Windows and Linux OS are the main packets contributors in the network. Trying to generalize this fact through Internet, we checked some stats sources about OS from [48] and we found similar results26.

For these reasons and searching the initial values of TTL for those OSs within [45] or [47] we decided that our initial set of possible TTL values were 32, 64, 128 and 255. For example, if the observed TTL is greater than 128 we will infer an original TTL of 255 and if less than 32 we will infer 32.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Bayesian</th>
<th>WF-Bayesian</th>
<th>Rule-Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows:</td>
<td>76.9</td>
<td>77.8</td>
<td>77.0</td>
</tr>
<tr>
<td>Linux:</td>
<td>19.1</td>
<td>18.7</td>
<td>18.8</td>
</tr>
<tr>
<td>Mac:</td>
<td>0.8</td>
<td>1.5</td>
<td>0.8</td>
</tr>
<tr>
<td>BSD:</td>
<td>0.8</td>
<td>0.1</td>
<td>1.6</td>
</tr>
<tr>
<td>Solaris:</td>
<td>0.7</td>
<td>1.3</td>
<td>0.5</td>
</tr>
<tr>
<td>Other:</td>
<td>1.7</td>
<td>0.6</td>
<td>0.2</td>
</tr>
<tr>
<td>Unknown:</td>
<td></td>
<td></td>
<td>1.3</td>
</tr>
</tbody>
</table>

Table 4 – Inferred Operating System Packet Distribution (Source [44])

24 Passive fingerprinting leverages the fact that different operating systems implement different TCP/IP stacks, each of which has a unique signature. Even between versions or patches of an operating system there exit subtle differences as developers include new features and optimize performance.

25 “The classifier examines the initial TCP SYN packets, but determines the probabilistic likelihood of each hypothesis, i.e. operating system, and selects the maximum-likelihood hypothesis” ([44]).

26 We compared these results with Table 1 “Inferred Operating Systems Distribution” within [44].
The drawback of limiting the possible initial TTL values is that packets from end systems that do not use contemplated values will get a wrong estimation of their initial TTL and accordingly, a wrong hop count estimation. However, this method works correctly nowadays in 90% of the cases at least.

We implemented a C program (see Appendix A) which takes an input dump file from the data repository and classifies each TCP conversation with the hops number between the two endpoints of such a conversation. As we previously processed those dump files with tcptrace, we only have to match the RTT samples with the appropriate TCP conversation, whose hops number is known. We did this with another simple C program which processes two text files.

### 3.4.2 Previous Discussion

Before starting to deal with the data from the repository, we are going to discuss a little bit about the relationship between delay and hops number. Intuitively, we think that the more hops number of a packet to reach its destination are, the higher the delay is. Is this assertion always true?

Trying to get some knowledge about this issue, we previously did some active probes with ping and traceret\(^\text{27}\) tools. We started measuring RTT delays and hops number for each POP shown in Figure 1.2.1, from one of our computers in the University of Twente (Enschede, The Netherlands). The results are displayed in Table 5. We also performed other similar measurements to universities (web servers) all over the world (Table 6). From these measurements, we extract the next conclusions:

- Even though the tendency of the delay is to increase when the number of hops do the same, there are some endpoints which need much more hops to be reached and their delay is lower than other endpoints which need less hops to be reached (e.g., University of South Africa or Ohio Valley University versus University of Cádiz). In the path to those endpoints, there are a lot of routers in not too much distance (maybe in the local area) and it is possible that those routers were not indispensable.
- We observe that universities inside The Netherlands are reached between 2 and 8 hops. All the POPs are reached with 6 hops as maximum. So networks directly connected to SURFnet (as the ones of the universities are) should add between 1 and 2 hops more. Then we can say that most of the sites belonging to The Netherlands are reached in less than 10 hops and the first hops belong to the SURFnet network. Anyway, in order to have a geographical criteria as in Table 2 for RTT Figures, we will say that hosts located in The Netherlands and some in Europe are reached in the range 1-12 hops, the rest of Europe and most part of the world (America, Africa, etc) in the range 13-20 hops and finally the farthest places are reached within 21-31 hops.

\(^{27}\) Tracert or traceroute is a TCP/IP utility which allows the user to determine the route packets take to reach a particular host (www.traceroute.org).
As we said before, very few hosts within Internet are reached with more than 30 hops. University of South Australia is reached in 21 hops, which is quite indicative of this.

