UNIVERSITY OF OSLO
Department of Informatics

Predicting TCP congestion through active and passive measurments

Master thesis

Ismail A. Hassan

May 2005

Abstract

The Transmission Control Protocol (TCP) has proved to be a reliable transport protocol that has withstood the test of time. It is part of the TCP/IP protocol suite deployed on the Internet, and it currently supports a variety of underlying networking technologies such as Wireless, Satellite and High-Speed networks.

The congestion control mechanism used by current implementation of TCP (known as TCP-Reno/new Reno) is based on the Additive Increase Multiple Decrease (AIMD) algorithm that was first introduced by Van Jacobsen in 1988[1] after the Internet experienced heavy congestion which subsequently led to a phenomenon called *congestion collapse*. The algorithm assumes no prior knowledge of end-to-end path conditions and blindly follows the same routine at the beginning of every connection namely, a slow start phase, a congestion avoidance phase and in the event of a lost segment reduces the transmission rate accordingly.

The network will experience different conditions depending on the amount of traffic exerted on it. At times it will endure heavy load while at other times there will be small amount of traffic. In the event that the end-to-end path characteristics are known and the amount of traffic generated is predictable, the AIMD algorithm does not take advantage of that information. In this thesis we investigate ways of predicting the available bandwidth between two hosts frequently in contact with each other through the deployment of bandwidth estimation tools. We would like to explore the possibility that AIMD can take advantage of bandwidth measurements collected between these hosts.

Acknowledgments

I would like to thank my adviser, Dr. Tore Møller Jonassen for his patience and confidence in my abilities. Giving me the freedom to conduct this project while at the same time been available for me if needed.

My gratitude goes to Prof. Mark Burgess for always encouraging us to seek higher goals and become better scientist. He has been a source of inspiration for me and I hope all the best for him.

I would also like to thank the staff of the master program: Kyrre Begnum, Kirsten Ribu, Siri Fagernes, Hårek Haugerud, Simen Hagen and Frode E. Sandnes for their support and encouragement throughout the 2 year period of the master program.

My sincere thanks go to my fellow graduate students in the MSc in Network and system administration: Fatima, Claudia, Ellef, Håvard, Stig Jarle, Weng Seng, Raheel, Trond, Le, Phong, Bård and Ole for their support and friendship.

Last but not least I would like to thank Uninett for giving me the necessary equipment and account to conduct my research without any obstacles.

Dedication

To my beautiful wife and kids for their everlasting love and support.

Preface

This Master thesis is written in partial fulfillment of the requirements for the degree of Master of Science in Network and system administration at Oslo University College. The master program is a co-operation between Oslo University College and University of Oslo.

Network and System administration is about designing, running and maintaining a networked systems consisting of hardware (computers, routers and switches), software and the most important part of all users. Having gained the necessary practical and theoretical skills of system management through the Master degree course at Oslo University college it was natural for me to choose a more technical topic related to my academical background.

The thesis studies TCP and its congestion control mechanism in detail. The latest research into the the area of bandwidth estimation techniques is also explored. This project as a whole has been very instructive and I have acquired a lot of knowledge through it.

Contents

1 Introduction **5**
- 1.1 Motivation . . . 6
- 1.2 Research objectives . . . 6
- 1.3 Thesis Structure . . . 6

2 Background **7**
- 2.1 TCP/IP history . . . 7
  - 2.1.1 The layering concept . . . 8
- 2.2 The Transmission Control Protocol (TCP) . . . 10
  - 2.2.1 Connection oriented . . . 10
  - 2.2.2 Reliable delivery . . . 10
  - 2.2.3 Flow control . . . 11
  - 2.2.4 Congestion control . . . 12
- 2.3 Congestion control issues in TCP . . . 15
  - 2.3.1 Interpreting loss as congestion . . . 15
  - 2.3.2 Fast long-distance networks . . . 15
- 2.4 TCP implementation that address the problem . . . 16
  - 2.4.1 TCP Vegas . . . 16
  - 2.4.2 TCP Westwood . . . 17
  - 2.4.3 ECN & RED . . . 17
  - 2.4.4 SACK . . . 18
  - 2.4.5 PFLDnet . . . 18

3 Related work **21**

4 Methodology **22**
- 4.1 Measurement methods . . . 23
  - 4.1.1 Passive measurements . . . 23
  - 4.1.2 Active measurements . . . 23
- 4.2 Bandwidth measurements . . . 23
- 4.3 The methods used to estimate bandwidth . . . 24
  - 4.3.1 Variable packet size (VPS) . . . 24
  - 4.3.2 Packet Pair technique . . . 25

4.3.3 Trains of Packet Pairs(TOPP) . . . 25
4.3.4 Self-loading Periodic Streams (SLoPS) . . . 26
4.4 The equipment & software tools used . . . 27
4.4.1 Equipment . . . 27
4.4.2 Tcpdump . . . 27
4.4.3 Tcpslice . . . 28
4.4.4 Traceroute . . . 28
4.4.5 Pathrate . . . 28
4.4.6 Pathload . . . 28
4.5 Experimental Setup . . . 29
4.5.1 Examining network traffic patterns of users . . . 29
4.5.2 Is the route taken by the packet stable ? . . . 30
4.5.3 The bandwidth estimating process . . . 30

5 Results analysis **32**
5.1 Type 1 . . . 32
5.2 Type 2 . . . 33
5.3 Type 3 . . . 34
5.3.1 Analyzing the results of pathrate . . . 34
5.3.2 Analyzing the results of pathload . . . 35

6 Proposals **39**
6.1 Proposed solution . . . 40
6.1.1 The Web100 kernel . . . 40
6.1.2 Network Tool Analysis Framework (NTAF) . . . 40
6.1.3 Work Around Daemon (WAD) . . . 41
6.2 Limitations . . . 41
6.2.1 Operating system (OS) related . . . 41
6.2.2 Access to end-2-end hosts . . . 41

7 Conclusions and Discussion **42**
7.1 Future work . . . 43

Appendices **43**

A **44**

B Abbreviations **47**

List of Figures

2.1 Layering concept 8
2.2 Frame 9
2.3 Connection establishment 11
2.4 TCP header 11
2.5 AIMD congestion control behavior 13
2.6 Slow start 14

4.1 Overview of bandwidth metrics 24
4.2 Packet pair technique 25
4.3 Topp 26
4.4 SLoPS 27
4.5 Setup 1 29
4.6 Setup 2 30
4.7 Setup 3 31

5.1 Available bandwidth week 1 35
5.2 Available bandwidth week 2 35
5.3 Frequency distribution 36
5.4 Average available bandwidth 38

List of Tables

2.1 HighSpeed TCP parameters 18

4.1 Equipment specification 27

5.1 Results of host monitoring 32
5.2 Output from traceroute 1 33
5.3 Output from traceroute 2 33
5.4 Pathrate results 34
5.5 Frequency distribution 2 36
5.6 Predicted bandwidth 37

A.1 Measured bandwidth 1-1 44
A.2 Measured bandwidth 1-2 45
A.3 Measured bandwidth 2-1 45
A.4 Measured bandwidth 2-2 46

Chapter 1

Introduction

The Transmission Control Protocol (TCP) has proved to be a reliable transport protocol that has withstood the test of time. It is part of the TCP/IP protocol suite deployed on the Internet, and it currently supports a variety of underlying networking technologies such as Wireless, Satellite and High-Speed networks. From its original standard [2], the protocol has undergone several face lifts[3] in order to cope with emerging technological demands. Most of the changes made to TCP primarily addressed ways of dealing with or avoiding congestion on the Internet. Congestion arises when intermediate devices such as routers and switches cannot cope with the traffic generated by the end systems.

The congestion control mechanism used by current implementation of TCP (known as TCP-Reno/new Reno) is based on the Additive Increase Multiple Decrease (AIMD) algorithm that was first introduced by Van Jacobsen in 1988[1] after the Internet experienced heavy congestion which subsequently led to a phenomenon called *congestion collapse*. AIMD was later supplemented with two more algorithm[3] called Fast retransmit and Fast recovery to further enhance the performance in certain situations. The algorithm assumes no prior knowledge of end-to-end path conditions and blindly follows the same routine at the beginning of every connection namely, a slow start phase, a congestion avoidance phase and in the event of a lost segment reduces the transmission rate accordingly.

The network will experience different conditions depending on the amount of traffic exerted on it. At times it will endure heavy load while at other times there will be small amount of traffic. In the event that the end-to-end path characteristics are known and the amount of traffic generated is predictable, the AIMD algorithm does not take advantage of that information. If congestion normally occurs at a particular time of the day between host A and host B, there is a high probability that congestions will happen at similar times in the future also. Having such knowledge about the long time behavior of an end-to-end path is indispensable information that is not incorporated in current implementations of TCP congestion control.

1.1 Motivation

Research conducted at Oslo University College in[4, 5] indicate that system activity is influenced by the behavior of users. Users exert a random influence on the system leading to fluctuating levels of demand and supply for resources. In such systems, one finds a number of regularities that reflect the social work and leisure patterns of the users over time. The amount of network traffic exerted on to the network is obviously influenced by these fluctuating demand and supply for resources. Taking that into consideration, we would like to investigate how the social work and leisure patterns impact the available bandwidth in a network?

1.2 Research objectives

- Investigate the behavior of the network traffic and distinguish if there exists a long term relationship between two peers. If a user has a preference in such way that the individual more often connects to certain hosts compared to others. The definition of what we mean by a long term relationship is described having contact with each other on a daily basis for a minimum of a week.

- Whether the path taken by a packet from host A to host B is stable. Does the packet take the same route every time?

- Find out ways to estimate available bandwidth and explore ways that these estimations can be helpful to TCP.

In this thesis we investigate ways of predicting the available bandwidth between two hosts frequently in contact with each other through the deployment of bandwidth estimation tools. We would like to explore the possibility that congestion control mechanism in TCP can take advantage of bandwidth measurements collected between these hosts. The aim is to have a good initial estimate of the starting transmission rate based on predictions calculated from long periods of measurements rather than resorting to slow start. We are also interested in predicting slow start threshold *(ssthresh)* from these calculations.

1.3 Thesis Structure

An overview of TCP fundamentals and issues encountered with its algorithms are presented in Chapter 2. The purpose of this chapter is to give the reader an understanding of the basics of TCP. Chapter 3 describes some related work. The methodology deployed in this thesis and experiments conducted are described in chapter 4. The results obtained through the tests are presented in Chapter 5. Proposals and limitations are discussed in Chapter 6. Conclusion and further work are presented in Chapter 7.

Chapter 2

Background

2.1 TCP/IP history

The Transmission Control Protocol/Internet Protocol (TCP/IP) is composed of a number of protocols that together specify how data can be exchanged between machines. A descendant of the Advanced Research Project Agency Network (ARPANET) which was established and funded by the U.S. defense department in 1969, it has become the de facto standard that is used on the Internet today. TCP/IP deploys the concept of layering in which each layer has a distinct task to perform. The idea of having separate layers stems from the computer programming world, where programs are divided into smaller functions/methods and structs/classes communicating with each other through calls (message passing). The aim is to let software developers concentrate on a particular layer and not be bothered by the other layers.

Application programmers for example do not require to know about the underlying infrastructure and are provided with the library/interface needed to pass a message to the layer below. The Berkley Socket API which is a set of C function calls that support network communication is such a library. Hardware developers benefit also from this separation of layers since they can produce a product specifically tailored for a defined layer quickly and cheaply. Cisco routers and switches are such products that mainly deal with the Network and the Data Link layers respectively see figure 2.1 for description on layers.

Another advantage with the layering concept is the ability to cope with technological changes. A new and better performing protocol can easily be adapted at any of the layers without having a profound impact on the other layers.

2.1.1 The layering concept

In the 1980's the International Standards Organization (ISO) began developing a model for a network system, called Open Systems Interconnection (OSI) model. This model had 7 layers and is mostly used as a reference in the networking world. The model never appealed to a wide audience and instead TCP/IP became the widely accepted and deployed model. We will now describe the different layers of TCP/IP and how they interact with each other. This is also depicted in figure 2.1.

The Application layer

The application layer is where data starts its journey. At this layer we have the popular applications such as the Web (HTTP), Mail(SMTP) and File Transfer Protocol(FTP). After data is generated by the application, it passes it to the layer below in this case the transport layer. Application programmers decide in advance the type of transport protocol that is suitable for the particular application. Depending on whether a reliable transport protocol is needed or not, the programmer either chooses TCP or UDP as means of transporting the packet.

