TCP over military networks.

Workman, Russell

TCP over Military Networks

Russell Workman

Thesis submitted to the University of Wales
in candidature for the
Degree of Doctor of Philosophy

Department of Electrical and Electronic Engineering
University of Wales Swansea
Sponsored by QinetiQ
September 2003

Pages: Cover + xii + 161

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed.....(Candidate)

Date.....29/9/03.....

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography and appended.

Signed.....(Candidate)

Date.....29/9/03.....

Statement 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed.....(Candidate)

Date.....29/9/03.....

PREFACE

The study in this thesis considers four layers of the ISO stack, i.e. physical, link, network and transport. The most important aspect to be considered is the interaction between the layers rather than the individual performance of each layer. The thesis has been broken down into three subject areas, physical, link and transport; the network and transport layer being combined into a single subject. Information gathered from the investigation into one layer is used as background research in the next layer; starting with the physical layer and finishing with the transport layer.

ACKNOWLEDGEMENT

First thanks to all those people who kept nagging me to finish this thesis, the list is rather long, but you know who you are.

Thanks to work colleges that made this possible (no particular order):
Clive Harding for allowing me to do this PhD with funding.
Roger Edward for insanely volunteered to mark the drafts with red pen.
Brian Ray for providing the information that could not be found in a book.
Fred Bell for making sure the money came from somewhere.
Steve Creed for providing the captured radio data.

Thanks to friends and fellow students:
Adam Tacy for showing me there is a harder way to do a PhD.
Martin Fricker, looks like we're in this boat together.
Simon Littlejohns for showing it is possible to enjoy doing a PhD.
Jerzy (Jurik) Wechta whose enthusiasm was quite scary.

A special thanks to my parents:
My Dad for proof reading my thesis.
My Mom for her incessant nagging (sorry encouragement).

And finally my supervisors (date order):
Professor Fred Halsall
Professor Bob Cryan
Professor Jaafar Elmirghani

SUMMARY

The major issues with existing military communications are the requirement to improve connectivity between systems. At present there are many bespoke systems used by co-operating nations that are unable to share information because of inter-connectivity problems. The solution requires the use of a common standard that the different nations are willing to implement and use. One network technology that is able to address these problems is the Internet protocol suite.

Tactical communications rely heavily on radio links to provide connectivity across the network. Radio links are inherently less reliable than fixed. The aim of this thesis is to study degrading effects at the radio link layer and see how this affects the performance of TCP/IP.

The study of the radio link effects concentrated on the distribution of errors (error patterns). The results from radios were compared against standard models and revealed a discrepancy between radios and the models. The military radio showed characteristics that were very different from any of the standard models hence a new burst error model was developed that could mimic these error patterns.

The next stage of the study was to observe how these burst errors affect the data link layer protocols. A hypothesis was proposed that burst errors would improve the performance of unprotected packet but adversely affect FEC. This was proven to be true, but the effect is only really noticeable at the point where the throughput collapses. The results also showed that the worst case to study for unprotected packets was random errors not burst errors.

The IP layer has no recovery mechanism, it is Transmission Control Protocol (TCP) that provides the reliability. TCP contains several mechanisms to handle lost packet detection, retransmission and congestion management. The performance of each of the algorithms was tested in the presence of errors to determine which combination produced the highest throughput. A number of network restrictions where also detailed, the round trip time delay should be less than 9 seconds and the residual bit error rate should be better than 1 in 10^5.

Various ways of separating congestion control and corruption were considered to try and improve the performance of TCP in the presence of errors. These mechanisms were TCP Vegas, a modified version of Vegas and Packet Pair measurement. Both the Modified Vegas and Packet Pair Control produced throughputs that were better than standard TCP in the presence of errors. The Modified Vegas produced very good results under congestion, again better than standard TCP.

The other approach that was considered for improving the performance of TCP was to change the links to reduce the errors that effect the performance rather than changing TCP to handle errors. The use of ARQ greatly improved the throughput. The use of ARQ increased the variance in the round trip time which has a major impact on the time delay sensitive algorithms such as Vegas and Modified Vegas. It is recommended that a combination of Modified Vegas and ARQ is not used.

From the results it is clear that there could be interaction effects between different protocol layers. Burst bit error may or may not produce burst packet losses. Pulse errors can have a dramatic impact on the throughput of TCP. The time delay variance introduced by ARQ can greatly effect time sensitive congestion control mechanisms such as Vegas. Hence with any system it is necessary to consider all layers not just the performance of a single layer under some arbitrary condition.

CONTENTS

Tables

Figures

ACRONYMS

ARP Address Resolution Protocol
ARPA Advanced Research Projects Agency
ARPANET ARPA Network
ARQ Automatic Repeat reQuest
ASK Amplitude Shift Keying
BASK Binary Amplitude Shift Keying
BCH Bose-Chaudhuri-Hocquenghem
BER Bit Error Rate
BSD Berkley System Distribution
CCITT International Telegraph and Telephone Consultative Committee
CERT Computer Emergency Response Team
CRC Cyclic Redundancy Check
DoD Department of Defence
EMC Electromagnet Compatibility
EHF Extremely High Frequency
FEC Forward Error Correction
FTP File Transfer Protocol
HDLC High-level Data Link Control
HF High Frequency
HTTP HyperText Transfer Protocol
IAB Internet Activities Board
ICMP Internet Control Message Protocol
IETF Internet Engineering Task Force
IP Internet Protocol
IRTF Internet Research Task Force
ISO International Standards Organisation
ITU International Telecommunication Union
JANET Joint Academic Network
LAN Local Area Network
LW Long Wave
MAC Media Access Control
MOD Ministry of Defence
MP3 Motion Picture 3 Audio format
MSS Maximum Segment Size
MTU Maximum Transmitted Unit
MW Medium Wave
NATO North Atlantic Treaty Organisation
NCP Network Control Protocol
OFDM Orthogonal Frequency Division Multiplexing
OSPF Open Shortest Path First
PC Personal Computer
POP3 Post Office Protocol version 3
PPP Point to Point Protocol
PRBS Pseudo Random Binary Sequence
PSK Phase Shift Keying
PSTN Public Switch Telephone Network
QAM Quadrature Amplitude Modulation
QoS Quality of Services
RFC Request for Comment
RIP Routing Information Protocol
RS Read Solomon
RSRE Royal Signals and Radar Establishment

RTO Retransmission Timeout
RTT Round Trip Time
SACK Selective Acknowledgements
SHF Super High Frequency
SMTP Simple Mail Transfer Protocol
SW Short Wave
SWS Silly Window Syndrome
TCP Transmission Control Protocol
TCP/IP short form of Internet Protocol Suite
TOS Type of Service
UDP User Datagram Protocol
UHF Ultra High Frequency
URL Universal Resource Locator
VoIP Voice over Internet Protocol
VHF Very High Frequency

1. INTRODUCTION

There is significant effort in the military to move toward a unified communication system across all strategic, tactical and combat areas. Many of the systems at the moment are bespoke and do not inter-operate with each other. This applies not only to UK forces, but also within NATO. For instance, there were several nations participating in peace keeping in the Balkans but only some of the systems were truly inter-operable. There are as many political problems as technical. Russia and US would still have limited information exchange even if their two systems could be plugged together.

The reason for the push for inter-operable communications is to enable a more effective force, and reduce the amount of resources needed to be deployed by each country. At the moment a large amount of the communications infrastructure used in NATO operations is of US and UK origin. There is a large amount of funding by NATO countries going into the development of gateways to provide connectivity between the two separate systems. This requires a standard interface that both sides of the gateway need to support.

Providing connectivity between two systems is one problem, allowing information exchange between applications on each side of the interface is another. Different systems even use different voice encoding standards. Most of the data systems are bespoke design for one purpose, hence are unlikely to inter-operate. Most of the day-to-day command and control data information is handled by e-mail or ftp. The data is normally in the form of word documents or power point presentations. These documents can be large and congest legacy systems. The UK tactical data network is supported by using a small number of fixed time slots in legacy circuit switching systems. The amount of data capacity in the trunk system is very limited and is only about 32 kbit/s per link (slower than modern standard home PC modem). Combine this with the fact that commanders are trying to transfer large files and the network will have a problem with congestion.

There is a major initiative to combine the data and voice network into one system to reduce the number of assets, manpower and network management. The voice would then be handled as another data application. This would mean the underlying system would be a data network not a voice network. One of the obvious choices is Internet Protocol Suite (IP suite). Most of the applications already use IP (e-mail and FTP) and there are many good reasons for using IP, but there may also be some problems.