<table>
<thead>
<tr>
<th>Destination POP</th>
<th>Hops' number</th>
<th>Min. RTT (ms)</th>
<th>Max. RTT (ms)</th>
<th>Avg. RTT (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ms1.amsterdam1.surf.net</td>
<td>6</td>
<td>6</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>ms1.delft1.surf.net</td>
<td>6</td>
<td>6</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>ms1.denhaag1.surf.net</td>
<td>6</td>
<td>5</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>ms1.eindhoven1.surf.net</td>
<td>6</td>
<td>7</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>ms1.enschede1.surf.net</td>
<td>3</td>
<td>1</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>ms1.groningen1.surf.net</td>
<td>5</td>
<td>9</td>
<td>19</td>
<td>12</td>
</tr>
<tr>
<td>ms1.hilversum1.surf.net</td>
<td>5</td>
<td>6</td>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>ms1.leiden1.surf.net</td>
<td>6</td>
<td>6</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>ms1.maastricht11.surf.net</td>
<td>6</td>
<td>8</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>ms1.nijmegen1.surf.net</td>
<td>5</td>
<td>7</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>ms1.rotterdam1.surf.net</td>
<td>6</td>
<td>5</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>ms1.tilburg1.surf.net</td>
<td>5</td>
<td>9</td>
<td>19</td>
<td>11</td>
</tr>
<tr>
<td>ms1.utrecht1.surf.net</td>
<td>5</td>
<td>6</td>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>ms1.wageningen1.surf.net</td>
<td>5</td>
<td>8</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>ms1.zwolle1.surf.net</td>
<td>5</td>
<td>8</td>
<td>17</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 5 – Relation RTT vs. Hops Number for each POP

<table>
<thead>
<tr>
<th>University</th>
<th>Hops' number</th>
<th>Min. RTT (ms)</th>
<th>Max. RTT (ms)</th>
<th>Avg. RTT (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universiteit Twente</td>
<td>2</td>
<td>7</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>Universiteit Utrecht</td>
<td>6</td>
<td>13</td>
<td>16</td>
<td>13</td>
</tr>
<tr>
<td>Universiteit Leiden</td>
<td>7</td>
<td>10</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>Technische Universiteit Delft</td>
<td>8</td>
<td>13</td>
<td>16</td>
<td>13</td>
</tr>
<tr>
<td>University of Cambridge</td>
<td>14</td>
<td>23</td>
<td>28</td>
<td>25</td>
</tr>
<tr>
<td>Ohio Valley University</td>
<td>14</td>
<td>105</td>
<td>137</td>
<td>120</td>
</tr>
<tr>
<td>Universität Dortmund</td>
<td>15</td>
<td>30</td>
<td>79</td>
<td>36</td>
</tr>
<tr>
<td>University of South Africa</td>
<td>16</td>
<td>269</td>
<td>291</td>
<td>271</td>
</tr>
<tr>
<td>University of Cádiz</td>
<td>18</td>
<td>65</td>
<td>68</td>
<td>65</td>
</tr>
<tr>
<td>University of South Australia</td>
<td>21</td>
<td>356</td>
<td>359</td>
<td>356</td>
</tr>
<tr>
<td>California Institute of the Arts</td>
<td>22</td>
<td>158</td>
<td>200</td>
<td>163</td>
</tr>
</tbody>
</table>

Table 6 – Relation RTT vs. Hops Number for some Universities all over the world

Keeping in mind these facts, now we are ready to analyze the data repository more clearly.

### 3.4.3 TTL Distribution

We start our analysis with the study of the TTL values extracted from the IP packets. Figure 3.4.1 shows the frequency distribution of the TTL value in location 1\(^{28}\). We appreciate two big groups of values, one of them near 128

---

\(^{28}\) As the results are very close to the rest of locations, we will only analyse the data from location 1.
and the other one near 64. However, not many values are in the zone of 32 or 255. The figure’s shape is something that we should expect and it justifies our simplification (the limitation of the number of initial TTL values). Moreover we can see that one of the peaks of the distribution is located in 64 (and not in 60), so the ambiguity problem is solved in that case. We cannot say too much in the case (30, 32).

![Figure 3.4.1 - Frequency distribution of the TTL values (Location 1)](image)

The big two peaks located in 128 and 64 are due to packets captured in the source endpoint, just in the same point where the packet monitor is located (zero hops between them), so those values are exactly their initial TTL values. However, this fact is not always like that. It could happen that the packet monitor was one or more hops away from the source host (we would observe a peak in 63 and not in 64, for example). This is not really a problem, we only have to be careful in the hops number computation.