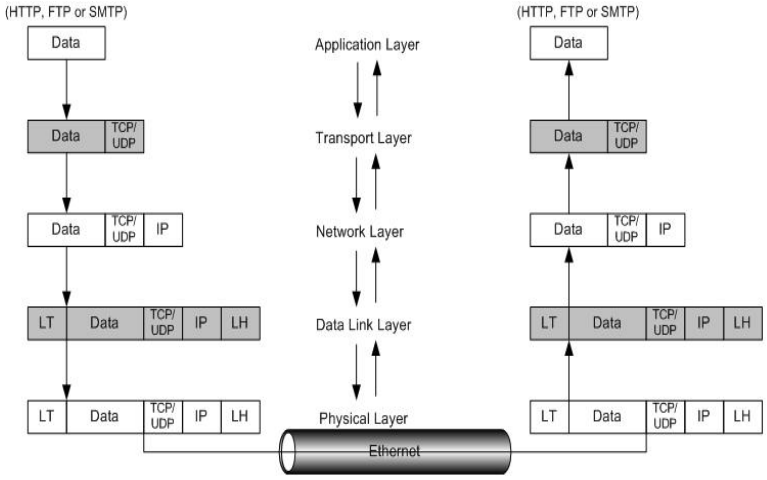

Figure 2.1: TCP/IP layer structure

Transport layer

When the transport protocol is handed a packet by the application, it extracts the necessary information from it. The most important been the port number that identifies the receiving application. This information is populated into a header see figure 2.4 and the data is encapsulated with this header. The header is nothing more than a marker which indicates the transition from the bits that belong to the data part and the ones that belong to TCP/UDP see figure 2.2. The header + data is called a segment at the transport layer, and this segment is passed to the Network layer.

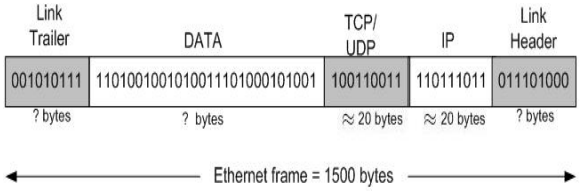

Figure 2.2: Frame

Network layer

The Network layer (Layer 3) is responsible for moving data across an internetworking. All the information needed to do so is contained in its header. IP is the protocol that operates at this layer. The protocol encapsulates the segment with its header and passes the datagram (segment + IP header) to the data link layer.

Data Link layer

Just before data is injected into the transmission media, the Data Link layer receives the datagram from IP and encapsulates it with a header and a trailer. The task of the Data Link layer is to move data across a link. A link is usually a direct connection between two devices that operate at layer 2 or 3.

Protocols that operate at this layer are among others, Point-to-Point Protocol (PPP) and Serial Line Interface Protocol (SLIP). Ethernet/IEEE 802.3, Token Ring / IEEE 802.5 and Fiber Distributed Data Interface (FDDI) are also Data Link protocols, but they also stretch to layer 1. After the header and the trailer are put in place, the frame is passed to the physical layer.

Physical layer

This is the actual physical media that is in use by the host. It could be a wired media such as Ethernet, Frame -relay or Fiber Optics. It could be wireless i.e IEEE 802.11, IEEE 802.16 or Bluetooth. The Network Interface Card (NIC) installed on the machine decides the type of media it can communicate with.

2.2 The Transmission Control Protocol (TCP)

The TCP/IP protocol suite includes two transport protocols, namely TCP and UDP. Applications such as web browsing (HTTP), e-mail (SMTP) and file transfer (FTP) all use TCP as the underlying transport protocol. TCP provides a reliable, connection oriented transport service that guarantees the delivery of data in order before gracefully terminating the connection. It also has a way of controlling the amount of traffic injected into the network through a process known as flow control and congestion control. We will further elaborate on how TCP-Reno/new Reno (refer to as TCP in this thesis) accomplishes this complex tasks in the following sections.

2.2.1 Connection oriented

Before an application using TCP as a transport protocol can transfer any data, it has to establish a connection with the receiving end by going through a process called the three-way handshake. The sender initially sends a SYN packet, waits for a SYN-ACK reply packet from the receiver and then replies with an ACK to complete the process. This is depicted in 2.3.

During the three-way handshake the sequence numbers are initialized randomly, thereafter each segment that is exchanged by the peers is assigned a unique sequence number which also indicates the amount of bytes sent. Upon completion of the three way handshake, the connection is established and data transmission can start. When all the data is sent, the sender and the receiver exchange FIN and ACK in both directions to terminate the connection.

2.2.2 Reliable delivery

TCP guarantees the delivery of data in order and without any duplication. It achieves this through the use of positive acknowledgment and retransmissions. The TCP sender assigns a sequence number to each byte sent, and expects an acknowledgment to be returned by the receiver. There is a sequence number field and acknowledgment filed in the TCP header which allows the sender and the receiver to exchange this information see figure 2.4. It also associates a timer with the sent data and if acknowledgments are not received within the given time period, it retransmits the data. TCP has another

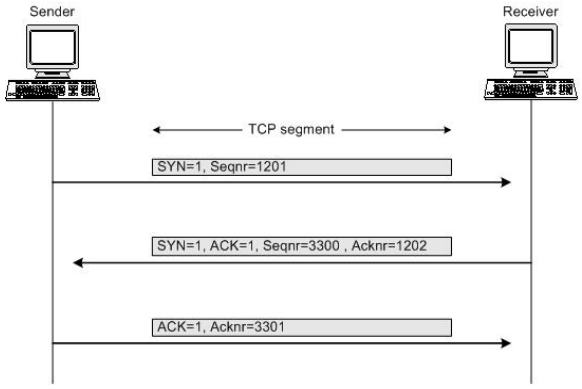

Figure 2.3: Connection establishment

mechanism which can trigger retransmission and that is duplicate acknowledgments (DACK) see section 2.2.4. When bytes arrive out of order at the receiver, it would generate DACK to inform the sender of the situation asking for the next expected bytes in order.

2.2.3 Flow control

Flow control ensures that the sender does not overwhelm the receiver with too much traffic. Buffers are allocated during the three-way handshake phase at both the sender and the receiver to accept incoming packets. To avoid buffer overflow which can lead to packet drops, messages are exchanged between the sender and the receiver indicating the buffer size. The sender is informed explicitly about available buffer size at the receiver through the window field of the TCP header (see figure 2.4). The receiver side

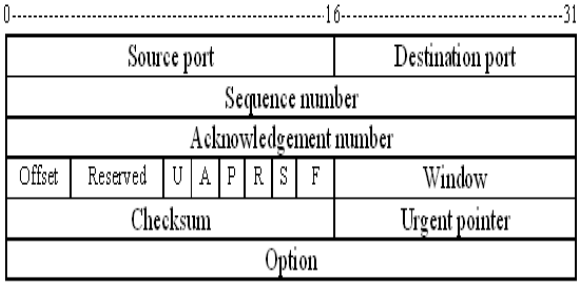

Figure 2.4: TCP header

of the connection keeps count of the number of the last byte read (lastByteRead) by the receiving application i.e. HTTP, FTP, SMTP and etc. It also keeps count of the number of the last byte that has arrived in-order from the network and placed in the buffer (lastbyteRcvd). Buffers for incoming packets (rcvBuffer) is adjusted as follows:

$$lastbyteRcvd - lastByteRead \leq rcvBuffer.$$

The window that is advertised to the sender is calculated based on the following:

$$window = rcvBuffer - (lastbyteRcvd - lastByteRead).$$

The sender side of the connection must also ensure that the amount of data it sends is accordance with the size of the *window*, therefore it maintains the variable advertisedWnd which is calculated from:

$$lastByteSent - lastByteAcked \leq advertisedWnd.$$

Apart from congestion window which we will elaborate on in the next section, the amount of data the sender can send is limited by its effective window (effWnd).

$$effWnd = advertisedWnd\ (lastByteSent - lastByteAcked).$$

2.2.4 Congestion control

Congestion control mechanism in TCP plays an essential role both for the application requesting the service and the Internet as whole. The mechanism provisions the amount of traffic that can be injected into the network thus governing the overall performance of the communicating processes. Congestion occurs when demand for network resources is larger than what the network can supply, thus leading to some packets been dropped along the way. For TCP, dropped packets means retransmissions.

Routers along the path of a connection cannot explicitly inform TCP of any congestion, due to the separation of layers. Intermediate routers operate at the network layer and therefore do not interact with the transport layer. This preserves the layered structure of the TCP/IP suite where every layer has a distinct function. Congestion is therefore detected through the loss of a packet. A packet is considered lost if an ACK is not received within a time frame (RTO) or upon the reception of duplicate ACKs. The algorithms that dictate congestion control are respectively Slow Start (SS), Congestion Avoidance (CA), Fast Retransmit and Fast Recovery [3]. Further explanation of these algorithms will be done in the following sections.

Slow start & Congestion avoidance

TCP has no knowledge of the paths bandwidth capacity upon establishing a new connection, and must therefore resort to an estimation of the available bandwidth. It accomplishes this by going through a process known as Slow Start and Congestion Avoidance. In Slow Start, the sender side of the connection maintains a variable called Congestion Window (CWN) which is initialized to 1 Maximum Segment Size (MSS). CWN determines the maximum number of outstanding packets that have been transmitted but not yet acknowledged. The TCP sender also keeps track of the receiver's advertised window size (RcvWnd) and calculates its transmission rate (sendWnd) as follows:

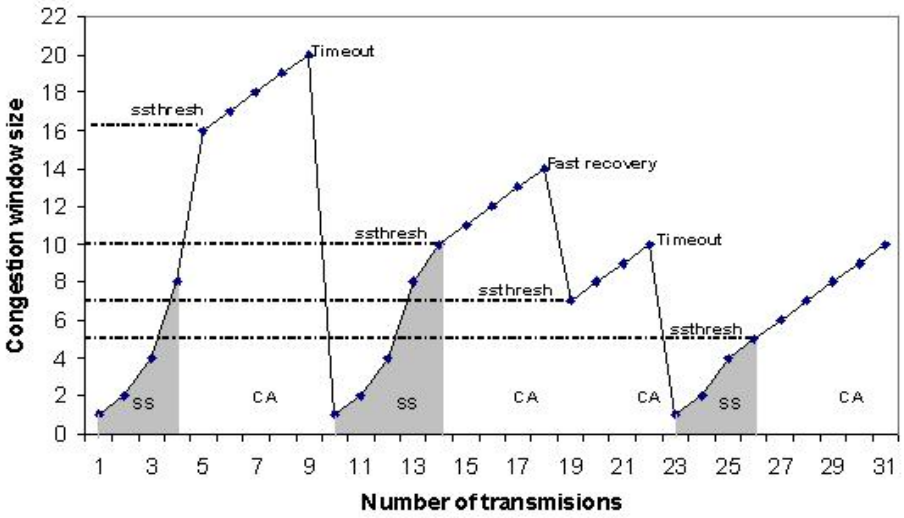

Figure 2.5: AIMD congestion control behavior

$$SendWnd = min(CWN, RcvWnd).$$

Where RcvWnd is the receiver advertised window through the TCP header see figure 2.4. Whenever the sender receives an ACK that confirms the reception of a sent packet, CWN is increased accordingly see figure 2.6. Thus, the CWN is increased exponentially during the SS phase.

Obviously if this increase continues without provisioning, the congestion window could exceed the available capacity and thereby overflow intermediate routers. To overcome this particular problem, the sender also maintains another variable called slow start threshold (ssthresh). This variable determines the transition from Slow Start to Congestion Avoidance. Congestion Avoidance phase is entered if CWN becomes larger than ssthresh. In the congestion avoidance phase, the CWN is increased linearly (1 segment) for every round trip time (RTT). This process continues until congestion occurs and is detected either through timeout or duplicate acknowledgments. Detection through duplicate acknowledgments will force TCP to invoke the Fast retransmit & Fast recovery algorithms. The details of these algorithms are explained in the next section.

