Evaluating TCP/IP Performance over Satellite Networks

Richard Thompson

Thesis presented in partial fulfillment of the requirements for the degree of
Masters of Science in Electronic Engineering Science
at the University of Stellenbosch.

Study Leader: Prof. S. Mostert

April 2004

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature:

Date:

Abstract

Interest in TCP/IP as a communication protocol for use in space communication has increased substantially with the growth of the world wide web (WWW). TCP/IP is a relevant communication protocol for space based communication systems that need to access the broader terrestrial communication network.

Low Earth Orbit(LEO) satellites have very short delay times between themselves and the ground, and correspondingly very short connection times. Staying in contact with a LEO satellite continuously through a space-based network requires large constellations of satellites and complex routing strategies.

Connectivity with the world wide web using a widely accepted protocol such as TCP/IP is desirable because it would make communication with the satellite over a terrestrial station possible, were it to route communication onto the WWW.

This thesis looks at the expected TCP/IP performance over satellite network links, identifies problem areas for current TCP/IP technologies, and makes suggestions for optimizing TCP/IP over such links.

The thesis also introduces a new performance benchmark, the equivalence level, allowing for the simplified description of TCP throughput behaviour over a broad range of network parameters. The performance of the Linux kernel release 2.4.18 TCP/IP stack is also evaluated.

Opsomming

Die belangstelling in TCP/IP as 'n kommunikasie protokol vir gebruik in die ruimte het kenmerklik toegeneem met die groei van die wêreld wye web (WWW). TCP/IP is 'n relevante protokol vir kommunikasie stelsels in die ruimte, veral met die doel om toegang tot land gebaseerde kommunikasie netwerke te kry.

Lae wentelbaan sateliete het baie kort vertragingstye tussen die aarde en die sateliet, en gevolglik baie kort verbindingstye. Groot sateliet konstelasies en komplekse verbintenis strategië word benodig om 'n lae wentelbaan sateliet deurentyd in kontak te hou met 'n ruimtegebaseerde netwerk.

Verbinding met die wereld wye web deur die gebruik van 'n wyd aanvaarde protokol, soos TCP/IP, is wenslik, want dit sal kommunikasie met die sateliet oor 'n aardgebaseerde stasie moontlik maak, sou dit kommunikasie oor die wêreld wye web stuur.

Hierdie tesis kyk na die verwagte werking van TCP/IP oor sateliet netwerk konneksies, identifiseer probleme met huidiglike TCP/IP tegnologie, en maak voorstellings vir die optimale funtionering van TCP/IP oor sulke konneksies.

Hierdie tesis stel ook 'n nuwe werkverrigtings maatstaf, die gelykheidsvlak, wat die vereenvoudige beskrywing van TCP/IP data tempo gedrag oor 'n groot variasie van netwerk parameters toelaat. Die werking van die Linux Kernel 2.4.18 TCP/IP stapel word ook geevalueer.

Acknowledgements

I would like to thank the TCP/IP research community, especially the creators of the *ns-2* simulator and the NISTnet emulator. Without their work none of my own work would be possible.

I would also like to thank Professor Sias Mostert and the ESL laboratory, for providing me with the opportunities that they have. Without their support this thesis would not have been possible. I am especially gratefull to Francois Retief, who held my hand as I learned some of the inner workings of Linux and whose ear I bent mercilessly.

I am also eternally grateful to my parents for their lifelong support. Finally I would also like to thank my wife, Renate. The fact that she is still my wife at the end of this project is surely a testament to her strength, courage and love for me.

List of Acronyms

ACK	Acknowledgement
BER	Bit Error Rate
BSD	Berkeley Software distribution
DBP	Delay Bandwidth Product
FACK	Forward Acknowledgement
FEC	Forward Error Correction
FIFO	First In, First Out
FTP	File Transfer Protocol
GEO	Geosynchronous Earth Orbit
HEO	High Earth Orbit
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISO	International Organization for Standardization
ISOC	Internet Society
LAN	Local Area Network
LEO	Low Earth Orbit
MTU	Maximum Transmission Unit
MEO	Medium Earth Orbit
NIST	National Institute of Standards and Technology
ns	Network Simulator
PAWS	Protection against wrapped sequence numbers
PER	Packet Error Rate
RFC	Request for Comment
RTO	Retransmission time out
RTT	Round Trip Time
SACK	Selective Acknowledgement
SNACK	Selective Negative Acknowledgement
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
MTU	Maximum Transmission Unit
ITU	International Telecommunications Union
MSS	Maximum Segment Size
AF	ACK filtering
ACC	ACK congestion control
CCSDS	Consultative Committee for Space Data Systems

cwnd	congestion window

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Practitioners using TCP/IP over satellite links are often under the temptation to use non-standard methods that could lead to dramatic performance gains. This means that all communication over the link must then be shielded from shared networks, or risk congestive collapse of the network.

Mechanisms that improve TCP performance over space based communication networks are not necessarily compliant with terrestrial TCP/IP. It is often best to only use accepted TCP/IP mechanisms over shared networks, including space-based ones. The Consultative Committee for Space Data Systems (CCSDS)[1] is a standards organization that looks at communication protocols specifically optimized for satellite communication. The Space Communications Protocol Suite Transport Protocol (SCPS-TP), which tries to comply with CCSDS standards is however not recommended for TCP/IP communication over shared networks for a number of reasons [33].

The aim of this thesis is to evaluate the current state of TCP/IP technology in space communication. This Thesis also makes recommendations as to how it can be optimized without compromising its use over shared networks.

The thesis focuses primarily on Low Earth Orbit (LEO) Satellite communication, with emphasis on its application on a satellite such as the Sunsat satellite [43], although communication with Medium Earth Orbit (MEO) and High Earth Orbit (HEO) satellites are also addressed.

[1] http://www.ccsds.org/

1.1 Sunsat and AX.25

AX.25 [59] was originally a requirement for Sunsat, the LEO micro satellite launched by the University of Stellenbosch in 1999. The reason for this is the fact that it was and is a standard for amateur packet-radio. AX.25 has, however, become severely dated [60], as the Internet revolution in the last part of the 1990s gathered steam.

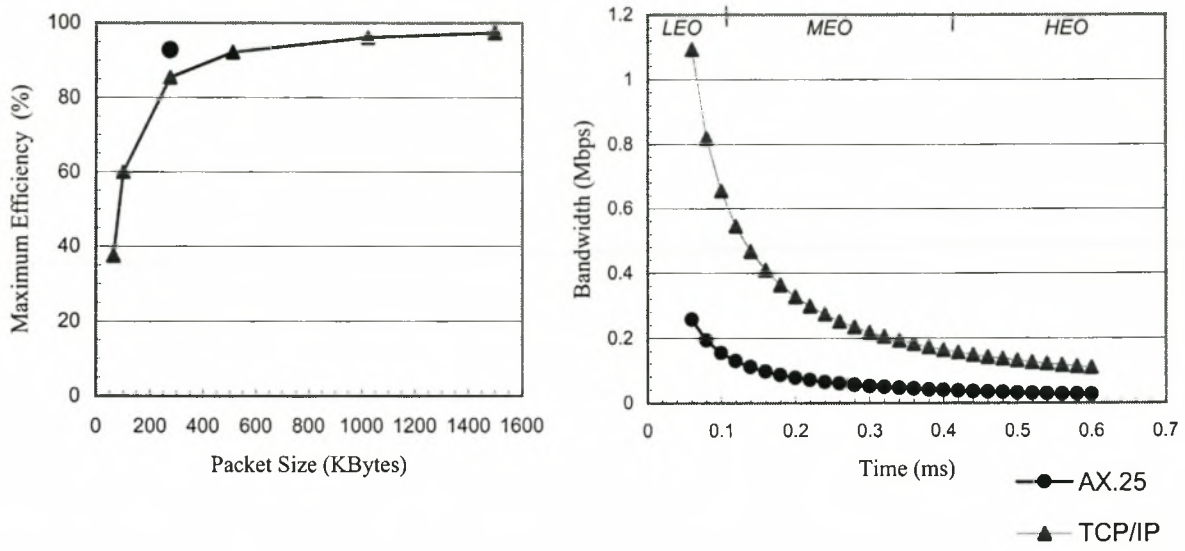

Figure 1.1: *Comparison between AX.25 and TCP/IPs theoretically throughput performance. (a) Maximum theoretical channel utilization. Notice that AX.25 has a fixed packet size. (b) Maximum theoretical bandwidth utilization.*

Although AX.25 is still in use in packet radio applications, newer developments in that field have focused on Internet connectivity as the primary requirement of alternative amateur radio protocols. AX.25 also does not cater for problems commonly experienced over a shared medium, such as congestion, or specific packet-radio problems such as partial collisions or high transmission error rates [27].

There are also various problems with the efficient use of AX.25 in a packet-radio network [45]. These fall into two categories, efficiency and functionality. The broadcast of amateur call signs with every frame leads to a loss of efficiency of 16 octets per frame. There are also problems with the flexibility of AX.25 because some functions implemented by AX.25, such as multiple repeater and source routing, should rather form part of the network layer and not the link layer.

The theoretical throughput of AX.25 and TCP/IP is compared in Figure 1.1. In the first graph the overhead for the two protocols are compared. The maximum efficiency axis indicates the theoretical maximum that a pipe can be filled with data, given a set

packet size. TCP/IP has a higher overhead if using similar size segments as the AX.25 protocol due to the fact that it uses a larger header size. Should network conditions make this favourable however packet sizes can be modified in TCP/IP to allow the protocol to adjust to levels comparable or even better then AX.25. The second graph illustrates the maximum bandwidth utilization of AX.25 and TCP/IP. This is the maximum amount of bandwidth that can be utilized as the time taken by a packet to travel from the sender to the receiver increases. TCP/IP extensions exist that allow TCP/IP to perform above this theoretical limit(see RFC 1323 [24]), yet no such extensions exist for AX.25.

1.2 Alternatives to AX.25

Van der Merwe [60] proposes four possible alternatives to AX.25.

AX.25 version 2.2 This new version of AX.25 addresses some of the problems associated with AX.25, but has not been widely accepted by the packet radio community.

TCP/IP This protocol was designed for use over terrestrial networks, where problems such as congestion and the fair use of bandwidth are coupled with the need for reliable data transmission. As its use has grown, much research has been done in its application in wireless and space communication.

DUAL The Digital Unreliable Amateur Link (Dual) was proposed to fill the gap between AX.25 and the need for TCP/IP interconnectivity.

Optimized Solution The final alternative is the design of a new protocol specifically designed for the space communication systems need.

These four options neatly summarize the way forward on developing satellite communication protocols, namely to continue developing an old protocol, to use a stopgap measure to provide the desired connectivity, to switch to a new protocol or to develop a new protocol for the specific situation.

The dwindling importance of AX.25 as a communication protocol, and its inherent problems, does not make it not a good choice. Similarly, the option of DUAL would not be a good option given that it has not widely been accepted. The design and implementation of a new communication protocol is both time consuming and error prone, and even then might not satisfy one's connectivity requirements. The use of an existing protocol such as TCP/IP, which is widely used and whose behaviour is well known, holds many benefits.

TCP/IP is a relatively old protocol, as is AX.25 is. Unlike AX.25 TCP/IP has continued to develop. TCP/IP is the most widely used protocol in existence and is supported by

a large research community. A large research effort into using TCP/IP in space is also underway, with the Internet Engineering Task Force (IETF) forming the working group *TCP/IP over Satellite Working Group*[2].

Many of the same problems that plague AX.25 also affects TCP/IP,but solutions have been found, implemented and accepted by the general community. Examples are larger packet sizes for better channel utilization, selective acknowledgement to allow better recovery from lost packets due to noise, and larger window sizes and sequence numbers to deal with larger bandwidth and longer distances. The available implementations of TCP/IP are however not perfectly suitable, and will probably never be, since they have to be applicable to a wider user base.

1.3 Problems with TCP/IP

TCP/IP was designed for use in terrestrial networks. The following characteristics of satellite networks differ from terrestrial networks [5], and might cause degradation in the performance of TCP/IP.

1. Long feedback loop
2. A large delay bandwidth product (DBP)
3. Transmission errors
4. Asymmetric link, be it bandwidth, bit error rate (BER) or delay.
5. Variable Round Trip Times
6. Intermittent connectivity

As has been noted in [17], when viewed from the transport protocol layers, satellite and mobile or wireless communications have many similarities. Only a subset of the above characteristics is normally found in a system however. Two approaches have been followed in adapting TCP/IP to the new network conditions: Either existing TCP mechanisms have been enhanced, as in the use of a larger window size to deal with the larger DBP of satellite communication [1], or special extensions are added, such as selective acknowledgement (SACK) to better recover from packet errors [38].

[2]http://www.isi.edu/tcpsat/

1.4 Work Methodology

Researching Transmission Control Protocol (TCP) [52] behaviour and properly evaluating the results can be a difficult task. Looking at the performance of a specific mechanism (i.e. algorithm or strategy) of TCP and comparing its performance with other mechanism under controlled circumstances do not always produce valid results. It is in the dynamics of the protocol, how it interacts with competing flows, that the true behaviour of TCP can be determined.

There are primarily three methods of assessing possible TCP behaviour. These are *Analysis*, *Simulation* and *Live Testing*. Each one of these methods has its advantages and disadvantages, with the ultimate aim being a more efficient reliable communication protocol. Allman *et.al.*[3] recommends that more then one method be used when testing a TCP/IP protocol mechanism.

Analysis Mathematical analysis is used to verify some aspect of the protocol, or to work out the theoretical limits of performance.

Simulation The network and protocols used in communication are entirely simulated by software.

Live Testing Tests are carried out over a real network using TCP analysis programs.

Emulation A hybrid of simulation and experimentation. An existing protocol stack implementation can be tested in a test bed using software to emulate network parameters.

This thesis will use simulation to investigate the properties of TCP/IP over a broad range of network conditions related to satellite communication. Emulation is then used to both test the performance of the Linux TCP/IP stack, as well as provide 'real-world' results with which to compare the simulation results.

A number of assumptions are made, the most important being a simplistic view of the number of flows that would traverse a satellite link at any time.

1.5 Thesis Structure

The thesis is divided into the following chapters. Chapter 2 and 3 provide a background study into the field of TCP/IP research. In Chapter 2 an overview is given of the current

state of TCP/IP. In Chapter 3 enhancements and changes for satellite TCP/IP communication are discussed. In Chapter 4 we look at the simulator used and the results of various experiments that were carried out. In Chapter 5 a new performance metric for TCP is proposed. In Chapter 6 emulation is discussed as well as the performance of the Linux TCP/IP Stack.

1.6 A Note on Bits and Bytes

The convention of the smallest data unit, *Bit* or *Byte*, is one that changes regularly with context. If not stated otherwise we will use bit instead of bytes when referring to channel characteristics. A *link* will refer to the data channel between two computers. A *connection* will refer to the end-to-end flow of data, over one or many computers.

Chapter 2

An Overview of TCP/IP

2.1 Introduction

The TCP/IP [1] protocol suite [56] is the most widely used protocol suite in existence. It has been in development since the late 1960's and is also one the most mature protocols available. The Internet Society (ISOC) currently controls its future development, and publishes the specifications for TCP/IP in so-called *Request For Comments* (RFC) documents. The Internet Engineering Task Force (IETF) also funds the *TCP over Satellite working group.*

2.2 TCP/IP Layers

The most important protocols in the TCP/IP suite is the Internet Protocol (IP), the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). In the layered International Organization for Standardizations(ISO) Open Systems Interconnection(OSI) standards view IP forms the network layers, while TCP and UDP are transport layers protocols (see Figure 2.1). The other protocols in the suite are built on these three protocols and most are found in the application layer.

2.2.1 The Link layer

The Internet Protocol (IP) is directly connected to the link layer, which is responsible for sending and receiving IP datagram's for the IP module. TCP/IP supports many different

[1] Also known as the Internet Protocol Suite

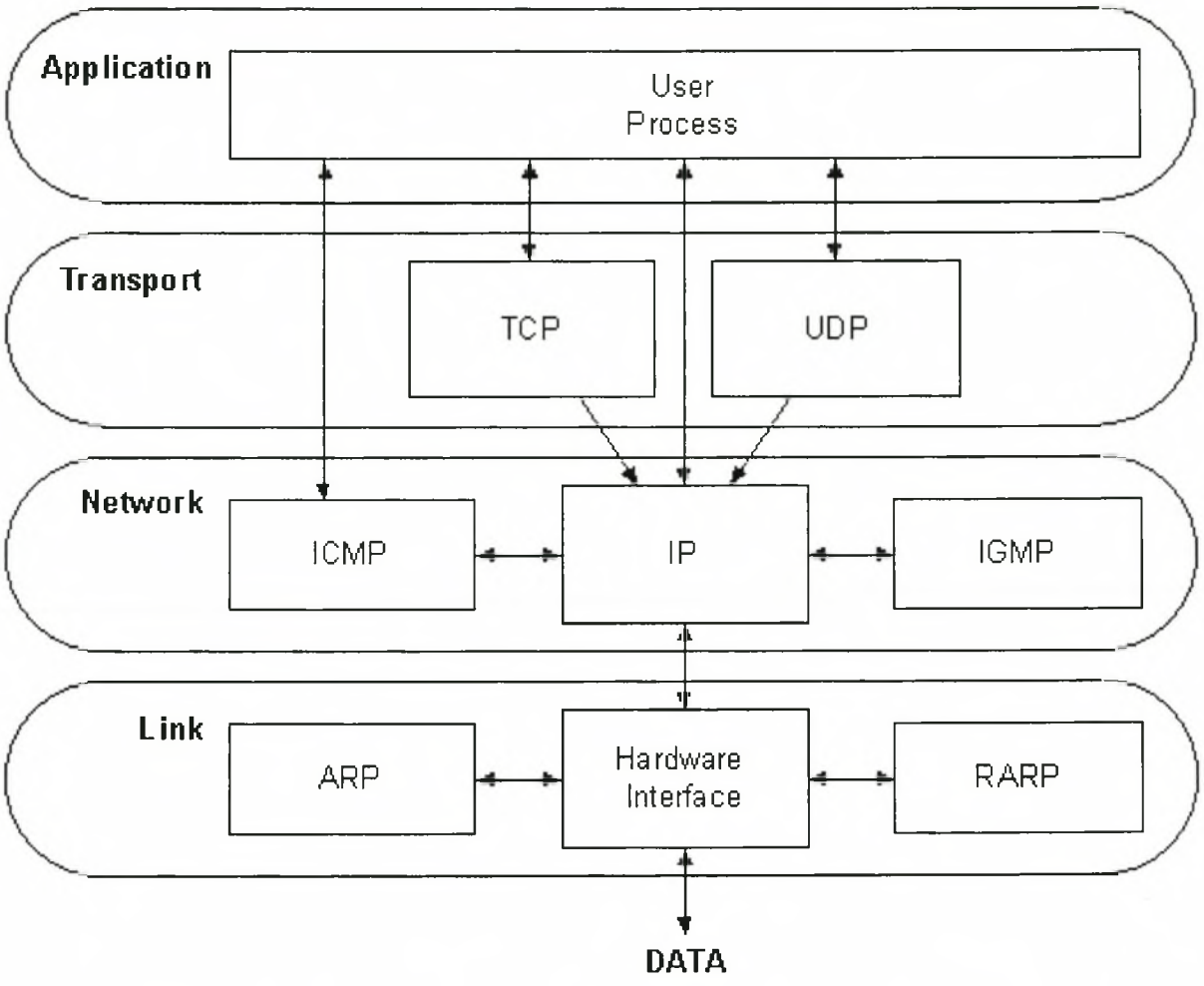

Figure 2.1: *The TCP/IP Protocol Suite as seen i.t.o. the layered ISO protocol view*

link layers, such as Ethernet and token ring.

The effect of the link layer on the TCP/IP protocol suite performance can be significant, and it can affect the TCP/IP protocol suite in three ways. Firstly there is delay that is introduced by the underlying hardware of the system. Secondly, the bit error rate of a channel can either be improved or worsened, and lastly it specifies the Maximum Transmission Unit (MTU), which determines the maximum segment size for TCP (For Ethernet this is 1500 bytes).

2.2.2 The Internet Protocol

The Internet Protocol (IP) is the base protocol of the TCP/IP protocol suite and all data is transmitted as IP datagram's. It is specified in [51]. IP is an *unreliable connectionless datagram delivery service* between two communication end points. Datagrams consists of an IP header and optional data of variable size.

IP packets are *routed* across a network, which consists of one or more links. An IP packet is routed independently on a hop-by-hop basis. This means that packets do not know the route to their destination (it is assumed that the next hop is closer to its destination), and that separate packets can follow different routes. This method of routing introduces behaviour that impacts on the performance of the protocol suite, namely

Packet Drops This is by far the most common occurrence. Routers that are too busy drop packets. This is used by TCP to identify congestion on the network, but causes unnecesary performance degradation when packets are lost due to errors in the physical and link layers.

Packet Re-ordering Packets can travel by different routs across a network. When there is a major route change (i.e. a faster route has been found) packets might become reordered.

Packet Corruption A packet might only be corrupted by link layer errors, but still be received at the destination. These packets are then dropped at the destination, causing the same problems for TCP/IP as with packets that are lost during transmission.

Packet Replication This occurs when multiple copies of the same packet are received. This is a rare and not well-understood phenomenon, and has been blamed on the failure of token ring networks [49].

A new version of IP, IPv6, is being planned to replace the current IPv4. It uses a larger IP address space (from 32 to 128 bit), but will be compatible with IPv4 via encapsulation. This will lead to more overhead however, negatively impacting on the performance of TCP, especially over low bandwidth channels.

2.2.3 Transmission Control Protocol

The Transmission Control Protocol (TCP) [52] provides a *connection-oriented, reliable, byte stream service* built on top of the IP layer. It is used by a number of Internet services, such as FTP [53] and HTTP [10] to communicate.

TCP provides a *reliable* service through the use of a positive acknowledgement(ACK) mechanism. If a sender does not receive an ACK signal for a given segment within a certain amount of time the segment is resent. The amount of time that a server waits is determined by the *re-transmit timeout* (RTO) [28] value.

TCP also provides flow control by using a sliding window. This allows the sender to only transmit a given number of segments before receiving an ACK. Each TCP segment sent contains a *window advertisement* which advertises the receiver's upper bound for the sender's sliding window. The sliding window size is allocated 16 bits in the header, thus allowing for a maximum sliding window size of 65 535 bytes [52].

A number of *congestion control* algorithms are also used to ensure that data is transmitted at a rate appropriate to the network resources available. This is important as a network can suffer from *congestive collapse* [46]. This is a state where communication is highly inefficient, with too much data being sent across the network, but little arriving at their intended end destination.

Congestion control algorithms attempt to prevent this congestive collapse by detecting congestion and then reducing the transmission rate. This might have a negative impact on the performance of TCP as performance is sacrificed for fairness. This is especially true for TCP links with long round trip times (RTT) [31].

Slow Start and Congestion Avoidance

Congestion control is implemented using a variable called the *congestion window (cwnd)*. The congestion window is the size of the sliding window used by the sender, and cannot exceed the receivers advertised window. TCP can therefore not inject more unacknowledged segments into the network than the receiver can process. TCP uses the slow start mechanism to quickly probe network capacity during start-up, and uses the more conservative congestion avoidance mechanism to probe for additional capacity later.