Figure 3.4.2 exhibits the overpowering of 128 as estimated initial value of the TTL (almost 80%). In second place and practically covering the rest of the cases is 64. It manifests, as it was expected, the dominion of the Windows and Linux OSs in the hosts distribution, which use these initial TTL values.
Anyway, these graphs are not saying nothing about the network’s health.

### 3.4.4 Hop’s Number Distribution

In order to know how the distribution of the hops in each location is, we can take a look to the Figures 3.4.3 a), b) and c). As we said in section 3.4.2, the relationship between delay and hops’ number is not always clear, but we test that within location 1 and 3 the percentage of hops lower than 12 (so local connections) is higher. Almost a 6% of connections measured in location 1 are between hosts separated by 1 hop. However, the distribution for location 2 seems to be a gaussian with mean 14 hops, which is coherent because we have to remember that location 2 belongs to a research center and we said that most of its connections were external to The Netherlands (in Table 6 we check that with 14 hops you can reach the University of Cambridge or Ohio Valley University, for example).

In all the locations we also see that it is rare to find connections between endpoints separated more than 23 hops, so as we previously asseverated, it is really infrequent to need 30 hops to reach a destination.

This kind of figures give us an idea of the hosts remoteness, but we think that you can learn more about the hosts’ geographical distribution with the RTT Figures, because they are directly related to the delay and the hops distribution can be deceitful.
Figure 3.4.3 a) - Hops' number distribution (Location 1)

Figure 3.4.3 b) - Hops' number distribution (Location 2)
3.4.5 RTT vs. Hop’s Number

The minimum RTT per hop during two different days (26-05-2002 and 25-06-2002) at different hours (11:15h and 04:15h) is represented in Figure 3.4.4 a). Similarly, the average RTT per hop is displayed in Figure 3.4.4 b). Both, minimum and average RTT are the median of all the collected samples for each hop. With this procedure, we notice about the increasing tendency of the delay with the hops’ number. In this case, the delay of each hop in the local zone (under 12 hops) is lower at 04:15h than at 11:15h, but curiously it is the opposite between 12 and 22 hops. One possible explanation of this is the hours’ difference between the end hosts, because in sites very far away from The Netherlands (more hops are needed) there is more activity at 04:15h than at 11:15h (local hour in The Netherlands).

Figure 3.4.5 shows the minimum and the average RTT per hop in location 1. It is interesting to observe that at 21 hops the delay increases considerably. This fact can be due to a satellite link for really long distances, but we have to say that the amount of valid samples from 20 hops is not very big, and could be that some outliers were giving us a false behaviour of the delay. It was also expected that the delay of 3 and 4 hops was lower than the figure’s displays, which indicates a probable congestion situation there (there are a lot of local connections in location 1).

---

29 Due to the big size of the available files for location 1, we mixed the data only for two files, 26-05-2002 (11:15h) and 25-06-2002 (04:15h), which is quite representative of the general behaviour.
Comparison of the Median of the Minimum RTT per hop (Location 1, 11:15h vs. 04:15h)

Figure 3.4.4 a) - Min. RTT vs. hop's number during two different days at different hours (Location 1)

Comparison of the Median of the Average RTT per hop (Location 1, 11:15h vs. 04:15h)

Figure 3.4.4 b) - Avg. RTT vs. hop's number during two different days at different hours (Location 1)
We followed the same process to evaluate the delay during a week of May within location 2, first at two different hours and later joining all the data to generate a general vision of the delay in location 2.
From Figures 3.4.6 a) and b), we discovered the same fact about the hourly difference beginning with 13 hops that we commented before. Figure 3.4.7 also certifies the increasing tendency of the delay with the hops number as the abrupt ascent of the same one starting at 21 hops. Comparing to Figure 3.4.5, location 2 seems to have less congestion in the first hops than in location 1.
In order to complete the study of the three locations, we will also add the graphs for the location 3 during a week in October (Figures 3.4.8 a) and b) and Figure 3.4.9). Previous comments are also valid here.