Fast retransmit & Fast recovery

Segments can arrive out-of-order at the TCP receiver. When that occurs, the receiving host cannot deliver these segments to the receiving application, and must therefore buffer them until the correct sequenced data arrive. According to RFC2581 [3], the

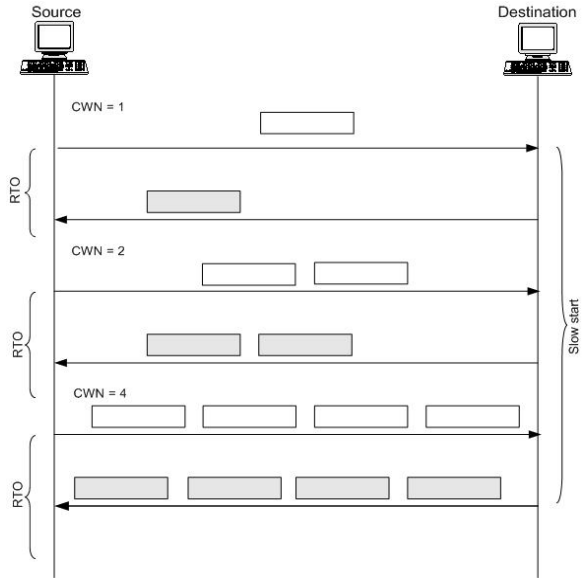

Figure 2.6: Slow start

TCP receiver should send an immediate duplicate ACK to inform the sender about the out-of-order segment received and the sequence number that is expected. This will cause the sender to receive duplicate acknowledgments (DACK) for previously sent data. RFC2581 [3] requires the sender to use the Fast retransmit algorithm upon the reception of 3 DACK. The goal of the Fast retransmit algorithm is to retransmit a lost segment immediately after 3 DACK rather than waiting for the retransmission time out (RTO) which could be relatively long in some situations.

The standard [3] specifies that the sender should use the Fast recovery algorithm instead of Slow Start (see section 2.2.4) if a lost packet is retransmitted due to 3 DACK rather than RTO. Fast recovery reduces the congestion window to halve and not to 1 segment as Slow Start would. The reason for that is the arrival of 3 DACK indicates that the network is capable of delivering some segments regardless of congestion, and therefore does not need to take drastic measurements such as reducing congestion window to 1 segment.

2.3 Congestion control issues in TCP

2.3.1 Interpreting loss as congestion

The AIMD algorithm used by TCP considers the loss of packets as a signal of network congestion. This in turn triggers the congestion control mechanism which leads to a reduced transmission rate. The fundamental assumption, on which the algorithm is designed around, causes performance deterioration for TCP when packets are lost due to other reasons than congestion. In [6] the authors identify 40 distinct error classes in TCP that are not all related to congestion along a path. Their investigation also revealed that 1 packet in 1,100 and 1 packet in 32,000 fails the TCP checksum.

Wireless networks (WLan) and cellular systems (3G) are gaining popularity. These networks suffer from significant losses due to bit errors and handoffs (When a mobile unit moves from one base station to another) [7]. The bit error rate in wireless networks can be in the order of $\approx 10^{-6}$ compared to wired networks which are normally in the order of $\approx 10^{-12}$ [8]. The need for better TCP performance in situations where loss is not due to congestion has long been discussed by the research community and a variety of solutions has been proposed some off which we will look into in section 2.4.

2.3.2 Fast long-distance networks

Fast long-distance networks are networks that operate at capacities in the range of 622Mbps to 10 Gbps. These networks are been deployed at ISP's backbone networks and between research institutions around the globe. At the First International Workshop on "Protocols for FastLong-Distance Networks in 2003", TCP's shortcomings in these environments were discussed.

Sally Floyd pointed out in her paper [9] that a standard TCP with a 1500-byte packets and a 100 ms round-trip time, would need an average congestion window of 83,333 segments and a packet drop rate of at most one congestion event every 5,000,000,000 to achieve a steady state throughput of 10 Gbps. Floyd also points out AIMD's inability to recover quickly from loss triggered event on networks with high Bandwidth Delay Products (BDP). An experiment conducted at California Institute of Technology [10] on the responsiveness of current TCP in various networks confirmed its poor performance on networks with high BDP. Their results show that it took TCP between 15min to 1h 30 min to recover from loss on a 10 Gbit/s link with a RTT of 120ms and a MSS of 8,960 bytes or 1,460 bytes.

Several solutions have been proposed to address these issues. They vary from extending the current TCP implementation, tuning the TCP parameters at the sending host and totally creating a new transport protocol. We will not describe in details these

different approaches, but some of the promising high-speed transport protocols will be mentioned in section 2.4.

2.4 TCP implementation that address the problem

2.4.1 TCP Vegas

TCP Vegas [11] was the first TCP implementation to branch from the traditional congestion control mechanism implemented in the widely used TCP Reno. Instead of relying on packet loss as an indication of network congestion, Vegas deploys a more sophisticated bandwidth estimation scheme to predict potential congestion on the network. It calculates the difference between the expected rate and the actual rate to estimate the available bandwidth.

TCP Vegas aim is to detect congestion that occurs between the source and the destination before packet loss happen and lower its transmission rate linearly (rather than multiplicatively as AIMD does) when loss is detected. The algorithm used by Vegas is as follows:

1. Define a base round trip time (*baseRTT*). This is determined by the minimum of all measured RTT during a connection.

2. Calculate the expected transmission rate (*expectedRate*) as:

   expectedRate = congestion Window / baseRTT

   Where *congestion Window* is the amount of data sent but not yet acknowledged for.

3. Calculate the difference (*diff*) between *expectedRate* and *actualRate*, where *actualRate = congestion Window / RTT* and RTT is the actual round trip time. In other words:

   diff = congestion Window / RTT.

4. Define the thresholds α and β, where α indicates too little data and β indicates too much data. (Currently $\alpha = 1 and \beta = 3$

5. If *diff* < α, then *congestion Window = congestion Window + 1*.

6. If *diff* > β, then *congestion Window = congestion Window - 1*.

7. If α < *diff* > β, then congestion Window is unchanged.

2.4.2 TCP Westwood

One of the new comers in to the arena of congestion control algorithms is TCP Westwood (TCPW). The idea behind TCPW is to perform an end-to-end estimate of the bandwidth available (BWE) along a TCP connection by measuring the rate of returning acknowledgments [12]. The algorithm behaves just like TCP Reno in slow start and congestion avoidance phase, but reacts differently to recover from lost. If timeout triggers loss the algorithm sets the values of ssthresh and CWN as follows[12].

- ssthresh = (BWE * RTTmin) / seg_size. In the event that ssthresh is less than 2, then ssthresh is set to 2.

- cwin = 1

On the other hand if duplicate ACKS are the cause of the loss, the algorithm behaves as follows [12].

- ssthresh = (BWE* RTTmin) / seg_size

- if CWN is larger than ssthresh which is an indication congestion avoidance phase, then set CWN = ssthresh.

In [13], they show TCP Westwood to perform well not only in wired networks but also in many wireless scenarios.

2.4.3 ECN & RED

Explicit Congestion Notification (ECN) and Random Early Detection (RED) are efforts to make the intermediate routers get involved in the managing of congestion control. ECN which is specified in RFC 2481 [14] takes advantage of fields in the IP headers so that router can mark a packet with the Congestion Experienced (CE) bit to inform the sender of the situation. The aim is to make TCP ECN capable so that it would reduce its transmission rate.

Active Queue Management is a scheme for routers to detect congestion before their queues overflows by deploying the RED algorithm. RED which was first presented in [15] basically monitors the average queue size for the output and drops packets based on a probabilistic calculations. It sets some thresholds for when to start dropping packets and when to mark them. End nodes as we explained in the previous section can benefit from this by getting an explicit notification about potential congestion build up along the path.

2.4.4 SACK

Selective Acknowledgment Options (SACK) described in RFC 2018 [16] addresses the problem of poor performance when multiple packets are lost from one window of data. TCP has the option of accumulation acknowledgments which means that rather than acknowledging every received packet, the receiver side of TCP can accumulate several packets and acknowledge them at one go. The drawback with this is that in the event of a packet loss, the sender has to retransmit the lost packet and all of the subsequent packets that were not acknowledged. For that reason SACK was proposes as a standard option to be used by the sender and the receiver.

The receiver side of TCP can take advantage of the SACK options to inform the sender of the missing sequence number. The sender, receiving a SACK option can then send the missing packet only and no other packets need to be retransmitted. Less retransmission mean better throughput and in that sense, SACK improves TCP performance.

2.4.5 Protocols for Fast long-distance networks

Traditional TCP (TCP-Reno/New Reno) with AIMD congestion control algorithm is showing signs of age and poor performance on high speed gigabits networks in [9, 10]. The two most promising algorithms to deal with the issue that are proposed today are HighSpeed TCP and FAST TCP. We explain these new protocols n the coming sections.

HighSpeed TCP

Sally Floyd proposed HighSpeed TCP (HS-TCP) [17, 18] to improve the performance of standard TCP in fast-long delay networks. HS-TCP modifies the congestion control mechanism so that the congestion window parameter is set based on the values in table 2.1: HS-TCP response will depend on the size of congestion window (cwnd) compared

window parameter	**value**
Low_Window	31
High_Window	83000
High_P	0.0000001

Table 2.1: HighSpeed TCP parameters

to the parameter Low_Window, thus:

- cwnd $\leq$ Low_Window, then HS-TCP use the same response function as standard TCP.

- cwnd > Low_Window, then HS-TCP uses the HighSpeed response function.

When cwnd is large (greater than 38 packets), the modification uses table 2.1 to indicate by how much the congestion window should be increased when an ACK is received, and it releases less network bandwidth than 1/2 cwnd on packet loss [19].

FAST TCP

Fast TCP is a descendant of TCP Vegas. Unlike Highspeed TCP, FAST TCP is a delay-based congestion control algorithm which uses queuing delay as a measure of congestion to adjust its window [20]. The protocol consists of four components, each one performing a separate task but collaborating together as one algorithm. The four components are:

- **Estimation**
- **Window Control**
- **Data Control**
- **Burstiness Control**

The **estimation** component keeps track of arriving acknowledgments (ACK) for sent data and calculates the RTT. If the received ACK is not a duplicate one, the **estimation** component calculates the minimum RTT, an exponentially smoothed average RTT and passes this information to the **window control** component. Loss detected either through a timeout or a duplicate ACK causes the **estimation** component to pass the information to the **data control** component.

Decision regarding how many packet to transmit is controlled by the **Window control** component. The algorithm used by FAST to update its window under normal conditions is [20]:

$$w = min\left\{2w, (1-\gamma)w + \gamma\left[\frac{baseRTT}{RTT}w + \alpha\right]\right\}$$

Where RTT is the current average RTT, baseRTT is the minimum RTT noticed so far and γ is a number between 0 and 1. α is parameter used by FAST to measure fairness and the number of packets each flow buffered in the network.

Data control . Data control component determines which packets to transmit next. It chooses from the following categories:

- New packets (The next packet to send is totally new)
- Retransmission of a lost packet

- Packets transmitted but not yet acknowledged.

The component keeps sending new packets when old ones are acknowledged and there is no indication of loss. In the event that loss is detected, the component has the alternative of choosing to send one of the categories mentioned earlier.

Burstiness control determines when to transmit the packets. The burstiness control component smooths out transmission of packets to track the available bandwidth. Networks with large BDP can in periods experience burstiness because of the large amount of data in flight. For example, a single acknowledgment can acknowledge several thousand packets creating a large burst. The operating system at the sender side could be occupied with other matters, thus accumulating packets at the output queue which can lead to transmitting in a large burst when CPU becomes available. bursty data increases the likelihood of massive data loss as router queues might not cope with such burtiness.

Chapter 3

Related work

The first attempt to use a bandwidth estimation technique to improve the congestion control in TCP can be traced back to TCP Vegas. Vegas estimation was based on measuring bandwidth from the number of sent packets see section 2.4.1 for more information on Vegas. The author of [21] proposed using the Packet Pair method explained in section 4.3.2 as a way of initializing slow start threshold (ssthresh) in TCP. Allman and Paxson [22] used a method known as Packet Bunch Mode (PBM) developed by Paxson in 1997. Their goal was to estimate available bandwidth with the PBM technique to set ssthresh correctly, such that the bandwidth available to a given connection could be fully utilized but never exceeded.

Enable (Enabling network-aware applications) [23] is an architecture which aims to provide information about network parameters to a number of hosts. The Enable service consists of three components. The Enable Server which keeps track of network parameters between itself and other hosts, a protocol for clients to communicate with the server and an API that makes querying the Enable Servers easier for the developers. The Network Weather Service (NWS) [24] is a distributed system that periodically monitors and dynamically forecasts the performance various network resources can deliver over a given time interval. The system has among other things sensors for measuring end-to-end TCP/IP performance such as bandwidth and latency.

The work that is closely related to ours is [25], where three different bandwidth estimation tools are evaluated and a method to infer TCP parameters are proposed. A modified version of TCP-SACK that includes the bandwidth estimation tool Initial Gap Increasing (*IGI*) [26] is used to set the ssthresh and congestion window (cwn) in TCP. We use a similar approach in terms of using a bandwidth estimation tool in our case *pathload* see section 4.4.6 for more information. Our work differs from theirs in the sense that we are interested in the long term behavior of a path and possible ways of setting ssthresh/cwn based on those measurements.