The slow start algorithm is used to gradually increase, from an initial size of one segment, the amount of unacknowledged data injected into the network. By gradually increasing the size of the sliding window intermediate routers are not overwhelmed when a connection is opened. For each ACK received, TCP increases the value of cwnd by 1 segment, providing exponential growth for the TCP connection.

Slow start continues until *cwnd* reaches the *slow start threshold* (*ssthresh*), initialized to the receiver's advertised window, or an ACK is not received. If TCP's RTO expires for a given segment, TCP retransmits the segment. The value of *ssthresh* is set to half that of *cwnd* and *cwnd* is reset to 1[2]. Once again TCP enters into slow start, this time until *cwnd* reaches half the previous value of *ssthresh*.

[2]The *cwnd* is actually specified in bytes, so *cwnd*'s size will depend on the segment size. For simplicity we just refer to the number of segments.

Congestion avoidance follows slow start, and during this phase the value of *cwnd* is greater than or equal to *ssthresh*. The value of *cwnd* is increased at the rate of $1/cwnd$, adding roughly one segment to the value of *cwnd* every Round Trip Time (RTT) until the maximum of the receivers advertised window is reached.

Fast Retransmit, Fast Recovery

Fast retransmit and fast recovery [23] [57] are methods that TCP can use to more effectively recover from segment drops. Instead of relying solely on the RTO, a smoothed average of RTT plus some variance, TCP can retransmit a segment and adjust the sending rate before the RTO timer expires.

TCP must generate an immediate acknowledgement [52], or duplicate ACK, when an out of order segment is received. The purpose of this is to let the sender know that a segment was out of order and what sequence number is expected.

The sender however, does not know whether it was a lost or out-of-order segment that caused a duplicate ACK. TCP therefore assumes that if only one or two duplicate ACKs are received that there was only a reordering of data received, but should three or more duplicate ACKs be received TCP will assume that a segment has been lost and proceed to retransmit the lost segment. This is known as the *fast retransmit* mechanism.

Next, TCP enters congestion avoidance rather than slow start. The reason for this is that for fast retransmit to be activated a data segment must have reached the receiver to trigger the duplicate ACKs. There is thus no reason to halt the flow of information abruptly. This is the *fast recovery* mechanism.

Fast recovery sets *ssthresh* to one-half of the minimum of the current *cwnd* and the receivers advertised window and sets *cwnd* to sstresh plus 3. Each time another duplicate ACK is received cwnd is increased by 1. When an ACK arrives to acknowledge new data, the *cwnd* is set to the value of *ssthresh*. The flow is reduced to half its previous speed, and this is called the Congestion avoidance mechanisms.

As already discussed, TCP must retransmit a segment if the RTO expires before the segment is acknowledged. This can only work well however if the timer granularity used in the operating system is less or equal to the RTO. The BSD Unix operating system's timer granularity however is 500ms [62], which is not sufficient to trigger retransmission on most terrestrial networks (RTT less than 500ms). Timer granularity on the Linux kernel however is much finer, and in the newer kernels (release 2.5 and later), micro granularity will be an option. It is an unresolved question however if the ability to make better RTT estimates will nullify some of the performance gains achieved by using fast retransmit and

fast recovery.

Selective Acknowledgement

A mechanism of cumulative acknowledgement is used by TCP, which only acknowledges segments at the edge of the receive window. This means that the sender must either wait the full RTT to discover whether a packet has been lost, or start to retransmit successfully received packets [18]. When multiple segments are dropped, the ACK-based clocking mechanism used by TCP is disrupted, which has a negative impact on TCP performance.

The Selective Acknowledgement (SACK) option [38] is a strategy that corrects this behaviour. It allows the receiver to notify the sender as to which segments have arrived. This allows the sender to only retransmit those segments that have been lost.

If the return path of a channel were loss less, one block per SACK option would always be sufficient. However, since this is not the case normally, the SACK option is defined to include more then one SACK block in a single packet. This increases TCP robustness in the presence of lost ACKs. With the use of the time stamp option [24] there is room for three SACK blocks. The use of other TCP options will decrease the number available of blocks, but with even only one block SACK TCP is more robust than normal TCP implementations [20] and suffers from less unnecessary retransmission.

Selective Negative Acknowledgement was proposed in [17]. This combines the concept of Negative Acknowledgement [21], which allows the receiver to report a block of data that has *not* been received and SACK, which provides the ability to report that a number of blocks are missing. This mechanism has not been widely accepted [6].

Forward Acknowledgement

The Forward Acknowledgement algorithm [39] uses the additional information provided by the SACK option to keep a measure of the number of bytes outstanding in the network. This is done by introducing two new state variables, *snd.fack* and *retran.data*, and keeping information on what data has been received and what retransmitted. FACK has a small performance advantage over SACK, but as with SACK can only help improve the performance of the congestion control algorithms.

2.3 TCP/IP performance measurement

2.3.1 TCP Reliability

The reliability of TCP is at all times of paramount importance when considering either the effects of channel characteristics or the use of new mechanisms for better performance. Reliability is affected by factors such as wrap around sequence numbers [24], which can occur when the *delay, bandwidth product* (DBP) is too large.

2.3.2 TCP/IP Metrics

The performance of TCP depends on the product of the transfer rate and the round trip delay. The DBP measures the amount of data needed to fill the pipe and the amount of unacknowledged data that TCP must be able to handle.

Influential factors in TCP performance are, for example, the window size limit, packet losses and the reliable measurement of round-trip time (RTT).

The question of what metrics to use is important, since performance is relative to the application. Various metrics have to be used to measure TCP/IP of which the most important are:

Throughput A Base measurement of the size of data that can be moved across a link in a fixed amount of time. The time taken to transmit the data is measured at the sender from the time that the first segment is transmitted to when the last ACK is received.

Latency A measurement of how quickly TCP/IP can respond. Typically it is the time taken for an ACK to be received on a data segment.

Fairness Is an indication of how TCP/IP flow interacts with other flows. It is not a well defined metric, and there are different views on how to measure fairness [8].

Quality of Service This metric tries to descibe to what extent the flow across a channel is guaranteed. As is the case with the fairness metric, the Quality of Service metric is still a metric that requires further research.

Some of these metrics are functionally opposed in that an improvement in one could have a negative impact on another. An example of this would be using a smaller data segment size to improve the latency of a TCP connection. Latency is improved due to the fact

that it takes less time to transmit the packet, and for an acknowledgement to be received. This can have a negative impact on the throughput performance of the TCP connection however, as the ratio of header information to the data carried by packets is increased.

The primary needs of a LEO Satellite such as Sunsat are threefold. Firstly none or few competing flows; secondly an onboard management control system that does not require real time input, and lastly large amounts of photographic data that needs to be transmitted. This makes throughput the most important metric for these purposes.

2.4 Conclusion

This chapter looked at the effect of the link layer and the Internet protocol on TCP performance. The basic mechanisms of TCP was discussed. Finally the metrics used to measure the performance of TCP/IP were introduced.

Chapter 3

Enhancing TCP/IP Over Satellite Channels

3.1 Introduction

Two approaches have been followed in adapting TCP/IP to satellite network conditions: either existing TCP mechanisms have been enhanced, as in the use of a larger window size to deal with the larger delay, bandwidth product of satellite communication [1], or special extensions are added, such as selective acknowledgement (SACK) used to better recover from packet errors [38]. In this chapter we look briefly at the link characteristics of satellite communication and discuss some mechanisms that have been proposed to improve performance.

3.2 Satellite Channel Characteristics

One characteristic dominates satellite communication, and that is bandwidth. The Bandwidth characteristics, along with the selection of the frequency band and the power flux density are all regulated by the International Telecommunications Union (ITU). Table 3.1 [32] contains a list of frequency bands assigned to Satellite communication.

Several characteristics of satellite channels that differ from those encountered in traditional wired channels that might impact on the performance of TCP/IP have been identified [5]. These characteristics are,

Long Feedback Loop Some satellites have a long propagation delay (approximately 250 ms one way for a geostationary satellite [36].) This can negatively impact on the latency (see Section 2.3.2) as well as some of TCP's congestion control mech-

Table 3.1: *Satellite Frequency Limitation Bands as established by the ITU*

Frequency Band	**Uplink** *(GHz)*	**Downlink** *(GHz)*
C-band	5.9 to 6.4	3.7 to 4.2
X-band:	7.9 to 8.4	7.25 to 7.75
Ku-band	14.0 to 14.5	12.5 to 12.75
Ka-band	27.5 to 31	17.7 to 19.7

anisms (see section 2.2.3). Figure 3.1 compares the delays commonly experienced by satellites in four orbit groups. It should be noted that there can be a further delay of between 20 to 40 ms on both the sender and receiver side due to a hardware delay. (Such as the time it takes serial modems to transmit for example.)

Large delay, bandwidth product The DBP defines the amount of data a protocol *should* theoretically be able to transmit without acknowledgement. TCP's 16-bit window field limits the effective bandwidth however to $2^{16}/RTT$ [42].

Transmission errors [1] Satellites generally have a low signal-to-noise ratio because of the distances involved. Some frequencies are also particularly susceptible to atmospheric effects such as rain attenuation, while mobile users are susceptible to multi-path distortion and shadowing.

Transmission Errors are problematic for TCP because TCP sees all dropped packets as signals of congestion in the network, and accordingly (See section 2.2.3) reduces the send rate unnecessarily.

Asymmetric use A network is said to be asymmetric with respect to TCP performance when throughput is affected not only by the forward link, but by the reverse link as well [9]. Asymmetry can affect various aspects of a satellite channel, such as bandwidth, delay and BER.

Variable round trip times The propagation delay of satellite orbits changes with time. Whether this will affect the performance of TCP is still an open question. The handover of a connection from one satellite to another can also lead to large variable round trip times.

[1] Whole numbers are commonly used when referring to the number of transmission errors, or Bit Error Rate (BER), of a link. It is an indication of the order of data loss over a link. For example a BER of 6 means that there is one error for every million bits transmitted (10^{-6}).

Intermittent connectivity Satellite constellations require that TCP connections be transferred from one node to another. This could impact on TCP/IP performance.

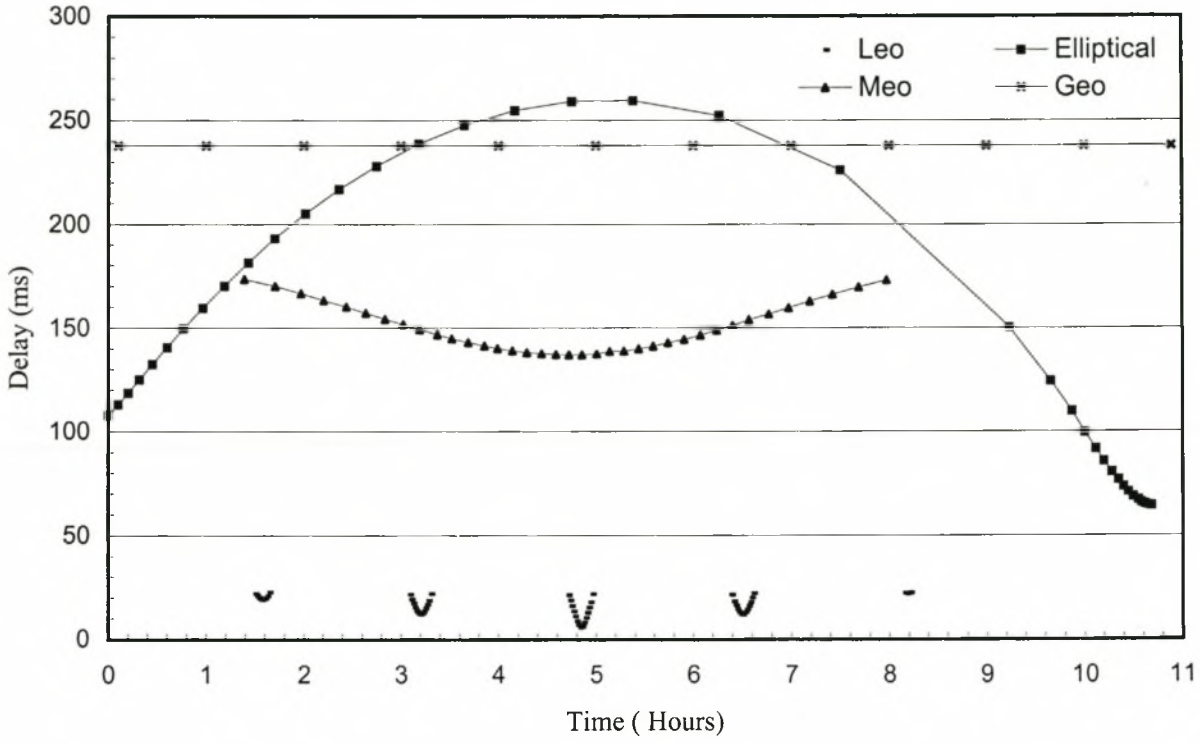

Figure 3.1: *Satellite Delay Profiles*

As has been noted in [17], satellite and mobile or wireless communications have many similarities when viewed from the transport protocol layers. Normally however, only a subset of the above characteristics are found in any one particular network.

3.3 Satellite Orbits

Satellites can be classified according to their altitude above the earth. Satellites can thus be devided into four orbital groups; Low Earth Orbit (LEO), Medium Earth Orbit (MEO), High Earth Orbit (HEO), and Elliptical Orbit satellites. Elliptical orbit satellites is the special case where the altitude of a satellite varies dramatically over the period of an orbit.

It is also possible to classify orbits according to their inclination to the earth. Polar orbits circle at near-polar inclination (a true polar orbit has an inclination of 90 degrees), while geostationary satellites orbit along the equatorial plane of the earth. Inclined orbits fall between these two extremes.

The view of using the altitude of a satellite to classify it is of much more use to communication research given the impact the delay between two points can have on communication.

3.3.1 Low Earth Orbit

A satellite is generally seen to have a low earth orbit if its altitude is between 300 kilometers, to around 5000 kilometers. The low altitude of the LEO satellite translates into very high velocities in relation to the ground. Connection times between these satellites and fixed terrestrial points are therefore short, at around 8 to 12 minutes per orbit for a satellite such as Sunsat.

The footprint (i.e. the size of the region that the satellite can view at any given time) of a LEO satellite is also small due to the low altitude, requiring a larger number of satellites to continuously cover a region. Examples of existing LEO satellite constellations are the Macrocell constellation with 96 Satellites, and Teledesic, with 288 satellites.

Another aspect of LEO satellite orbits is radiation. High-energy particles, trapped in the earth's magnetic field, the magnetosphere, can be very disruptive of space operation and lower the lifetime of a satellite considerably.

It should be noted that satellite operating below an altitude of 1000 km are relatively safe from the ravages of the magnetosphere. Of interest is the South Atlantic anomaly, a region where the protective Van Allen belt is lower then at other areas.

LEO satellites are ideal for earth observation, and for low latency communication networks.

Figure 3.1 illustrates the delay profiles for various satellites. The SUNSAT LEO satellite was used to illustrate the LEO delay profile. The delay between the ground station and the satellite changes rapidly during the course of one pass by the satellite.

What is not included in this delay profile however is the delay caused by the hardware used. It is assumed for the sake of simplicity that a hardware delay of 20 ms is to be found in both the sender and the receiver of any wireless communication. One way delay for the LEO satellite such as SUNSAT can thus vary between 25 and 45 milliseconds.

The effect such a change in the delay of a TCP connection would have on performance was investigated. See Chapter 6.4.2.

3.3.2 Medium Earth Orbit

Medium Earth Orbit satellites circle at an altitude of roughly between 5000 and 10000 kilometers. These satellite suffer from very high levels of radiation, and as a result this is not a very popular orbit.

MEO satellites do have a number of advantages though. Due to the higher altitude, and corresponding larger footprint, fewer satellites are needed to cover a region. The latency of a connection to a MEO satellite is also a lot less when compared to that of a geostationary satellite.

MEO satellites make for a good compromise between the high latency of higher satellites and the small footprint of lower satellites, if radiation effects can be dealt with. They are also useful as bridges in LEO constellations, reducing the required number of satellite hops in a connection.

3.3.3 High Earth Orbit

High Earth Orbit satellites circle at altitudes above 10000 kilometers, and are usually above the magnetosphere where radiation is at its worst. The most common high altitude satellites are geostationary satellites, which orbit the earth at a fixed point relative to the Earth.

Geostationary satellites have a large footprint, but cannot cover areas in the higher or lower altitudes very well since the orbit is along the equatorial plane. The latency of geostationary satellites is very high, due mainly to the large distances (approximately 36000 kilometers).

3.3.4 Elliptical Orbits

Elliptical orbits do not maintain relatively fixed altitudes. Due to the nature of their orbit, they are very useful for covering a specific region continuously with very few satellites. It is also possible to cover regions that are very far north and south of the equator. A good example of this would be the Molniya constellation, used by the Soviet Union military to provide satellite communication for their forces in Siberia, close to the North Pole.

3.4 Data Link Layer Mechanisms

The data link layers can have a large impact on the performance of TCP/IP. If the link layer protocol implements measures such a floating window and a retransmission mechanism, performance can suffer just as is the case with TCP (see Section 3.5.1). Two mechanisms are however recommended for implementation. These are Path MTU discovery, and Forward Error Correction (FEC).

3.4.1 Path MTU Discovery

Path MTU Discovery [44] [30] can be used to determine the maximum packet size that can be used over a network channel without incurring fragmentation in the lower layers. This allows TCP to safely use the largest possible segment size, thus lowering its overhead. It also allows TCP to increase the congestion window *cwnd* more rapidly (i.t.o. bytes) as this mechanism is segment-based.

The disadvantage of Path MTU discovery is a delay that might occur when data is first sent. However, given that the MTU in many networks is standardized, this does not occur often in practice.

Whether the use of the largest possible segment size is optimal for TCP over satellite is an unresolved question, as the BER properties of a channel might be affected by the segment size. With the use of methods such as Forward Error Correction (FEC) however, larger segments should provide better performance [37].

3.4.2 Forward Error Correction

TCP/IP currently has no mechanism to distinguish between packet loss due to errors in the channel, or packets dropped due to congestion. Performance is thus negatively affected when TCP reduces the congestion window size unnecessarily.

The use of (FEC) coding can improve the BER characteristics of a satellite channel. FEC will however not solve all bit error problems on a TCP channel as problems such as jamming, deep space mission and rain fading will still persist. FEC also requires additional hardware, as well as lowering the available bandwidth. It can also add both delay and a timing jitter to the channel.

3.5 Accepted TCP mechanisms

Standard TCP mechanisms are those that have been accepted by the ISOC standards committee, and as such are guaranteed to not cause congestive collapse on shared terrestrial link.

3.5.1 Larger Initial Window

The initial window size indicates the number of segments to be transmitted when a connection is initially established. Currently the initial window size is set to one, but [1] argues that by increasing this initial window size better performance can be attained over satellite links. By sending more packets initially the congestion window can open more rapidly, wasting less bandwidth during the slow start algorithm. Sending at *least* 2 packets initially also allows TCP not to wait for an ACK timeout because the receiver only sends an ACK every *second* full-sized segment (the delayed ACK mechanism [11]). Increasing the initial congestion window *cwnd* requires changes to the sender side TCP layer however.

[4] proposes that the equation $min(4 \times MSS, max(2 \times MSS, 4380bytes)$ be used to determine the value of *cwnd*.

RFC 2581[7], a standards tracking document, allows TCP to use up to 2 segments, but the use of an initial congestion window size of 3 or 4 is not expected to present any danger of congestion collapse. It may however lead to degraded performance over some network channels. The ideal value for *cwnd* is however still an unresolved question and will probably need to be adjusted specifically for each given situation.

3.5.2 The Window Scale Option

A 16 bit field is used to report the received window size to the sender, therefore the largest window that can be used is 2^{16} or 64 kilobytes. The Window Scale option allows window sizes of 32 bits by defining a scale factor which is used to find the true window size. This is an important mechanism for high DBP links if a link is to be properly utilized.

3.5.3 Round Trip Time Measurement Option

TCP needs accurate and current RTT estimates to adapt to changing network conditions. Getting the correct estimates may be difficult in both theory and practice, especially in

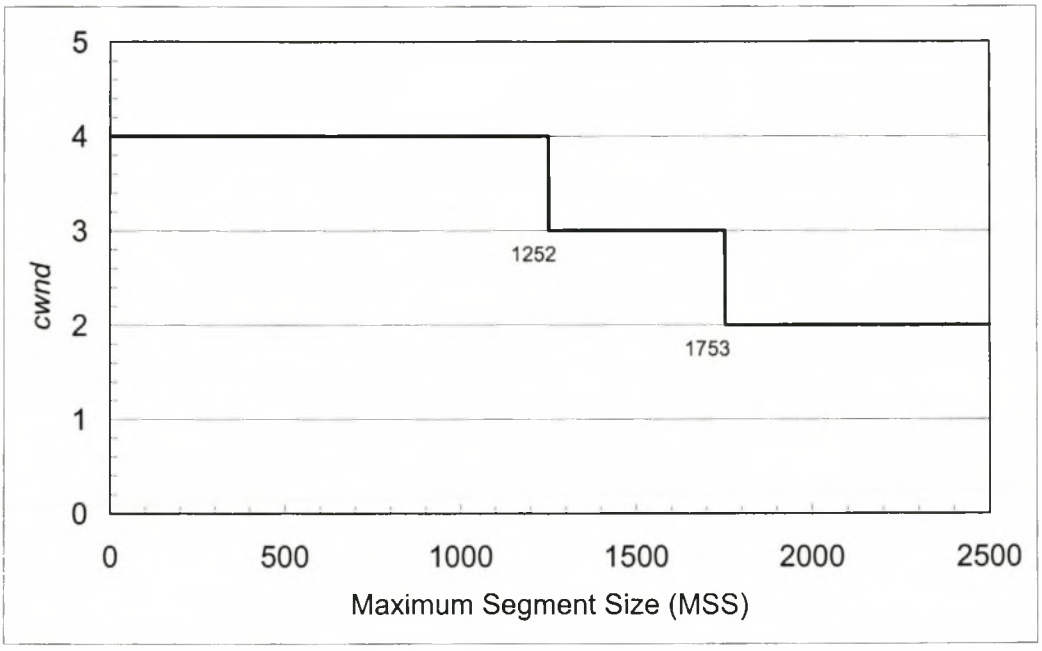

Figure 3.2: *Initial Congestion Window Size*

high DBP links such as found in GEO satellites (see [24], p11, for a discussion on problems with measuring RTT).

A TCP timestamp option has been proposed as a solution. The sender side marks a packet with the current time, which the receiver then reflects back to the sender in the acknowledgement. A single subtraction then gives an accurate return time measurement.

Measuring RTT in high DBP links is difficult, and it is doubtful whether the overhead of the RTT Measurement Option will be justified in LEO satellites, or even in some MEO satellites. Elliptical satellites pose a different problem with their changing RTT.