![Figure 3.4.8 a) - Min. RTT vs. hop's number during a week at different hours (Location 3)](image1)

![Figure 3.4.8 b) - Avg. RTT vs. hop's number during a week days at different hours (Location 3)](image2)
Now we are in conditions to put the obtained data for all the locations together and to try to understand better their performance. Figure 3.4.10 displays the minimum RTT per hop for all the locations. These locations, which with the RTT Figures seem to have quite different distribution of the delay, here they have the same behaviour, as the curves are practically corresponding (chiefly locations 2 and 3). With the exception of location 1 for 3 hops, the curves are particularly similar between 1 and 12 hops, because all of them have the use of SURFnet network in common or the destination endpoints are not far away from The Netherlands. All of them also exhibit an increasing trend of the RTT with the hops’ number, and an abrupt increment beginning in 21 hops, but curiously in 22 hops there is a drop of the delay again, specially strong for location 2 (we have to remember again that this behaviour could be due to the presence of outliers in the data).
Looking at the average RTT (see Figure 3.4.11) the feeling is that the network in location 2 is working worse than in the other ones, because this metric is the biggest one in most of the hops. On the other hand, it is in location 3 where the network seems to be better.
3.4.6 Other Related Figures

But trying to see this issue more clearly, we compute the subtraction between average and minimum RTT, which can indicate the presented congestion in the path (Figure 3.4.12). For the first 6 hops, location 2 presents the best performance, while locations 1 and 3 present peaks of congestion. This effect can be due to the traffic behaviour of the users (mainly local traffic in location 1 and 3, and external traffic in location 2). From there, location 2 presents the worst delay performance, while location 3 barely suffer from congestion.

Figure 3.4.13 represents the ratio minimum RTT/hop’s number per hops count of the intended destinations. We also observe an increasing trend of this ratio with the number of hops. This fact makes sense, because for farther destinations the space between hops is supposed to be bigger (physical distance), and the propagation delay increases. The three represented curves are quite similar, unless in the third hop within location 1, which the value of the ratio is high and indicate a situation of congestion. We also observe that the range of RTT introduced per hop is 1-20ms. This fact could be useful for characterizing the network.

![Figure 3.4.12](image-url)

*Figure 3.4.12 - Comparison of the Avg. RTT less Min. RTT vs. hop’s number for all the locations*
3.4.7 Conclusions about RTT/FNH Figures

After knowing more about RTT as a Function of the Number of Hops Figures we can asseverate that they provide a good indicator about how the network is working. We think that this kind of graphs can help better to identify in which part of the network we have more problems, as we have separated the connections following the hops’ number that they have needed to reach the endpoints and in the other class of figures the data were more mixed.

If we want to characterize the SURFnet’s delay, this groups of figures are more appropriate than RTT Figures or RTT Variation Figures, because actually we are measuring the delay within connections that have one end in the SURFnet network, and the measured latency does not depend too much of this part for farther endpoints.

The TTL and hops distribution figures are not very indicative of the network’s health; on the other hand all the figures shown in sections 3.4.5 and 3.4.6 give us a quite clear idea about the distribution of the latency in each part of the network, its variability and the possible points of congestion.
Chapter 4

Conclusions and Future Work

4.1 Conclusions

The goal of the project was to get more insight about the latency inside the networks, particularly inside the SURFnet network, but with the use of passive measurements (TCP/IP packet monitoring) to obtain the user perceived performance. Our research question was: “Is it possible to determine ‘network health figures’ with the use of passive measurements of delay?”

Let’s do a small summary first. We started the searching for an answer to this question by investigating the necessary background information within Chapter 1. Thereby, we presented our network under study (SURFnet), the delay definition and the reasons that make necessary its measurement. We explained the differences between active and passive measurements as well.

In Chapter 2, we defined the basic metrics to evaluate the delay (RTT, OWD and jitter) and the reasons to choose RTT as a main metric in our work. We investigated the state-of-the-art in passive RTT measurements which gave us the initial approach to our work and we introduced our data repository from where we took the files to process the data. We also presented the tool to extract valid RTT samples: tcptrace.

From this previous work, we defined in Chapter 3 three different groups of figures to evaluate the health of the network related to the latency: the RTT, RTT Variation and RTT as a Function of the Number of Hops Figures. How does each figure contribute to solve our problem?