The aim of this thesis is to look at how well TCP/IP would perform as a military communications system. However, just looking at TCP/IP in isolation would not give a clear understanding of how well it would perform. The Physical and Link layer are also considered. However network management and routing are not considered, as they are large subject areas in their own right. Some of the routing and management systems use TCP as a transport layer, therefore a study of TCP is needed before considering the effects on these other protocols. IP network is also capable of carrying real-time traffic, but the important issue covered in the thesis is the reliable transport of data, hence the thesis concentrates on the performance relating to getting information from one point in the system to another, rather than real-time issues to support multimedia applications. The effects of the link's characteristics and link protocols are also considered to see how they might affect this information transfer.

1.1 Thesis Outline

Several different areas of work are covered by the chapters in the thesis. Chapter 2 covers general background information. Chapter 3 covers the technical details of TCP. The following chapters give more in depth studies of particular areas. Each of these chapters is broken down further into background, results and conclusions relative to the subject being covered. Chapter 4 covers the effects at the physical layer such as bit error patterns. Chapter 5 details the implementation of a new error model. Chapter 6 covers the effects on link layer protocols, such as the effect of bit errors on data packets and the use of FEC and ARQ to combat these problems. Chapter 7 covers TCP/IP, and looks at the performance of TCP under the conditions generated by the link layer. Chapter 8 looks at possible improvements to TCP. Chapter 9 draws conclusions from the results and makes recommendations as to what can be done to get the best performance from a system based on TCP/IP.

1.2 Motivation and Objectives

I have spent several years working in the field of military communications. Some of the presently fielded equipment is getting old and has over run its intended service life. There is pressure from command officers to support services such as Email. IP systems are being shoehorned into the existing equipment. From talking to users and network administrators it was clear that this shoehorn approach was not delivering an acceptable performance. My motivation was to look at the problem in detail to find what was causing the problems. The main objective was to be able to make list of recommendations that if applied would deliver the best performance from a standard IP network. I took this a little further to try and find a way of modifying TCP without breaking interoperability, which could improve the performance even further.

1.3 Original Contributions

This thesis covers a number of topics and extends existing knowledge and understanding in a number of areas. At the physical layer an approach used to look at error statistics is extended to look at conditional bit errors (Chapter 4.2.1, Appendix A). The analysis of conditional errors makes it possible to distinguish between random and environmental noise and man made errors. The man made errors could be caused by the design of the radio or interference such as pulse errors. During the study of the bit errors from two different radios there was a clear indication that the statistics from one of the radios did not match the statistics from standard errors models. A new error model was produced to match the error statistics from this radio (Chapter 5).

A study was completed to look at how physical layer errors affect link layer processes. The author was unable to find existing information on the subject of interaction between bit errors and link layer. Part of the study looked at the effect of burst errors on packet dropping statistics (Chapter 6.3.1). The other area was the way burst errors had different effects on forward error correction (FEC) protected packets and non-protected packets (Chapter 6.3.2).

A lot of the information about the transmission control protocol (TCP) is relative to the performance under congestion or over long delay links such as satellite. There is very little material on the performance of TCP in high error environments. Chapter 7 is a detailed look at the performance of TCP when subjected to corruption (Chapter 7.5.1, Chapter 7.5.3), the interaction between different TCP algorithms (Chapter 7.5.2), and the effect of pulse errors (Chapter 7.5.3). From the results in this chapter a number of recommendations are made to get the best performance from TCP in a high error environment (Chapter 7.6.1).

From Chapter 7 it is clear that there is room for improvement in the TCP algorithm for dealing with corruption. Two approaches were taken to alter TCP, a modification to TCP Vegas (Chapter 8.1.2) and using Packet Pair measurements (Chapter 8.1.4). The results for these algorithms were compared against the optimum configuration of TCP from Chapter 7 and unmodified TCP Vegas (Chapter 8.3). A study was also carried out to look at the effect of automatic repeat request (ARQ) on the performance of TCP. This showed that there are interaction effects between various TCP algorithms and the ARQ (Chapter 8.3.4).

2. BACKGROUND

2.1 Basic overview of military communications

The Military Network can be split into three areas, strategic, tactical, and combat. The strategic network is normally very static and uses landlines or dedicated satellite links. The combat area consists of multiple fragmented components and normally has to be lightweight and mobile. In between these areas is the tactical. The tactical area provides command and control, logistic support, etc. It provides a means of communication between separated combat areas, and a link from the combat area back to the high level strategic decision making process. The tactical network needs to be able to move, deploy quickly and provide reliable communication to a large number of users.

To provide mobility and rapid deployment the separate nodes in the network are connected with radio links of some description (Figure 2-1). The backbone of the system normally uses line of sight, point-to-point links, in the UHF/SHF band. These do not have to be truly line of site, they can still work through light woodland. Satellite links are normally used to connect the tactical trunk network back to the strategic network. Omni-directional radios are used to connect to the mobile combat area.

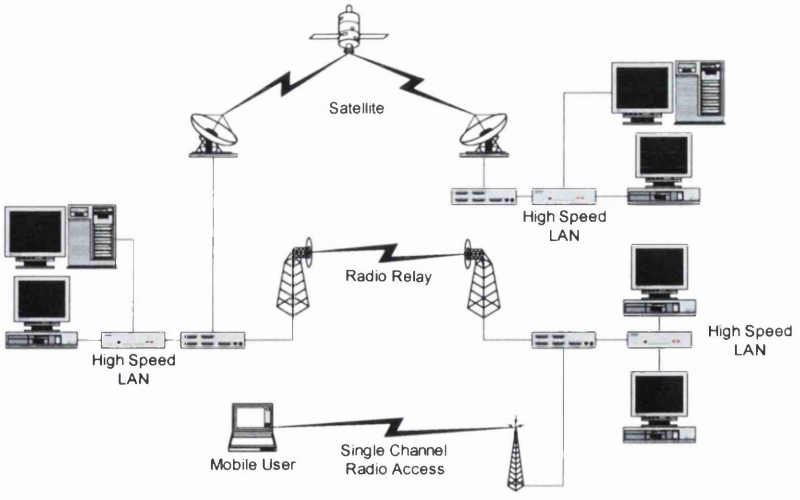


Figure 2-1 General Network Architecture

This means communication between different areas could pass across multiple different link technologies; from high-speed reliable fibre in a command centre to a low capacity intermittent HF radio link to a mobile unit. The communication protocols need to be able to handle connections that change from one technology to another.

The NATO forces now undertake more peace keeping roles than large scale war that the systems were original designed for. The line between strategic, tactical and combat starts to blur. It is no longer possible to draw a geographical front-line and hence this has an impact on the communications. The trunk network now becomes important to provide reliable communication.

2.1.1 What makes military and commercial systems different

One of the major differences between military and commercial systems is in the network architecture. A commercial system such as a telephone network has a large number of users on low bandwidth circuitry connected with a high capacity backbone. Even on the Internet most users are connected though relatively low speed modems. A military network will have users on high bandwidth circuits connected with a low capacity backbone. This makes congestion management a problem in military networks. Commercial networks use call tariffs to control the amount of use users make of the network. This is not an option for military networks. Military tactical networks use unreliable communication links, without the enemy purposely causing interference. A commercial network would just invest money to provide better communications such as using underground fibre connections. Tactical systems are dynamic, as network components are always being added and removed, sometimes on purpose sometimes not. The military cannot have prolonged downtime while the network is changed. Commercial systems can be shut down at points of low usage such as a weekend or overnight, which can be planed to minimise disruption. The military do not have this luxury.

Another issue that differs between commercial and military systems is security. The military impose a far more stringent security policy. There are several types of security: physical security to stop access to equipment, access security to prevent unauthorised access to the network and services, encryption to prevent information from being understood even if the encrypted data can be accessed, profile hiding to prevent determination of key assets such as command posts. Due to the sensitive nature of some of the military communication security it will not be covered in this thesis.

2.1.2 Military System Requirements

The military networks need to support varied services, from real-time video to simple file transfer. Some of the services require complete uncorrupted data, others such as voice can handle a percentage of corruption. Some services require timely delivery. Some of the services may depend on the actual situation (mark with 'O' in (Table 2-1)). For example the exact location of an enemy position will need to be accurate and timely so that the enemy can be caught before they have chance to move. The position of a friendly Land Rover travelling between bases may only need updates every half an hour. When services are competing for network bandwidth the more important service needs to be given preference. In legacy systems the primary service is voice, most of the command structure was based around it. The legacy systems provide pre-emption by senior staff, meaning the users have control of the importance of the information. These types of features are only just beginning to appear in commercial systems. There is a big effort by IETF groups to produce Quality of Service (QoS) controls in the Internet. These are being looked at to provide priority controls for military systems.