3.5.4 Protection Against Wrapped Sequence Numbers

Each TCP segment is identified with a 32-bit sequence number. Packets with duplicate sequence numbers can therefore be received in high DBP links. The Protection against Wrapped Sequence Numbers (PAWS) mechanism uses the same timestamp option as the RTTM.

A packet can be discarded as a duplicate if the timestamp from a received packet is less

than that from a previously received packet. The PAWS mechanism is highly recommended for links with a DBP above the theoretical maximum performance of TCP (see Figure 1.1).

3.5.5 Selective Acknowledgement

See Section 2.2.3

3.5.6 Forward Acknowledgement

See Section 2.2.3

3.5.7 Acknowledgement Strategies

RFC 1122 [11] states that TCP *should* ACK every second 'full' segment received. There is also a timeout mechanism by which if no second full segment is received within 500ms an ACK is sent.

In links where the delay is especially long this would mean that the time taken by TCP to fill the pipe is unnecessarily long, since *cwnd* is only increased for each received ACK. While it would be beneficial to switch this mechanism off, it is not recommended by [5] for shared networks.

Figure 3.3 illustrates the time taken to fill the pipe for various connection delays with a fixed bandwidth. The delay to fill large pipes can be substantial, and the need to avoid a restart should packets be lost is obvious.

3.6 Experimental TCP/IP Mechanisms

3.6.1 Explicit Congestion Notification

Explicit Congestion Notification (ECN) [19] [54], an enhancement to IP, allows routes to signify that they are experiencing congestion without dropping packets. Routers can instead set a Congestion Experienced (CE) bit in the packet header of ECN-enabled transport protocols.

ECN may be especially beneficial to connections with small congestion window sizes because the sender can avoid many of the re-transmissions and associated timeouts [22].

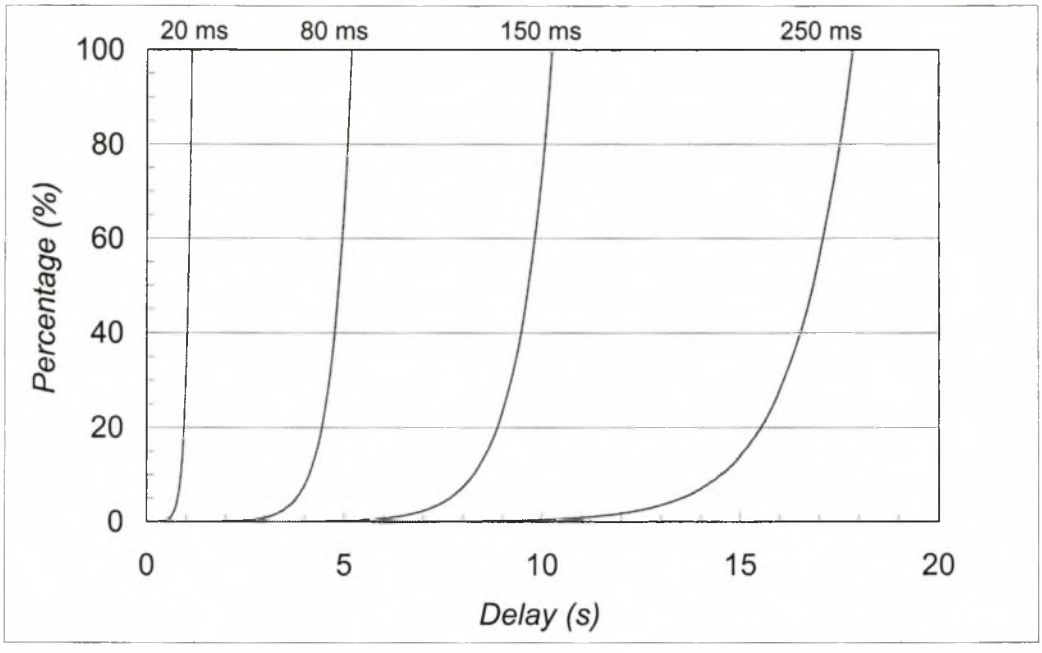

Figure 3.3: *Time taken to fill pipe.*

The relative performance benefit of ECN for bulk transfers is the greatest when on average each flow has 3 to 4 outstanding packets during the flow's round-trip time [63].

3.6.2 Pacing TCP Segment

Connections with a high DBP take longer to fully open up during slow start, and with short, bursty TCP connections this can have a very undesirable impact on performance. By implementing rate-based pacing (RBP) on the sender side, TCP can avoid entering slow start after an idle period and the related performance difficulties.

RBP is a technique whereby the data sender temporarily paces TCP segments so as to restart the ACK clock, in the absence of incoming ACKs. This is discontinued as soon as the first ACK is detected.

RBP would be especially valuable for HTTP transfer over high BDP connections. It does require sender side modifications however, but has been shown to improve performance for WWW-like traffic [61].

3.6.3 TCP Header Compression

TCP Header Compression, described in [16], uses the fact that information in TCP/IP headers remains relatively constant to compress header size from around 40 bytes for IPv4 to 5 bytes (3 bytes for some common cases).

This can have a positive impact on the bandwidth utilization performance of TCP/IP connections with low or medium bandwidth. Used in conjunction with the 'twice' algorithm, a header request mechanism and a cache for packets that do not uncompress successfully (see [16] for a discussion of these mechanisms), the header compression mechanisms has been shown to improve throughput by over 10-15%.

Because header compression can reduce the packet size, there is a related benefit of a decreased packet error rate and an increase in the reliability of the connection (if the bit error rate is uniform).

If synchronization between the sender and receiver is lost however, many packets may be lost. Losing synchronization too frequently might have very adverse effects on the performance of the protocol.

3.6.4 ACK Congestion Control and Filtering

Asymmetric TCP/IP links can have a negative impact on performance. The reason for this is because TCP uses ACKs to pace the rate at which data is sent. Asymmetric links, whether the link is bandwidth asymmetric, delay asymmetric or BER asymmetric, interfere with the ACK pacing used by TCP.

One method of dealing with this is to reduce the number of ACK's through either ACK filtering (AF) or ACK congestion control (ACC) [9]. AF is the process whereby redundant cumulative ACKS are removed from the return path. It requires sender side adaptation however, or the use of ACK reconstruction. Given the relatively simple path taken over a dedicated satellite-ground station link, it would not be recommended. It would however be a good option for connections that use asymmetric satellite links as part of a larger connection.

ACC is an extension of the 'delayed ACK' strategy, whereby a router or sender can control the pace at which packets are transmitted. Once again the strategy requires either sender-side adaptation or intervening routers. Both AF and ACC are good strategies for multiple link connections, but simpler network topologies would perhaps be better served by a rate-based pacing strategy.

3.7 Other mechanisms

3.7.1 Multiple Data Connections

Using multiple TCP/IP connections to fully utilize the bandwidth of a connection is more of an application level strategy, but it is used to increase performance over high bandwidth links. It is not efficient, and it has limited applicability (bulk data transfer primarily), but it is effective and the use of this strategies will likely increase as bandwidth increases [40].

There is an alternative strategy of using multiple TCP connections that connect to multiple data sources to complete bulk data transfers (see the freeware program Xolox[2] as an example). Data can be downloaded, and different sources can be used to provide different parts of the data. This is possibly a very powerful strategy, and could be a very useful technique over LEO satellite constellations. The large numbers of interconnected satellites, and the relatively short connection times can make for a very complex routing problem. By looking at individual LEO satellites in a constellation as independent data servers, information can be promulgated through the network, bypassing the need for routing connections through a network. Data could then be given a 'time to live' counter that would allow data to be purged from the network after a reasonable time. This strategy would obviously consume much more bandwidth, and therefore power, than setting up single transaction connections. It could certainly be useful however for those constellations that can only provide partial coverage or that provide content to multiple destinations.

3.7.2 Spoofing and Proxy Servers

Spoofing is the method whereby a router close to the satellite link sends back acknowledgements to the sender for data received. The router is then responsible for the transmission of data to the satellite.

A Proxy server is very similar to spoofing, but a connection is made to the proxy server instead of the satellite directly. The proxy server can then use whichever protocol it wishes to transmit data to and from the satellite.

Spoofing is a relatively simple solution, but has a number of problems associated with it such as the need to buffer data segments before transmitting them to the satellite. It also

[2] http://www.xolox.nl/

breaks the end-to-end semantics of a TCP/IP connection. The use of a Proxy server is therefore recommended instead of spoofing.

The fundamental flaw with both spoofing and a proxy server is the need to direct traffic through these connections. If the route was to change, or a connection made that does not go through the router or proxy machine, these methods would be useless.

Nevertheless the use of a proxy server is recommended if the channels of communication with the satellite are controlled. A thorough discussion of the use a proxy server in satellite networks can be found in [15].

3.8 Conclusions and the Future of TCP/IP

The bandwidth requirements of communication systems are set to grow rapidly in coming years. As the global and near-earth communications network grows, and better performance will be expected of the protocols that are to be used.

TCP/IP has become the standard for communication over the global network, and systems that wish to connect to this network are required to use TCP/IP. The varied requirements of services however, will lead to conflicting mechanisms being incorporated into the TCP/IP protocol suite. An example of this is the generally small amounts of traffic generated by HTML pages, compared to the large data traffic generated by multimedia services. If TCP/IP is to remain the leading protocol, its success will depend on its ability to adopt new mechanisms to deal with these conflicted requirements. The success of new methodologies however, will depend on their ability to be integrated into the existing TCP/IP framework.

A good example of the way in which TCP is heading is Fast TCP [25]. The requirement for high bandwidth connections over long delay paths has led to the investigation into the reasons behind network instability and to the boundaries to which the performance of the TCP protocol can be pushed. The ability to detect congestion, to recover successfully from errors while keeping the pipe filled, to avoid the burstiness associated with shared networks and the ability to resume normal operation quickly after an interruption, are however still the main concerns.

Chapter 4

Simulation and Results

4.1 Introduction

Simulation can be used to imitate segments of a communication network, or the entire network. For example, the satellite to ground station link can be simulated while the computers and networking hardware used in the experiment can be real. This latter option is discussed in Chapter 6 and is referred to as emulation. Alternatively a network simulator can be used to simulate not only the link parameters such as bandwidth and delay, but the network and protocol as well. This is the definition we will use when referring to simulation.

There are three advantages to using simulation to evaluate TCP's performance. Firstly, because the network is simulated in its entirety, only a single workstation is needed to run the experiment, and rare or complex networks (such as satellite networks) can be investigated without requiring access to these technologies. Secondly, a wide variety of scenarios can be investigated relatively quickly, allowing general conclusions about performance to be drawn [50]. Lastly, simulation allows researchers the ability to easily investigate new TCP/IP mechanisms (i.e. new algorithms, routing strategies, etc.) or verify other researcher's results.

Simulation seems to be the ideal solution to researching problems in network communication, but it should always be noted that simulators can only be used to point out problems in a TCP/IP protocol stack [3], or show that a new mechanism is promising. Because simulation adds a level of abstraction to any experiment, the conclusions drawn from the simulation of TCP/IP stack models might not accurately reflect the real world implementations. Also non-network elements such as the operating system and network cards can influence behaviour, and assumptions made in the implementation of the simulator will be reflected in the results.

4.2 The Simulator

There are various simulators that are being used by researchers. Two examples are *x-sim* [12] and *ns-2* [41]. The *ns-2* simulator was chosen because of the functionality of the program and its wide use and acceptance by the research community[1].

The *ns-2* simulator is part of the Virtual InterNetwork Testbed (VINT) project, which is based on the *ns* simulator built by the Lawrence Berkeley National Laboratory (LBNL). *ns* was in turn based upon the REAL [29] simulator. University of Southern California's Information Sciences Institute (USC/ISI) currently maintains *ns-2* and for the sake of simplicity will simply referred to as the *ns* simulator.

The *ns* simulator is a discrete event simulator for researching packet networks. It models a network as events, which happen in an instant of virtual time, but can take an arbitrary amount of real time. The *ns* simulator does not use real TCP/IP implementations, and therefore does not reproduce their exact results. Rather, it can be used to investigate the underlying TCP mechanisms.

The *ns* simulator models various aspects of a network, which can be divided into four categories.

- Traffic models and applications such as the internet, constant bit rate and FTP.
- Transport Protocols models such as TCP and UDP.
- Routing and queuing strategies.
- physical media such as wired (point-to-point, LAN's etc.) or wireless networks.

The *ns* simulator is object oriented (written in C++), with a scripting front end (OTcl). This two-language approach allows *ns* to have the speed and power of a systems language such as C++, while having the flexibility of a scripting language to model ever changing network topologies.

4.2.1 Transport Protocol Implementations

The first widely available TCP/IP stack was the 4.2BSD in 1983 [56]. The BSD refers to *Berkeley Software Distribution*, as the University of Berkeley distributed it, although many people contributed. Many TCP/IP implementations have been based on this original stack, and are available in the *ns* simulator.

---

[1] http://www.isi.edu/nsnam/ns/ns-research.html

Tahoe The 4.3BSD release in 1986, which implemented some performance improvements for TCP, was followed by the Tahoe implementation in 1988. This release implemented new algorithms such as Slow Start, Congestion Avoidance and Fast Retransmit [23]. These changes were aimed at controlling network congestion while maintaining throughput.

Reno The 4.3BSD Reno release implemented Fast Recovery (see Section 2.2.3), preventing the channel from being cleared after a packet drop and forcing TCP/IP to go into slow start again. Other additions were TCP header prediction, SLIP header compression and routing table changes. The *ns* simulator also uses a model named NewReno that uses a slightly modified fast recovery mechanism.

Sack Selective Acknowledgement has been widely implemented in many TCP/IP stacks, including Linux and Windows. (see section 2.2.3).

Fack A model implementing Forward Acknowledgement (Fack) (see Section 2.2.3).

Vegas Vegas is a modification to the TCP/IP protocol stack proposed in [13]. It does not use new mechanisms, but rather implements a modified retransmission and congestion mechanism. It has not been widely accepted, even though it has shown promise.

The various TCP/IP stacks refer to different congestion control mechanisms used. In practice, there would also be differences in implementation, but one of the assumptions made during simulation is that implementation does not significantly impact performance.

4.3 Parallel Processing

One of the strengths of simulation is the power to relatively quickly and thoroughly investigates the general performance of TCP/IP over a broad range of channel parameters. The simulator was found to be very processor-intensive however, leading to experiments that would run for weeks given only a simple network model.

The time taken by the *ns* simulator to complete an experiment is proportional to the time that the network is to be simulated for. If a network is to be simulated for a 10 min TCP connection, ns might complete the experiment in 1 minute. As processor power is increased the time taken to complete a simulation does not improve dramatically. Given the large number of simulations that need to be completed to investigate only a small range of channel parameters, experiments will take an unacceptable long time to complete.

Under *ns* each simulation is totally independent. This means that experiments can be spread over a number of computers, and if post-processing is performed on these machines as well, very little data needs to be transferred to a central point at the end of an experiment. This makes network simulations an ideal problem for cluster computing. A Mosix cluster[2] was available, but given the complex nature of the *ns* architecture it was not possible to use this. Instead, scripts were used that broke the experiment up among the available computers, loaded the necessary software (necessary only for any files that might have changed), executed the experiment and returned data back to a central computer (which can be anywhere on the network).

An experiment that consists of a hundred thousand simulations, can now be broken into 40 completely separate experiments consisting of 2500 simulations. This led to a dramatic increase in the speed with which large scale experiments could be completed in reasonable amounts of time (As the computers were only available over weekends, an experiment had to be concluded in two to three days).

4.4 TCP/IP Performance Parameters

The performance of a TCP/IP stack is dependent on a number of external and internal factors, from rain fading to the number of packets initially broadcast. Some of the variables that are considered are,

4.4.1 External Parameters

Bit Error Rate (BER) Uniform plus temporary effects such as rain fading and jamming. This thesis investigates the uniform error rates, ranging between an error rate of one is ten thousand (10^{-4} to one in a hundred million (10^{-8}). Temporary disturbances, such as rain fading, can have a very negative effect on the performance, but these will not be addressed in this thesis beyond the fact that it can be viewed as lowering the average error rate for a limited time.

Delay Dependent on the speed of light, but also hardware delay, ranging between 0 ms to 300 ms. Hardware delay is especially important in LEO satellites, where the physical delay might only be a few milliseconds, but the hardware delay 20 to 40ms.

Bandwidth Dependent on hardware. Values of between 1 kbps to 10 Mbps are investigated.

[2]http://www.mosix.org/

4.4.2 Internal Parameters

Packet Size An internal TCP/IP variable. The packet error rate (PER) can be found by multiplying the value for the BER with the packet size. The PER indicates TCP/IP's perceived error rate. Packet sizes of between 10 and 1500 bytes are investigated.

cwnd This value is the size of the initial window, and can impact on how fast a TCP/IP connection will be able to fill the pipe.

Window Size Sets the window size used by TCP, and plays the role that of the receivers advertised window in real world TCP.

The possible bandwidth levels usually found in satellite communication varies between 9,600 kbps for small LEO satellites to 100 Mbps connections for large communication satellites. Delay for satellite links vary from 2 ms up to 250 ms. The bit error rate for a satellite connection can be expected to vary from 10^{-10} up to 10^{-4}.

4.5 Simulation Models

The *ns* simulator uses the scripting language *tcl* to build models of the networks to be simulated. These models can be very detailed, since it allows you to use various link layer protocols (both traditional and wireless), various queuing mechanisms, different error models, and a host of other features.

Initially two models were evaluated. The first model consists of two nodes connected to each other representing the satellite link, and then nodes connected to these nodes representing the satellite and ground station computers. The second model was that of two nodes, connected using a wireless link protocol.

These have been abandoned because the added complexity would have necessitated the investigation of issues outside the scope of this thesis (the queuing model used by the satellite link in the first case, and the wireless link protocol in the second).

A simpler model was finally chosen however given that the underlying behaviour of TCP was of interest. TCP sees the world through the blinkered eyes of bandwidth, delays and dropped or erroneous packets. A simpler model of two nodes connected over a single link is therefore seen as adequate.

4.6 Simulation Results

In this section a number of experiments that were conducted are discussed. The goal of these experiments was to investigate both the bahaviour of TCP/IP over a broad range of network parameters, as well as to illustrate the usefullness of the tools that were built.

4.6.1 TCP Performance

Figure 4.1 (d) illustrates the general form that TCP/IP performance takes as the bandwidth and delay is varied. A TCP stack implementing forward acknowledgement, using a packet size of 1000 and experiencing a bit error rate of 10^{-6} is shown. As is illustrated in Figure 4.1 (d), TCP/IP performs very well at low delays or low bandwidth, but performance falls rapidly as the DBP increases.

Figures 4.1 (a),(b) and (c) show the same data as Figure (d), but at a 90 degree angle, and for different Bandwidths. The colour is an indication of the relative performance levels that can be expected.

4.6.2 Congestion Control Comparison

The performance of various congestion control algorithms can be investigated by comparing their performance to a baseline TCP. Tahoe is chosen as this baseline because it is the oldest and simplest of the TCP/IP stacks. In Figure 4.2 Tahoe is compared with Reno, Newreno and Sack by subtracting the performance graph (such as the one seen in Figure 4.1) of Tahoe from that of the stack to which is to be compared. The experiment is conducted with a BER of 10^{-6} and a packet size of 1000. It can be seen that the newer stacks (Reno, Newreno and Sack) improve the performance as the bandwidth and delay is increased.

Of interest is the fact that there is such a large difference between Newreno and Reno (See the top two graphs in Figure 4.1). It illustrates well the point that small changes in TCP/IP can have dramatic consequences. It should also be noted that for low bandwidths, Tahoe performs very well and sometimes even better then the newer stacks.

4.6.3 TCP Performance as the Error Rate is varied

Figure 4.3 depicts the improvement in channel utilization of a TCP stack as the average transmission error rate is decreased. Once again the general form of TCP performance as

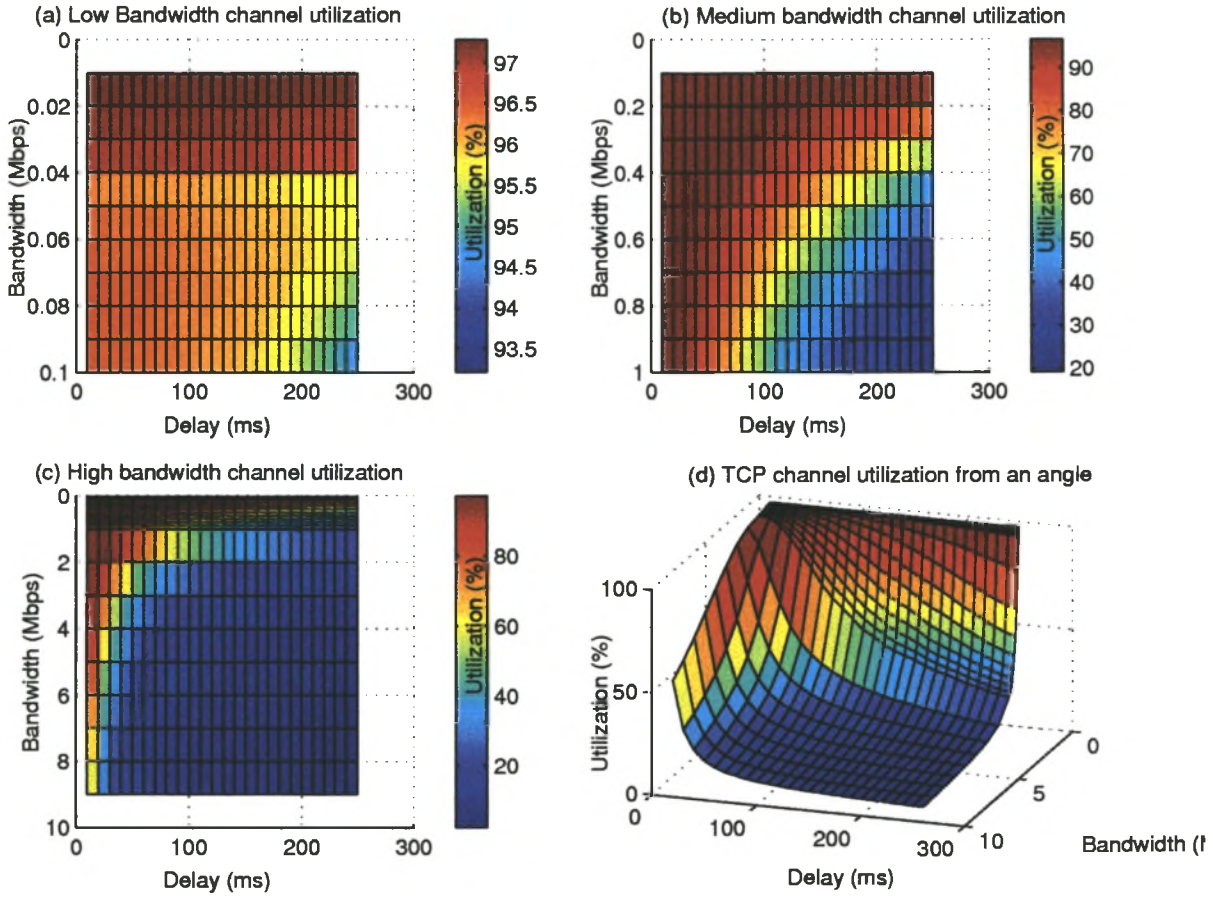

Figure 4.1: *TCP channel utilization with TCP implementing forward acknowledgement, using a packet size of 1000 and experiencing a bit error rate of* 10^{-6} *(a) Low Bandwidth 10 to 100 kbps (b)Medium Bandwidth 100 kbps to 1 Mbps (c) All bandwidths tested 10 kbps to 9 Mbps (d) Figure (c) viewed from an angle*