The RTT Figures represent the CDF of the RTT samples in terms of TCP connections. This figure can help us in the following way:

- It characterizes the effect of geographical location of each connection’s end-points. We observe this issue perfectly in Figure 3.2.1 e). We clearly distinguish four zones in that figure (from the minimum RTT), one of them belongs to local connections, and the rest to places far away from The Netherlands. This fact allows us to understand the behaviour or habits of the users of that location in terms of usual endpoints destinations, which can help to forecast where it is more likely to suffer from congestion or to design the links to optimize the performance.
- It helps us identify the changes of the traffic with the time within a location. This can serve as a method to estimate the maximum and minimum usage’s level of a link at different hours (e.g. see Figure 3.2.5) and this can be useful to plan the network’s requirements. Or taking a look to the Figure 3.2.7 we are able to check the technology changes in the month’s time scales (we can imagine that we changed a router in
the network in order to improve its performance and we observe the
requested result in July). We could also detect temporal bad
performance due to a problem (e.g. route change).
• We can also appreciate that the range of RTTs experienced by TCP
segments is extremely large (from 1 ms to 10 s), which allows us to have
an idea of the RTT extremes.
• It gives us an approximation of the congestion in the network, if we
observe the difference between the minimum and the average RTT.

The RTT Variation Figures show the variability within TCP connections, and on the
whole we have learned that:

• Connections with smaller minimum RTT show a greater variability in RTTs
  (Figure 3.3.1).
• Connections with higher median RTTs also exhibit a larger disparity in the
distribution of RTTs (Figure 3.3.4).
• The average RTT is likely to be between 1 and 4 times the minimum RTT.

However, these affirmations are always applicable in whatever IP network, so
they do not give us too much information about the actual performance of the
network. It is our measurement of jitter (Figure 3.3.6), which can serve us better
for our aims. This study of the worst case of variability can be used to design the
buffers to correct such jitter or to decide if it is possible to run a determined
application in the network.

Finally, we studied the RTT as a Function of the Number of Hops. We explained
the way to obtain such figures from the TTL field of the IP packets and the
problem of the initial values that depend of the OS. From these figures, we have
concluded that:

• The hop’s number distribution is indicative of the geographical
distribution of the connection’s end-points.
• It is rare to find connections between end-points separated more than
  23 hops, and it is really infrequent to need more than 30 hops to reach a
destination.
• The median of the RTT samples in each hop presents an increasing trend
  when the number of hops grow, as we expected previously.
• The first 10 hops give us an indication of the SURFnet performance and
  with these figures we can study better different parts of the network.
• If we compare the minimum and average RTT at different times in the
  monitored link, we can know when the network is working better.
• Figure 3.4.12 gives us an approximation of the average congestion in
  each hop, so we are able to determine more exactly the point where
  the network is not working properly.

Within sight of these results, the feeling is that we have really found suitable
figures to characterize the network’s delay. We do not have a “winner figure”,
because all these graphs complement each other and we found different
nuances of the same fact, which can help us understand better the network
performance. The use of passive measurements is very appropriate for
modeling Internet traffic and as all the information that we obtain is real (not
from probe traffic), we obtain the best approximation to the network performance perceived by users. Although the passive measurements depend entirely on the presence of appropriate traffic on the network to extract the desired data, in the case of the delay it is not very difficult and we are able to infer the performance of the network. In this case, the major limitation could be the big amount of data that need to be stored to extract accurate measurements.

4.2 Future Work

Now we know that we are able to infer the performance of the network with the use of passive measurements of the delay. The next step would be to build an application (e.g. a web application) which gets all these figures together and gives us the option to compare the results in different moments of the time. It could take measurements at certain times and later update the statistics automatically.

We could make, for example, a table similar to Figure 1.2.1, but using the number of hops and the minimum, maximum and average RTT and jitter as well. Then we would need to find an appropriate threshold value for each metric to decide if the network is going well or not (in the same way of the green, yellow and red colors of that figure). The first hops would help us gauge the current SURFnet performance, and in the future, when SURFnet6 is available, we will be able to compare between them. It is expected that connections that use light paths will reduce the latency, specially when the delay is not dominated for the propagation time (e.g. transatlantic path) and instead of having a big amount of routers, now we have a direct light path. The jitter will be improved as well.

It could also be interesting to compare these results with the same ones obtained with active measurements, and then determine when it is more appropriate to use each method and we could check if the provided results are parallel.

Nevertheless, the imminent emergence of next generation networks as SURFnet6 implies the necessity of providing tools and insight to benchmark hybrid networks, and this will probably be the next challenge.
References

http://www.surfnet.nl/info/en/home.jsp

http://www.gigaport.nl/info/en/home.jsp

http://www.netherlight.net/info/home.jsp


http://m2c-a.cs.utwente.nl/repository/

http://www.tcpdump.org/

http://www.tcptrace.org/

http://www.glif.is/

http://www.ietf.org/html.charters/ippm-charter.html


http://www.mathworks.com/

Analysis of the Delay in the SURFnet Network


[17] Inferring TCP Connection Characteristics Through Passive Measurements (Sharad Jaiswal†, Gianluca Iannaccone§, Christophe Diot§, Jim Kurose†, Don Towsley; INFOCOM 2004).