Services	Uncorrupted	Timely
Map data	✓	
Pictures	O	
Voice		✓
Video		✓
Logistic	✓	
Report	✓	
Sensor data	✓	✓
Commands / Orders	✓	✓
Location Updates	✓	O

Table 2-1 Types of Services

The military systems not only need to support these services while the network is stable but also while the network is changing. The network needs to be able to recover quickly from component failure. This requires a distributed system, with no single point of failure. In the Internet if a single name server was used and failed, then no connection can be established using a Universal Resource Locator (URL), even though the packet network could still be working. All data such as name servers, routing tables and informational databases need to be distributed across the system reliably. The network needs to be a mesh so that a single link failure does not cause network fragmentation. If network fragmentation does occur, the fragments must be able to act autonomously.

2.2 Why use TCP

The most important point to cover is why use TCP/IP. What does it offer, does it have any obvious problems. TCP/IP was developed as part of the ARPANET funded by the US Department of Defence (DoD) Advanced Research Projects Agency (ARPA). The original aims of the program were to share computer resource across a wide geographical area [Leiner00]. The development of the protocols had to deal with network failure. This was not a goal but more a necessity as computers and communications were unreliable. The protocols can handle network component failure, but assume the communications are stable if available. The development of the protocols had to handle the interconnection of separate networks all running different protocols. The Internet is a network of heterogeneous networks. The Internet protocol suite provides a standard mechanism for getting dissimilar systems to communicate with each other. Why reinvent the wheel, TCP/IP already provides the mechanism for providing interoperable communication.

The TCP/IP protocols were built into the development of the Berkley System Distribution (BSD). BSD was released with source code, (the original open source software). This allowed anyone using BSD to suggest changes to the software and protocols. The protocols were developed as part of the operating system, hence were designed as an integral part of the system rather than an add-on. Almost all modern operating systems have some form of TCP/IP built in, from UNIX main frame systems to Windows on a home PC. It is even possible to get TCP/IP support for embedded systems. This makes it possible for any nation to build an interoperable system if TCP/IP is chosen as the common protocol.

The Internet protocol suite is not a fixed protocol, it continually benefits from extra features added. TCP/IP was developed as a research tool. It provides enough scope and flexibility to allow the existing protocols to be extended. The Internet Engineering Task Force (IETF) is the standards body that controls the addition of new features to make sure old and new protocols can inter-operate. The IETF is self and US Government funded, meaning no one company can force the direction of the development. The standards are free to look at unlike the ITU and CCITT standards. This means that anyone from the large corporate company and governments to self-funded researches can access and suggest changes to the protocols.

The Internet protocol suite is ideal for military inter-operable communication, as it is an available open standard designed to provide intercommunication between different types of network technologies. It has the additional bonus of already having support for network component failure. The potential problem is that the original design of the Internet protocol suite assumed that the underlying communication system could provide reliable communications links. TCP was not added to the protocol until the system had been tried across a radio network [Kuo95]. The protocol has a very simple error checking process that is not strictly enforced. Most of the recent design changes to the protocol are to deal with congestion not corruption. This does not mean that TCP does not work. The amateur packet radio network runs a TCP/IP network with its own IP address space [Taurus][RFC790]. Research needs to be done to show the limits to the level of corruption that TCP/IP can acceptably handle.

2.3 Modelling Techniques

There are three ways to study networks: analytical modelling, simulation and practical testing and characterisation. All these techniques have their pros and cons.

Practical testing and characterisation will give the exact result as long as the measuring process does not affect the working network. It can sometimes be quite difficult to be completely passive. Normally in an investigation there is a goal, e.g. to show how a network responds under certain conditions. To study this, the conditions have to be generated in the system. This in its self is not going to be passive. To be completely passive means waiting until the condition occurs naturally. This means recording and processing vast amounts of data waiting for the event to occur. Introducing artificial conditions in an operating system may affect the users on the system. Having an isolated system is one way to prevent interruption to users, but now the network is not being used in the normal manner. Building large isolated test network requires a large number of assets and is expensive. Testing with real equipment is the only way to guarantee a particular process works in the real world. The real world suffers inaccuracies such as the accuracy of clocks and tolerances of manufacturing.

The other extreme is analytical modelling. There are different types of models from Cartesian mathematics to Flow modelling. Normally to build an analytical model requires making many assumptions in order to simplify the model to something that can be studied. The simplification means the model will not give exact results, compared to results from the real system. The analytical models give a feel for how the real system would behave.

Simulations give a good balance between analytical models and real system measurement. Like analytical models, it is sometimes necessary to make assumptions. The assumptions do not have to be as drastic. The assumptions in simulations tend to be timing related. The aim of a simulation is to mimic as closes as possible the real system. The advantage of simulations is that there is more control than in a real system.

It is possible to give some level of confidence that the results are representative by comparing the results from three different approaches. The results can only be accurate for the given condition, the important point is to use representative conditions.

2.3.1 OPNET

The chosen simulation environment is OPNET (www.opnet.com). OPNET provides a graphical interface for laying out and connecting different network components. There are pipes and packet stream handlers used to interconnect different components, and there are mechanisms for logging events and recording statistics from the simulation. There are three layers to the model: network, node and process. The network layer is used to interconnection different network components such as terminal and switches with links (Figure 2-2). The network components have a node layer. This layer shows how the individual processes are interconnected (Figure 2-3). There is a selection of special processes to provide interfaces with other nodes. These include packet senders and receivers. There are queuing and processes for controlling the actions of the node. These processes have a state diagram that controls what actions are taken in a particular event (Figure 2-4). The individual states have 'C' code functions that define the actions. The actions are 'C' calls to OPNET specific functions. There are a great number of functions including queuing, timing and logging.

Figure 2-2 Network model

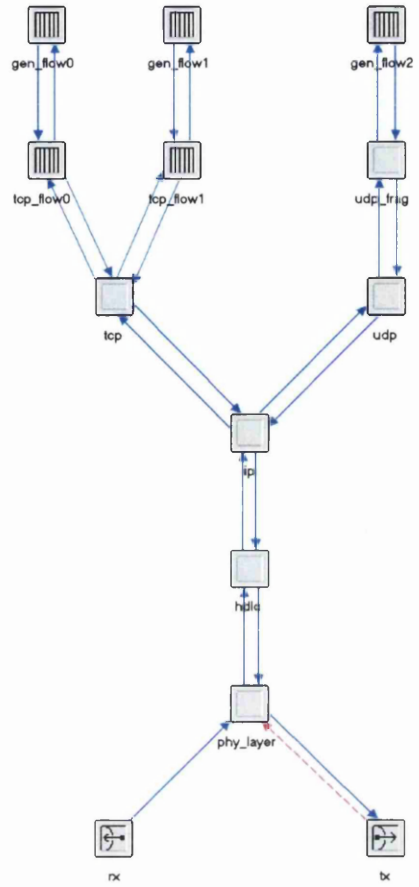


Figure 2-3 Process model

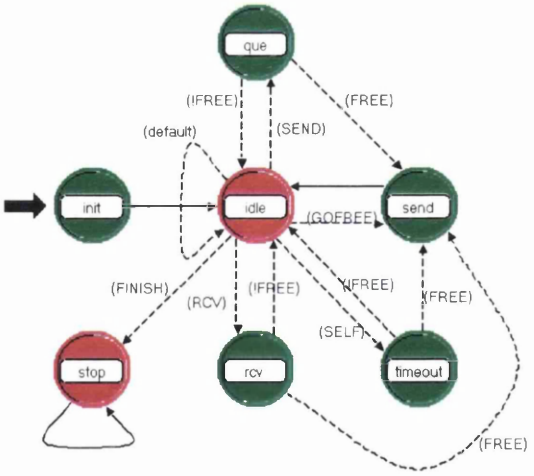


Figure 2-4 State diagram

OPNET uses an event driven model rather than time driven. All the events are queued, and handled in time order. On handling an event the appropriate process is triggered. The handling of the event can cause further events that are queued and handled in the appropriate order. The simulation time of an event is based on when it should occur. The simulation time only increments for time delay events such as timers and transmission across links. The simulation time does not increment for processing of events. Hence if an event causes itself to be triggered with zero delay, the simulation will continuously run the same event and time will never progress.

2.4 Conclusions

TCP/IP seems a good candidate for providing inter-operable communications over the multitude of technologies that are used in a military network. This assumption needs to be addressed. The performance has to be evaluated and any potential problems have to be assessed. If any problems are identified they can be analysed in order to identify any solutions that would work in a military context.