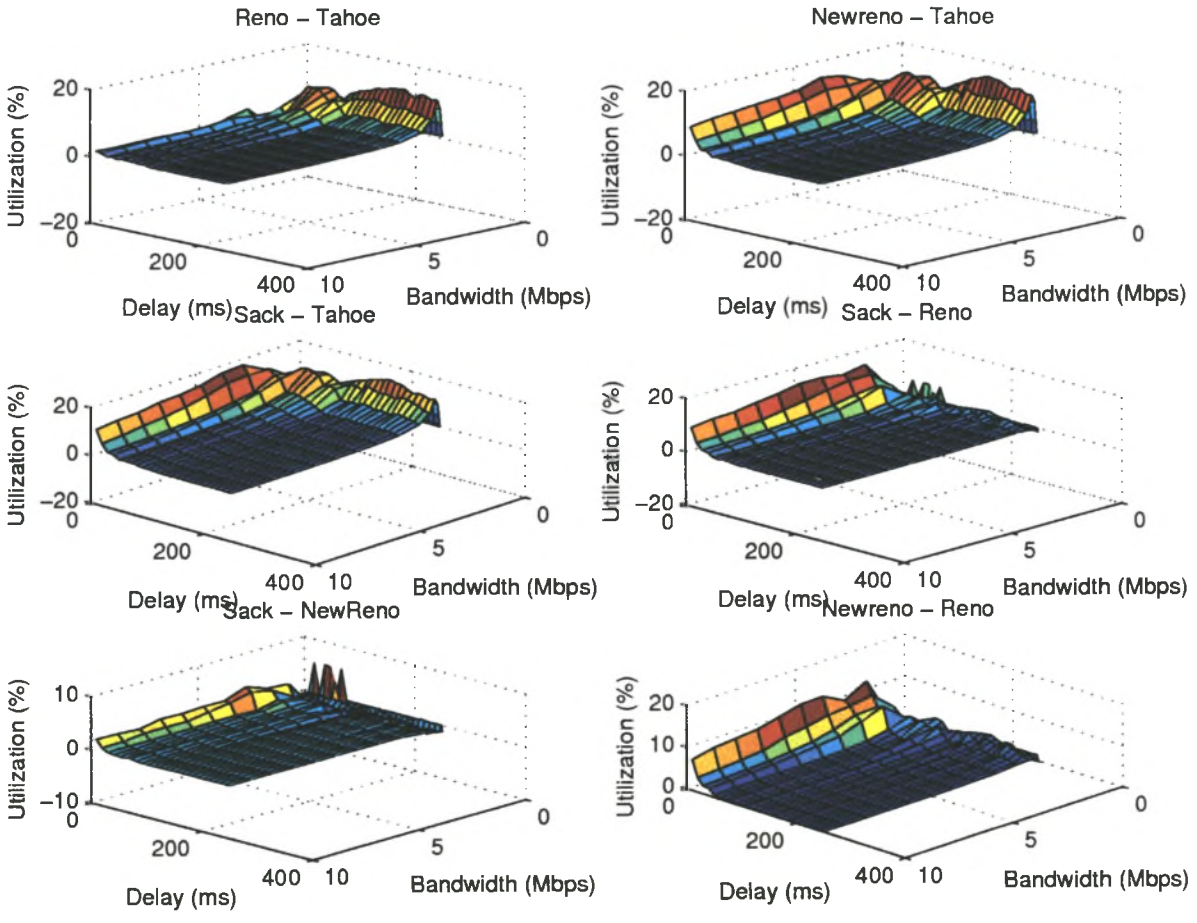

Figure 4.2: *Congestion Control Mechanism Comparison*

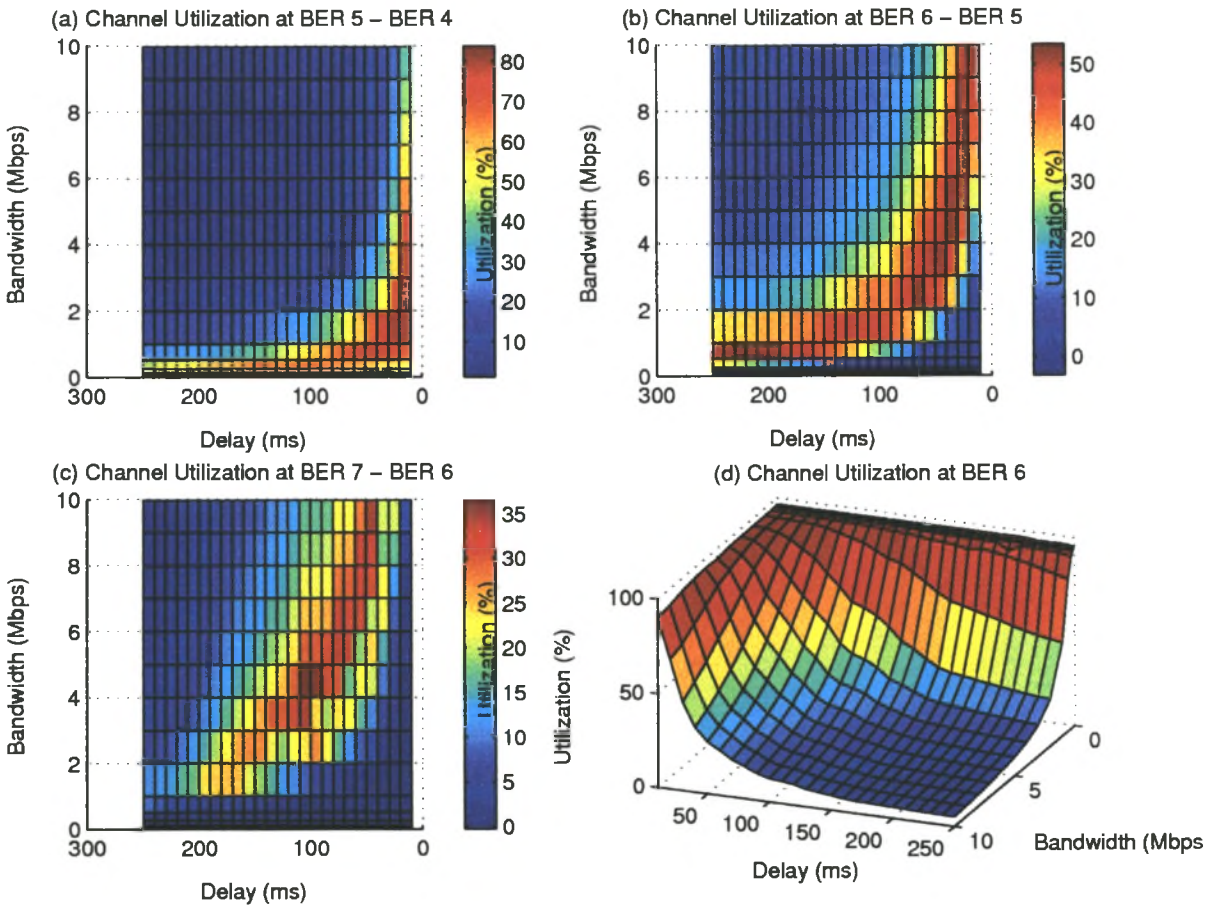

Figure 4.3: *TCP channel utilization improvements as transmission error rates are decreased from Bit error rate (BER)* 10^{-4} *to* 10^{-7}.

bandwidth and delay is varied is illustrated by Figure 4.3(d). Figures 4.3(a),(b) and (c) are obtained by subtracting the channel utilization data of TCP connection at a higher transmission error rate from that of a TCP connection with a lower transmission error rate. A TCP stack implementing selective acknowledgment and using a packet size of 1000 is shown.

Figure 4.3(a) illustrates channel utilization gains as transmission errors decrease from 10^{-4} to 10^{-5}, Figure 4.3(b) illustrates channel utilization gains as transmission errors decrease from 10^{-5} to 10^{-6} and Figure 4.3(c) illustrates channel utilization gains as transmission errors decrease from 10^{-6} to 10^{-7}.

These graphs illustrate that the bandwidth utilization for high DBP links improves as the BER is decreased. The less obvious result is that low delay and bandwidth links do not necessarily have improved bandwidth utilization with a lower BER. The conclusion can thus be drawn that it is unnecessary to reduce the average bit error rate of links beyond a certain level, after which there will be little if any gain in bandwidth utilization for a given range of delays.

4.6.4 Packet Size

The effect of size of the data segment transmitted by TCP/IP on performance is illustrated in Figure 4.4. This experiment was conducted with a delay of 40ms using the FACK stack, at four different bandwidths (Figures (a)1000 kbps, (b)256 kbps, (c)54 kbps, and (d)18 kbps). The header size of TCP/IP imposes an overhead of 40 bytes per packet, and this is taken into consideration.

A number of conclusions can be drawn from these graphs,

- As the data segment size is decreased, the overhead involved increases. This is illustrated in Figure 1.1(a) and is confirmed here.
- Channel utilization starts to decrease rapidly at some point as the number of transmission error increase. It can be seen however that by using a lower data segment size performance can be increased up to levels comparable with links with much less errors. The reason for this is that TCP is a packet based protocol, and by lowering the packet size the packet error rate is decreased.
- The packet size is of no use in improving performance over links with error rates below a certain level. In these cases, the larger the data segment the better.

These graphs are usefull, in that they give a good indication as to the optimal packet size for a given TCP/IP connection.

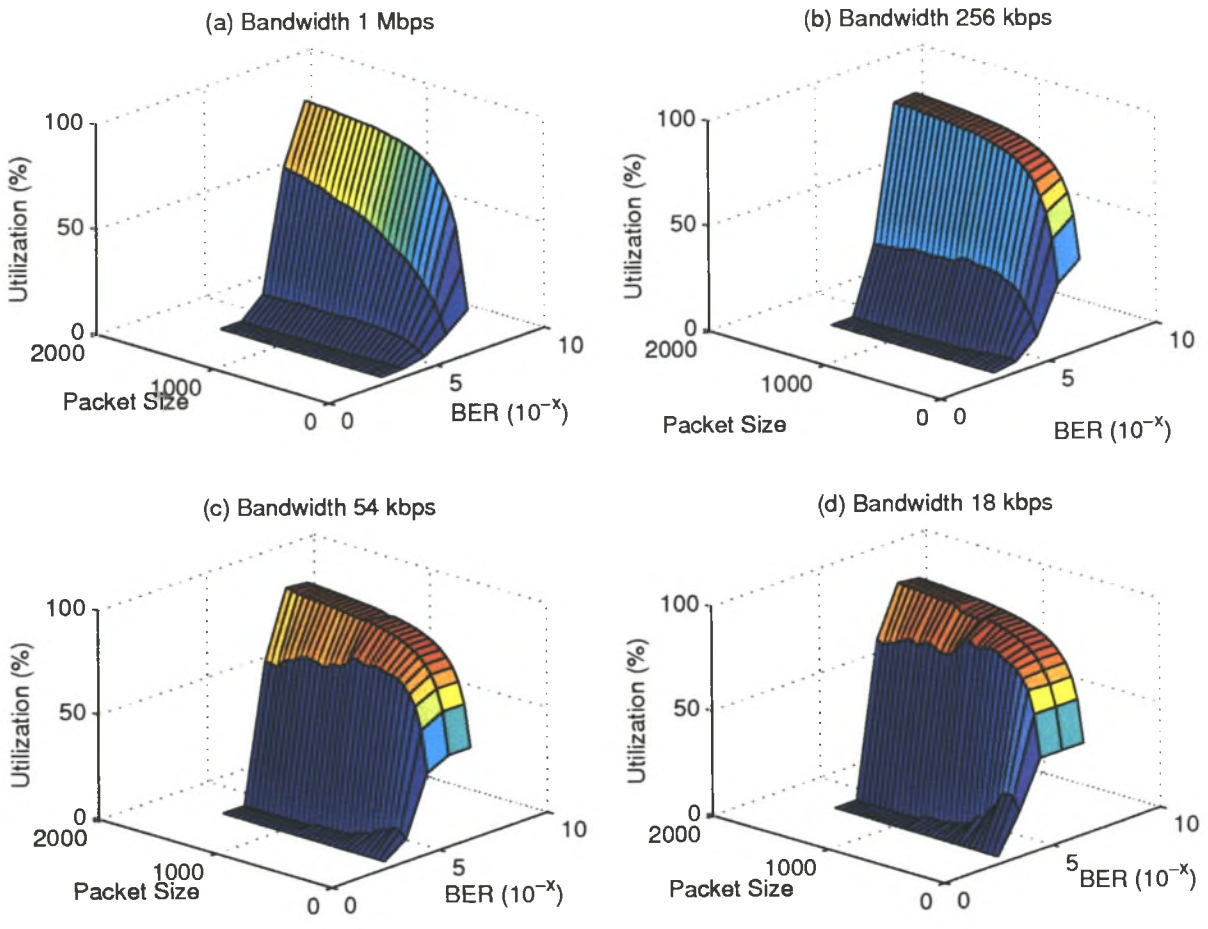

Figure 4.4: *TCP channel utilization i.t.o Bit Error Rate (BER) and Packet Size at various bandwidths. TCP stack implementing Forward acknowledgement experiencing a delay of 40 ms.*

4.6.5 The Relationship between Bandwidth and BER

As the bandwidth is increased the BER needs to be increased to avoid a sharp decline in throughput performance.

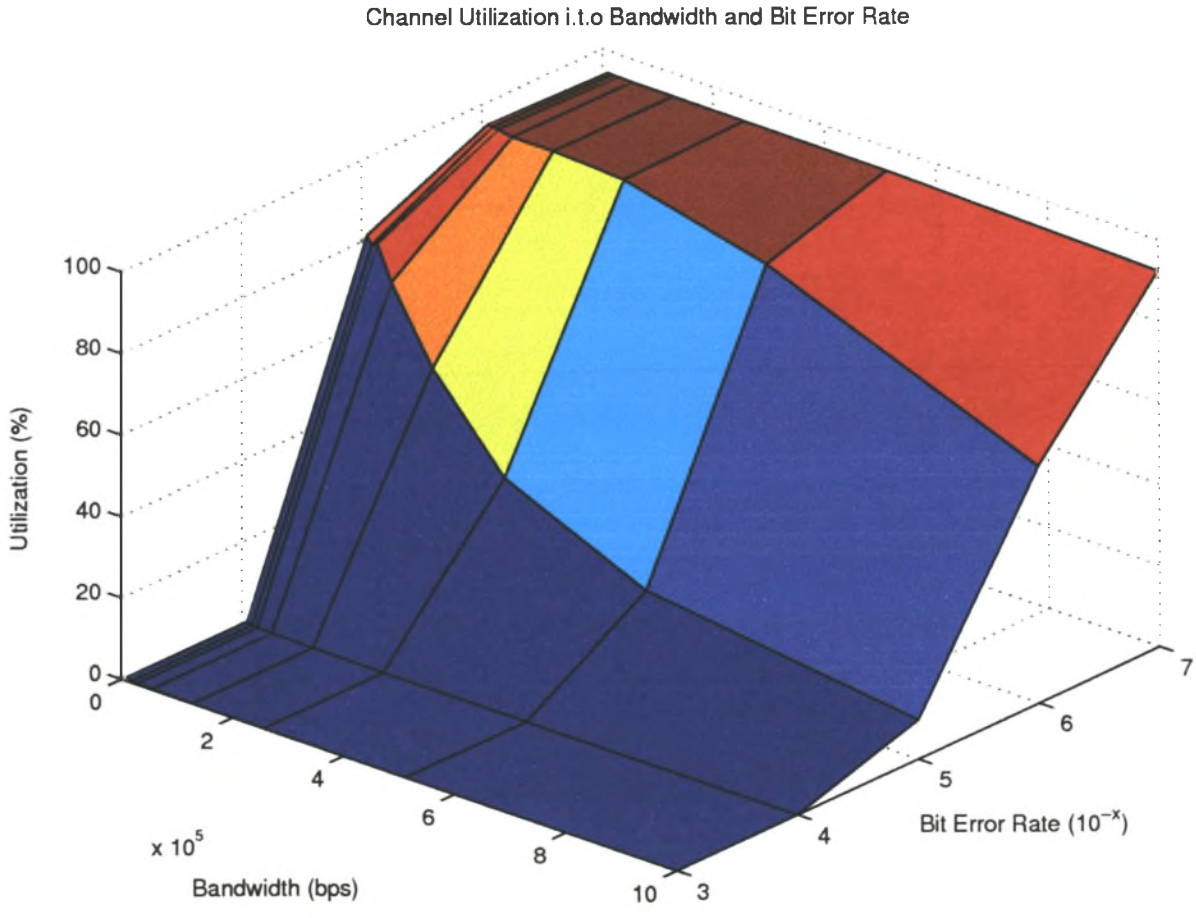

Figure 4.5: *Performance comparison when looking at Bandwidth and Bit Error Rate*

Figure 4.5 illustrates this concept very clearly. The performance degradation encountered as bandwidth increases can clearly be noticed in the figure.

4.7 Conclusions and Future Work

This chapter discussed the simulation program used to investigate TCP behaviour, as well as the various components that affect the experiments conducted using the simulator. The use of parallel processing and the improvements in experiment time was also discussed. This chapter then looked at some external and internal parameters that affect the performance of TCP. Finally the effect of some of these parameters was investigated and results presented.

The *ns* Simulator can be a powerful tool when investigating TCP/IP behaviour. Care must be taken however to not lose sight of what is being investigated, as too much detail can unnecessarily complicate the results. The correct level of abstraction is important.

A number of observations were made on the results of simulations carried out. Fundamentally TCP is a terrestrial protocol. TCP that has not been enhanced for high-end performance does not perform well over high DBP links, which many satellites in MEO and HEO orbits have. LEO satellites have networks parameters that are very similar to terrestrial networks however. Consequently the use of TCP/IP can be recommended for use in LEO satellites as a communications protocol. MEO and HEO satellites, depending on their bandwidth, can also use TCP and achieve high levels of utilization. As the delay and bandwidth increase however utilization will decrease dramatically.

It is seen that the bit error rate of a link has a significant impact on performance. High DBP TCP links experience a significant increase in performance as the BER is increased. The performance of TCP only increases significantly up to a point however, after which any further decrease in the number of bit errors have little impact.

It was further observed that Packet Size is an effective tool in increasing the performance of links that are on the borderline of performance.

Some recommendation can be made as regards future work with the simulator.

- The use of parallel processing is definitely recommended if a large number of parameters are to be investigated. This brings in a number of complexities, of which the large number of data points can be the most frustrating. If the TCP/IP stack is too be investigated further in this manner, the implementation of an integrated experimental front end to manage this complexity is recommended.
- A large number of computers are needed to effectively use parallel processing as a research tool. Access to the required number of computers, for long enough periods of time, was found to be problematic however. The need to properly manage available computer time is also an issue that is recommended for future study.

A number of recommendations can also be made as regards to future simulation experiments.

- Investigating the various mechanisms discussed in Chapter 3. Parallel simulation would be an ideal solution for discovering the right combination of TCP/IP parameters for a given network channel, as the optimal values can be found through automated experiments. Because of the large number of parameters and variables, the number of data points generated would be difficult to analyze. The use of new

metrics, such as the equivalence metric proposed in the next chapter, could hopefully help reduce this data burden.

- The effect of more complex network flows on the performance of TCP needs to be investigated.
- The modeling of more complex network topologies, especially terrestrial to space network routing, is of interest. Previous studies have mostly focused on either space based or terrestrial networks exclusively.

Chapter 5

A New Performance Metric for TCP

5.1 Introduction

Figure 5.1 (a) illustrates the case of a TCP stack implementing forward acknowledgement congestion control, using a packet size of 1500 and experiencing a bit error rate of 10^{-6}. Interest in reducing the number of channel parameters that was being investigated (bandwidth, delay and BER) led to the formulation of an equation that would combine parameters as they relate to TCP channel utilization. Reducing the number of parameters is usefull because it simplifies the performance comparison between different TCP/IP mechanisms over a broad range of network conditions. An equation describing whether the channel utilization of a TCP connection for all bandwidth and delays will be above or below a certain performance level was found, and is called the equivalence metric. The surface represented by the formula

$$log((Bandwidth \times Delay)^{-1}) + Constant$$

that is illustrated in Figure 5.1(b), is compared to the channel utilization data of a TCP connection (as is seen in Figures 4.1(c) and Figure 5.1(a)). This formula is called the equivalence formula.

The equivalence level is found by comparing the surface areas of the TCP channel utilization at a given percentage, and the surface represented by the equivalence formula.