Chapter 4

Methodology

This chapter describes the test methodology for this thesis and the actual experiments carried out. It further explains the type of tools used and the test bed at which the experiments were conducted. A taxonomy of measurement methods and different bandwidth estimation techniques are also covered in this section.

Our methodology consisted of performing three types of experiments at different time periods. The goal of the first type of experiment which we will refer to as Type 1 was to establish the existence of user preferences. The aim was to see if users usually connected to certain hosts each day in such way that a long term relationship is established between the user's machine and the preferred host.

The second type (referred to as Type 2), looks at the stability of the path taken by a packet. The fundamental idea behind a packet switched network is the possibility for a packet to take the best route to the destination. Routers along a path decide what constitutes to be the best route with the help of some routing protocol. Two packets originating from the same source, can in theory take different paths. Path asymmetry can also exist between two host. What this means is that the route from:

$$Source \Rightarrow destination \neq Destination \Rightarrow Source$$

It is therefore interesting for us to gain the necessary knowledge from our chosen path as it would play a crucial role in our final and third experiment.

The third type of (referred to as Type 3), measures the available bandwidth between two chosen hosts to see the long time trends of bandwidth fluctuations caused by supply and demand of bandwidth recourses. Our aim with this measurement is to identify patterns which could indicate periods of congestion. Before we delve into detailed description of tools, equipment and setups used in the three experiments we would like to clarify some concepts regarding measurement methods, bandwidth metrics and techniques for estimating bandwidth.

4.1 Measurement methods

Measurement methods can be categorized to two types namely active and passive measurements. Depending on the metric of interest, active, passive or both can be deployed.

4.1.1 Passive measurements

Passive measurements means capturing packets transmitted by applications through a packet sniffing tool. Passive monitoring can be done on-line, but often one is interested in capturing the traffic and storing it for later analysis. Our first experiment is categorized as a passive one. We capture all the packets sent by a number of machines we chose to monitor. More on Type 1 experiment is described in section 4.5.1. Popular passive monitoring tools are tcpdump, ethereal and Cisco NetFlow. We use tcpdump (see section 4.4.2 for more information) in this thesis.

4.1.2 Active measurements

Active measurements inject artificial packets into the network. These are usually small amount of probes sent from a source to a destination to calculate some metric. The probes are meant to imitate traffic generated by the metric of interest so that realistic value can be calculated. The idea behind this method is to deploy a well defined sample to reach conclusions about the overall behavior of the network. In our quest to estimate the available bandwidth, we use pathrate and pathload described in section 4.4.5 and 4.4.6 respectively. Both tools are categorized as been active measurement tools as they send probes into the network to collect their data.

4.2 Bandwidth measurements

Two metrics that are important when referring to bandwidth are capacity and available bandwidth. Capacity is a metric that is associated with a link bandwidth. It is a measure of the maximum throughput the link can provide to an application when there is no competing traffic [27]. Bear in mind that the capacity of a link is a fixed parameter that does not change and does not decrease when the amount of traffic passing through increases see figure 4.1 for an overview of bandwidth metrics. The bandwidth capacity of a path is defined as:

$$C \equiv \min_{i=1...H} C_i$$

Available bandwidth on the other hand is measure of the unused/utilized bandwidth at a particular time see figure 4.1 for details.

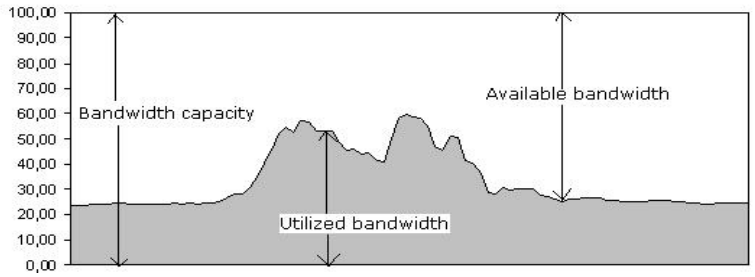

Figure 4.1: Overview of bandwidth metrics

4.3 The methods used to estimate bandwidth

Bandwidth is a commodity that is playing a crucial role in the networking world. Internet Service Providers (ISP) offers its customers a bandwidth metric to fulfill a Service Level Agreement (SLA), certain applications such as adaptive streaming require measurement of available bandwidth and congestion control mechanism need a method for bandwidth estimation (BE). It is no surprise then, that a method to estimate bandwidth has long been the interest of research community. A comprehensive and detailed description of the different bandwidth estimating techniques is covered in [28]. The article focuses on four major techniques (see following sections) used by the majority of BE tools todays.

4.3.1 Variable packet size (VPS)

VPS probing method was first presented in [29] and is considered among the earliest methods of trying to characterize the bandwidth of a link. It behaves much the same way as traceroute see section 4.4.4 does in the sense that the technique deliberately starts with a small Time-To-Live value in the IP header so that the packet is dropped by an intermediate router. The aim of VPS is to exploit the ICMP time-exceeded error massage returned to a sender when a router drops a packet to calculate the Round Trip Time. The RTT to each hop consist of:

- serialization delays: the time it takes to transmit the packet on the link which will depend on the packet size.
- propagation delays: the time it takes for each bit of the packet to traverse the link.
- queuing delays: processing time at the intermediate routers.

VPS sends several equally sized probing packets along a path and expects one of the packets together with its ICMP reply does not experience any queuing delays. Thus

computing the capacity from measured RTT and taking into account the three delays mentioned above.

4.3.2 Packet Pair technique

The Packet Pair technique was first defined in [30]. The idea is based on sending two back-to-back packets from source to a destination with a spacing that is very short, and then calculating the bandwidth capacity at the receiver see figure 4.2. The Packet Pair algorithm [31] is based on the assumption that a link with a capacity C_i and with a packet size L the transmission delay τ_i is:

$$\tau_i = \frac{L}{C_i}.$$

If there is no cross traffic in the path, the packet pair will reach the receiver spaced by the transmission delay in the narrow link such that:

$$\tau_i \equiv \frac{L}{C}.$$

The receiver can then calculate the capacity C from the measured dispersion Δ as:

$$C = \frac{L}{\Delta}.$$

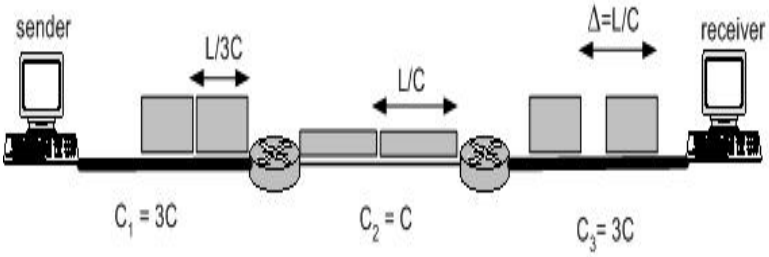

Figure 4.2: Illustration of the packet pair technique

4.3.3 Trains of Packet Pairs(TOPP)

TOPP, first presented by Melander [32] is a an extension to the Packet Pair technique in section 4.3.2. TOPP probing mechanism is based on sending many pairs of equally sized packets with a starting transmission rate o^{min} to a receiver. After the first pairs of packets are sent, the same amount of packets are sent again increasing $o^{min} + \Delta o$. This process continues until a o^{max} is reached and the probing ends.

The receiver timestamps all the received packets and sends it back to the sender. TOPP is based on the assumption that packets that are separated with time ΔS transmitted over a link with service time $\Delta Q_b > \Delta S$, then the packets will be separated by $\Delta R = \Delta Q_b$. Having a packet of size *b* and time separation of ΔR then the available bandwidth can be calculated as:

$$avail = \frac{b}{\Delta R}.$$

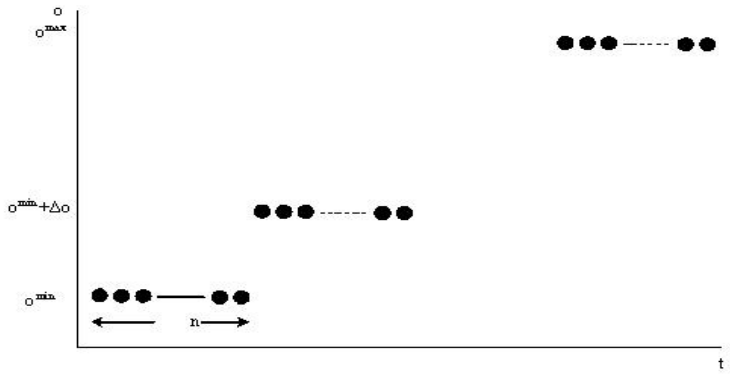

Figure 4.3: Topp

4.3.4 Self-loading Periodic Streams (SLoPS)

SLoPS [33] is based on the idea that one-way delay of a periodic packet stream can indicate the available bandwidth along the path. A source Src sends periodic packet streams to a receiver Rcv, time stamping each sent packet. The receiver on the other hand checks the time stamp of the arriving packets and calculates the one-eay delay. SLoPS has the following metrics:

- **K** is the number of packets a stream consists of.
- **L** is the size of a packets in bits
- **T** is the transmission period of the packet
- **R** is the transmission rate where **R = L/T**
- **D** is delay time of the packet

When the transmission rate **R** of the packet stream is larger than available bandwidth, the one-way delay experiences a temporary increasing trend. The SLoPS algorithm is able to measure an approximate available bandwidth by sending several packet streams with different **R** , this is depicted in figure 4.4.

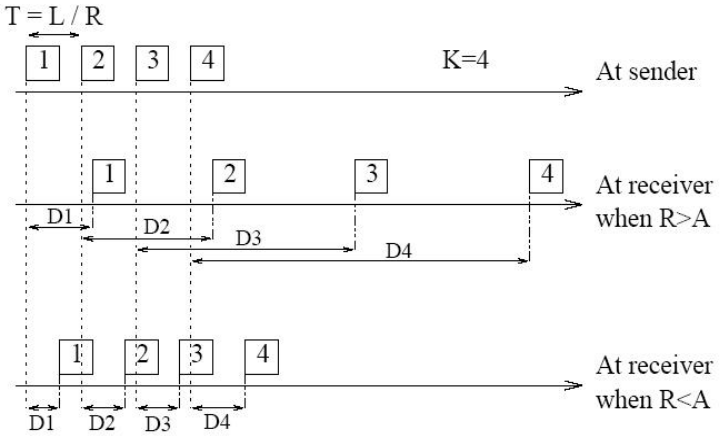

Figure 4.4: Self-loading periodic streams courtesy of [34]

4.4 The equipment & software tools used

4.4.1 Equipment

Our testbed consisted of two Gnu/Linux machines. One located at Oslo University college, department of engineering and the other located at Uninett in Trondheim. Table 4.1 shows the equipment specification.

	Startreck at Oslo	**Scampi1 at Trondheim**
Operating System	Debian GNU/Linux	Debian GNU/Linux
Version	2.6.8-1	2.4.27
Nr. of CPU	1	3
CPU type	Pentium 4 - 1.90GHz	XEON - 2.20GHz
Memory	512 MB	2 GB

Table 4.1: Equipment specification

4.4.2 Tcpdump

Tcpdump [35] which is included in most Unix/Linux distributions is a powerful packet capturing tool that sniffs and gathers network traffic. Written by Van Jacobsen in 80's, the tool is text based and can only be accessed through the command line. It has a filtering capabilities so that one can filter out packets based on protocol, IP address, network address, port number, TCP flags such as SYN, FIN, and ACK. Tcpdump was very handy in our pursuit to identify two communicating host. It allowed us to passively capture traffic from a specific number of hosts to later analyze them off-line.

4.4.3 Tcpslice

Tcpslice is a tool that can break down a large packet trace file captured by tcpdump see section 4.4.2 into smaller files. It also has a built in function to merge several tcpdump files into one. There is time option which one can specify to break the file depending on a start time and an end time. We found the tool to be helpful since most of our traces spanned over several days and needed to be sliced. The tool is available for most Unix/Linux distributions but can also be downloaded from [36]

4.4.4 Traceroute

Traceroute [37] is a tool that is included in most operating systems. Its primary function is to trace the route a packet would take between the source computer and a specified destination on the Internet. It accomplishes this by sending UDP packets with a small Time-To-Live (TTL) value (initialized to one) and then listens for an ICMP "Time exceeded" message generated by intermediate routers. Every time Traceroute receives "Time exceeded" message, it will resend the packet increasing the TTL value by one.