The approach chosen to assess TCP/IP is a mixture of real measurements, analytical models and simulations. The simulation tool chosen is OPNET as it provides a wealth of functions for building and running tests on packet-based networks. Analytical models are used as a check to aid in the validation of the simulations and to provide a means of reproduction statically equivalent to the real world measurements.

3. THE INTERNET PROTOCOL

3.1 Introduction

3.1.1 Brief history of The Internet

General reference material is taken from then following: [Cerf93] [Cerf00] [Roberts00] [Zakon00a] [Zakon00b] [Leiner00].

The Internet started as a small test bed to look at ways of building a packet network. The project was funded by the American Department of Defence (DoD) Advanced Research Project Agency (ARPA). J.C.R. Licklider spelt out the use of computers in the real world and the way they could interact with humans [Licklider60], this spurred others including Larry Roberts to try and build a network of computers. The original designs for the ARPA Network (ARPANET) where published by Larry Roberts in 1967 [Roberts67]. The first part of the network connected in 1969 consisted of 4 nodes, UCLA, Standford Research Institute, University of California Santa Barbara and University of Utah. The first packet radio network (ALOHAnet) developed in Hawaii was added to the network in 1972. At this time Network Control Protocol (NCP) [RFC60] did not support reliable data transfer over radio, so work was started on what became Transmission Control Protocol (TCP) [CerfKahn74]. The first international connection to ARPANET was in 1973 to University College London (UCL) via satellite link from Norway. By 1975 some of the fundament protocols had started to appear, the File Transfer Protocol (FTP) and Email was also available. In 1975 TCP was tested over a double satellite hop from Hawaii to the UK. On the 26 March 1976 Queen Elizabeth II sent an Email from the Royal Signals and Radar Establishment (RSRE) in Malvern. In 1979 mobile packet radio network experiments were started.

On the 1 January 1983 the whole of the ARPANET changed from using NCP to TCP [RFC801]. In this year part of the ARPANET was split off into the MILNET which later became part of the US Defence Data Network. The UK Joint Academic Network (JANET) that now connects most UK Universities was connected in 1984. In 1986 both the Internet Engineering Task Force (IETF) and Internet Research Task Force (IRTF) where formed under the Internet Activities Board (IAB). In 1987 the Internet suffered from several congestion collapses [Jacobson88]. This brought about a major change in the TCP protocol [Jacobson88]. On the 3rd of November 1988 an Internet worm virus [Seeley88] written by Robert Morris caused a major system failure. This caused the creation of the Computer Emergency Response Team (CERT) which announces vulnerabilities in networking software, the first advisory being Morris's Internet worm. The 1980s saw a great expansion in the Internet with networks being added by academic and research institutions, business and governments from all around the globe. Due to the large investment and expansion of the Internet (Figure 3-1)[Zakon00a], the original ARPANET was decommissioned in 1990.

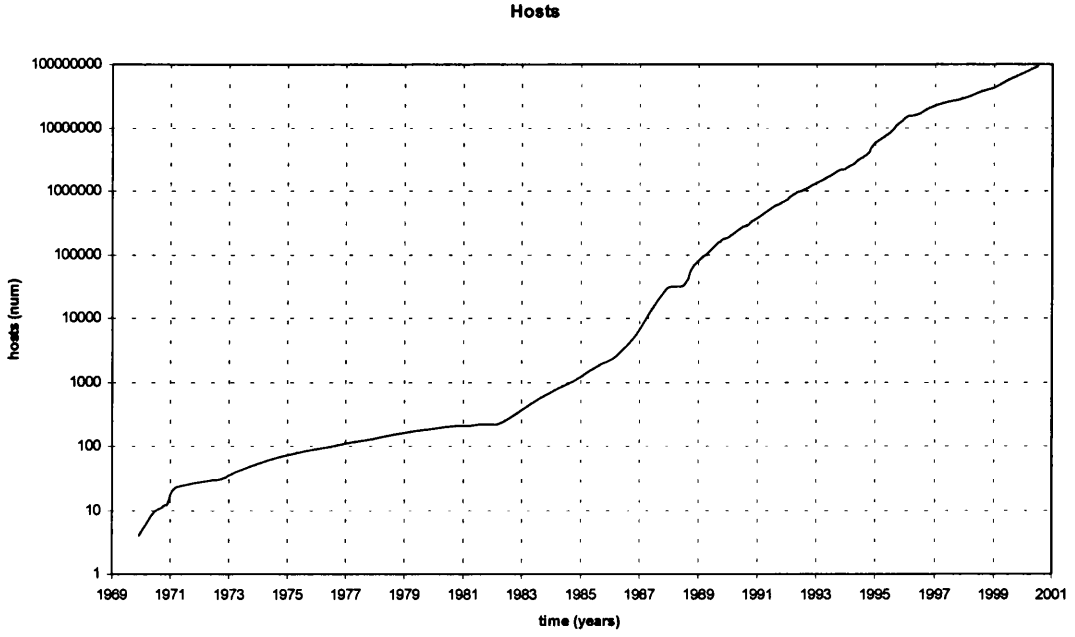


Figure 3-1 Internet growth

3.1.2 Initial goals of the ARPANET

At the start of the ARPANET program the initial objectives were for a resource sharing computer network [Roberts70]. The aim was to provide remote access to the limited number of computing facilities that were available at the time. In this process, techniques were developed and experience was obtained in interconnecting computers. The protocol development was part of a large investigation of an approach to open-architecture network environment. Bob Kahn set some ground rules to work to [Kahn72]. Each distinct network should be self-contained and be able to connect to Internet with internal modifications. Communication would be best effort retransmission for the source to recover lost packets. A black box would be used to connect separate networks. The boxes would be kept simple by not storing information of the state of the data flows. There would be no global control.

This was a purely research objective there were no military oriented objectives. This can be seen by the fact there was no dynamic routing. This was not available until 1988 when the Routing Information Protocol (RIP) [RFC1058] was introduced. The first data transfer protocol Network Control Protocol (NCP) had no reliability built into it. It was assumed that interconnection would be 100% reliable. However, it was not until the work with packet radio was started that it was realised that a more resilient protocol was needed.

Because the ARPANET program was a research study involving several different groups, information was readily passed between lots of different people. This openness was possible due to government funding rather than corporate business. Corporate business would have kept the knowledge in house to give them a competitive edge. Major changes to the protocols could be made because it was a research program not a commercial network with paying customers.

Berkeley Software Distribution (BSD) played a leading role in the extensive use of TCP/IP. BSD were more interested in developing a processor architecture independent operating system [McKusick99]. This made it an ideal candidate for the ARPANET program as the different research sites were using different hardware. Rather than pay the vast cost to make all the sites consistent with the same hardware the unification was made at the operating system level. BSD was asked by the ARPANET program to provide the operating system and network software. The BSD software was always released with source, this gave users the ability to feed their own bug fixes and improvements back into the system. In 1989 BSD released the networking software under a free licence. This gave anyone the ability to add TCP/IP network to any other operating system. It was this simple fact that led to the vast number of systems that can be networked together. BSD eventually released most of the operating system used, with a similar free licence.

Berkeley Software Distribution (BSD) released their software with their version of TCP, and has done so ever since the ARPANET program was started. The most commonly used version at the beginning of the ARPANET was BSD 4.1. This was, until recently, the most commonly used distribution until BSD 4.3 Tahoe was released. There are several open source distributions of the BSD software (OpenBSD, FreeBSD). These are not just TCP software but whole operating systems. The main objective is to be the most secure operating system available [OpenBSD].

3.2 Basic Structure of TCP/IP

The TCP/IP stack is divided into different layers (Figure 3-2). The main protocols are the Internet Protocol (IP), User Datagram protocol (UDP) and Transmission Control Protocol (TCP). The Internet standards also cover protocols such as File Transfer Protocol (FTP), Telnet and Hypertext Transport Protocol (HTTP). HTTP is normally implemented as part of the application, whereas UDP, TCP and IP are implemented in the operating system.

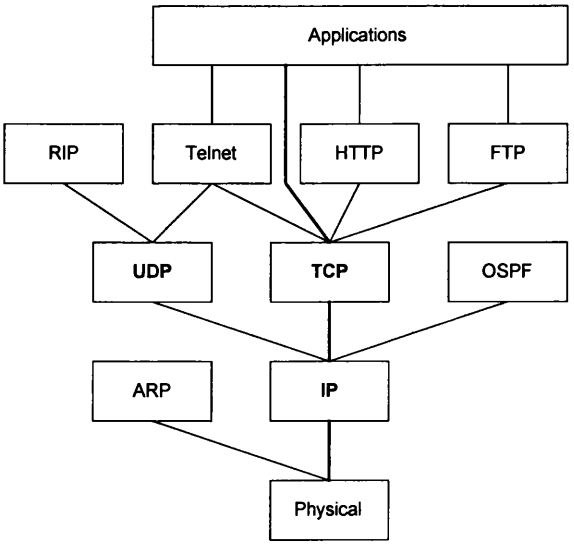


Figure 3-2 Internet Protocol stack

There are other supporting protocols to make the network work, Routing Information Protocol (RIP) and Open Shortest Path First (OSPF) for dynamically routing the IP traffic. The Address Resolution Protocol (ARP) is used to search for the hardware address (physical layer) that matches the required IP address.