[21] Internet End-to-End Performance Monitoring at SLAC. http://www-iepm.slac.stanford.edu/

[22] CAIDA, the Cooperative Association for Internet Data Analysis. http://www.caida.org/


[28] M2C Deliverable D1.5 (Remco van de Meent; University of Twente, 2005). http://arch.cs.utwente.nl/projects/m2c/m2c-D15.pdf


http://www.winpcap.org/


[34] Understanding Delay in Packet Voice Networks (Copyright © 1992-2005 Cisco Systems).  


http://surfstat.surfnet.nl/rttpl/

http://en.wikipedia.org

[38] One-way Delay Measurement Using NTP (Vladimír Smotlacha, CESNET, Prague, Czech Republic).  


[40] New Methods for Passive Estimation of TCP Round-Trip Times (Bryan Veal, Kang Li, and David Lowenthal; PAM 2005).

http://stat.cwru.edu/~nidhan/onlinepaper/nettop.pdf

[42] RTT Stats (tcptrace).  
http://www.tcptrace.org/manual/node9_mn.html

http://www.cs.wm.edu/~hnw/courses/cs780/papers/ccs03.pdf


[45] Default TTL Values in TCP/IP.  
http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html
http://www.ouah.org/incosfingerp.htm

[47] Lists of fingerprints for passive fingerprint monitoring (Lance Spitzner; May 2000).  
http://www.honeynet.org/papers/finger/traces.txt

[48] Browser News (Stats).  
Appendix A

Source Code of tcphops.c

We present in this appendix the C source code of the program that we have called tcphops.c. In the documentation section of [32], we can find the requirements to run this application under Windows. This program read all the TCP segments of a dump file (created with tcpdump) and computes the hop’s number for each TCP conversation.
Analysis of the Delay in the SURFnet Network

```c
#include <stdio.h>
#include "pcap.h"
#include "netinet/ip.h"
#include "string.h"
#include "stdlib.h"

/* 4 bytes IP address */
typedef struct ip_address{
    u_char byte1;
    u_char byte2;
    u_char byte3;
    u_char byte4;
    }ip_address;

/* IPv4 header */
typedef struct ip_header{
    u_char vers_hl;    // Version (4 bits) + Internet header length (4 bits)
    u_char tos;       // Type of service
    u_short id;       // Total length
    u_short identification; // Identification
    u_short flags_f;  // Flags (3 bits) + Fragment offset (13 bits)
    u_char ttl;       // Time to live
    u_char proto;     // Protocol
    u_short src;      // Source address
    u_short dadr;     // Source address
    u_short daddr;    // Destination address
    u_int op_pad;     // Option + Padding
    }ip_header;

/* TCP Header */
typedef struct tcp_header{
    u_short sport;    // Source port
    u_short dport;    // Destination port
    u_int seqnum;     // Sequence Number
    u_int acknum;     // Acknowledgment number
    u_char dflag;     // Reserved
    u_char rflag;     // Urgent pointer
    }tcp_header;

/* Struct to store all the incoming packets */
struct packetlist {
    ip_address saddr; // Source address
    ip_address daddr; // Destination address
    u_short sport;   // Source port
    u_short dport;   // Destination port
    u_char ttl;      // TTL field of the IP header
    struct packetlist *next; // Next element in the list
};
typedef struct packetlist Packetlist;

/* Struct to store packets that belongs to the same TCP conversation */
struct conversationlist {
    ip_address saddr; // Source address
    ip_address daddr; // Destination address
    u_short sport;   // Source port
    u_short dport;   // Destination port
    u_char hopnumber; // Hops' number computation
    u_int flags;     // To decide if it is a valid data or not
    struct packetlist *packets; // Packet's list of this conversation
    struct conversationlist *next; // Next element in the list
};
typedef struct conversationlist Conversationlist;

#define PNULL ( (Packetlist *)NULL )
#define CNULL ( (Conversationlist *)NULL )
```
Analysis of the Delay in the SURFnet Network