Traceroute determines when the packet has reached the destination host by using a port number that the targeted host normally do not use thus when an ICMP message "port unreachable" is received, it confirms the end of the trace. Three probes are sent at each TTL and the results are displayed at the end of the probe. If a response is not received within 5 seconds, a timeout interval indicated by "*" character is printed for that probe. We this tool to identify the route between the machine at our school campus Oslo and another machine located at Uninett in Trondheim.

4.4.5 Pathrate

Pathrate is an end-to-end capacity estimation tool developed by [38]. It uses packet-pairs and packet-trains method described in 4.3.2. The tool can be run as normal user and does not require any privileged accounts, though access to both the sender and the receiver is required. The tool served our purpose well in estimating the capacity between the two test machines.

4.4.6 Pathload

Pathload [33] is a tool for estimating the available bandwidth of an end-to-end path. It uses the SLoPS method described in section 4.3.4. Pathload consists of two processes, one running at the sender while the other is running at the receiver. The process at the sender sends periodic streams of UDP packets to the receiver at a rate higher than the available bandwidth in the path. It monitors variations in one way delays of probing packets and deduces available bandwidth based the on the behavior of these delays.

4.5 Experimental Setup

4.5.1 Examining network traffic patterns of users

The setup for this experiment is depicted in figure 4.5 and as you will notice the IP addresses are represented with letters. This is due to the fact that data protection laws prohibit us from exposing the real IP addresses or any part of it that could identify the user. The first three small letters represent the network address while the last capital letter stands for the host number.

We begun with selecting 10 hosts that we considered to be eligible for monitoring. Our selection was mainly based on the hosts been active (generating network traffic) on a daily bases. The fulfillment of this criteria was established prior to our actual experiment through a test carried out by a fellow student who used the network traffic probe ntop [39] to visualize network characteristics. To capture traffic from these host we used the popular packet capturing tool tcpdump see section 4.4.2 for more information on the tool. The 10 hosts were connected to a switched network which created a slight problem for us in terms of tcpdump not been able to capture packets from all of the hosts. We solved this by asking our network administrator to divert the network traffic to a Switch Port ANalizer (SPAN).

Since our goal with these experiments was to identify the long term relationship between two hosts, we filtered out all other traffic except UDP and TCP through the use of BPF (Berkeley Packet Filter) facility provided by tcpdump. We also concentrated on traffic to other destinations than our Local Area Network (LAN) as this was more in line with our research. The experiment run for a period of 1 week. We would have

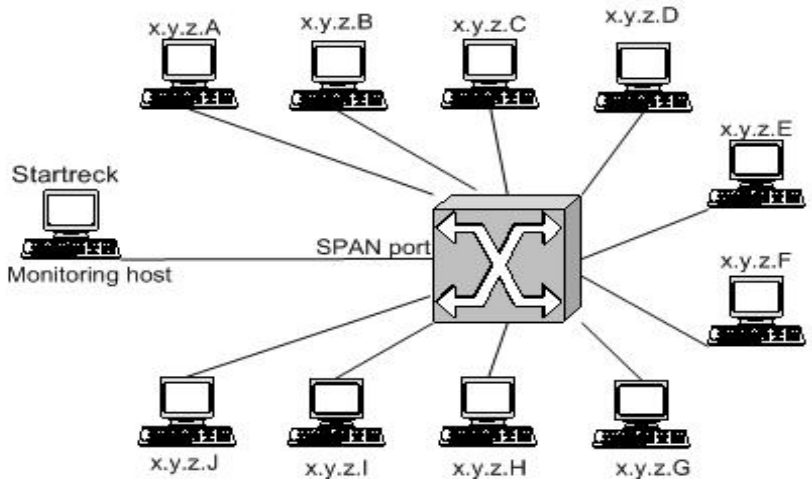

Figure 4.5: Setup 1

liked to run the experiment for a longer period but unfortunately could not because of the SPAN port been tied to other experiments conducted by fellow students. The results obtained from the experiment was written to separate file for each host and later analyzed. The outcome of this experiment will be described in more details in section 5.1.

4.5.2 Is the route taken by the packet stable ?

The primary function of a router is to transport packets across an internetwork and determine the best path for that packet. Depending on the routing protocol used, the best path can be determined either through hop counts or other metrics such us bandwidth and latency. Change in these metrics will lead to the router selecting an alternative route for a packet. There are lots of factors which can influence these metrics such as congestion and link outage. The route taken by a packet might therefore vary if the network conditions change. In this experiment, we are interested in finding out the stability of our selected path since this has a significant role in our next experiment namely estimating the available bandwidth.

Figure 4.6 shows our setup for this experiment where the two hosts involved are Startreck and Scampi1 respectively. Startreck is located at our school campus in Oslo Norway while Scampi1 is at located Uninett's site in Trondheim Norway. There are approximately 5 routers between the two hosts and the link speeds vary between 100 Mbits/s and up to 2.5 Gbits/s. The tool we selected for this experiment is traceroute see section 4.4.4 for more information. We run the experiment for period of 1 week at a 3 minutes interval. This time interval is adequate for our purpose compared to the intervals used by other monitoring sites such as TRIUMF [40].

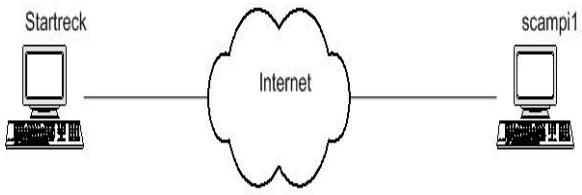

Figure 4.6: Setup 2

4.5.3 The bandwidth estimating process

In this part of the experiments, we were interested in finding out about the bandwidth capacity and available bandwidth of the path between Startreck and scampi1 and for that purpose we used pathrate and pathload. Various tools exist today that can perform the same type of measurements as pathrate and pathload. We chose these two tools

in particular because of a test conducted by [41] where it was confirmed that both of them gave a good estimate of the capacity and available bandwidth.

To estimate the capacity, we run first pathrate every 1/2 hour in 15 times. The capacity of a link/path is a fixed parameter and does not change with the load, therefore we found it unnecessary to conduct more than 15 measurements. Having estimated the capacity, our interest know leaned toward estimating the available bandwidth. For that we run pathload for a period of 1 week first and another week at a later stage, all together 14 days. Pathload was run every 15 min for 24 hours.

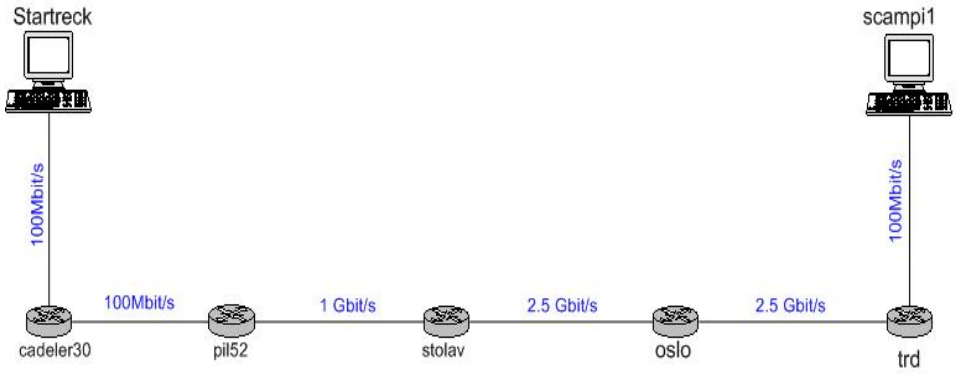

Figure 4.7: Setup 3

Chapter 5

Results analysis

5.1 Results from Type 1 measurements

In table 5.1, we have the results obtained from Type 1 experiment. It clearly shows that 4 of the hosts had contacted at least 1 host every day of the week. Of course we acknowledge hat data collected in one week, cannot represent the whole picture. We cannot assume that these hosts will continue to have contact with those host in the future either. What we can note though is that the results are promising and further investigation based on a longer data collection period is needed.

Source	Average# of unique hosts connected to per day	Unique hosts contacted more than once in a week	Unique hosts contacted every day (1 week)
x.y.z.A	36	5	1
x.y.z.B	40	7	0
x.y.z.C	25	3	0
x.y.z.D	46	6	1
x.y.z.E	36	6	0
x.y.z.F	120	15	3
x.y.z.G	32	4	0
x.y.z.H	30	4	0
x.y.z.I	102	11	1
x.y.z.J	42	7	0

Table 5.1: Results of host monitoring

5.2 Results from Type 2 measurements

The route taken by packets sent from StarTreck $\Longleftrightarrow$ scampi1 is shown in table 5.2 and 5.3. After analyzing the output generated by traceroute during the measuring period which lasted for 1 week, we did not find any route changes. We can also see from the results that there are no path asymmetry. The path from source to destination is the same as the way back. We can then conclude that the path between Startreck and scampi1 is fairly stable and therefore we could proceed with the bandwidth estimation process.

ismail@StarTreck:$ traceroute scampi1.uninett.no
traceroute to scampi1.uninett.no (158.38.0.233), 30 hops max 38 byte packets
1 cadeler30-gw.uninett.no (128.39.89.1) 0.968 ms 0.676 ms 0.615 ms
2 pil52-gw.uninett.no (158.36.84.21) 0.839 ms 0.710 ms 0.741 ms
3 stolav-gw.uninett.no (128.39.0.73) 0.551 ms 0.555 ms 0.566 ms
4 oslo-gw1.uninett.no (128.39.46.249) 0.723 ms 0.691 ms 0.616 ms
5 trd-gw.uninett.no (128.39.46.2) 8.379 ms 8.409 ms 8.311 ms
6 scampi1.uninett.no (158.38.0.233) 8.543 ms 8.340 ms 8.340 ms

Table 5.2: Output from traceroute: StarTreck $\Longrightarrow$ scampi1

ismailah@scampi1:$ traceroute StarTreck.iu.hio.no
traceroute to StarTreck.iu.hio.no (128.39.89.251), 30 hops max, 38 byte packets
1 trd-gw (158.38.0.226) 0.180 ms 0.131 ms 0.090 ms
2 oslo-gw1 (128.39.46.1) 7.823 ms 7.764 ms 7.790 ms
3 stolav-gw (128.39.46.250) 7.890 ms 7.834 ms 7.854 ms
4 pil52-gw (128.39.0.74) 8.117 ms 8.113 ms 8.090 ms
5 cadeler30-gw (158.36.84.22) 8.710 ms 8.596 ms 9.635 ms
6 startreck.iu.hio.no (128.39.89.251) 8.657 ms 8.279 ms 8.879 ms

Table 5.3: Output from traceroute: scampi1 $\Longrightarrow$ StarTreck

5.3 Results from Type 3 measurements

5.3.1 Analyzing the results of pathrate

There are several links along the path between Startreck and scampi1 with different capacity ranging from 100Mbps to 2.5Gbps see figure 4.7. This information is widely available at Uninett's (The ISP that owns the path) home page. The capacity of a path will always be dictated by the link with the lowest capacity and therefore we were interested in seeing how pathrate would estimate the capacity of our chosen path. Results from pathrate states that our path has a capacity of approximately 86 Mbps see table 5.4 below. We know in reality that our path has a lowest capacity of 100 Mbps, so the estimates from pathrate is close and will suffice for us now.

Nr.	Final capacity estimate
1	86 Mbps to 86 Mbps
2	86 Mbps to 87 Mbps
3	84 Mbps to 86 Mbps
4	86 Mbps to 86 Mbps
5	86 Mbps to 87 Mbps
6	86 Mbps to 86 Mbps
7	86 Mbps to 87 Mbps
8	86 Mbps to 86 Mbps
9	85 Mbps to 86 Mbps
10	86 Mbps to 86 Mbps
11	86 Mbps to 87 Mbps
12	86 Mbps to 86 Mbps
13	86 Mbps to 86 Mbps
14	86 Mbps to 87 Mbps
15	86 Mbps to 86 Mbps

Table 5.4: Output from pathrate

5.3.2 Analyzing the results of pathload

Initial observation (Raw data)

To get an overview of what the data looked like, we plotted our observation against time. Our aim with this method was to see if important features of the time series such us trend, seasonality and cycles could be seen. Our initial intuitive deduction from the figures 5.1 and 5.2 is that the available bandwidth fluctuates dramatically at some time of the day. We also noticed that these fluctuations had a cyclic pattern which usually occurred in the middle of the day. Measurements start at 02:40 am and are repeated every 24 hours.