The part of the stack that is of interest is from the application through TCP and IP. This is the protocol that most applications use to provide reliable transport of data between two end terminals. It is necessary to look at the details of both the IP and TCP protocols. TCP provides the mechanism for implementing a reliable data transfer and is considered in detail.

3.3 The IP Protocol

The Internet Protocol (IP) [RFC791] is responsible for getting the data from the sources to the destination. The IP header has fields that contain information used by intermediate nodes to get the packet to the destination in an appropriate manner. IP is stateless, nodes do not hold information relative to individual flows. This reduces the memory and process requirements of individual nodes. This means each packet has to hold all the information required to get that packet to the destination.

3.3.1 Routing

Internet traffic is routed based on the destination address. A router uses the destination address and other information to find the best route in its routing table. Simple routing looks for a matching address in the routing table, and forwards the packet through the appropriate port indicated in the table. The match does not have to be exact. The hierarchical structure of Internet address [RFC1519] means addresses can be aggregated into a single match pattern using wild cards and sub-addressing.

Other information useful for routing is in the Type of Service (TOS) field. This indicates whether the packet is delay sensitive or part of a bandwidth demanding connection. The router can use this information to use alternative routes or apply priority control.

3.3.2 Maximum Transmitted Unit

Data networks normally have a limit on the largest packet that can be transmitted to the Internet protocol suite. This is known as the maximum transmitted unit (MTU). The default size is that of the connected interface, for Ethernet this is 1.5 kbytes. Other links in the network may only support smaller packets. Routers must support packet sizes up to 576 bytes [RFC1122]. It is optional to support large packets, though most will support Ethernet size packets. There are two mechanisms for handling large packet over a link that uses small packets, these are fragmentation and an IP flag called don't fragment.

Fragmentation is the process of breaking a single IP packet into several small packets. The small packets are sent across the small packet link and continue through the network as small packets. The destination is responsible for recombining the fragments back into a single packet. The fragmentation process is inefficient. A source can prevent fragmentation by setting the do not fragment flag.

If a node needs to fragment a packet but the packet has the do not fragment flag set, the node discards the packet. The node then sends an Internet Control Message Protocol (ICMP) [RFC792] packet to the source indicating that a packet was discarded because it was too big. The source can then adjust its MTU for that destination and use smaller packets [RFC1256].

3.3.3 Internet Protocol version 6

There have been some major changes to the Internet protocol. These changes are to the format of the header, hence a new version is needed [RFC1883]. The reasoning behind the changes is to support large faster networks, with the addition of control for quality of service (QoS). The changes include increasing the address space from 32 bits to 128 bits (large enough to address every atom on the surface of the earth). Some of the fixed fields have been changed to options, one of these is the fragmentation information. The addition of a flow label field is to be used for quality control processes such as differentiated services. The flow label is to be used to identify a packet belonging to the same connection. The intermediate nodes can then identify and handle flows independently.

3.4 The Transmission Control Protocol

The Transmission Control Protocol (TCP) [RFC793] provides end-to-end flow control and reliable data transfer using an automatic repeat request (ARQ) scheme. The transmission window is counted in octets (bytes), rather than in packets. The TCP segments (packets) carry the sequence number of the first octet in the segment. The major difference between TCP and most link ARQs is that the timeout period and window size are both dynamic. The protocol uses information from previous transmissions to work out the best values for both. When TCP was developed there was flexibility built in so that the protocol could be extended. There are variable size options that can be added to the TCP header. These options carry extra information need to implement different control algorithms.

3.4.1 Reliable Data Transfer

Reliable data transfer is achieved with sequence numbers where the receiver keeps track of the next expected sequence number. When the correct sequence number is received, an acknowledgement is sent back indicating the next expected sequence number. This is used by the sender to determine how much data has been received correctly. TCP uses cumulative acknowledgements. Any data before the sequence number in the acknowledgement is assumed to have been received correctly. Even if it covers several segments worth. This can happen if an acknowledgement is lost. If the sender fails to get the correct response, the unacknowledged data can be retransmitted. TCP has several mechanisms for detecting corrupt or lost data, namely, duplicate acknowledgements and selective acknowledgements. The fall back mechanism is a timeout.

One of the important values to get correct estimation for, is the timeout. If the timeout is too long there will be extra delay before corrupt or lost data can be recovered. If the timeout is too short unnecessary retransmissions occur. Both of these cases can create a reduction in performance, one from not keeping the connection full of data, the other causing congestion. On a retransmission the value of the timeout is doubled and the retransmission rate is reduced to prevent congesting the network

3.4.2 Timeout Estimation

The original TCP algorithm for estimating the timeout is based on a smoothed average of the Round Trip Time (RTT). The RTT is measured by the delay between sending a segment and receiving an acknowledgement. The Retransmission Timeout (RTO) is calculated from smoothing the RTT. Two constants are used in the algorithm, α the smoothing constant and β the scaling factor (3.1), where α has a value of about 0.8 and β has a value between 1.5 and 2.

$$RTO_{n+1} = \alpha \cdot RTO_n + (1-\alpha) \cdot RTT$$
$$Timeout = \beta \cdot RTO \quad (3.1)$$

The smoothing process means that when the RTT increases quickly (e.g. when there is an increase in congestion) the RTO will lag slightly behind. This means the RTO value will be too small, and if used directly may cause unnecessary retransmissions. β is needed to make the timeout slightly larger than the smoothed RTT.

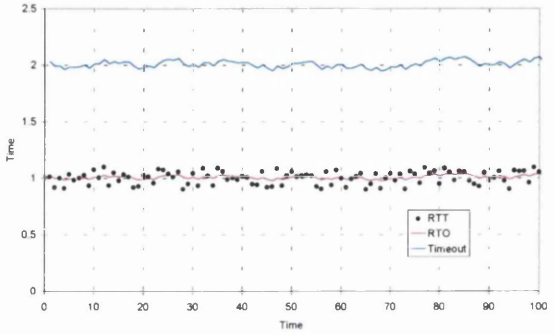


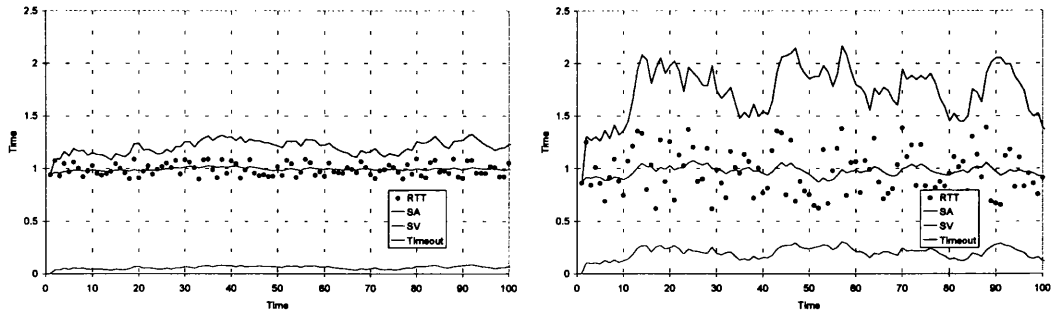
Figure 3-3 Original Timeout Calculation

The first algorithm over estimates the timeout value (Figure 3-3), because of this the process was changed to a second order average [Jacobson88](3.2). The constant α is 1/8 and β is 1/4. This algorithm can be implemented in integer arithmetic (Code 3-1). This new algorithm is better in that a lower value for the timeout is used without it being smaller than the RTT. The use of the variances allows the timeout value to adjust to different amounts of deviation in the round trip time (low deviation Figure 3-4(a), high deviation Figure 3-4(b)).

$$\Delta = RTT - sa$$
$$sa = sa + \alpha \cdot \Delta$$
$$\delta = |\Delta| - sv \quad (3.2)$$
$$sv = sv + \beta \cdot \delta$$
$$RTO = sa + 4 \cdot sv$$

```
m -= sa>>3
sa += m
if (m<0) m =- m
m -= sv>>2
sv += m
rto = sa>>3+sv
```

Code 3-1 Integer Timeout Estimation



(a) (b)

Figure 3-4 New Timeout Calculation

The algorithm is initialised to a known state when the connection is generated. The default timeout for the first segment is 3 seconds. The estimation state variables are initialised to give this value, the variance (`sv`) is initialised to zero.