The equivalence level, along with the equivalence formula, allows us to describe the channel utilization of TCP independent of bandwidth and delay. It also allows us to quantitively compare the channel utilization of different TCP/IP mechanisms. This is advantageous because the equivalence level can more accurately reflect TCP behaviour then single throughput measurements. The question arises whether the equivalence formula

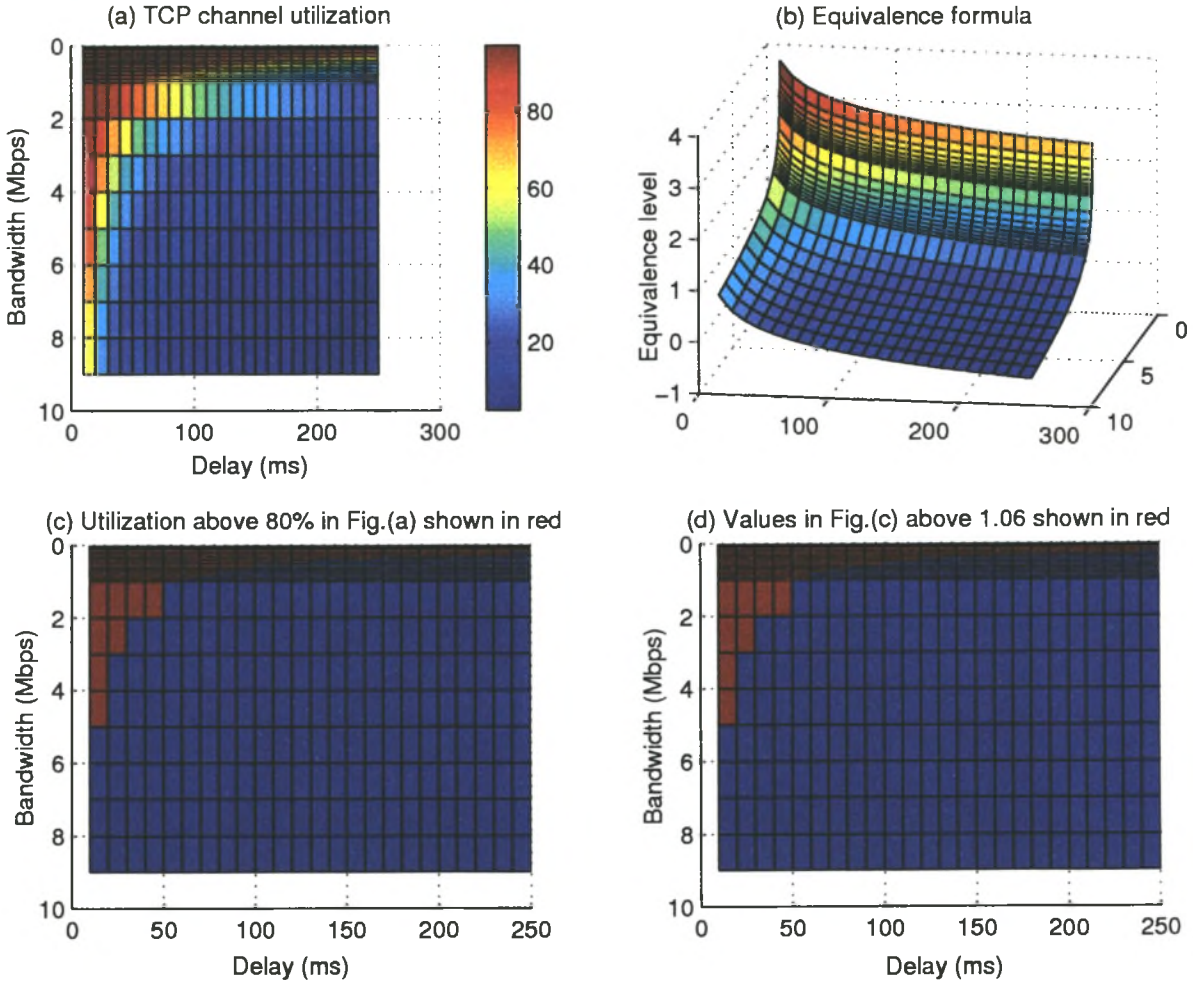

Figure 5.1: *(a)TCP channel utilization; (b)Equivalence formula values; (c)TCP performance above 80% in Figure (a) is indicated in red. Performance below 80% by blue; (d) Values above 1.06 in Figure (b) are indicated in red. Values below 1.06 by blue.*

also accurately represents different congestion control mechanisms, alternative channel utilization levels and different transmission error rates. These issues are addressed in the following sections.

5.2 Congestion Control

The equivalence level of the different congestion control mechanisms as represented by the Tahoe, Reno, Newreno and Vegas stack models are illustrated in Figure 5.2 and Figure 5.3. In Figures 5.2(a) and (c) and Figures 5.3(a) and (c) the surface area coloured white represents areas where channel utilization for the respective mechanisms are above 80%. In Figures 5.2(b) and (d) and Figures 5.3 (b) and (d) the surface areas in white represent points in the equivalence formula above the equivalence level given for each figure.

It can be clearly seen that the equivalence formula can accurately represent the channel utilization for different congestion control mechanisms. Small difference can be seen however (See Figure 5.2 (c) and (d) for a good example). The extent of these differences is investigated further later in this section.

Note also that the equivalence level *decreases* as channel utilization *improves*. This is counterintuitive, and would be solved by not inverting the delay, bandwidth product. The equivalence formula would however then also be inverted, and would not reflect the shape of the TCP channel utilization graphs. Inverting the delay, bandwidth product is thus purely a matter of preference.

5.3 Channel Utilization Levels

The equivalence level at different channel utilization levels is illustrated in Figure 5.4. The TCP stack implements the forward acknowledgement congestion control mechanism, uses a packet size of 1500 and experiences a bit error rate of 10^{-6}. The equivalence level is determined at channel utilization levels of 85% in Figures 5.4(a) and (b) and 90% in Figures 5.4(c) and (d). The surface areas in white represent channel utilization above the respective levels. It can be seen that the equivalence formula does give an accurate representation of channel utilization in Figures 5.4(c) and (d) at their respective equivalence levels.

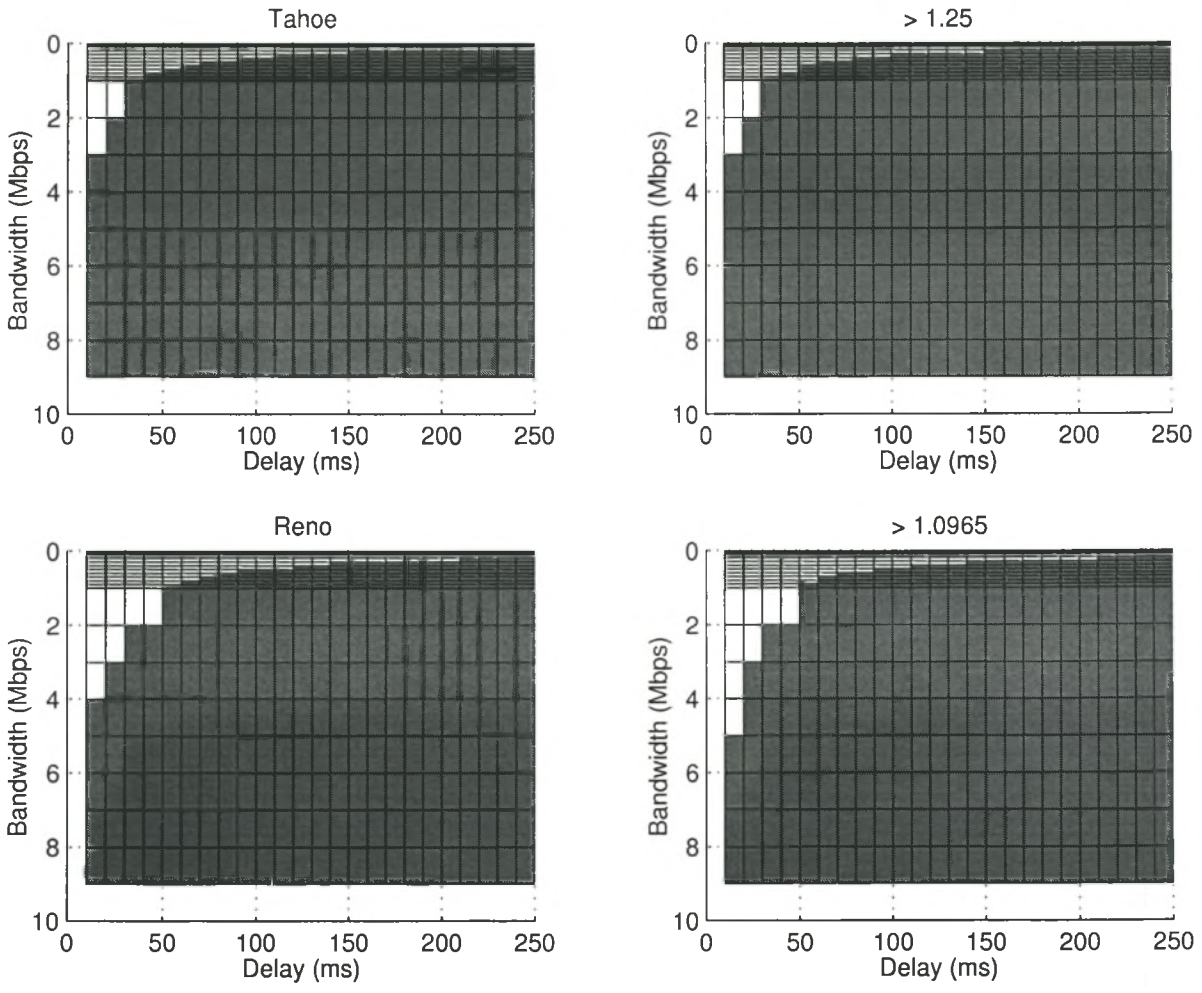

Figure 5.2: *(a) and (c) Tahoe and Reno congestion control mechanisms with points with channel utilization above 80% indicated in white, (b)and (d) Points above equivalence level 1.25 and 1.0965 respectively indicated in white.*

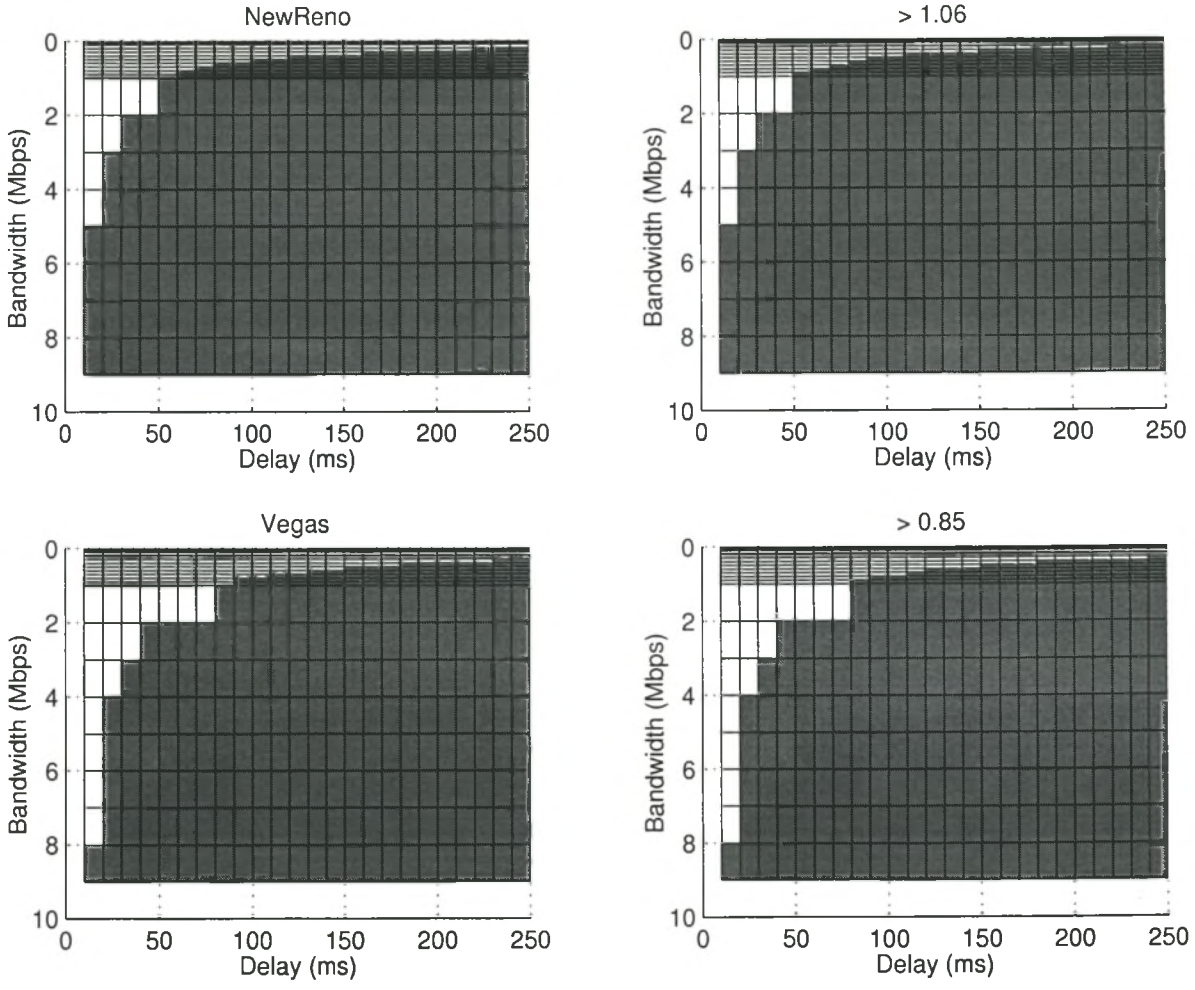

Figure 5.3: *(a) and (c) Newreno and Vegas congestion control mechanisms with points with channel utilization above 80% indicated in white, (b)and (d) Points above equivalence level 1.06 and 0.85 respectively indicated in white.*

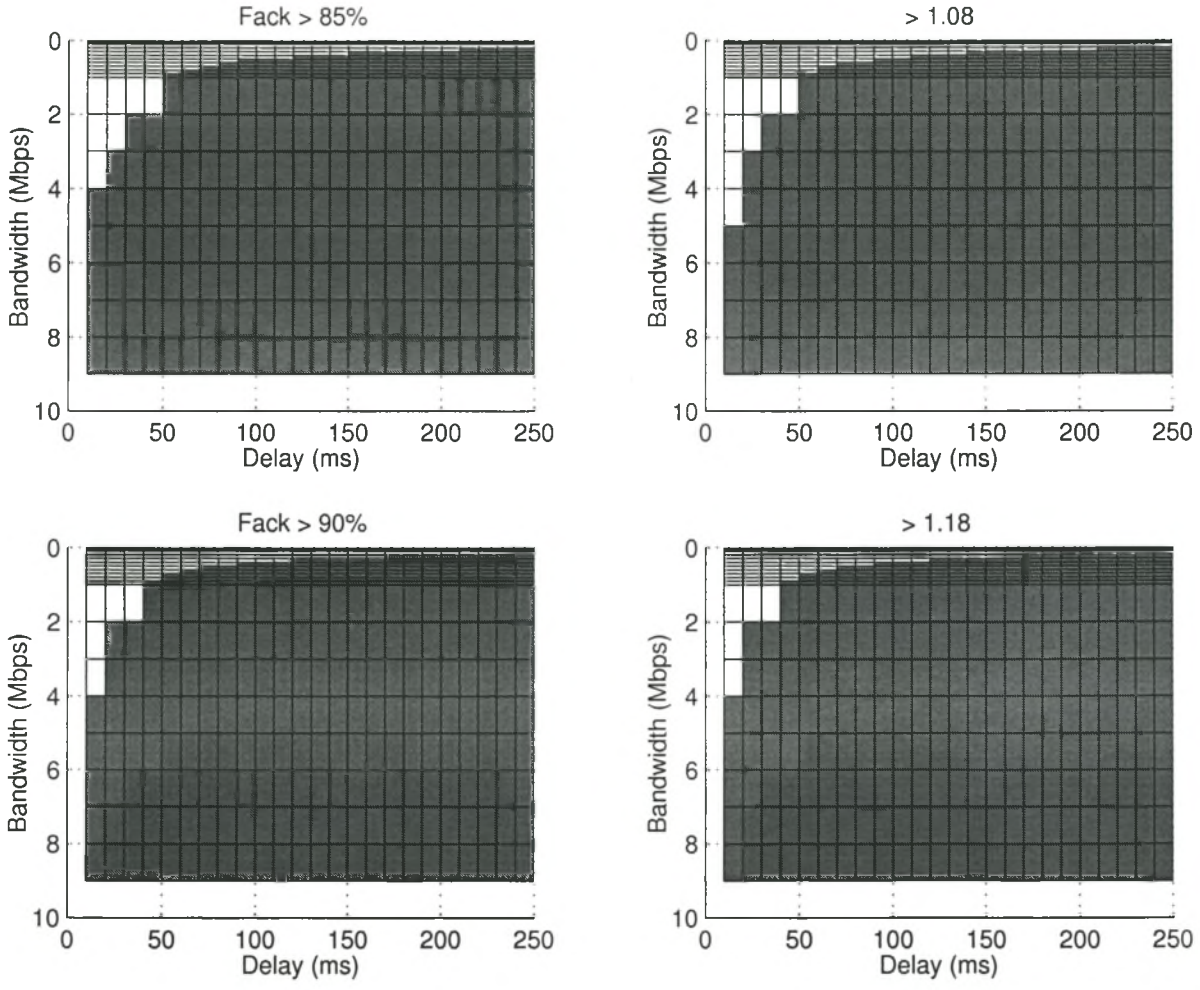

Figure 5.4: *(a) and (c) Channel Utilization above 85% and 90% respectively indicated in white, (b)and (d) Points above equivalence level 1.06 and 1.18 respectively indicated in white.*

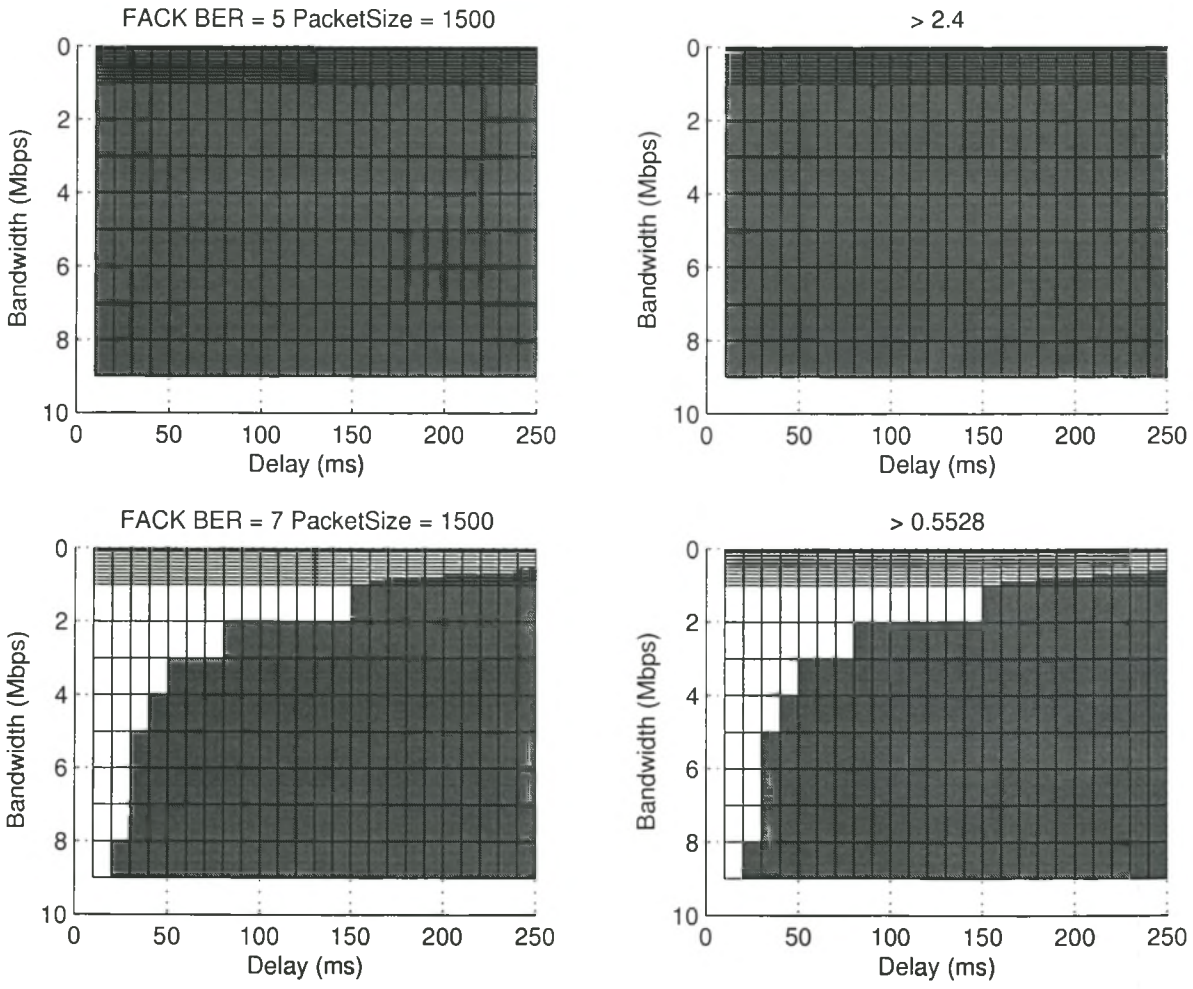

Figure 5.5: *Fack Utilization above 80% is shown in these graphs. Packet size is 1500.(a) and (c) Performance at BER 5; (b)and (d) Performance at BER 7*

5.4 Transmission Error Rates

The equivalence level for the TCP stacks implementing the forward acknowledgement congestion control mechanism, using a packet size of 1500 and experiencing an average bit error rate of $10^{-5} and 10^{-7}$ is illustrated by Figures 5.5 (a) and (c) respectively. The surface areas in white represent channel utilization above 80% in Figures 5.5(a) and (c). Figures 5.5(b) and (d) represents the area above the equivalence level of the equivalence formula.

(Note that the channel utilization for the case where the average bit error rate is 10^{-5} is not below 80% at all points. The points where it is above 80% is merely at very low bandwidths and are therefore not clearly visible on these figures, which look at a very broad range of values.)

Once again the equivalence formula, combined with the equivalence level, seem to be a very good representation of the observed TCP channel utilization.

5.5 Differences between the Equivalence Metric and Measured Results

It has been noted that the equivalence level, combined with the equivalence formula, has not been a 100% accurate reflection of channel utilization in all cases. Small differences, that are points that are above or below the level of channel utilization indicated by the Equivalence metric, were noticed. The extent of those differences was investigated.

Table 5.1 shows the equivalence values for 5 congestion control mechanisms, at three different bit error rates and at 4 different performance levels. There were 675 data points used when calculating each value.

Along with the respective equivalence level, the number of points that differ between the TCP channel utilization represented by the equivalence level and that as measured by the simulator was counted. The maximum as well as the average difference between the measured channel utilization and that represented by the equivalence metric was also recorded.

The largest number of differences was noticed at 90% utilization, although the average of these differences was relatively small (below 3%) and are therefore not significant. The largest differences were noted at a transmission error rate of 10^{-5}) (Up to 32% for some points, and the average difference for these points was also relatively high (around 10% for some). The number of differences seen are relatively low however (Around 20 points out

of 675, or about 3% of the measured points. This large difference is investigated further in the next section.

The average number of measured points that differs with the equivalence metric is relatively small (less then 1.5 % of the total points) and the average difference of these points where they do differ is also small (on average less than 2%). The Equivalence metric can thus be seen as a good representation of the TCP channel utilization.

5.6 Equivalence level anomalies

In the previous section it was noted that there are sometime small differences between the measured results and the equivalence metric used to represent the results. Sometimes these differences were significant.

Figures 5.6, 5.7 and 5.8 illustrate the use of the equivalence metric for a TCP stack implementing the forward acknowledgement congestion control mechanism, using a packet size of 500 and experiencing a bit error rate of 10^{-5}, 10^{-6} and 10^{-7} respectively. The equivalence level is calculated for a channel utilization of 80%.

The bottom graph in each Figure shows the position and size of the difference between the measured values and that represented by the equivalence metric. The differences illustrated by Figures 5.7 and 5.8 are relatively small and there are very few. Although differences are at the edge, they are spread out over a wide area. The differences in these two cases are not significant. In Figure 5.6 the largest differences can be seen, but it is also seen that these differences are limited to a very small region at the edge of the equivalence metric. It is still a serious anomaly.

This experiment has shown that anomalies do exist, but that they appear to be localized and found only where channel utilization is at the extremes of performance. It is a problem that must still be guarded against.

5.7 Conclusions and Future Work

This chapter introduced the concept of a new performance metric for TCP channel utilization. The chapter went on to show that the performance metric applied to a range of congestion control mechanisms, at various levels of channel utilization and at various bit error rates. Small differences between the equivalence metric and measured results were noticed, and investigated further. These differences were not found to be significant however, except in one case where a significant localized anomaly was detected.