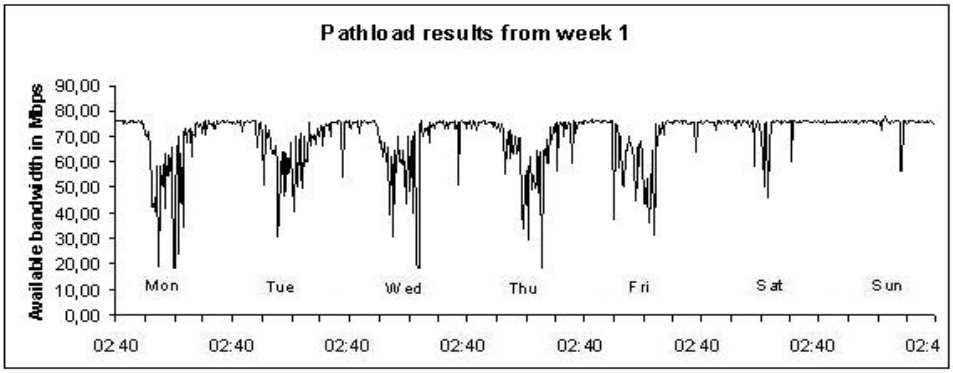

Figure 5.1: Measurement from pathload (first week)

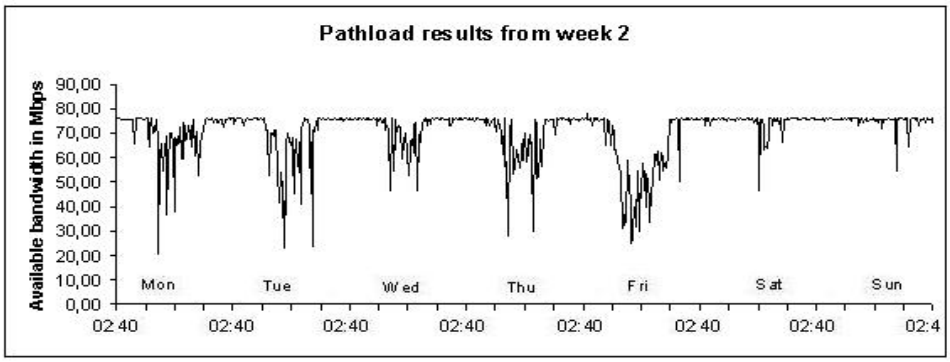

Figure 5.2: Measurement from pathload (second week)

Frequency distribution

In table 5.5, the frequency distribution of the measurements taken by pathload for 2 weeks are shown. We can see from these tables and the figure 5.3 that almost 90% of the time the available bandwidth is in the range og 60 Mbits/s - 80 Mbits/s. This is close to the paths available bandwidth capacity measured by pathrate in section 5.3.1. What this tells us is that the path from Startreck to Scampi1 is only congested 10% of the time.

	Week 1			Week 2	
Range in Mbits/sec	**Occurrence**	**Percentage %**	**Range in Mbits/sec**	**Occurrence**	**Percentage %**
10.1 - 20.0	0	0.0%	10.1 - 20.0	5	0.74%
20.1 - 30.0	8	1.18%	20.1 - 30.0	4	0.59%
30.1 - 40.0	12	1.77%	30.1 - 40.0	10	1.47%
40.1 - 50.0	17	2.50%	40.1 - 50.0	26	3.83%
50.1 - 60.0	43	6.33%	50.1 - 60.0	48	7.07%
60.1 - 70.0	82	12.08%	60.1 - 70.0	93	13.70%
70.1 - 80.0	517	76.14%	70.1 - 80.0	493	72.61%
Total	679		Total	679	

Table 5.5:

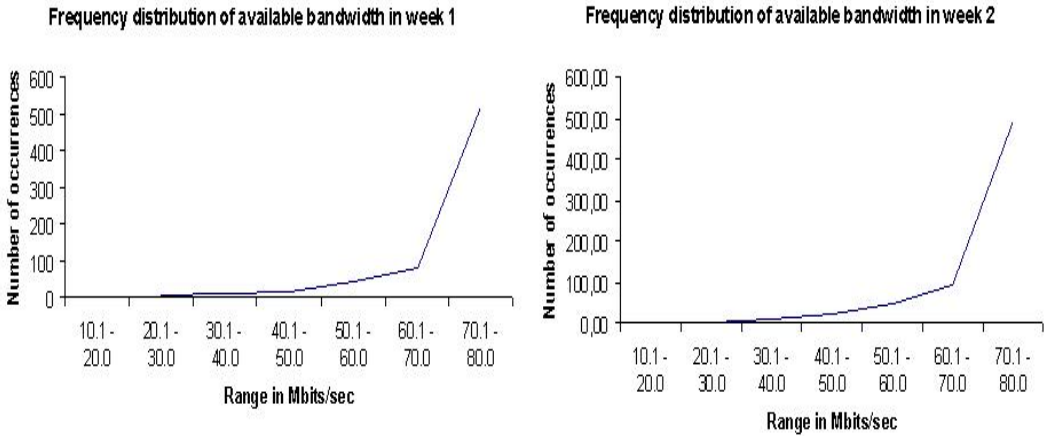

Figure 5.3: Illustration of the frequency distribution

Bandwidth predictor

In this section we will estimate a value to predict the available bandwidth. Since we have measured the available bandwidth at equally spaced interval (every 15 minutes per day), we take the simple approach of calculating the arithmetic mean of all the measurements taken at an interval (for example 02:40 AM on every day of the week).

The Auto Regressive Integrated Moving Average (ARIMA) model developed by Box and Jenkins would have been more appropriate in analyzing our time series. Due to time shortage, we were unable to apply this powerful statistical tool and therefore settled for the arithmetic mean and standard deviation as means of predicting our desired variable. Basically our simple formula for predicting the available bandwidth is given as:

$$\text{available\_bandwidth(t)} = \mu(t) - \sigma(t). \,,$$

Where the arithmetic mean μ and the standard deviation σ of **N** observations are defined as:

$$\mu = \frac{\sum x}{N} \qquad\qquad \sigma = \sqrt{\frac{\sum (x-\mu)^2}{N}}.$$

What we do is take the arithmetic mean and subtract the standard deviation from it so that we have a fairly good estimate of the available bandwidth at time *t*.

Time	μ	σ	Predicted bandwidth = $\mu - \sigma$
02:40	75.55	1.13	$75.55 - 1.13$
02:55	73.37	1.11	$73.37 - 1.11$
03:10	75.62	0.47	$75.62 - 0.47$
03:25	75.72	1.28	$75.72 - 1.28$
03:40	75.89	0.46	$75.89 - 0.46$
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
02:25	75.75	0.50	$75.40 - 0.50$

Table 5.6: predicted bandwidth

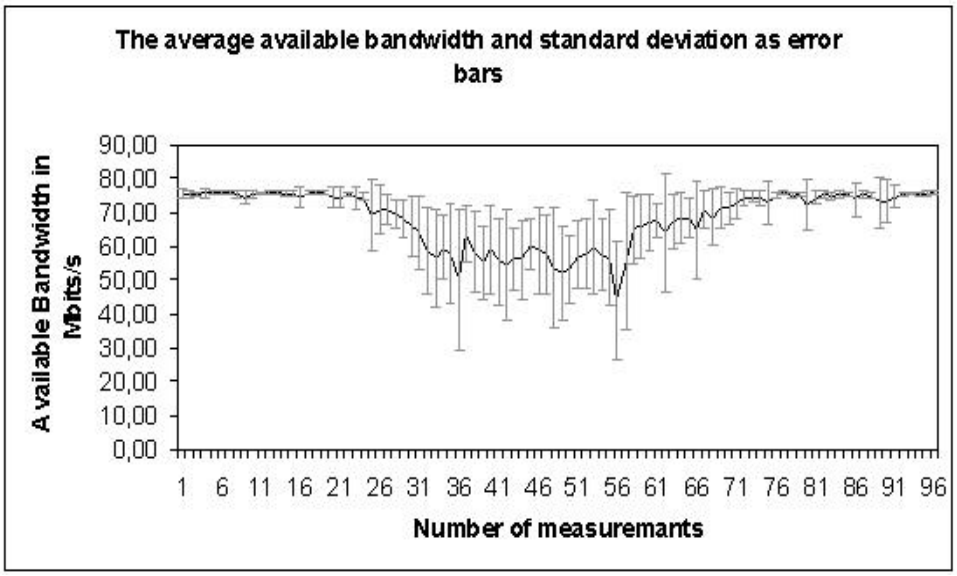

Figure 5.4: average available bandwidth and standard deviation as error bars

Chapter 6

Proposals

This chapter explores how the results obtained in the previous chapters can be harnessed and utilized. Finding out about a path characteristics is easy. One can use a tool like the ones deployed in this thesis and get pretty good estimates of the desired metrics. The harder part is to use the results in a way that is beneficial. The natural question to ask is where can our results be useful? Several candidates are listed below:

- Daily backups which involves backing up a large quantity of data between two machines located at separate locations.
- A large corporate that has offices at different locations that exchange huge amount of data periodically.
- Collaborating research institutions that need large bulk data transfer.
- Government agencies
- Educational institutions

The list can go on. The point is that as long as two machines have a constant need to exchange data periodically, knowledge about the behavior of the path between them is essential.

TCP is the primary component that decides how much traffic a sender can inject into the network, it is then a natural to investigate how our findings can help TCP. Results obtained from the bandwidth estimation experiments showed that 80%- 90% of the time the path between Startreck and scampi1 was not utilized.

6.1 Proposed solution

One of the main objectives of this thesis was to find out ways to estimate available bandwidth and explore its helpfulness to TCP. The challenge was how the information gathered by our bandwidth estimation tool could be passed to TCP. Changing the TCP code and tuning the operating systems kernel were solutions that crossed our minds. The possibility of that approach been beyond the scope of this thesis was also looked into. Fortunately we did not need to reinvent the wheel and found much what we needed has already been provided by others. In the next sections we present three such solution when combined together will be suitable and appropriate for our purpose.

6.1.1 The Web100 kernel

The Web100 project [42] was created to produce a complete host-software environment that would run common TCP applications at close to the available bandwidth. The aim of Web100 was to develop software that interacts with the operating system (As of date, only Linux is supported) to automatically optimize performance for all TCP transfers by implementing a set of instruments in the TCP/IP stack. The TCP Kernel Instrument Set (TCP-KIS) is the the document which defines these instruments. TCP-KIS allows a user to view many of the variables that make up a TCP connection such as current congestion window size, ssthresh and the number of retransmits and tune them accordingly. The Web100 software is composed of two components:

- A patch to the Linux kernel, which is responsible for collecting and exposing the instruments.
- A shared library with a set of utilities which allows the easy reading and manipulation of these instruments

6.1.2 Network Tool Analysis Framework (NTAF)

NTAF [43] developed by The Data Intensive Distributed Computing Research Group (DIDC) at Lawrence Berkeley National Laboratory is a framework for running network test tools and storing the results in a relational database for later retrieval. The basic function performed by the NTAF is to run tools at regular intervals and send their results to a central archive system for later analysis or for use by other programs such as WAD see next section 6.1.3 for description. Tools such as pathload, pathrate and ping can be configured to run periodically.

NTAF original goal was to evaluate various bandwidth and capacity estimation tools, but was later extended to testing new network protocols. When NTAF is used with Web100 modified Linux kernel, TCP parameters such as congestion window, number of congestion events and slow-start threshold can be monitored and recorded.

6.1.3 Work Around Daemon (WAD)

The Work Around Daemon (WAD) developed by Dunigan [44] uses an extended version of the Web100 modified kernel. It allows end users to tune certain TCP parameters. TCP's Additive Increase and Multiplicative Decrease algorithm can also be tuned through this daemon. WAD uses a configuration file that specifies what flows (source, source port, destination, destination port) are eligible for tuning. The configuration file also indicates if static values found in a table are to be used or if dynamic tuning from the NTAF is needed.

When a new connection event is triggered by the kernel, the WAD daemon checks the configuration file to see if the connections flow needs to be tuned. WAD provides tuning for the send and receive buffer sizes and AIMD values. These values are retrieved from the NTAF database.

6.2 Limitations

6.2.1 Operating system (OS) related

The solutions presented so far in this chapter are operating system dependent. The need to tune and modify the kernel is an essential part of the work around mentioned above. Most of the proposals apply to the Linux OS, so Microsft Windows is out of the question. We have not looked into whether such a solution exists for Microsofts OS and therefore would have to be dealt with in future work.