Timeout estimation can only be made for unambiguous segments, i.e. segments that have only been sent once. If a segment is retransmitted, it is not possible to determine which of the segments triggered the acknowledgement. The time stamp option [RFC1072] was added to increase the number of measurements available to estimate the best timeout value.

3.4.3 Time Stamp Option

The time stamp option [RFC1072] [RFC1323] works by including a time stamp with every segment that is sent. Retransmitted segments include a new time stamp. The receiver echoes the time stamp back when acknowledging the segment. The sender can then make an estimate for retransmitted segments as the acknowledgement has a record of the time the received segment was sent, irrespective of which of the retransmitted segments was received. The time stamp value is from a monotonic clock. Different end systems can use different clock rates, as the other end of the connection only echoes the values back.

3.4.4 Maximum Segment Size

As already mentioned (Section 3.3.2), data networks have a limit to the maximum size of a packet. TCP uses the MTU to calculate the largest block of data that can be sent in a single segment. This is referred to as the maximum segment size (MSS). This value is the size of the largest packet (MTU) minus the size of the TCP and IP headers including any additional options. This gives a size of 536 bytes (default router MTU of 576 - 40 bytes for the headers) to guarantee the segments will not to be fragmented. The MSS is used to control the size of the packets and used in adjusting the control windows.

3.4.5 Window control

One of the major features of TCP is the way it controls the size of the congestion window. The window size is a prime flow control mechanism. The size of the congestion window represents the amount of unacknowledged data that is allowed in the network. The original TCP opened the window to an arbitrary size, and then did flow control. This causes congestion on the start of every connection. This was changed so that TCP starts with a window of one segment.

The congestion window prevents the sending of too much data into the network. New data can only be sent if there is space in the window. This space is only released for more data when an acknowledgement is received. As the connection proceeds the segments should be spaced in time and lead to the self clocked nature of TCP. The received segments trigger acknowledgements and the acknowledgements trigger the sending of new segments.

The congestion window is not the only variable that has control over the amount of data that can be sent into the network. The receiver sends information back to the sender indicating the amount of buffer space available for receiving new data. This is known as the advertised window and prevents the sender from overloading the receiver with data.

There are two stages to the flow control, one is slow start and the other is congestion avoidance [Jacobson88]. They increase the window in different ways to achieve the best performance. The dividing line between the two is the threshold. If the congestion window is less than the threshold slow start is used, if above then congestion avoidance is used. The threshold value changes with the state of the connection. The initial value is set artificially high, this prevent the connection being in the congestion control state when the connection is started.

3.4.6 Slow Start

As already stated the connection is initiated in slow start [Jacobson88] with a congestion window of one segment. During slow start, the congestion window is increased by one segment for every received acknowledgement (Figure 3-5). This means the amount of data in the network slowly grows as the connection proceeds. The inherent delay between packets on the network means the window of data is spread more evenly rather than being sent in a large single burst. The increase is actually exponential, which is not slow, but is slower compared to opening a large window all at once.

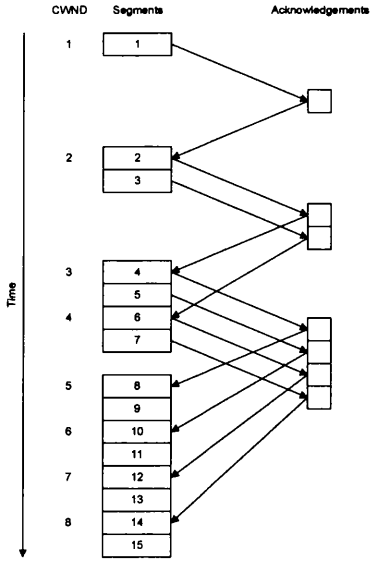


Figure 3-5 Slow Start CWND increments

3.4.7 Congestion Avoidance

In the congestion avoidance state [Jacobson88] the window is opened much slower than in the slow start state. This allows the connection to test for bandwidth without causing congestion. The aim is to open the window (*cwnd*) by one segment (Maximum Segment Size *mss*) per round trip time, or one segment for each complete window of data sent (3.3), rather than suddenly increased the window by one segment. The window is increase by a fraction of a window for each acknowledgement.

$$cwnd = cwnd + mss / cwnd = \frac{cwnd \cdot cwnd + mss}{cwnd} \quad (3.3)$$

3.4.8 Setting the Threshold

The aim of the threshold is to indicate the largest possible congestion window that does not cause congestion. The threshold is reset every time there is a retransmission. The threshold is set to half the current congestion window or half the advertised window which ever is smaller. During a retransmission, the congestion window is also reset. There are two main types of retransmission a timeout and a fast retransmission. These are triggered by different events, and have different control over the congestion window.

3.4.9 Timeout Retransmission

The timeout retransmission is triggered by a timeout. The response to this is to reset the congestion window back to one segment [Stevens96]. The window is then increased according to slow start until the threshold is reached. The threshold is now half the size of the original congestion window. On reaching the threshold, the window is increased according to the congestion avoidance algorithm (Chapter 3.4.7).

3.4.10 Fast Retransmission

Fast retransmission [Jacobson90] is triggered by receiving duplicate acknowledgements. These duplicate acknowledgements are generated by the receiver when out of sequence segments are received. After a segment is lost, all the following segments are considered out of order. The receiver still sends an acknowledgement but indicates the sequence number it was expecting, not the one received. The acknowledgements generated by receiving out of order segments are known as duplicates as they carry the same acknowledgement number.

Three duplicate acknowledgements are used to trigger the retransmission of the missing segment. As out of order packets are received, hence the acknowledgements, it can be assumed that the connection is not heavily congested. To avoid slowing the connection the congestion window is not reset to one segment. This mechanism is called fast recovery.

3.4.11 Fast Recovery

Fast recovery [Jacobson90] follows a fast retransmission. As in all retransmissions, the threshold is set to half the congestion window. New data cannot be sent, as the amount of unacknowledged data is most likely equal to the congestion window. The congestion window is set to the new threshold value plus three segments. The three segments are to account for the three segments that have already left the network, causing the duplicate acknowledgements. During the fast recovery state the congestion window is increased by one segment for consecutive duplicate acknowledgements. Each duplicate acknowledgement indicates that a segment has left the network. The increase in the congestion window allows one new segment to be sent into the network. This does not increase the congestion, as the amount of unacknowledged data in the network remains the same, because the unacknowledged segment prevents the window from progressing. When the missing segment is acknowledged the congestion window is set to the threshold. TCP then leaves the fast recovery state and goes back to normal flow control, starting in the congestion avoidance state.

3.4.12 Silly Window Syndrome

The silly window syndrome (SWS) [RFC813] is an effect associated with measuring the congestion window in octets rather than packets. The congestion window is not always a whole number of maximum sized segments. This leads to sending of segments that are less than the maximum. When a small segment is acknowledged this causes yet another small segment to be sent. If the send process is unable to keep the connection full, TCP will send small segments when they are not filled in time. The number of small segments increases, leading to a high overhead and heavy congestion.

Several modification were made to TCP to compensate for this problem; the Nagle algorithm and Delayed Acknowledgements alter the sending process and change the receiver advertised window.

3.4.13 Nagle Algorithm

The Nagle algorithm [RFC896] is a simple solution to prevent many small packets being sent when using Telnet. When using Telnet each key press could generate a separate TCP segment. The algorithm simply prevents TCP from having more than one small (less than MSS) unacknowledged segment. The first key press generates a segment, but the following keys are stored until either there is enough to send a whole MSS or the small segment is acknowledged.

3.4.14 Delayed Acknowledgements

The problem that causes silly window syndrome is that the available space in a window is updated very quickly with the slightest change. Delayed acknowledgements [RFC813] reduce the number of updates and increase the size the updates cover. A delayed acknowledgement waits a short time after receiving a segment to see if any more will be received. This allows the receiver to acknowledge more than one segment at a time. An acknowledgement must be sent if there is at least two MSS worth of data that needs to be acknowledged.

The delayed acknowledgement has other effects. It reduces the number of acknowledgement packets that are sent reducing the load on the network. The reduced number of acknowledgements means the congestion window does increase as quickly, again this helps reduce congestion. It also enables acknowledgements to be piggybacked on data packets. Most higher level protocols such as Hypertext Transport Protocol (HTTP) and Telnet, are request and response processes. The acknowledgement for TCP segment carrying the request can be sent with the response to the request. This again reduces the number of packets in the network and reduces the process overhead on the end systems.