Table 5.1: *Equivalence values showing stack performance at three Bit Error Rates. Tests were conducted with a Packet Size of 500.* ***D/M/A*** *stand for; (D) The number of measured points that differ from that given by the equivalence metric, (M) The maximum size (in percentage) that points differ by and (A) The average size (in percentage) of the difference. 675 Data points are compared. A Constant of 6 is used.*

Stack	BER 5	D/M/A	BER 6	D/M/A	BER 7	D/M/A
Tahoe						
50%	1.6198	17/23.12/10.0	0.95078	2/1.35/0.83	0.39794	12/6.65/3.07
60%	1.7447	19/30.8/10.17	1.0555	1/1.45/1.45	0.49485	11/6.54/2.75
70%	1.9208	15/30.6/9.96	1.1804	0/0.00/0.00	0.6197	7/4.12/2.28
80%	2.2218	15/24.1/4.50	1.3565	11/3.04/1.26	0.7447	11/3.11/1.28
90%	3.7	0/0.00/0.00	1.9208	30/4.19/1.72	1.3768	31/1.22/0.48
Reno						
50%	1.5654	19/24.5/9.61	0.87942	2/0.34/0.18	0.37675	5/3.72/2.24
60%	1.7167	15/32.0/13.5	0.96657	3/2.31/1.12	0.44369	11/6.12/3.21
70%	1.8665	18/32.1/9.15	1.0757	3/0.99/0.68	0.5528	8/4.36/1.76
80%	2.1427	25/25.2/4.30	1.1804	2/0.86/0.58	0.6819	3/0.99/0.78
90%	3.7	0/0.00/0.00	1.4437	6/0.63/0.29	0.92081	90/4.44/1.67
NewReno						
50%	1.5171	15/15.1/6.77	0.86012	0/0.00/0.00	0.37675	5/9.52/3.27
60%	1.6498	13/25.1/10.1	0.95078	3/2.88/2.11	0.44369	11/19.52/4.11
70%	1.7959	17/28.7/8.75	1.0457	3/2.86/1.33	0.5528	5/29.5/7.44
80%	2.0000	15/26.0/5.90	1.1675	0/0.00/0.00	0.6576	5/39.5/9.14
90%	2.7447	14/5.89/2.79	1.4437	3/0.3/0.15	0.92081	40/1.27/0.20
Sack						
50%	1.5406	12/16.7/8.44	0.86646	0/0.00/0.00	0.37675	4/2.90/1.73
60%	1.6576	15/26.7/10.27	0.96657	1/1.15/1.15	0.44369	10/4.98/2.59
70%	1.7959	18/31.2/9.59	1.0555	1/0.21/0.21	0.5528	4/3.68/1.95
80%	1.9830	15/28.45/6.7	1.1675	0/0.00/0.00	0.6576	4/2.90/1.56
90%	2.4202	32/7.41/2.67	1.3979	48/2.47/1.42	0.92081	5/1.27/0.46
Fack						
50%	1.4949	14/15.1/6.25	0.84163	5/1.95/1.25	0.35654	5/4.56/2.47
60%	1.6234	13/20.42/6.50	0.94309	2/0.82/0.68	0.44369	8/3.89/1.93
70%	1.7694	13/23.24/8.15	1.0362	1/0.60/0.60	0.5228	9/5.70/2.57
80%	1.9508	16/21.53/4.8	1.1427	3/1.43/1.00	0.6198	15/5.30/2.10
90%	2.301	46/13.9/2.68	1.3768	7/0.78/0.39	0.89279	3/0.74/0.40

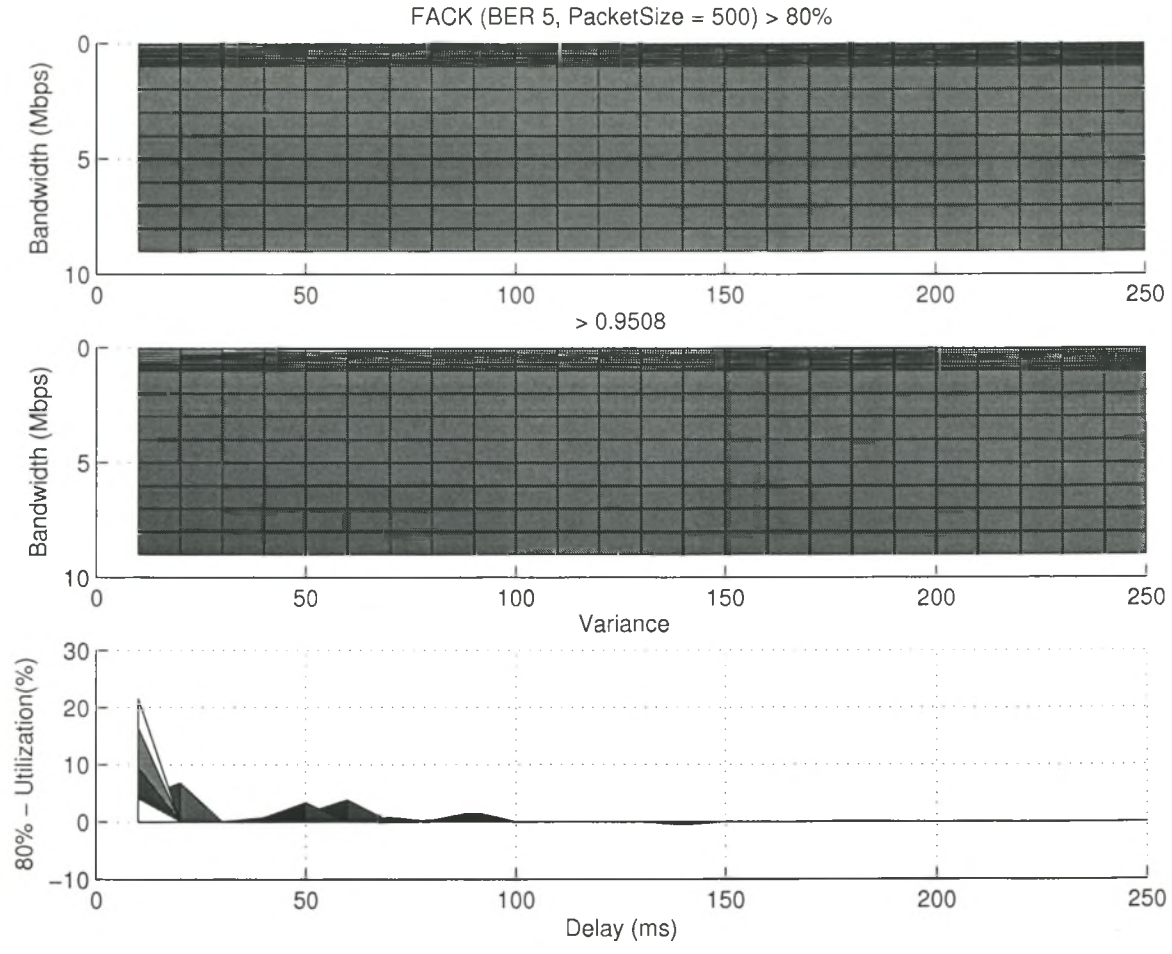

Figure 5.6: *TCP channel utilisation above 80% is illustrated in these graphs. Packetsize is 500. BER* 10^{-5} *(a) simulations data (b) the equivalence metric (c) size of difference at points that do not agree between (a) and (b)*

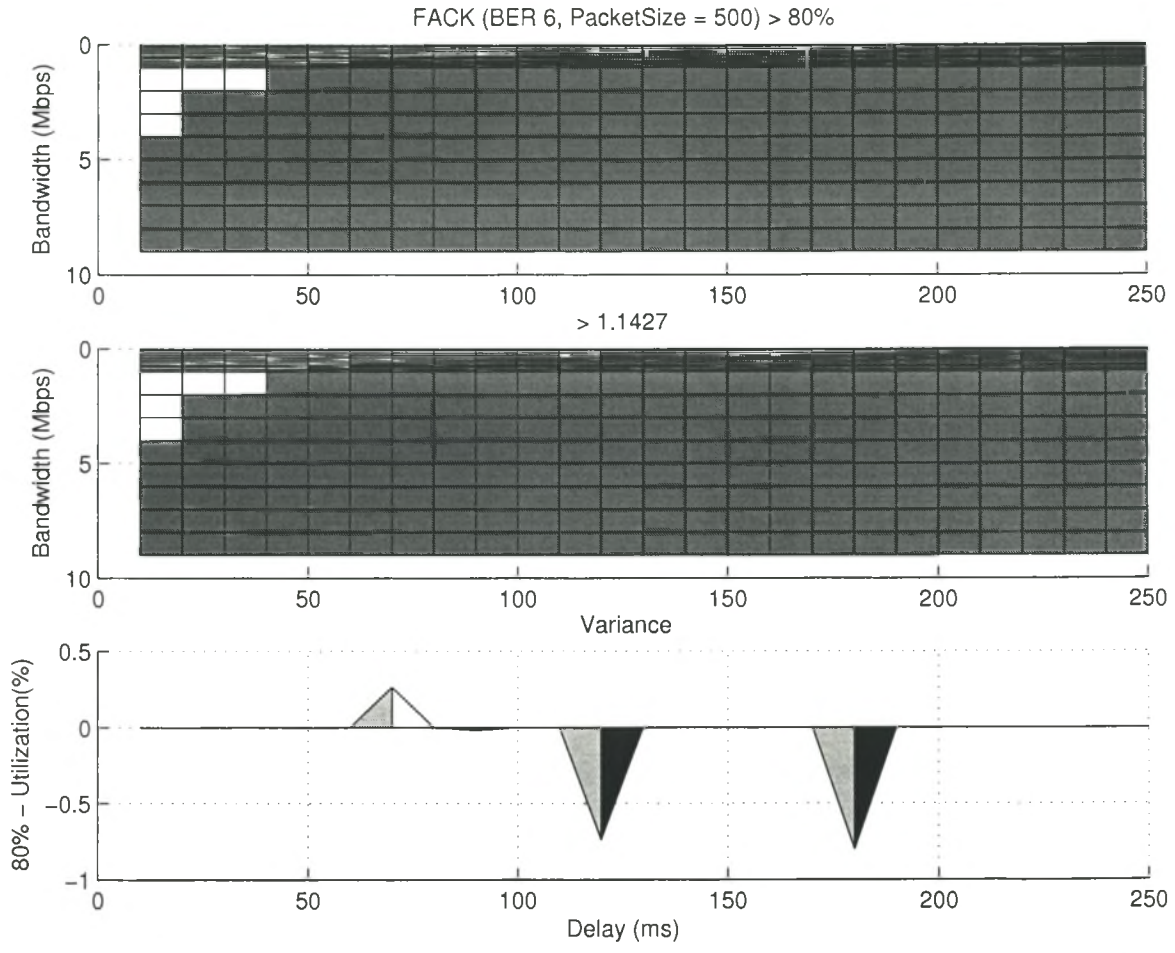

Figure 5.7: *TCP channel utilization above 80% is illustrated in these graphs. Packet size is 500. BER* 10^{-6} *(a) Simulations data (b) the equivalence level (c) Size of difference at points that do not agree between (a) and (b)*

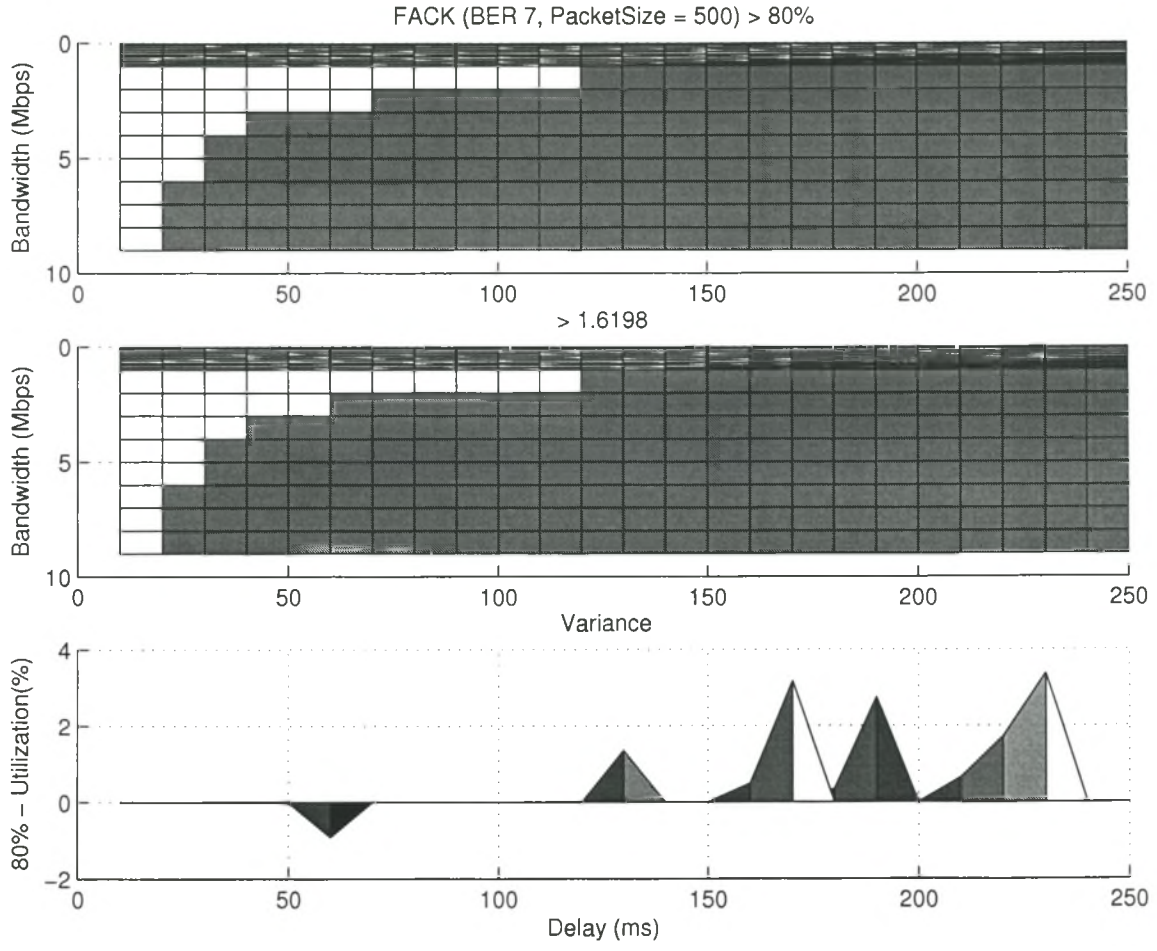

Figure 5.8: *TCP channel Utilization above 80% is illustrated in these graphs. Packet size is 500. BER* 10^{-7} *(a) Simulations data (b) the equivalence level (c) Size of difference at points that do not agree between (a) and (b)*

The equivalence metric can be used to describe the channel utilization of a TCP stack independent of the bandwidth and delay of a connection. It allows researchers to quantitively compare the channel utilization of different TCP/IP mechanisms. This is advantageous because the equivalence level can more accurately reflect TCP behaviour then single throughput measurements.

The equivalence metric could also be a useful tool for benchmarking. System designers interested in knowing whether the channel utilization of TCP will be sufficient over proposed communication links can use the equivalence metric, irrespective of the bandwidth or delay of their system.

The equivalence metric is by no means an exact representation of the performance of a TCP/IP stack, but by calculating the equivalence level of a TCP stack a large amount of performance data can be neatly summarized.

There are a number of factors that must be kept in mind when using the equivalence metric

1. The metric only represents the channel utilization with a fixed level of performance.
2. The metric can only indicate whether channel utilization will be above or below a certain level of performance. Using more than one equivalence level measurement can refine this however.
3. To calculate the equivalence level requires many discrete performance measurements. This is difficult to do in real world situations.
4. Equivalence levels at the extremes of TCP stack channel utilization could possibly not be a good representation of channel utilization. (At high bit error rates for example. See Section 5.6.)

Despite the limitations of the equivalence metric, it is a promising technique for measuring TCP performance.

A number of issues as regards the equivalence metric require further study.

The equivalence metric requires further testing not only to prove its validity, but also to show that it can be a useful metric in TCP research. To this end the equivalence metric should also be used to study known TCP performance issues and mechanisms. The results obtained by using the equivalence metric can then be compared to results obtained through other metrics. This would serve as a test to the validity of the metric, as well as proof of its usefulness to the research community.

The equivalence level is currently calculated by comparing the surface area above a certain performance level and comparing it to the surface area of the equivalence formula above the equivalance level. A method whereby the equivalence level can be determined with less measurement would be extremely advantageous. One possible method would be to only measure points along a narrow strip of bandwidths and delays. This method would not be as accurate as the current method, but could possibly be sufficient to obtain a reasonable approximation. The most serious problem to overcome would be the detection of significant anomalies, such as the one seen in Figure 5.6.

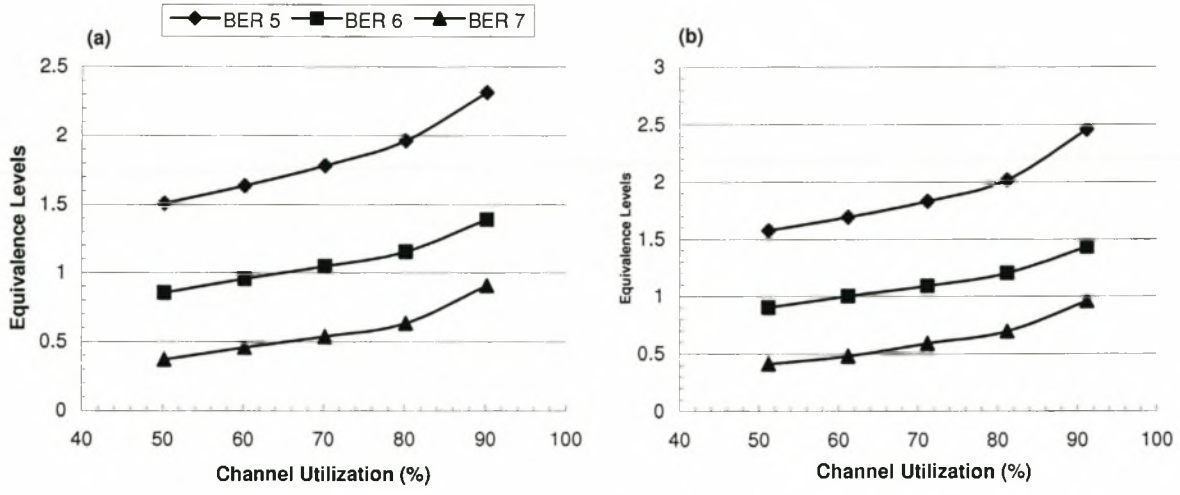

Figure 5.9: *The relationship between the equivalence levels of TCP connections with different bit error rates. (a) The equivalence levels for a TCP stack implementing forward acknowledgement. (b) The equivalence levels for a TCP stack implementing selective acknowledgement.*

In Figure 5.9 the relationship between the equivalence levels for different bit error rates is illustrated. There appears to be a relationship not only between the equivalence levels with different bit error rates, but also between the equivalence levels with different levels of channel utilization. Should this be true, the equivalence metric of TCP channel utilization could be extended to apply to bit error rate, as well as a range of channel utilization levels.

Chapter 6

Emulation

6.1 Introduction

Emulation represents a hybrid of network simulation and live testing. It models a particular piece of the network path between two real hosts. Emulation is important because it serves as a verification check on the results obtained through simulation.

The main strength of emulation is that it allows for the testing and evaluation of TCP/IP implementations without the cost and complexity of building a real test bed. (for satellite channels especially, this cost can be prohibitively high). Another advantage is the ability to modify the router characteristics, which is usually not possible with commercial routers.

Emulation suffers from three disadvantages however, firstly, that it 'does' simulate a part of the network, and therefore suffers from the same problem as simulation (see Section 4.1 on NS simulation problems). Secondly, emulation is constrained by the hardware used in the emulator, as is the case with live testing. Finally, emulators might not be able to represent complex or changing topologies properly.

By turning a computer into a router, a software emulator can adapt communication over this link to assume the characteristics of whichever network is chosen. Example applications include *dummynet*[55], ONE [2] and more recently NISTnet [47].

6.2 The Emulator

A network emulator requires a host of different hardware and software system. This section will briefly discus these systems.

6.2.1 Emulator Test bed Design

The Emulator Test bed design (see 6.1) allows for single or multiple TCP connection to be made across the emulator. It consists of three computers. The 'Emulator', one 'Sender' and one 'Receiver'. The Emulator simulates the channel characteristics we are interested in, while the sender/receiver computers run test software over the network.

A Real-time Dynamic Space Segment Emulator (RDSSE) [58], is an example of an emulator that does not follow this model. The RDSSE system emulates not only the transport and network layers, but also the data and link layers of a satellite system. This system is certainly much more realistic than emulators that only model the higher layers, but accordingly suffers from more of the problems associated with live testing (such as higher cost and experiment turn over time).

Simpler forms of emulation (such as the serial method used in [15]) impose severe restrictions in terms of bandwidth and are of limited use.

The advantages of using an emulator such as NIST Net are firstly the low cost of the test bench. Secondly, from the perspective of the transport layer emulation is very realistic and it allows for the investigation of a wide array of channel characteristics. Thirdly, experiment turnover time is relatively fast.

6.2.2 Emulator Software

NIST Net [47] is a network emulation package for IP networks. It allows a single Linux PC to be set up as a router to emulate a number of channel parameters. It allows for controlled, reproducible experiments in a laboratory environment. Because NIST Net operates at the IP level, it is independent of the rest of the test bed, and this allows a multitude of IP technologies to be tested over it.

NIST Net is implemented as a Linux kernel module extension. It has both a graphical and command line interface and allows for various network channel characteristic to be emulated. These include packet delay, packet loss and bandwidth limitation. NIST Net can also be driven by data produced from measurements of actual networks, and support user defined packet handlers for functions such as time stamping and data collection, interception and diversion of flows, and the generation of protocol responses from an emulated client.

6.2.3 Channel Characteristics

Three satellite channel parameters dominate communication and can all be emulated with NIST Net, namely These are:

Packet Delay Packets passed through the emulator can either be delayed by a fixed or variable amount of time. If this is variable, then the delay will vary around a set point determined by a given probability distribution. The emulator has a minimum delay for all connections because of hardware constraints.

Bandwidth Limitation NIST Net can limit the throughput, by creating a bottleneck at the emulator computer (see 6.1). Once again hardware constraints limit the maximum bandwidth of the emulator.

Bit Error Rate Packet loss can be simulated over the link by the emulator. It can either be a uniform Packet Error Rate (PER) or be congestion based. When uniform PER is used, the Bit Error Rate (BER) for a channel can vary, depending on the packet size. PER must therefore be varied according to what the *average* packet size is of the connection over the emulator if a given BER is to be simulated (see Table 6.1). The congestion based packet loss allows the emulator to control the length of the router queue. The Bit Error Rate of the emulator is not constrained by the hardware, but some of test software packages (such as *netperf* [26]) will not function properly if it is set too high.