6.2.2 Access to end-2-end hosts

Some of the tools deployed in this thesis such as pathrate and pathload require access to both ends of the connection. The NTAF framework described in this chapter would also require access to end nodes. This limits the scale at where the solution can be applied to. Nevertheless, the list of candidates we presented at the beginning of the chapter is suitable environment. In such environments, both end of the connection are usually owned by a single entity or organization.

Chapter 7

Conclusions and Discussion

TCP, a robust and reliable transport protocol has been in use for the last 2 decades. The protocol was not originally designed for the sort of environment its operating on today. When TCP was first created, the network was operating at 9.6 Kbps, while today we have 10 Gbps and Wireless networks. Over the years, researchers have solved TCP shortcomings in an ad hoc manner. This has created many different flavors of TCP (Tahoe, Reno, New Reno, Vegas, SACK, Westwood, FAST and HighSpeed) some of which we have described in this thesis. Having so many varieties of TCP implementation only substantiates the problems magnitude and the need to further research on TCP to find an optimum solution.

In this thesis we highlighted common features of TCP, outlined some issues with the congestion control mechanism and conducted experiments to measure user preference, path stability and available bandwidth estimation. The results we obtained from the bandwidth estimation tools are promising in terms of understanding path characteristics. Our findings show that the path chosen for the experiment was congestion free 80% - 90% of the time.

A transport protocol such as TCP, would greatly benefit from this findings if adjusted properly so that it fully utilizes the paths capacity. Therefore, in chapter 6 we presented some proposals that could help the AIMD algorithm in TCP so that it has knowledge of path characteristics collected through bandwidth estimation tools. The proposals do not change the TCP standard in any way, but merely presents an alternative implementation.

7.1 Future work

The results obtained in these experiments appear to be promising in terms of applying it to congestion control. It should be noted though that the network we studied consisted of only one path and further study on other paths is necessary. we would also like to verify our proposal either through simulation or an actual implementation.

The research community is heavily involved in developing bandwidth estimation tools and several of these tools exist today. In our experiments, measurement tools played a crucial role and one area we would like to investigate more in the future is the accuracy of the bandwidth estimation tools deployed in this thesis. Due to time constrains we were unable to compare them with other similar tools.

Deploying more statistics on our finding is also needed. We have looked into The Box-Jenkins ARIMA model and found it suitable for analyzing the time series presented in this thesis and therefore would like to investigate its application.

Appendix A

Time	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Time	Mon	Tue	Wed	Thu	Fri	Sat	Sun
02:40	76,00	76,30	76,60	75,40	75,60	75,60	75,40	08:40	75,80	75,20	75,40	74,10	70,40	75,40	75,40
	76,50	73,80	75,40	75,40	75,80	75,90	75,60		76,50	76,60	76,60	70,50	68,20	75,30	76,60
	75,80	75,40	75,40	75,20	75,40	75,40	75,40		70,40	71,80	74,40	75,20	63,40	75,40	75,40
	75,40	76,30	75,60	76,50	78,00	75,40	75,40		64,50	71,00	75,60	72,60	63,60	75,40	76,50
03:40	75,60	76,00	76,50	75,60	75,40	76,30	75,20	09:40	73,00	71,20	72,80	69,80	59,00	75,40	75,00
	75,60	76,70	75,40	76,30	75,60	72,80	75,40		76,50	70,30	76,60	61,20	56,80	75,00	76,50
	75,40	76,60	76,60	74,80	75,90	76,40	76,30		69,50	52,40	71,20	76,30	53,40	76,20	76,20
	75,40	75,40	75,40	75,40	72,60	76,50	75,00		71,00	70,00	72,80	57,20	31,24	75,40	76,10
04:40	75,40	75,40	76,60	76,00	75,80	72,90	76,40	10:40	75,60	69,20	71,50	56,60	31,61	76,60	76,50
	75,60	73,10	76,40	76,60	75,40	75,40	75,40		67,20	71,30	46,60	58,40	41,40	76,60	76,60
	75,40	75,40	75,40	76,50	76,50	75,40	76,60		69,10	66,10	76,50	28,22	32,00	75,20	72,80
	75,40	75,80	76,60	75,40	75,80	75,40	76,50		20,83	67,40	54,60	71,20	33,50	75,40	75,40
05:40	75,40	75,40	75,40	75,40	76,50	76,60	75,40	11:40	60,70	71,40	70,50	75,20	58,90	75,60	74,20
	75,40	75,40	76,50	75,40	75,00	75,40	75,40		65,60	54,80	73,00	54,00	47,30	75,60	76,10
	75,40	75,40	75,40	75,40	75,40	75,40	75,40		54,80	41,80	75,40	60,90	43,80	76,60	74,60
	65,50	75,60	75,40	73,60	75,40	76,10	75,40		65,60	54,20	70,90	53,40	24,75	76,00	75,40
06:40	75,80	75,40	76,60	75,40	76,50	75,20	75,40	12:40	63,30	47,20	72,80	55,60	26,24	75,60	75,90
	76,60	76,60	75,40	76,00	76,50	75,80	75,40		68,80	22,82	72,50	63,10	30,87	75,40	74,80
	76,50	76,50	75,40	75,40	76,50	75,40	75,40		36,80	62,10	59,40	60,80	40,00	75,20	74,70
	76,50	76,50	76,50	73,60	64,80	76,50	75,60		56,80	36,80	66,20	55,70	31,61	76,40	75,40
07:40	76,30	75,40	75,40	74,90	75,20	75,40	75,60	13:40	70,40	65,10	67,40	60,80	54,40	74,20	75,40
	76,10	75,40	76,30	75,40	75,00	75,40	76,60		70,40	69,90	66,70	65,50	29,48	75,60	75,20
	75,40	75,40	75,80	76,30	64,80	74,70	75,40		66,80	67,10	69,60	58,80	45,50	73,90	76,60
	75,40	75,90	72,80	74,40	73,40	76,50	76,30	14:25	37,80	64,90	62,50	64,80	43,60	75,80	75,00

Table A.1: Pathload available bandwidth measurements in week 1, from 02:40 am until 14:25 pm

Time	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Time	Mon	Tue	Wed	Thu	Fri	Sat	Sun
14:40	68,10	68,20	52,30	70,20	45,30	72,80	75,40	20:40	70,80	75,60	76,60	75,40	75,60	75,40	75,40
	64,80	60,40	62,40	58,50	57,70	46,50	75,40		75,60	75,40	75,40	69,20	74,40	75,40	76,50
	67,40	45,10	67,30	67,60	52,80	75,40	74,60		76,50	76,00	76,50	76,10	75,40	75,40	76,50
	65,50	59,20	67,60	60,80	39,70	75,40	75,20		75,40	75,60	75,40	75,40	75,60	76,20	75,40
15:40	72,30	67,60	61,80	69,20	52,90	72,60	75,40	21:40	75,40	75,40	76,60	75,40	76,50	75,80	64,60
	59,40	56,60	72,60	63,80	50,40	63,20	75,40		75,40	75,40	75,40	76,30	75,40	75,80	75,40
	74,40	66,40	74,60	60,80	41,60	64,60	71,90		75,60	76,60	75,40	76,20	75,60	76,30	76,60
	59,40	41,00	46,20	29,48	33,90	63,80	75,60		75,80	75,40	75,40	75,40	49,80	76,10	76,20
16:40	73,40	72,10	69,20	75,60	45,60	69,10	75,40	22:40	74,40	75,40	75,40	76,10	75,40	75,40	76,00
	73,50	73,60	64,60	73,90	50,00	76,00	75,40		75,40	75,40	76,00	75,40	75,80	76,40	75,40
	67,60	75,50	67,20	51,30	61,40	76,10	75,40		75,40	75,40	73,00	75,20	75,40	75,40	76,50
	73,20	76,40	74,60	52,20	56,90	75,40	76,50		74,60	75,80	72,80	75,40	75,40	75,40	76,20
17:40	69,60	75,60	73,70	63,70	59,80	73,00	75,40	23:40	74,50	76,00	74,80	76,30	76,50	76,30	72,80
	64,30	75,60	75,40	67,70	62,20	76,50	75,40		75,40	75,40	76,50	74,00	75,40	75,80	75,40
	71,40	72,10	75,80	56,20	51,20	75,00	75,50		75,60	75,40	75,40	75,40	75,60	75,80	75,40
	75,60	55,80	73,60	70,60	54,60	75,80	75,60		74,50	76,60	76,00	75,40	75,80	75,40	75,40
18:40	60,90	66,20	75,40	75,20	61,80	75,20	76,10	00:40	72,30	75,40	75,40	76,00	75,60	75,60	75,20
	67,20	23,22	75,40	74,80	60,60	74,00	54,60		75,40	76,30	76,60	75,40	76,00	75,40	76,30
	70,60	71,40	73,90	75,40	54,80	76,20	75,40		75,40	75,00	75,40	75,60	75,40	75,40	74,60
	52,60	70,50	74,80	75,10	58,00	75,40	76,50		75,40	75,60	76,70	75,40	75,60	76,50	76,40
19:40	67,30	75,00	75,80	75,40	55,90	66,00	75,40	01:40	76,50	75,40	75,60	74,90	75,40	76,20	75,40
	64,80	71,40	75,20	75,20	60,70	75,60	76,30		76,10	76,60	75,40	75,40	75,40	76,50	75,20
	68,80	76,60	76,40	75,40	61,90	75,40	76,50		75,40	75,40	76,30	75,40	76,50	75,60	76,50
	71,60	75,80	76,50	75,40	75,40	76,50	75,40	02:25	75,80	75,40	76,40	76,40	75,40	75,60	73,90

Table A.2: Pathload available bandwidth measurements in week 1,from 14:40 pm until 02:25 am

Time	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Time	Mon	Tue	Wed	Thu	Fri	Sat	Sun
02:40	76,60	75,00	74,60	72,80	76,60	76,60	76,10	08:40	71,40	69,80	67,80	73,70	37,90	75,40	76,20
	76,50	75,40	73,00	75,40	76,60	75,20	75,40		69,80	51,40	68,40	75,20	75,00	76,60	75,60
	76,50	75,20	75,40	76,50	75,40	75,40	75,40		70,40	74,60	62,10	72,00	74,40	75,20	76,30
	76,50	75,40	75,40	72,70	75,40	76,30	76,50		72,10	68,00	63,40	74,60	69,30	73,10	75,40
03:40	76,50	76,60	75,40	75,40	75,90	76,40	75,10	09:40	65,00	70,70	67,80	75,00	57,80	75,40	75,40
	76,30	75,60	75,60	76,50	76,90	75,40	75,40		48,50	72,50	60,20	71,80	66,90	76,50	76,30
	76,30	76,30	76,60	75,40	75,40	75,40	76,30		42,30	72,20	61,60	73,20	69,80	75,40	75,40
	75,20	75,80	76,50	75,40	75,40	75,40	75,60		44,00	67,20	65,60	55,40	51,60	76,00	76,60
04:40	76,10	71,80	71,20	72,60	75,40	75,40	75,40	10:40	40,90	63,40	39,40	66,10	50,40	74,40	75,20
	75,20	75,40	76,60	75,40	76,60	75,40	75,40		52,00	63,40	61,90	71,90	63,40	76,60	75,40
	75,40	75,60	75,60	75,40	76,10	75,60	75,60		58,60	63,70	56,40	61,40	66,20	76,20	76,50
	75,60	76,50	76,30	75,40	76,50	72,60	75,40		19,01	66,00	30,87	72,60	64,80	75,60	76,10
05:40	76,60	76,30	76,60	76,30	75,40	76,60	75,40	11:40	45,90	58,80	56,10	70,10	69,50	75,40	76,60
	76,60	75,40	75,40	76,50	75,40	76,30	75,40		58,70	30,87	64,40	64,40	70,20	75,40	75,60
	76,50	75,40	75,20	76,30	76,50	76,30	76,50		51,40	38,20	55,70	63,20	66,20	76,60	76,50
	75,60	75,40	75,40	74,80	75,40	75,90	76,30		58,80	56,40	70,40	70,60	65,90	74,20	75,40
06:40	75,40	76,30	75,40	75,20	76,50	74,80	75,60	12:40	41,70	63,90	60,60	63,60	61,10	75,20	76,00
	75,40	76,50	75,40	75,40	76,50	76,00	76,60		63,00	46,80	61,20	69,50	44,80	76,10	76,50
	75,80	76,50	76,50	75,40	75,60	75,60	75,40		54,60	64,00	59,80	65,60	59,80	67,60	76,60
	76,60	75,60	75,40	75,40	75,40	76,30	75,40		62,40	62,90	59,00	63,60	65,80	58,30	76,60
07:40	75,40	65,90	76,30	75,20	75,40	76,30	75,40	13:40	55,00	48,40	64,40	69,40	49,40	72,40	76,50
	75,90	75,20	75,00	75,40	75,40	75,70	75,40		57,80	61,40	56,40	41,00	65,60	74,00	75,40
	75,00	73,20	74,80	74,20	76,50	74,80	76,40		65,50	55,70	44,00	34,20	68,40	76,40	75,60
	74,80	75,40	72,80	75,50	76,60	76,30	75,40	14:25	10,29	67,70	69,60	53,00	62,90	75,20	75,20