3.4.15 Window solutions to SWS

Another approach to avoid SWS is for the receiver to only increase the advertised window when there is a large change [RFC813]. This avoids advertising small windows, hence stops the sender from sending small segments. The simplest solution to the problem is to actively prevent the sender from sending small segments and for the sender to wait until the window is large enough to send a full segment.

3.4.16 Selective Acknowledgements

Selective acknowledgement (SACK) [RFC1072] [RFC2018] is a TCP option that allows the receiver to give the sender more information about what data has been received. The receiver uses this option when out of order segments are received (Figure 3-6). The option includes indexes to the start and end of the out of order data blocks. The blocks could include a number of sequential segments. The sender can use this information to determine exactly what data has been received. Giving an accurate measure of how much data has left the network and indicating which segments need resending. This mechanism was introduced to handle multiple dropped packets from the same window. Fast retransmission can only handle one dropped packet per window without falling back to using timeouts.

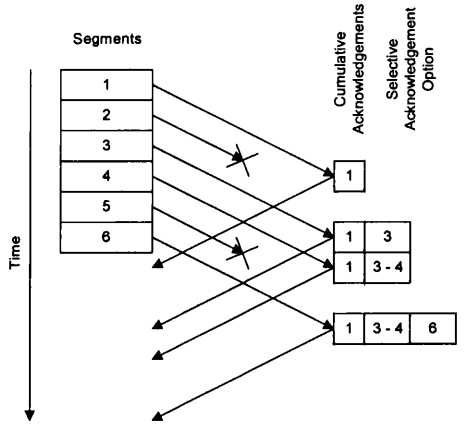


Figure 3-6 Selective Acknowledgement Option

3.4.17 Window Scaling

The maximum window size in a standard TCP segment is 65536 octets (64k). This is because the advertised window and sequence numbers in the segment are stored as 16 bit values. This is normally limited to 32k because some systems use signed instead of unsigned values. To solve this problem window scaling was introduced [RFC1072] [RFC1323]. The TCP option indicates the number of right bit shifts in the header sequence number. A value of 32 could be stored as 2 with 4 right shifts. This puts a restriction on the size of the segments. They need to be multiples of 2^n, where n is the number of right shifts. The bit shifting of the sequence can extend the congestion window up to 1G octets. This is useful for sending large files over a connection with high bandwidth delay products such as satellites.

3.5 Conclusions

Since the initial Internet there has been many modification and improvements to the TCP/IP protocol suite. Most of the major ones occurred before 1990. TCP/IP network is based around transferring data between computers on different networks, not about providing a communications system. The network support within an operating system is tuned to provide a usable interface to applications, rather than tuning the application to meet the requirements of the networking technology.

The layered structure of TCP/IP allows different types of protocol. TCP for reliable file transfer, unconditioned UDP for real-time data. The structure allows the use of raw IP packets, used by protocol such as OSPF. This allows for a mixture of protocols to work coherently on the same network infrastructure.

The best effort approach enables the use of different network technologies. The only functions that the Internet protocol assumes is packet framing and the ability to deliver the packet to the next part of the connection in the network. The underlying network does not need to provide reliable data transfer, hence TCP/IP can be used over error prone links such as a modem over a telephone line.

Most of the original modifications were implemented to improve the performance of the operating system. This was achieved by reducing the computational loading in the network components (Delayed Acknowledgements). The modifications over the last few years have been to improve the performance over newer fast technologies. These modifications include SACK and timestamp options that enable TCP to work over connections with high bandwidth delay products, such as satellite links.

4. RADIO CHANNEL CHARACTERISTICS

4.1 Introduction

The military use a mixture of link technologies. Most existing systems use cables for internal connectivity and there is a move towards using fibre for more bandwidth capacity, better electrical isolation and electromagnetic compatibility (EMC). Military Headquarters are normally wired together as a normal office, with Ethernet for computers and telephone lines for phones. Precautions are taken when laying cables to avoid EMC problems such as not running data cables parallel to power cables. As the military telephone lines are digital twisted pairs the bit error rates on the cables internal within a vehicle or within a headquarters is kept very low, no worse than a normal office.

Due to the mobility that is needed in tactical communication systems, the network relies heavily on radio links to interconnect the geographically separated nodes. The military use the full range of the frequency spectrum (Table 4-1), choosing the best available band for the purpose. Some bands are suited to long distances some to short. Frequencies above Very High Frequency (VHF) tend towards giving only line of sight connections. By placing the antennas on high ground, large distances can be covered, but they become obvious targets. Certain frequency bands have special characteristics. The Ionosphere refracts High Frequency (HF). This allows a signal to travel far beyond line of sight, in some cases to the other side of the world. The problem is that the Ionosphere is constantly changing, causing high levels of distortion and fading. The oxygen absorption band prevents signals around 60 GHz travelling very far before the energy is absorbed. This allows for high capacity short-range radio links that do not interfere with each other.

Band	Frequency Band	Line Of Sight	Typical Usable Ground Wave Distance	Typical Channel Bandwidth
LW	100 - 300 kHz		750 km	10 kHz
MW	0.3 - 2 MHz		100 km	16 kHz
SW/HF	2 - 30 MHz		150 km	3 kHz
VHF	30 - 300 MHz	✓+	100 km	10 - 5000 kHz
UHF	0.3 - 3 GHz	✓+	60 km	10 - 8000 kHz
SHF	3 - 30 GHz	✓	20 km	25 MHz
EHF	30 - 300 GHz	✓	5 km	25 MHz
O_2 Absorption	60 GHz	✓	500 m	2 MHz

Table 4-1 Radio Bands

Satellite links provide high capacity data links over a large distance. The problem is that they are very vulnerable, have long propagation delays and are expensive to build, launch and maintain. They are very useful for providing connection to very remote locations, but not very effective for a large number of independent users. The military have a high concentration of users in command centres that can provide ideal points to connect the network together. They also provide a sensible place to connect mobile users to the network. Point-to-point UHF/SHF radios are normally used to connect the command centres together. This provides both the range and bandwidth needed. Sometimes a relay point is needed to cover larger distances.

Most of the errors and data corruption happens in the network while the data is being transmitted across the radio links. This means the chance of corruption increases with the number of links the connection has to use. This is different from a mobile phone network or Public Switch Telephone Network (PSTN) where most of the corruption occurs at the users connection.

4.1.1 Radio Channel Models

The characteristics of a radio channel depend on the frequency band used. Different models have been developed to mimic the radio channel characteristics. HF models try to account for different effects such as multi-path propagation and fading. There are also predictive models that try to estimate the best time to send a signal. The number of sunspots correlates highly with the ability to achieve long range HF communications.

One of the simplest models is additive Gaussian distributed noise. This is normally used as a reference standard for comparing modulation schemes. Noise received by a radio is unlikely to be a Guassian distributed. Effects such as multi-path can cause the same signal to be received from different directions with different amplitude and phase. This gives noise that correlates with the signal whereas Gaussian noise is uncorrelated. The Rayleigh fading model tries to mimic these effects by using two independent Gaussian distributions one for frequency and one for phase. There are other models based on the Rayleigh model (Rician Distribution, m-Nakagami [Nakagami60]) that adds extra parameters to provide more control of the effects.

4.1.2 Modulation and Bit Errors

The bit errors at the receiver are affected both by the characteristics of the channel model and also the modulation scheme. The simplest data carrying modulation is Binary Amplitude Shift Keying (BASK), the MFS60 Rugby clock signal is BASK. It is also possible to modulate the frequency or phase. Some schemes use a mixture, such as Quadrature Amplitude Modulation (QAM). Providing more levels means the symbol rate can be reduced (Figure 4-1). This has to be weighed against the decrease in the noise margin between symbols. One way to do this is to plot the signal to noise ratio against the Bit Error Rate (BER). The signal and noise are normalised to 1 Hz to give E_b/N_0, where E_b is the signal energy for 1 Hz of the bandwidth and N_0 is the noise power spectrum density. This can be used to compare different modulation schemes (Figure 4-2). Shannon's Law [Shannon48] gives the maximum information throughput for a noisy channel. This gives the theoretical maximum achievable performance, a target to aim for when designing a new modulation scheme. Shannon's Law assumes Gaussian noise, this does not hold true for radio channels. This means the theoretical curves for the modulation scheme are also inaccurate. To get more accurate results either a correctly matched fading model is need, or alternatively the modulation scheme can be tested over a real radio link.