Alberto Castro Hinojosa

void dispatch_handler(u_char *, const struct pcap_pkthdr *, const u_char *);
void deletePacketlist(PacketList * l);
void deleteConversationlist(ConversationList * l);
void print протокол(ConversationList * l);

int main(void)

int main(int argc, char *argv) {
pzp = 0;
char errorbuf[PCAP_ERRBUF_SIZE];
_u_char hops1,hops2;
int valid = 0;
int count = 0;
headPL = NULL;
next = NULL;
hopCL = CLNULL;
max = 0;
}

/* Check the input arguments */
if (argc != 2) {
    printf("Usage: %s filename", argv[0]);
    return -1;
}

/* Open a capture file */
if (!pcap_open_offline(argv[1], errorbuf) || NULL) {
    printf("stderr,\nError opening dump file\n")
    return -1;
}

/* Initial assert */
printf("pcap.hops is executing now. Please, be patient...\n");
/* Read and dispatch packets until EOF is reached */
/* We put all the TCP segments in a Packetlist structure within dispatch_handler */
/* pcap_loop: fp, 0, dispatch_handler, NULL; */
/* Now we want to separate the TCP segments in TCP conversations */
/* The first time, there isn't Conversationlist, we create it */
headCL = (ConversationList*)malloc(sizeof(struct ConversationList));
headCL->next = CLNULL; // Next element is empty yet
headCL->addr = headPL->addr;
headCL->length = headPL->length;
headCL->port = headPL->sport;
headCL->port = headPL->port;
headCL->port = headPL->port;
headCL->packets = (PacketList*)malloc(sizeof(struct PacketList));
/* We fill the fields of the first packet which belongs to this conversation */
headCL->packets->addr = headPL->addr;
headCL->packets->sport = headPL->sport;
headCL->packets->sport = headPL->sport;
headCL->packets->sport = headPL->sport;
headCL->packets->ttl = headPL->ttl; // Received TTL of this packet
headCL->hopcount = 0;
/* Initially hopcount = 0 in this conversation */
headCL->hopcount = 0;
/* Initially packets are not valid */
next = headCL->packets;
max = 0;
/* Delete the first element of the packet list, because was added to the conversationlist */
free(next);
hopCL->temp = next;
temp->temp = next;
}

/* We look over the packet list to classify the packets into their TCP conversation */
while (headPL != NULL) {
    while (next != NULL) {
        /* The current list is not finished */
        /* If the packet belongs to this TCP conversation... */
        /* We accept only conversations with samples of packets from both sides */
        if (next->addr->byte1 == next->addr->byte1) {
            if (next->addr->byte1 == next->addr->byte1) {
            }}
        }
    }
}
Alberto Castro Hinojosa

Analysis of the Delay in the SURFnet Network

```c

/* Now we have all the packets organized per TCP conversations, let's compute hops number */

aux3 = headCL;
while (aux3 != NULL) { // While we have TCP conversations
    aux = aux3->packet;
    /* The initial values of this variables are the ttl values */
    hop1=aux->ttl1; // hop1: number of hops from the source to the measurement point
    hop2=aux->ttl2; // hop2: number of hops from the destination to the measurement point
    valid = 0;

    /* Check the conditions to accept the segments for that TCP conversation */
    /* We only want the segments to compute the hops if we have segments in both directions */
    while (hop1 -- hop2 % & aux->next != NULL) { // Valid
        if ( (aux->saddr.byte1 != aux->next->saddr.byte1) ||
            (aux->saddr.byte2 != aux->next->saddr.byte2) ||
            (aux->saddr.byte3 != aux->next->saddr.byte3) ||
            (aux->saddr.byte4 != aux->next->saddr.byte4) ) {
            valid = 1; // Valid
        }
        hop1=aux->next->ttl1;
        aux = aux->next;
    }

    if (valid == 0) {
        hop1=hopcounter(hop1);
        hop2=hopcounter(hop2);
        aux3->hopnumber=hop1-hop2; // hopnumber is the sum
        aux3->valid = 1; // valid template
    }
    aux3 = aux3->next; // Next elements in the conversation list
} // End of list

/* Print results */
printf("\nTCP connections vs. number of Hops:\n");
print(hopcounter(hop1)); // Free memory
deleteConversationList(headCL);
return 0;
}
```

/* Dispatcher handler */
/* Read all the packets and store them in only TCP in a PacketList structure. */

void dispatcher_handler(u_char *tmpl, const struct pcap_pkthdr *header, const u_char *pkt_data) {
    ip_header = ih;
tcp_header = tph;
    u_int ip_len;