Table A.3: Pathload available bandwidth measurements in week 2, from 02:40 am until 14:25 pm

Time	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Time	Mon	Tue	Wed	Thu	Fri	Sat	Sun
14:40	21,16	47,40	48,60	57,00	44,00	69,40	76,50	20:40	75,40	74,60	75,40	72,00	75,40	75,20	75,40
	46,00	52,00	56,80	29,48	43,40	63,20	75,60		71,00	73,80	75,80	75,40	76,30	76,50	76,60
	63,00	40,90	68,40	55,10	46,00	50,50	72,60		76,10	66,00	75,40	56,40	75,40	59,80	76,50
	70,00	61,60	39,80	61,40	53,30	64,60	76,10		73,40	72,80	75,60	73,90	76,50	75,80	76,40
15:40	24,52	70,40	68,80	62,10	48,20	75,20	75,40	21:40	75,80	75,20	75,80	75,40	75,80	76,60	75,80
	63,00	50,50	71,80	49,30	36,40	75,40	76,20		75,00	74,60	72,80	74,60	75,40	75,40	76,10
	53,00	66,20	28,22	54,80	47,10	45,80	78,00		74,80	74,90	75,20	75,70	75,80	75,40	75,40
	34,60	71,60	10,33	49,50	64,80	66,00	76,50		74,80	74,60	75,40	69,80	75,60	75,40	75,40
16:40	71,00	55,40	12,20	51,20	31,61	75,50	75,40	22:40	72,80	69,80	75,20	75,20	75,00	76,50	76,50
	73,40	49,80	74,40	64,70	52,50	76,60	75,60		75,40	75,40	72,80	75,60	76,40	75,40	75,40
	67,70	70,20	76,20	48,20	75,20	75,60	75,20		72,80	75,40	75,40	73,60	76,60	75,40	76,30
	71,40	58,60	72,80	60,40	74,20	75,40	75,40		76,10	75,40	75,60	75,40	76,60	76,30	75,40
17:40	71,90	61,00	70,20	64,00	66,80	75,40	76,50	23:40	75,20	75,40	75,40	74,80	75,20	76,30	76,30
	73,60	60,00	73,30	14,03	74,60	76,60	75,60		75,40	76,60	76,50	59,40	75,00	76,60	75,40
	62,20	75,60	72,40	64,50	71,60	76,40	76,60		75,40	75,20	72,80	74,80	75,40	76,60	75,40
	72,00	66,80	74,20	65,20	75,40	75,40	76,50		76,50	76,60	72,80	76,60	74,80	75,60	75,80
18:40	71,00	70,20	73,70	59,40	71,00	75,00	76,20	00:40	75,40	75,40	75,60	75,40	75,40	75,40	76,50
	70,20	69,50	67,60	65,00	72,50	76,10	75,40		76,10	75,40	51,00	71,60	75,60	75,40	75,40
	74,60	69,20	75,60	69,00	72,90	76,30	56,80		75,60	54,20	75,20	74,20	75,40	75,60	75,40
	75,40	70,80	73,90	59,60	75,40	75,40	67,40		76,00	75,40	76,10	75,40	64,20	75,40	75,90
19:40	73,90	67,20	75,00	74,00	76,00	76,60	57,00	01:40	75,40	75,80	75,40	75,40	75,60	76,30	75,40
	75,20	73,30	72,80	68,60	75,60	75,40	76,20		75,60	75,60	75,60	75,20	75,40	75,60	75,00
	73,30	70,20	75,40	72,50	75,60	75,40	76,10		75,40	75,40	74,90	75,40	75,40	76,50	75,40
	71,60	69,60	75,60	75,00	75,20	76,50	75,20	02:25	75,40	76,40	75,40	75,00	75,40	75,90	75,60

Table A.4: Pathload available bandwidth measurements in week 2, from 14:40 pm until 02:25 am

Appendix B

Abbreviations

AIMD	-	Additive Increase Multiple Decrease
API	-	Application Programming Interface
ARIMA	-	Auto Regressive Integrated Moving Average
ARPANET	-	Advanced Research Project Agency Network
BDP	-	Bandwidth Delay Products
ECN	-	Explicit Congestion Notification
FDDI	-	Fiber Distributed Data Interface
FTP	-	File Transfer Protocol
HTTP	-	Hypertext Transfer Protocol
ICMP	-	Internet Control Message Protocol
IP	-	Internet Protocol
ISO	-	International Standards Organization
ISP	-	Internet Service Provider
LAN	-	Local Area Network
MSS	-	Maximum Segment Size
NTAF	-	Network Tool Analysis Framework
NWS	-	Network Weather Service
OSI	-	Open Systems Interconnection
PPP	-	Point-to-Point Protocol
RED	-	Random Early Detection
RFC	-	Request For Comment
RTO	-	Retransmission Time Out
RTT	-	Round Trip Time
SACK	-	Selective Acknowledgment
SLIP	-	Serial Line Interface Protocol
SLoPS	-	Self-loading Periodic Streams
SMTP	-	Simple Mail Transfer Protocol
SPAN	-	Switch Port ANalyzer
TCP	-	Transmission Control Protocol
TOPP	-	Trains of Packet Pairs
TTL	-	Time To Live
UDP	-	User Datagram Protocol
VPS	-	Variable packet size
WAD	-	Work Around Daemon

Bibliography

[1] Jacobsen Van. Congestion avoidance and control. *ACM Computer Communication Review*, vol. 18:pp. 314 – 329, Aug 1988.

[2] IETF. http://www.ietf.org/rfc/rfc0793.txt
Accessed betwenn Feb - May 2005.

[3] IETF. http://www.ietf.org/rfc/rfc2581.txt.

[4] Burgess Mark. Computer immunology. In *Proceedings of the Twelfth Systems Administration Conference (LISA 98)*, December 1998.

[5] Burgess Mark; Haugerud Haarek; Straumsnes Sigmund and Reitan Trond. Measuring system normality. *ACM Transactions on Computer Systems (TOCS)*, vol. 20:pp. 125 – 160, May 2002.

[6] Stone Jonathan and Partridge Craig. When the crc and tcp checksum disagree. *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols*, vol. 30, Aug 2000.

[7] Choon Chan Mun and Ramjee Ramachandran. Tcp/ip performance over 3g wireless links with rate and delay variation. *Proceedings of the 8th annual international conference on Mobile computing and networking*, Sep 2002.

[8] Cat'ceres Ramot'n and Liviu Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATION*, vol. 13, June 1995.

[9] IETF. http://www.ietf.org/rfc/rfc3649.txt.

[10] Jean-Philippe M. and Sylvain R.
http://netlab.caltech.edu/FAST/publications/caltech-tr-68-2398.pdf
Accessed May 2005.

[11] Brakmo Lawrence S.; OŠMalley Sean W. and Peterson Larry L. Tcp vegas: New techniques for congestion detection and avoidance. In *Proceedings of the conference on Communications architectures, protocols and applications*, volume vol. 24, May 1994.

[12] Ren Wang; Valla M; Sanadidi M and Gerla M. Using adaptive rate estimation to provide enhanced and robust transport over heterogeneous networks. In *Proceedings of the 10 th IEEE International Conference on Network Protocols (ICNP 02)*, November 2002.

[13] Mascolo S.; Casetti C.; Gerla M.; Sanadidi M. Y. and Wang R. Tcp westwood: Bandwidth estimation for enhanced transport over wireless links. *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages pp. 287 – 297, 2001.

[14] IETF. http://www.faqs.org/rfcs/rfc2481.html.

[15] S. Floyd and V Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, pages pp. 397 – 413, Aug 1993.

[16] IETF. http://www.ietf.org/rfc/rfc2018.txt.

[17] IETF. http://www.faqs.org/rfcs/rfc3649.html.

[18] IETF. http://www.faqs.org/rfcs/rfc3742.html.

[19] Evandro de Souza and Deb Agarwal. A highspeed tcp study: Characteristics and deployment issues. http://dsd.lbl.gov/ evandro/hstcp/hstcp-lbnl-53215.pdf Accessed May 2005.

[20] S.H.; Bunn J.; Choe H.D.; Doylle J.C.; Newman H.; Ravot S.; Singh S.; Paganini F.; Buhrmaster G.; Cottrell L.; Martin O. Cheng Jin; Wei, D.; Low and Wu chun Feng;. Fast tcp: from theory to experiments. *IEEE Network*, vol. 19:pp. 4 – 11, Jan - Feb 2005.

[21] Hoe Janey C. Improving the start-up behavior of a congestion control scheme for tcp. *Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages pp. 270 – 280, 1996.

[22] Vern Paxson Mark Allman. On estimating end-to-end network path properties. *ACM SIGCOMM Computer Communication Review*, vol. 1.

[23] Tierney B.L.; Gunter D.; Lee J.; Stoufer M. and Evans J.B. Enabling network-aware applications. *Proceedings. 10th IEEE International Symposium on High Performance Distributed Computing*, pages pp. 281 – 288, August 2001.

[24] Wolski Rich; Spring Neil and Peterson Chris. Implementing a performance forecasting system for metacomputing: the network weather service. In *Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)*, August 1997.

[25] Easwaran Yegyalakshmi and Labrador Miguel A. Evaluation and application of available bandwidth estimation techniques to improve tcp performance. *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCNŠ04)*, pages pp. 268 – 275, Nov 2004.

[26] Hu Ningning and Steenkiste Peter. http://gs274.sp.cs.cmu.edu/www/igi/
Accessed between Feb - May 2005.

[27] Jain Manish and Dovrolis Constantinos. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput. *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols*, vol. 32, Aug 2002.

[28] C.; Murray M. Prasad, R.; Dovrolis and K. Claffy. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE Network*, Nov-Dec 2003.

[29] Jacobson Van. Pathchar: A tool to infer characteristics of internet paths. ftp://ftp.ee.lbl.gov/pathchar/
Accessed May 2005.

[30] Keshav Srinivasan. A control-theoretic approach to flow control. In *Proceedings of the conference on Communications architecture & protocols*, volume vol. 21, August 1991.

[31] Dovrolis C.; Ramanathan P. and Moore D. What do packet dispersion techniques measure?
http://www.caida.org/outreach/papers/2001/consti/consti.pdf
Accessed May 2005.

[32] M. Melander, B.; Bjorkman and P. Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. *IEEE Global Telecommunications Conference, 2000. GLOBECOM '00.*, vol. 1:pp. 415 – 420, Nov - Dec 2000.

[33] Jain M. and Dovrolis C.
http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/pathload.html
Accessed betwenn Feb - May 2005.

[34] Jain M. http://www.cc.gatech.edu/j̃ain/pub/talk/ntg_fall.pdf
Accessed between Feb - May 2005.

[35] Jacobson Van; Leres Craig and McCanne Steven. http://www.tcpdump.org.

[36] Paxson Vern. ftp://ftp.ee.lbl.gov/tcpslice.tar.Z.

[37] Jacobson Van. ftp://ftp.ee.lbl.gov/traceroute.tar.gz.

[38] Jain M. and Dovrolis C.
http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/pathrate.html
Accessed betwenn Feb - May 2005.

[39] Deri Luca. http://www.ntop.org/ntop.html
Accessed between Feb - May 2005.

[40] TRIUMF Network Monitoring. http://sitka.triumf.ca
Accessed between Feb - May 2005.

[41] Shriram A; Murray M.; Hyun Y.; Brownlee N.; Broido A; Fomenkov M and Claffy K. Comparison of public end-to-end bandwidth estimation tools on high-speed links.
http://www.caida.org/outreach/papers/2005/pam-bwest/pam-bwest.pdf
Accessed April 2005.

[42] Web100.org. http://www.web100.org/
Accessed May 2005.

[43] Lawrence Berkeley National Laboratory. http://dsd.lbl.gov/DIDC/NTAF/
Accessed May 2005.

[44] Dunigan Tom; Mathis Matt and Tierney Brian. A tcp tuning daemon. *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, November 2002.