6.2.4 Test Software

Several packages are used in the Testbed to measure TCP/IP performance, namely

tcpdump [34] This package can be used to record network traffic over a network interface.

ttcp [14] This package can be used to measure throughput and latency. It is also used as a packet generator in the Testbed in conjunction with *tcptrace*.

netperf [26] It is used to measure the end-to-end throughput and latency of a channel. Useful when investigating TCP/IP behaviour over a number of channels.

tcptrace [48] This package analyses the output of *tcpdump*. It is used to analyze the dynamic behaviour of a TCP connection, but can also be used for end-to-end metric measurements.

Table 6.1: *Packet Error Rates (expressed as a percentage) used in various experiments*

Segment Size	Packet Size	BER 4	BER 5	BER 6
34	100	8	0.8	0.08
184	250	20	2	0.2
434	500	40	4	0.4
684	750	60	6	0.6
934	1000	80	8	0.8
1184	1250	x	10	1
1334	1400	x	11.2	1.12
1434	1500	x	12	1.2

6.2.5 Emulator Hardware

The emulator configuration used in this thesis consists of three computers, with an additional computer available through a 100 MB Ethernet hub connected to the Emulator. The ground and satellite computer are called thus purely for convenience.

Table 6.2: *Emulator Hardware Description*

Computer	Processor	RAM	Distribution	Linux Kernel	Network Card
Ground	Pentium90	32	Redhat 7.3	2.4.18	10 MB
Satellite	Pentium90	32	Redhat 7.3	2.4.18	10 MB
Emulator	Pentium500	128	Redhat 9	2.4.20	100 MB
Workstation	Pentium500	128	Mandrake 9.1	2.4.21	100 MB

The emulator has three 100 MB network cards installed. This allows it to connect to the two emulator computers via cross over cables, and to a network hub. The workstation computer is connected over this 100 MB hub, which is also connected to the local LAN. Traffic over the hub can therefore not be predicted and care must be taken to not have this uncertainty affect an experiment. The Ethernet hub is very fast in comparison with the emulator computer's network cards, and by using the workstation purely as a packet source or sink for the generation of competing flows this uncertainty can be minimized.

The fact that the hardware and software specifications of the satellite and ground station computers are the same can be both desirable and undesirable. It is desirable in that communication is independent of the hardware as to which direction it flows. It is undesirable because a satellite can be reasonably expected to have similar processing capabilities as those used in the emulator, the ground station can be expected to have

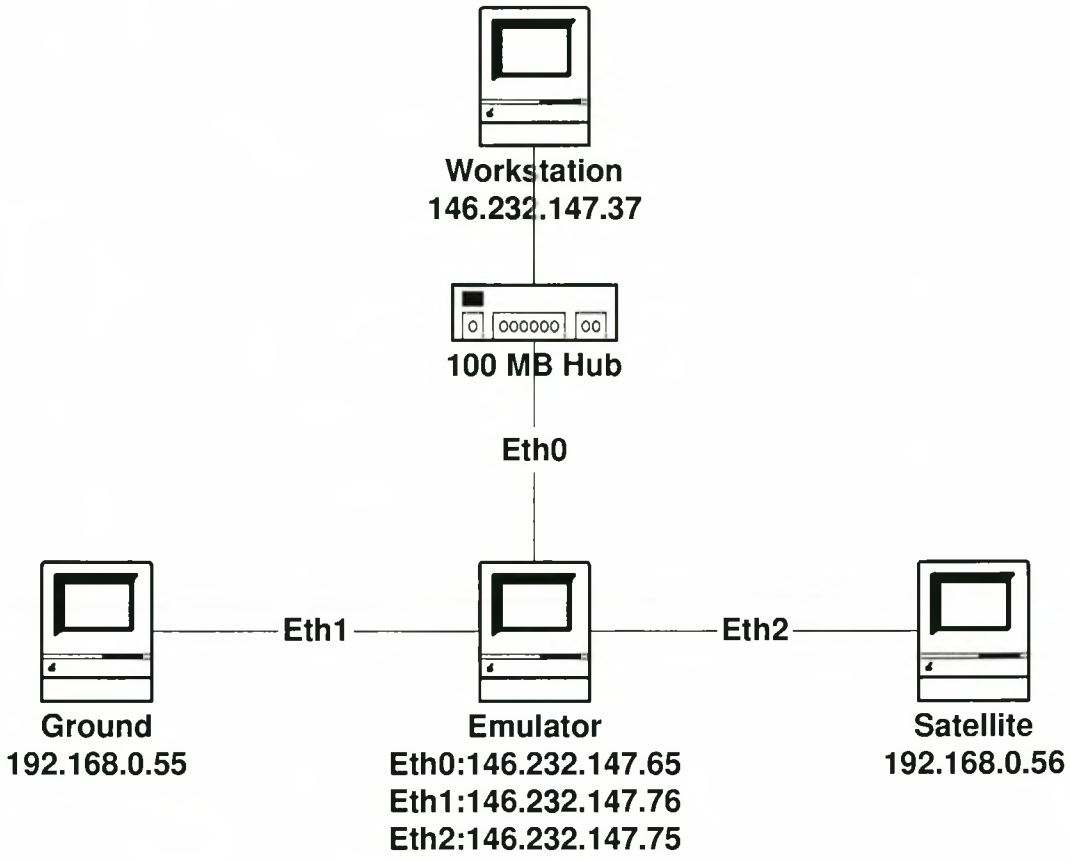

Figure 6.1: *Emulator network layout*

much higher-end capabilities. If hardware asymmetry will impact TCP/IP performance is an unresolved issue, but by using lower end systems the errors in the results can be biased on the conservative side.

See Appendix A for test bed limitations.

6.3 Implementations of the TCP/IP stack

The primary value of emulation over simulation is the ability to test a real world TCP/IP implementation. Every implementation has its own individual bugs, even if every effort is made to create a simulation model as accurate to an implementation as possible.

The simulation models found in the *ns* simulator are modeled on the Unix Vegas, Tahoe, Reno and SACK stacks. None of these implementations exist in the Linux kernel however. While the Linux TCP/IP stack might implement many or all of the mechanisms in any one of these distributions, it will not have exactly the same behaviour because of

implementation differences. In Table 6.3 a number of implemented stacks that are available are listed along with which mechanisms are implemented that are of interest to high performance TCP/IP [35].

Table 6.3: *TCP/IP stack implementation support for high performance.*

Operating System	FreeBSD 2.1.5	Linux 2.1.90 - 2.2.	Microsoft Win98
RFC1191 Path MTU Discovery	Yes	Yes	Unknown
RFC1323 Large Window Support	Yes	Yes	Yes
Default maximum socket buffer size	256kB	64kB	1GB(?!)
Default TCP socket buffer size	16kB	32kB	8kB
Default UDP socket buffer size	40kB	32kB(?)	Unknown
RFC2018 SACK Support	Available	SACK and FACK	Yes

6.4 Evaluating the Linux Stack

6.4.1 General Performance

The first experiment conducted is to gather data to look at the general behaviour of the stack, and to compare this with the simulations conducted. To this end the model of a LEO satellite with 80ms RTT delay and 20kbit/s bandwidth was chosen. This model specifically was chosen because it closely corresponds to the parameters that might be present in a LEO micro-satellite. Throughput results are given in Table 6.4.

These tests are conducted using *ttcp*, with throughput measured by the *ttcp* receiver. Tests ran for at least 60 seconds. The Packet Error Rate of the emulator needs to be adjusted to accommodate different packet sizes. Because the test consisted of TCP Data transfers in one direction, the average packet error rate will need to differ considerable as data is transferred in one direction and ACKs, which have much smaller package sizes on average, in the other. Notice that there are no results for some of the connections with high bit error rates and high packet sizes. This is due to the fact that the high bit error packet error rate makes TCP transactions impossible (See Table 6.1).

Table 6.4: *The measured throughput of the Linux stack over a connection with a return time of 80ms and bandwidth of 20 kbps*

Packet Size	BER 4	BER 5	BER 6
100	13.93	18.84	
250	0.25	18.86	18.70
500	x	18.66	18.85
750	x	16.72	18.85
1000	x	15.37	18.95
1250	x	15.91	18.87
1400	x	12.33	18.87
1500	x	12.25	18.86

It can be deduced from Table 6.4 that performance can be maintained by choosing a packet size of 500 as the transmission error rate increases from 10^{-6} to 10^{-5}. This result corresponds with the simulation results (see Figure 4.4(d)).

6.4.2 Rapidly changing Return Times

To investigate whether change in the delay would affect the performance of TCP the emulator was used to conduct some elementary tests. It was not possible to gradually alter the delay experienced by a TCP connection over the course of a connection, due to instability in the NISTnet program. It was however possible to change the delay experienced, so long as reasonable intervals were kept between changes.

A TCP connection was instantiated using ttcp as a packet source. Bandwidth was limited to 20 kbps and for simplicity the emulator introduced no errors. The delay of a connection was changed from 60 ms to 20 ms abruptly, and back again after 30 seconds.

A TCP trace analysis of this connection showed that no packets were assumed lost by the sender, and that TCP adjusted to the new network conditions smoothly. This can be expected from the knowledge that the Linux kernel uses a timing interval of 100 ms, and therefore would probably not detect such a small change in RTT (Δ RTT = 80ms). It also follows from simulation results that channel utlization would not be less, given that the change does not change network conditions to such an extent that it enter a less then optimal area (See Figure 4.1).

6.4.3 Asymmetric Behaviour

Asymmetry is a concern for TCP/IP performance (see Section 3.2). Bandwidth asymmetry is a common phenomenon with terrestrial and space links. Delay and BER asymmetry can also be observed.

A number of experiments was conducted using a model for a LEO satellite (i.e. 80 ms RTT, BER of 6) with a 20 kbps uplink and a 1 Mbps downlink. In this experiment *ttcp* is used as a packet generator, and network traffic is recorded using *tcpdump*. This is then analyzed using *tcptrace*. Unless otherwise stated all experiments use the network parameters described, and network traffic is recorded at the ground (or sender) side. The UDP streams used in the experiments are sent to the workstation computer, so as to minimize any possible hardware distortions. a Packet size of 1500 is used.

Experiment 1 A baseline experiment is conducted. Both links are 20 kbit/s and a TCP data transfer is conducted from the ground to the satellite.

Experiment 2 A TCP data transfer is conducted from the ground to the satellite.

Experiment 3 A TCP data transfer from the ground to the satellite is conducted while a UDP data transfer is simultaneously taking place from the satellite to the workstation.

Experiment 4 A TCP data transfer from the satellite to the ground station is recorded

Experiment 5 A TCP data transfer from the satellite to the ground station is recorded, while a UDP data transfer is simultaneously taking place from the satellite to the workstation.

The *tcptrace* analysis for these experiments can be seen in Table 6.5. The bandwidth asymmetry is not a factor that affects throughput performance in the case where data is transferred from the ground to the satellite and the return path is larger than the send path. Even where a UDP transfer is taking place in the opposite direction it does not affect the throughput utilization. In the case where data is transferred from the satellite to the ground, bandwidth utilization is affected more seriously (Utilization is 14%). Bandwidth utilization drops down further to 9.6% when a UDP transfer is conducted at the same time.

The implications of these results are that a highly asymmetric communications link could not be fully utilized by TCP. Highly symmetric bandwidth is common among Low Earth Orbit satellites however, as their is a large amount of data that usually needs to be moved to the ground, and relatively little command and control data that needs to go to the

satellite. This is not a problem for the TCP/IP protocol suite if the data does not need to be transmitted 100% reliably, as UDP can be used to send data down at the full bandwidth available. Should a reliable high bandwidth link be required however, a suitable return bandwidth must be provided.

The problem of finding a proper return path bandwidth size so as to fully utilize a high bandwidth TCP connection, would be better solved by using simulation rather than emulation.

Table 6.5: *Asymetric Experiment Results - Throughput*

Parameter	Exp 1		Exp 2		Exp 3		Exp 4		Exp 5	
	S → D	D → S	S → D	D → S	S → D	D → S	S → D	D → S	S → D	D → S
total packets:	1147	770	1153	828	1145	724	1134	757	4532	2942
ack pkts sent:	1146	770	1152	828	1144	724	1133	757	4531	2942
pure acks sent:	2	768	2	826	2	722	1	755	2	2940
sack pkts sent:	0	183	0	158	0	147	0	158	0	550
max sack blks/ack:	0	2	0	2	0	1	0	2	0	2
unique bytes sent:	1638400	0	38400	0	1638400	0	1638400	0	6553600	0
actual data pkts:	1144	0	1150	0	1142	0	1132	0	4529	0
actual data bytes:	1655776	0	63016	0	1652880	0	1638400	0	6553600	0
rexmt data pkts:	12	0	17	0	10	0	0	0	0	0
rexmt data bytes:	17376	0	24616	0	14480	0	0	0	0	0
zwnd probe pkts:	0	0	0	0	0	0	0	0	0	0
zwnd probe bytes:	0	0	0	0	0	0	0	0	0	0
outoforder pkts:	0	0	0	0	0	0	15	0	52	0
pushed data pkts:	52	0	109	0	49	0	76	0	398	0
SYN/FIN pkts sent:	01-Jan	01-Jan	01-Jan	01-Jan	01-Jan	01-Jan	01-Jan	01-Jan	01-Jan	01-Jan
req 1323 ws/ts:	Y/Y	Y/Y	Y/Y	Y/Y	Y/Y	Y/Y	Y/Y	Y/Y	Y/Y	Y/Y
adv wind scale:	0	0	0	0	0	0	0	0	0	0
req sack:	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
sacks sent:	0	183	0	158	0	147	0	158	0	550
urgent data pkts:	0	0	0	0	0	0	0	0	0	0

Table 6.5: *Asymmetric Experiment Results - Throughput*

Parameter	Exp 1		Exp 2		Exp 3		Exp 4		Exp 5	
	S → D	D → S	S → D	D → S	S → D	D → S	S → D	D → S	S → D	D → S
urgent data bytes:	0	0	0	0	0	0	0	0	0	0
mss requested(bytes):	1460	1460	1460	1460	1460	1460	1460	1460	1460	1460
max segm size(bytes):	1448	0	1448	0	1448	0	1448	0	1448	0
min segm size(bytes):	712	0	96	0	712	0	872	0	208	0
avg segm size(bytes):	1447	0	1446	0	1447	0	1447	0	1447	0
max win adv(bytes):	5840	63712	5840	63712	5840	63712	5840	63712	5840	63712
min win adv(bytes):	5840	5792	5840	5792	5840	5792	5840	5792	5840	5792
zero win adv(times):	0	0	0	0	0	0	0	0	0	0
avg win adv(bytes):	5840	62579	5840	62736	5840	62580	5840	62581	5840	63182
initial window(bytes):	2896	0	4344	0	2896	0	1448	0	4344	0
initial window(pkts):	2	0	3	0	2	0	1	0	3	0
ttl stream length(bytes):	1638400	0	38400	0	1638400	0	1638400	0	6553600	0
missed data(bytes):	0	0	0	0	0	0	0	0	0	0
truncated data(bytes):	1621456	0	28516	0	1618620	0	1604440	0	6417730	0
truncated packets:	1144	0	1150	0	1142	0	1132	0	4529	0
data xmit time(sec):	84.118	0	4.498	0	83.813	0	10.923	0	67.545	0
idletime max(ms):	3075.7	150.9	675.5	150.2	3048.2	294.2	404	408.9	131.5	133.6
throughput(Bps):	19285	0	19293	0	19262	0	147624	0	96675	0

Table 6.6: *Asymetric Experiment Results - Latency*

Parameter	Exp 1		Exp 2		Exp 3		Exp 4		Exp 5	
	S → D	D → S	S → D	D → S	S → D	D → S	S → D	D → S	S → D	D → S
RTT samples:	578	2	663	2	568	2	590	2	2362	2
RTT min:(ms)	82.7	0.2	82	0.2	113.4	0.4	0.2	81.4	0.2	108.5
RTT max:(ms)	2247.5	0.4	124.7	0.3	3300.2	0.4	39.6	88.3	57.7	126.9
RTT avg:(ms)	763.3	0.3	623.8	0.3	811.7	0.4	2.2	84.8	2.8	117.7
RTT stdev:(ms)	346.1	0	200.7	0	541.5	0	6.8	0	8.6	0
RTT from 3WHS:(ms)	82.7	0.4	82	0.3	119.2	0.4	0.3	88.3	0.4	126.9
RTT full_sz smpls:	576	1	661	1	566	1	588	1	2355	1
RTT full_sz min:(ms)	85.1	0.2	85.1	0.2	113.4	0.4	0.2	81.4	0.2	108.5
RTT full_sz max:(ms)	2247.5	0.2	124.7	0.2	3300.2	0.4	39.6	81.4	57.7	108.5
RTT full_sz avg:(ms)	764.5	0.2	625.3	0.2	812.3	0.4	2.2	81.3	2.8	108.5
RTT full_sz stdev:(ms)	345.6	0	199	0	541.6	0	6.8	0	8.6	0
post-loss acks:	12	0	17	0	10	0	15	0	51	0
segs cum acked:	543	0	454	0	555	0	528	0	2117	0
duplicate acks:	180	1144	148	1150	146	1142	152	1131	529	4529
triple dupacks:	11	0	14	0	10	0	13	0	48	0
max # retrans:	1	0	1	0	1	0	0	0	0	0
min retr time:(ms)	600	0	600.1	0	590.1	0	0	0	0	0
max retr time:(ms)	2325	0	199	0	3447.8	0	0	0	0	0
avg retr time:(ms)	1103.1	0	853.1	0	1166.5	0	0	0	0	0
sdv retr time(ms):	506.5	0	182.5	0	824.7	0	0	0	0	0

6.4.4 Comparison with Simulations

As stated before, one of the major differences between using simulation and testing in a test bed is the difference of time needed to run an experiment. In simulation, an average experiment will run for less than a minute, depending on processor power. In emulation however, most experiments are limited by the network being used (in this case Ethernet) or the channel characteristics being investigated.

A simple comparison is performed between the performance of the Forward Acknowledgment stack available in *ns* and the Linux stack used in the Test bed.

Table 6.7: *Comparing Channel Utilization LEO Simulation results with Emulation at BER* 10^{-4}

Packet Size	Simulation(%)	Emulation(%)	Difference(%)
100	52.19	69.65	-17.46
250	30.23	1.25	28.98
500	7.67	0.00	
750	2.25	0.00	
1000	0.60	0.00	
1250	0.58	0.00	
1400	0.65	0.00	
1500	0.70	0.00	

Table 6.8: *Comparing Channel Utilization LEO Simulation results with Emulation at BER* 10^{-5}

Packet Size	Simulation(%)	Emulation(%)	Difference(%)
100	98.55	94.20	4.35
250	97.88	94.30	3.58
500	94.83	94.25	0.58
750	91.65	83.60	8.05
1000	81.47	76.85	4.62
1250	79.33	79.55	-0.22
1400	77.43	61.65	15.78
1500	75.00	61.25	13.75

The differences between emulation and simulation can be explained by the fact that emulation introduces many more complexities into the experiment. Factors such the extra delay introduced by real world hardware and software as well as errors not present

Table 6.9: *Comparing Channel Utilization LEO Simulation results with Emulation at BER* 10^{-6}

Packet Size	Simulation(%)	Emulation(%)	Difference(%)
100	98.09	0.00	
250	98.30	93.50	4.80
500	98.60	94.25	4.35
750	98.85	94.25	4.60
1000	99.20	94.75	4.45
1250	99.33	94.35	4.98
1400	99.45	94.35	5.10
1500	99.60	94.30	5.30

in the simulation model impact on the maximum channel utilization of TCP, and worsen the performance of TCP under adverse conditions. Other factors include the link layer that might not be adequately simulated by the simulator, or implementation problems associated with the real world TCP/IP stack used by the emulator.

These differences must be expected however. This is also the reason why emulation should be used in conjunction with simulation.

6.5 Conclusions and Future Work

6.5.1 Early Results

This chapter discussed the design and implementation of the emulator, as well as the various component parts that influence the experiments conducted on the emulator. The performance of the Linux TCP/IP stack was tested for Low Earth Orbit condition, and that performance compared with results obtained through simulation. The effect of wildly varying Return Times in Low Earth Orbit conditions were also investigated and found not to influence performance. Lastly various experiments were conducted into the bandwidth asymmetric behaviour of TCP, as well as the influence of competing UDP data flows in such an environment. These last experiments showed that maximum channel utilization is affected by the asymmetric bandwidth to such an extent that the available bandwidth cannot be efficiently utilized . On the other hand the use of competing UDP flows, used in experiment 5, showed that the a relatively high, if not efficient or fare, TCP transfer rate can be maintained at the same time as a UDP tranfer. This result leads to the conclusion

that a small uplink and large downlink bandwidth can be efficiently utilized by using a combination of both TCP and UDP.

Emulation is an important tool when researching TCP behaviour. Although not as flexible as simulation, results obtained through emulation can be seen as more believable. This is because the level of abstraction that is present in emulation is less then that found in simulation. Emulation therefore presents a benchmark for the results obtained through simulation, allowing us to verify our results.

The Linux TCP/IP protocol stack channel utilization in Low Earth orbit condition was found to be very close to optimal. The bit error rate(BER) does have a serious impact on performance however, and a BER of 10^{-6} is recommended for similar connections. The use of smaller packet sizes can be used to improve channel utilization in the event of temporarily higher bit error rates (Such as might be experienced with connections at low angles to the horizon).

6.5.2 Emulator

The use of the Emulator to investigate a broad range of network parameters is problematic however. Some of these problems experienced were

- The use of old hardware causes maintenance and software updates to be difficult and time consuming.
- The Emulator software, NIST net, was found to have a number of bugs (requiring the emulator computer to be restarted for the emulator to function again).
- Turnover times for experiments are long, experiments are complex to set up, and experiments need constant supervision.

An attempt was made at automating emulation experiments using the netperf test program and the use of shell scripting. This was found to be unworkable due to errors in the netperf test program, as well as instability in the NIST net emulator software. These problems are not believed to be insurmountable, but would not be trivial to solve.

A number of recommendations can be made as regards future work with the emulator.

- The hardware of the emulator should be kept together between experiments. Setting up the emulator is not a trivial exercise and experiment time is wasted if a time consuming setup process needs to be completed every time.

- That the emulator and test software be integrated in some manner. The need to run different programs at different computers is very inefficient.
- The programs *tcpdump*, *ttcp* and *tcptrace* were found to work well and their use is recommended.
- The need exists to investigate the performance of wireless link layer protocols. The addition of more physical network media is recommended.

There is a number of areas where the emulator can be used in future research

- The validity of the equivalence metric discussed in Chapter 5 needs to be verified with the emulator.
- The performance of the Linux TCP/IP stack over a broader range of parameters needs to be investigated.
- Finally, it should be noted that the emulator would be a useful tool during the quality assurance phase of any future satellite projects.

Chapter 7

Conclusions and Future work

This thesis made use of two techniques, simulation and emulation, to study the behaviour of TCP/IP over Satellite networks. A new performance metric, the equivalence level, was proposed. The equivalence level, used in conjunction with the equivalence formula, can be used to summarize the channel utilization of TCP for any given bandwidth and delay. Although promising, further study of this metric is recommended.