    /* Retrieve the position of the ip header */
    ip_len = (ip_header?)((pkt_data + 14)); // Length of ethernet header
    ip_len = (ip->ip_hl * 4); // 4
    /* Retrieve the position of the tcp header */
    tcp_len = (tcp_header?)(pkt_data + 14 + ip_len);
    /* Only TCP packets are examined */
    if (tcp_len > 0) {
        /* Store all the packets in a PacketList */
        if (headFL != NULL) {
            // The first time, there isn't list, we create it
            headFL = (PacketList*)malloc(sizeof(struct PacketList));
        }
        ```
Analysis of the Delay in the SURFnet Network

```c
headPL->next = NULL; // Next element is empty yet
headPL->addr = th->addr; // Copy IP Dest from ip header
headPL->saddr = th->saddr; // Copy IP Src from ip header
headPL->ttl = th->ttl; // Copy TTL field from ip header
headPL->sport = tcp->sport; // Copy Src Port from tcp header
headPL->dport = tcp->dport; // Copy List Port from tcp header
aux2 = headPL;

} /* Otherwise, we create a new element at the end of the list */
else {
    aux = (PacketList*)malloc( sizeof(struct PacketList) ) ;
    aux->next = NULL;
    aux->addr = nh->addr;
    aux->saddr = th->saddr;
    aux->ttl = nh->ttl;
    aux->sport = tcp->sport;
    aux->dport = tcp->dport;
    aux->next = aux1;
    aux2 = aux;
}
}

/*******************************************************************************/
/* DeletePacketList */
/* Free memory of a created PacketList */
void deletePacketList( PacketList* l ) {
    PacketList* temporary;

    while ( l ) {
        temporary = l->next;
        free(l);
        l = temporary;
    }
}

/*******************************************************************************/
/* Free memory of a created ConversationList */

void deleteConversationList( ConversationList* l ) {
    ConversationList* temporary;

    while ( l ) {
        temporary = l->next;
        deletePacketList(l->packets);
        free(l);
        l = temporary;
    }
}

/*******************************************************************************/
/* Print hops */
/* Print the TCP connections with the hop number */

void printHops( ConversationList* l ) {
    ConversationList* temporary;
    int i;
    temporary = l;

    for(i=1;i<=32; i++) {
        printf("\n\n Hop Number = %d, ");
        printf("\n-------------------------\n'\n' );

        while (temporary)
            if (temporary->valid == 1 && temporary->hopnumber == i){
                printf("%s\n%d \rightarrow %d \rightarrow %d \rightarrow %d\n", temporary->add1.byte1,
                        temporary->add1.byte2,
                        temporary->add1.byte3,
                        temporary->add1.byte4,
                        temporary->add1.byte5,
                        temporary->add1.byte6,
                        temporary->add2.byte1,
                        temporary->add2.byte2,
                        temporary->add2.byte3,
                        temporary->add2.byte4,
                        temporary->add2.byte5,
                        temporary->add2.byte6,
                        temporary->add2.byte7,
                        temporary->add2.byte8,
                        temporary->add2.byte9,
                        temporary->add2.byte10,
                        temporary->add2.byte11
                        temporary->add2.byte12);
            }
        temporary = temporary->next;
    }
}
```
\begin{verbatim}
    temporary = 1;
    
    // Calculates the hops with the estimated initial TTL (32, 64, 128, 255) and the read TTL from packets */
    u_char hopcounter(u_char h) {
        if (h <= 32) {
            return (32-h);
        } else {
            if (h <= 64) {
                return (64-h);
            } else {
                if (h <= 128) {
                    return (128 - h);
                } else {
                    return (255 - h);
                }
            }
        }
    }
\end{verbatim}
Appendix B

Minimum RTT vs. SYN RTT

In order to verify if the SYN RTT may be used as a reasonable approximation of the minimum RTT, we used the data of two weeks (one in May and the other one in June) from location 2, and we plotted the CDF of the ratio minimum RTT/SYN RTT (see Figure AppB. 1). This figure presents a similar shape to Figure 2.2.1, but we do not obtain exactly the same results as in [16]. From our figure, we can say that in this case, only in 48.5% of connections the minimum RTT is equal to the SYN RTT. However, for more than 70% of connections the SYN RTT exceeds the minimum RTT by less than 10%, which really suggests that the SYN RTT may be used as a reasonable approximation of the minimum RTT.

Figure AppB. 1 - CDF of the Ratio Min. RTT / SYN RTT