For conclusions and recommendation on simulation, emulation as well as the equivalence metric, please refer to the conclusions of their respective chapters.

This Thesis looked to answer two questions. The first was the validity of TCP/IP as a communication protocol in satellite networks. There are also a number of problems associated with TCP/IP over satellite networks. Despite this, the use of TCP/IP over satellite networks is still recommended, as long as the delay bandwidth product of the connection is not prohibitively high.

The second question was how to optimize TCP/IP for satellite networks. In Chapter 3 of this thesis many of the enhancements and new mechanisms that have been proposed to improve TCP/IP communication over satellite networks was discussed. Although promising, most of these methods have been studied in isolation from each other. It is hoped that by using simulation a TCP/IP stack that uses a combination of methods and is optimized for satellite communication could be developed. This would be a computationally mammoth task however. The equivalence metric, with its ability to summarize large amounts of performance data, would be an ideal method by which the performance of a proposed stack could be evaluated. The validity of simulation results can then be confirmed by using the emulator and a Linux TCP/IP stack in which the optimizations has been implemented.

Appendix A

Test bed Capabilities and Limitations

A.1 Introduction

The test bed is used to simulate possible network channel characteristics. Both the software and hardware introduce limitations that restrict the range of tests that can be performed. Three characteristics dominate satellite channels: delay, bandwidth and BER. In addition to this, the Operating system (in this case Linux 2.4) introduces parameters that could influence performance.

A.2 Delay Limitations

Delay affects the network latency, which can be broken up into four parts :

Transmission delay is equal to the packet size divided by the bandwidth, and is the amount of time it takes a network node to transmit or receive a packet

Queuing delay is the amount of time the packet spends waiting in queues, either to be transmitted or received. It is thus also dependant on transmission delay

Propagation delay is the delay imposed by the speed of light, and varies according to the medium over which it is traveling.

Processing overhead is both software and hardware dependant.

In contrast to the ONE emulator [2], the NIST net emulator can only vary the propagation delay. The transmission delay, related queuing delay and processing overhead of the test

bed is therefore determined by the hardware and software used. A minimum latency is imposed on the test bed.

Using the *ping* program the delay for various send sizes was measured. The results can be seen in Table A.1 and Figure A.1.

Table A.1: *Minimum Delays (ms) over emulator for different send sizes using ping*

Send Size	Min	Avg	Max	Mdev
10	0.902	0.959	1.045	0.049
100	1.349	1.428	1.551	0.068
500	3.494	3.557	3.687	0.086
1000	6.174	6.243	6.418	0.122
1500	9.082	9.162	9.343	0.085
2000	9.645	9.683	9.772	0.143
4000	12.699	12.770	12.991	0.136
6000	16.736	16.786	16.898	0.158
8000	19.915	19.966	20.066	0.149
12000	27.092	27.165	27.307	0.158
16000	33.483	33.524	33.567	0.028

As can be seen in Table A.1, the minimum delay of the testbed is very low. A serial connection on the other hand (such as the one found in a satellite modem) would impose a much higher delay. The maximum amount of delay that can be emulated is still an unknown value because the hardware has a limited amount of memory.

A.3 Bandwidth limitations

The main bandwidth limitation of the emulator is imposed by hardware. Using 10 Mbps Ethernet cards in the emulator computers restricts the bandwidth to a level somewhere below this. Running a 60 sec *ttcp* stream across the emulator, with NISTnet switched off, and analyzing the network traffic with *tcptrace* gives the results presented in Table A.2 and Table A.3.

Table A.2: *Throughput analysis of emulator configuration with NIST Net switched off using tcptrace*

Parameter	Ground to Sat	Sat to Ground
total packets:	39471	20466

Table A.2: *Throughput analysis of emulator configuration with NIST Net switched off using tcptrace*

Parameter	Ground to Sat	Sat to Ground
ack pkts sent:	39470	20466
pure acks sent:	2	20464
sack pkts sent:	0	0
max sack blks/ack:	0	0
unique bytes sent:	56510768	0
actual data pkts:	39467	0
actual data bytes:	56510768	0
rexmt data pkts:	0	0
rexmt data bytes:	0	0
zwnd probe pkts:	0	0
zwnd probe bytes:	0	0
outoforder pkts:	0	0
pushed data pkts:	1993	0
SYN/FIN pkts sent:	1/1	1/1
req 1323 ws/ts:	Y/Y	Y/Y
adv wind scale:	0	0
req sack:	Y	Y
sacks sent:	0	0
urgent data pkts:	0 pkts	0 pkts
urgent data bytes:	0 bytes	0 bytes
mss requested:	1460 bytes	1460 bytes
max segm size:	1448 bytes	0 bytes
min segm size:	56 bytes	0 bytes
avg segm size:	1431 bytes	0 bytes
max win adv:	5840 bytes	63712 bytes
min win adv:	5840 bytes	5792 bytes
zero win adv:	0 times	0 times
avg win adv:	5840 bytes	63680 bytes
initial window:	2896 bytes	0 bytes
initial window:	2 pkts	0 pkts
ttl stream length:	65536000 bytes	0 bytes
missed data:	9025232 bytes	0 bytes
truncated data:	55326758 bytes	0 bytes

Table A.2: *Throughput analysis of emulator configuration with NIST Net switched off using tcptrace*

Parameter	Ground to Sat	Sat to Ground
truncated packets:	39467 pkts	0 pkts
data xmit time:	61.475 secs	0.000 secs
idletime max:	227.5 ms	176.5 ms
throughput:	**918989 Bps**	**0 Bps**

A.4 Error Limitations

During this study it has been found that NISTnet does not function properly for very low error percentage packet error rates. Throughput performance is observed to decrease for flows if the Packet Error Rate (PER) is set to below 0.1%, even though it should logically increase. These problems persists, even if the PER is then set back too a 0% drop rate, and requires a kernel restart too resolve.

The fact that TCP Packets size impacts on the PER of a link with a set BER, and that a number of test use one-way TCP data flows means that the PER for both directions will necessarily differ. The return flow, with its small packet size ACKs, will have a much smaller PER.

At BER 6 the PER of an ACK flow would be 0.05%. This is below the level that the problematic performance results are observed. Using 0.1% PER allows one to err on the side of caution.

For BER higher than 6, investigating performance with the addition of errors becomes almost impossible. At BER 7, the smallest packet size that can be investigated is 1250. At BER 8 this becomes 12500, which is of little interest.

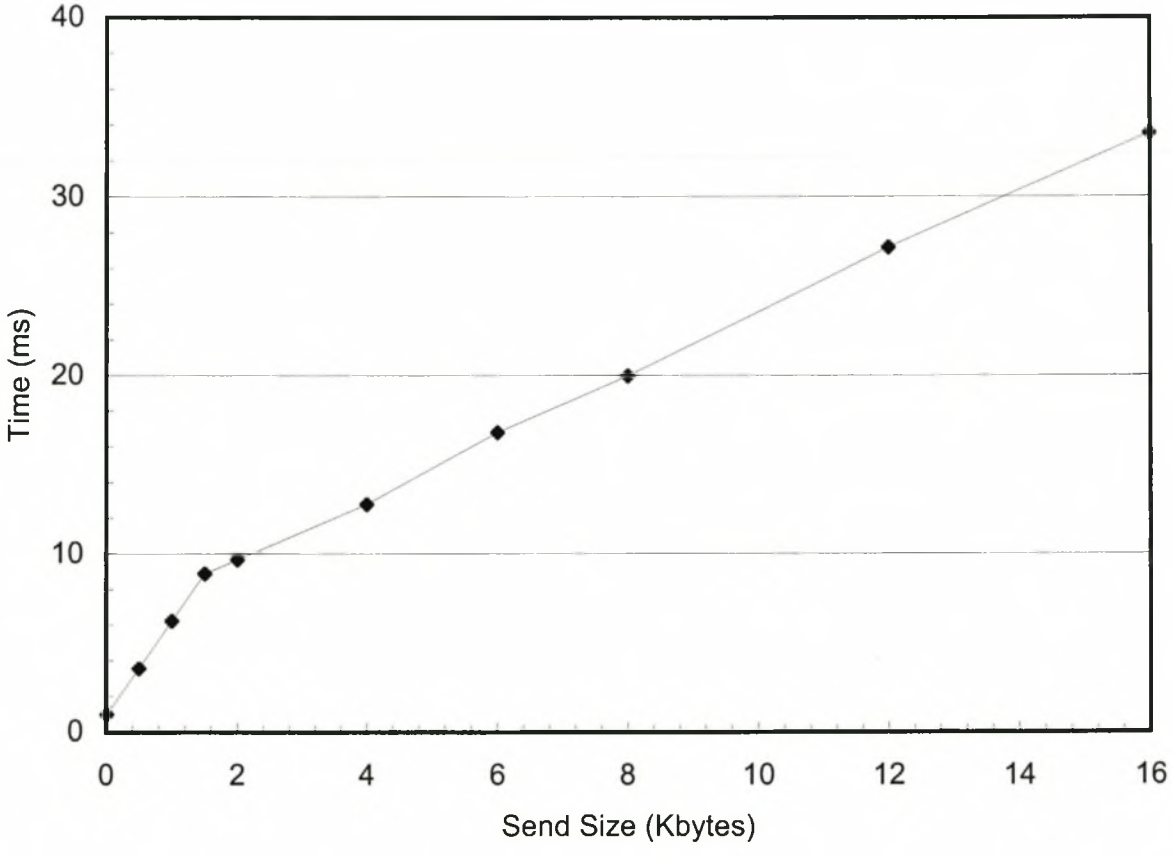

Figure A.1: *Minimum Delays (ms) over emulator for different send sizes using ping*

Table A.3: *Latency analysis of emulator with NIST Net switched off using tcptrace*

Parameter	Ground to Sat	Sat to Ground
RTT samples:	19493	2
RTT min:	2.4 ms	0.3 ms
RTT max:	103.9 ms	0.4 ms
RTT avg:	9.4 ms	0.3 ms
RTT stdev:	9.9 ms	0.0 ms
RTT from 3WHS:	2.4 ms	0.4 ms
RTT full_sz smpls:	18425	1
RTT full_sz min:	5.0 ms	0.3 ms
RTT full_sz max:	103.9 ms	0.3 ms
RTT full_sz avg:	9.4 ms	0.3 ms
RTT full_sz stdev:	10.1 ms	0.0 ms
post-loss acks:	0	0
segs cum acked:	19976	0
duplicate acks:	0	39468
triple dupacks:	0	0
max # retrans:	0	0
min retr time:	0.0 ms	0.0 ms
max retr time:	0.0 ms	0.0 ms
avg retr time:	0.0 ms	0.0 ms
sdv retr time:	0.0 ms	0.0 ms

Appendix B

TCP/IP Resources

B.1 Introduction

A number of resources are available to TCP/IP researchers or network engineers. This appendix hopes to give a short overview of what is available.

B.2 Societies

A number of interrelated societies control the development of the Internet and its underlying protocols. The most important are,

ISOC The Internet Society is a professional membership society that addresses issues confronting the future of the Internet. It is the organizational home for the groups responsible for Internet infrastructure standards, including the Internet Engineering Task Force (IETF) and the Internet Architecture Board (IAB). Their homepage can be found at *http://www.isoc.org/isoc/*.

IETF The Internet Engineering Task Force is an open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. The technical work of the IETF is done in its working groups, which are organized by topic into several areas (e.g., routing, transport, security, etc.). Their homepage can be found at *http://www.ietf.cnri.reston.va.us/home.html*.

IAB The Internet Architecture Board is chartered both as a committee of the Internet Engineering Task Force (IETF) and as an advisory body of the Internet Society

(ISOC). Its responsibilities include architectural oversight of IETF activities, Internet Standards Process oversight and appeal, and the appointment of the RFC Editor. The IAB is also responsible for the management of the IETF protocol parameter registries. Their homepage can be found at *http://www.iab.org/*.

IRTF The Internet Research Task Force is a composed of a number of small Research Groups. The Research Groups work on topics related to Internet protocols, applications, architecture and technology. The IRTF chairman is appointed but the IAB. Their homepage can be found at *http://www.irtf.org/*.

IANA The Internet Assigned Numbers Authority is responsible for keeping the numbers that allow the Internet to work. These include IP addresses, protocol number assignments and domain name services. Their function is being taken over by ICANN. Their homepage can be found at *http://www.iana.org/*

ICANN The Internet Corporation for Assigned Names and Numbers is a technical coordination body for the Internet. Created in October 1998 by a broad coalition of the Internet's business, technical, academic, and user communities, ICANN is assuming responsibility for a set of technical functions previously performed under U.S. government contract by IANA and other groups. In addition, ICANN coordinates the stable operation of the Internet's root server system. Their homepage can be found at *http://www.icann.org/*.

B.3 Online Resources

There are a number of online resources that are very useful. These include,

RFCs A document series, begun in 1969, which describes the Internet suite of protocols and related experiments. Very few RFCs describe Internet standards, but all Internet standards are written up as RFCs. A collection of all RFCs to date can be found at *http://rfc.sunsite.dk/*.

Resource list A very complete list of TCP/IP resources has been compiled by Uri Raz. It is available at *http://www.private.org.il/tcpip_rl.html*.

Citeseer CiteSeer is a scientific literature digital library that aims to improve the dissemination and feedback of scientific literature. It is a very complete archive of computer science research, and the ability to traverse references is an amazing tool. Documents are available for download in a number of formats, and most

references in this document can be found here. The homepage of this project is *http://citeseer.nj.nec.com/*.

FAQ The TCP/IP Frequently asked questions documents list for the comp.protocols.tcp-ip Usenet newsgroup. The FAQ provides answers to a selection of common questions on the various protocols (IP, TCP, UDP, ICMP and others) that make up the TCP/IP protocol suite. It is posted to the news.answers, comp.answers and comp.protocols.tcp-ip newsgroups on or about the first Friday of every month. An online copy is available at *http://www.itprc.com/tcpipfaq/default.htm*.

Bibliography

[1] ALLMAN, M., "Improving TCP performance over satellite channels." Master's thesis, Ohio University, 1997.

[2] ALLMAN, M., CALDWELL, A., and OSTERMANN, S., "ONE:The Ohio Network Emulator." *Ohio University Computer Science*, August 1997, Vol. Technical Report TR-19972.

[3] ALLMAN, M. and FALK, A., "On the Effective Evaluation of TCP." *ACM Computer Communication Review*, October 1999.

[4] ALLMAN, M. E. *et al.*, "Increasing TCP's Initial Window." *RFC 2414*, 1998.

[5] ALLMAN, M. E. *et al.*, "Enhancing TCP over Satellite Channels using Standard Mechanisms." *RFC 2488*, 1999.

[6] ALLMAN, M. E. *et al.*, "Ongoing Research Related to Satellites." *RFC 2760*, 2000.

[7] ALLMAN, M. E. *et al.*, "TCP Congestion Control." *RFC 2581*, 2001.

[8] ALTMAN, A., BARAKAT, C., LABORDE, E., BROWN, P., and COLLANGE, D., "Fairness Analysis of TCP/IP." in *Proc. of IEEE Conference on Decision and Control, Sydney, December*, 2000.

[9] BALAKRISHNAN, H., PADMANABHAN, V. N., and KATZ, R. H., "The Effects of Asymmetry on TCP Performance." in *Mobile Computing and Networking*, pp. 77–89, 1997.

[10] BERNERS-LEE, T., FIELDING, R., and NIELSON, H., "Hypertext Transfer Protocol - HTTP/1.0." *RFC 1945*, 1996.

[11] BRADEN, R., "Requirements for Internet Hosts –Communication Layers." *RFC 1122*, 1989.

[12] BRAKMO, L. and PETERSON, L., "Experiments with Network Simulators." *ACM SIGMETRICS*, 1996.

[13] BRAKMO, L. S., O'MALLEY, S. W., and PETERSON, L. L., "TCP Vegas: New Techniques for Congestion Detection and Avoidance." in *SIGCOMM*, pp. 24–35, 1994.

[14] CHESAPEAKE COMPUTER CONSULTANTS, I., "Test TCP (TTCP)." `http://www.ccci.com/tools/ttcp/`. 1997.

[15] COETZEE, C., "On Optimizing High-Latency Asymmetric Satellite TCP/IP." Master's thesis, Department of Electric and Electronic Engineering University of Stellenbosch, October 1999.

[16] DEGERMARK, M., ENGAN, M., NORDGREN, B., and PINK, S., "Low-Loss TCP/IP Header Compression for Wireless Networks.." in *ACM/Baltzer Journal on Wireless Networks, vol.3*, pp. 375–87, 1997.

[17] DURST, R., MILLER, G., and TRAVIS, E., "TCP extensions for space communications." *2nd ACM International Conferance on Mobile Computing and Networking (Mobicom)*, 1996.

[18] FALL, K. and FLOYD, S., "Simulation-based Comparisons of Tahoe, Reno and SACK TCP." *Computer Communications Review*, July 1996.

[19] FLOYD, S., "TCP and Explicit Congestion Notification." *ACM Computer Communications Review*, October 1994.

[20] FLOYD, S., "Issues of TCP with SACK." tech. rep., 1996.

[21] FOX, R., "TCP Big Window and Nak Options." *RFC 1106*, 1989.

[22] HADI SALIM, J. and AHMED, U., "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks." *RFC 2884*, 2000.

[23] JACOBSON, V., "Congestion Avoidance and Control." *ACM Computer Communication Review; Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988*, 1988, Vol. 18, 4, pp. 314–329.

[24] JACOBSON, V., BRADEN, R., and BORMAN, D., "TCP extensions for high performance." *RFC 1323*, 1993.

[25] JIN, C. *et al.*, "Fast TCP: From Theory to Experiments." *IEEE Communications Magazine*, April 2003.

[26] JONES, R., "Netperf: a network performance monitoring tool." `http://www.cup.hp.com/netperf/netperfpage.html`.

[27] KARN, P., "Toward new link layer protocols." *QEX*, June 1994, pp. 3–10.

[28] KARN, P. and PARTRIDGE, C., "Improving Round-Trip Time Estimates in Reliable Transport Protocols." *ACM Transactions on Computer Systems*, 1991, Vol. 9, No. 4, No. 4, pp. 364–373.

[29] KESHAV, S., *REAL: a Network Simulator. Technical Report 88/472*. University of California Berkeley, 1988.

[30] KNOWLES, S., "IESG Advise from Experience with Path MTU." *RFC 1435*, 1993.

[31] KRUSE, H., "Performance Of Common Data Communications Protocols Over Long Delay Links: An Experimental Examination." in *3rd International Conference on Telecommunication Systems Modeling and Design*, p. 409, 1995.

[32] LARSON, W. J. and WERTZ, J. R., *Space Mission Analysis and Design*. Second edition. Torrance, California: Microcosm, Inc., 1998.

[33] LAWAS-GRODEK, F. J., "Scps-tp, tcp and rate-based protocol evaluation for high delay, erro prone links."

[34] (LBNL), L. B. N. L., "Tcpdump." `http://www.tcpdump.org/`.

[35] MAHDAVI, J., "Enabling High Performance Data Tranfers on Hosts." `http://www.psc.edu/networking/perf_tune.html`.

[36] MARAL, G. and BOUSQUET, M., *Satellite Communications Systems*. Third edition. New York: John Wiley & Sons, 1999.

[37] MATHIS, M., SEMKE, J., MAHDAVI, J., and OTT, T., "The Macroscopic Behaviour of the TCP Congestion Avoidance Algorithm." *Computer Communications Review*, 1997.

[38] MATHIS, M., M., J., S., FLOYD, and ROMANOW, A., "TCP Selective Acknowledgement Options." *RFC 2018*, 1996.

[39] MATHIS, M. and MAHDAVI, J., "Forward Acknowledgement: Refining TCP Congestion Control." in *SIGCOMM*, pp. 281–291, 1996.

[40] MAZZUCCO, M., ANANTHANARAYAN, A., and GROSSMAN, R. L., "Merging Multiple Data Streams on Common Keys over High Performance Networks.." in *15th Supercomputing conferance on High performance networking and computing*, 2002.

[41] MCCANNE, S. and FLOYD, S., "NS(Network Simulator)." www-nrg.ee.lbl.gov. 1995.

[42] MCKENZIE, A., "A Problem with the TCP Big Window Option." *RFC 1110*, 1989.

[43] MILNE, G. *et al.*, "SUNSAT." tech. rep., 1991.

[44] MOGUL, J. and DEERING, S., "Path MTU Discovery." *RFC 1191*, 1990.

[45] MOHAMMADI, O., TOOMY, W., and BROWN, L., *DUAL- A packet format for a new amateur radio link layer*. The University of New South Wales: Department of Computer Science, February 1995.

[46] NAGLE, J., "Congestion Control in TCP/IP Internetworks." *RFC 896*, 1984.

[47] NIST, "NIST Net 2.0.12." http://snad.ncsl.nist.gov/itg/nistnet/. November 2002.

[48] OSTERMANN, S., "tcptrace - TCP dump file analysis tool." http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html.

[49] PAXSON, V., *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California, 1996.

[50] PAXSON, V. and FLOYD, S., "Why We Don't Know How To Simulate the Internet." in *Winter Simulation Conference*, pp. 1037–1044, 1997.

[51] POSTEL, J., "Internet Protocol." *RFC 791*, 1981.

[52] POSTEL, J., "Transmission Control Protocol." *RFC 793*, 1981.

[53] POSTEL, J. and REYNOLDS, J., "File Transfer Protocol (FTP)." *RFC 959*, 1985.

[54] RAMAKRISHNAN, K. and FLOYD, S., "A Proposal to add Explicit Congestion Notification (ECN) to IP." *RFC 2481*, 1999.

[55] RIZZO, L., "Dummynet: a simple approach to the evaluation of network protocols." *ACM Computer Communication Review*, 1997, Vol. 27, No. 1, No. 1, pp. 31–41.

[56] STEVENS, W. R., *TCP/IP Illustrated, vol 1: The Protocols*. Reading, Massachusetts: Addison-Wesley, 1994.

[57] STEVENS, W., "TCP Slow Start, congestion avoidance, fast retransmit and fast recovery algortihms." *RFC 2001*, 1998.

[58] TAAGHOL, P., AZIZ, H., NARENTHIRAN, K., TAFAZOLLI, R., and EVANS, B., *A real-time dynamic space segment emulator*. Hull Canada: IMSC, 1999.

[59] TAPR, "AX.25 Amateur Packet-Radio Link-Layer Protocol Version 2.2." `www.tapr.org/tapr/ax25.doc`. November 1997.

[60] VAN DER MERWE, B., "Micro-satellite Data Handling: A Unified Information Model." Master's thesis, Department of Electric and Electronic Engineering University of Stellenbosch, March 2001.

[61] VISWESWARAIAH, V. and HEIDEMANN, J., "Improving Restart of Idle TCP Connections." tech. rep., 1997.

[62] WRIGHT, G. and STEVENS, W. R., *TCP/IP Illustrated, vol II: The Implementation*. Reading, Massachusetts: Addison-Wesley, 1995.

[63] ZHANG, Y. and QIU, L., "Understanding the End-to-End Performance Impact of RED in a Heterogeneous Environment." tech. rep., 2000.