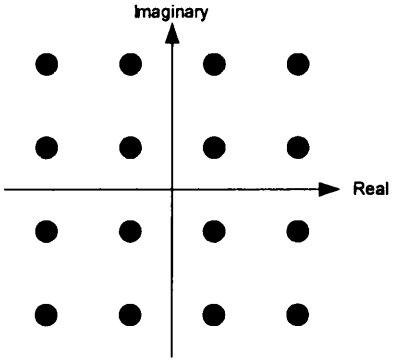


Figure 4-1 QAM 16 Constellation

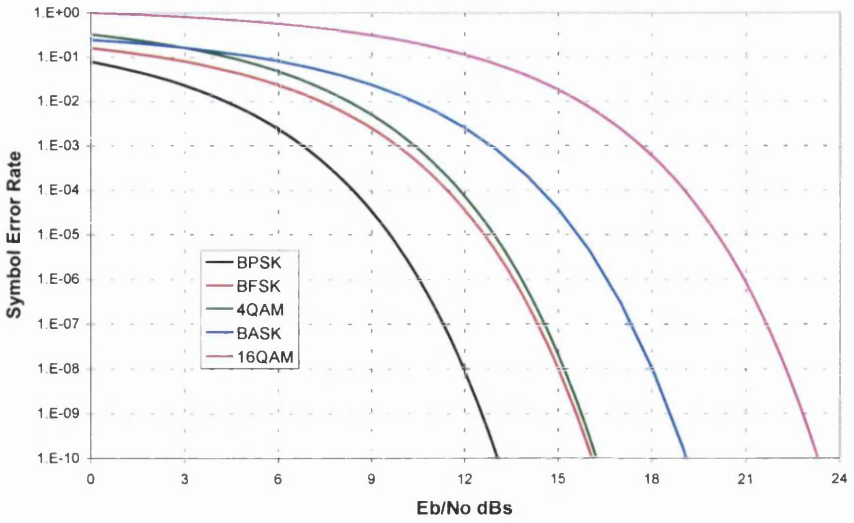


Figure 4-2 Comparison of modulation schemes

Bit errors are injected when the demodulator cannot correctly determine the transmitted symbol. This is caused by the noise moving the symbol beyond the recognition point for the transmitted symbol (Figure 4-3). BASK and Gaussian noise channels give random bit errors. With a fading channel more errors will be generated when the signal to noise ratio drops below a certain threshold and errors will be generated in-groups (bursts). Interference from other sources will cause different error patterns. A switch mode power supply could generate pulse interference every 10 or 20 ms. Inter-modulation effects would generate errors that also depend on the data being sent. The interaction between the modulation scheme and the type of noise makes it difficult to determine the actual pattern of errors.

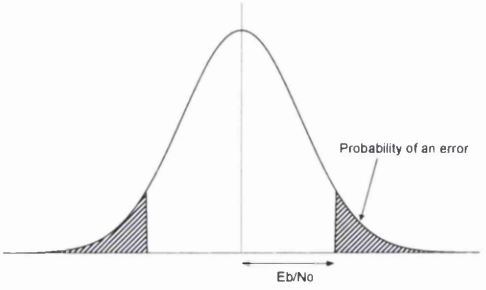


Figure 4-3 Probability of error with gaussian noise

4.1.3 Bit Error Models

There are different models to replicate different error patterns. The simplest model is pulse errors that are generated by injecting an error every N bits. This type of error pattern is normally only generated by man made interference. Random errors are more realistic for Gaussian noise. Random errors can be generated using a uniformly distributed random number generator and setting a threshold, the threshold being the same for each bit. A new random number is used for each bit, giving each bit the same probability of being in error.

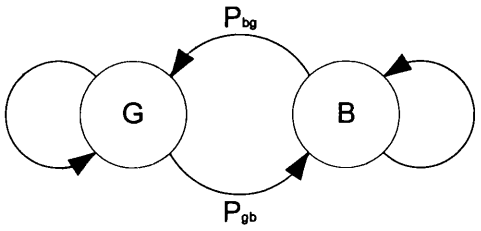


Figure 4-4 Gilbert & Elliot model

Research by Gilbert [Gilbert60] and Elliot [Elliot63], produced a burst error model that mimics the error patterns found on telegraph connections. The model consists of two states, a good state and a bad state (Figure 4-4). The probability of an error in the good state is 0, the probability of an error in the bad state is a 0.5. The pattern of errors is controlled by the probability of the state changing. The model can be extended by selecting the bit error probability for the two states. A set of equation can be derived to control the behaviour of the model (4.1). Keeping to the standard error probabilities per state (P_{eg}=0, P_{eb}=0.5) the model can be defined with two values, the size of the burst (P_w) and the average bit error rate (P_e). P_p can be derived with $P_w/2P_e$, hence all the parameters required to control the model.

$$
\begin{aligned}
P_e &= \frac{P_{eg}\cdot(P_p - P_w) + P_{eb}\cdot P_w}{P_p} = P_{eg}\cdot(1-D_c) + P_{eb}\cdot D_c \\
P_{gb} &= \frac{1}{P_p - P_w} \quad P_{bg} = \frac{1}{P_w} \quad D_c = \frac{P_w}{P_p} \\
P_{eb} &= \frac{P_e - P_{eg}\cdot(1-D_c)}{D_c} \\
P_p &= \frac{P_p = P_{eb} - P_{eg})}{P_e - P_{eg}}
\end{aligned}
\tag{4.1}
$$

where:
P_e is the average bit error rate
P_{eg} is the bit error rate in good state
P_{eb} is the bit error rate in bad state
P_{gb} is the probability of going from the good state to the bad state
P_{gb} is the probability of going from the bad state to the good state
P_p is the average spacing between burst
P_w is the average length of burst
D_c is the duty cycle

The Gilbert-Elliot burst error model is a two state Markov chain. A Markov chain is a series of states each with a fixed probability of moving to another state. Several researchers have used Markov chains to model particular channel models, such as the Rayleigh fading model [Chen99][Zorzi99]. The model was matched to data captured from a satellite link [ACTS]. The conclusion from most of these papers is that the Gilbert-Elliot model gives very good results. The increased accuracy when using a larger number of states is minimal.

4.1.4 Statistics from real links

Some researches have monitored links and looked at the statistics of the bit error patterns [ACTS][Ahrens00]. The results from a radio link in Germany show that there is a discrepancy between the Markov chain state models and the pattern of errors present on the link.

4.2 Error Patterns

The pattern of errors generated by the demodulator can cause different effects higher up the communications stack. Some FEC schemes are designed assuming random errors. Burst errors would cause the FEC to fail and possibly produce a lower throughput than expected. It is therefore necessary to study what types of patterns are generated.

4.2.1 Error pattern analytical software

What is needed is a way to show how the errors are distributed. One way is to look at the distribution of the spaces between the errors. Software was developed to analyse bit error patterns based on error spacings (Appendix A). The software takes a stream of errors and builds up a histogram of the number of occurrences of each error spacing. The software can treat each error spacing as an individual value or group error spacings together in a bin. The size of the bin can be selected to be linear or logarithmic. The software also builds a map of conditional probabilities for consecutive errors (i.e. if the previous error spacing was x the probability of the next error spacing being y is P_{xy}). This is for examining dependencies between errors. Several error models have been built into the software, as well as the ability to read error files captured from real measurements. This allows for the same analytical process to be used both on the models and real data.

4.2.2 Reference conditions

Figure 4-5 shows the distribution of errors (distance between errors) for a pulsed interference. The high peak at the beginning is caused by several errors during the error pulse. The peak at 300 bits is caused by the spacing between the pulses. Knowing the bit rate of the link it is possible to work back to the time interval for the pulse interference. The small peak at 600 bits is caused by some error pulses not causing any errors, as the error probability during the pulse error is not 1, hence double the time spacing between errors.

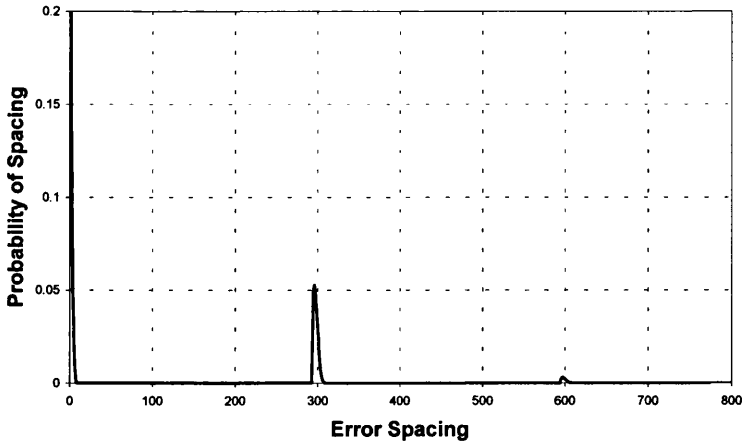
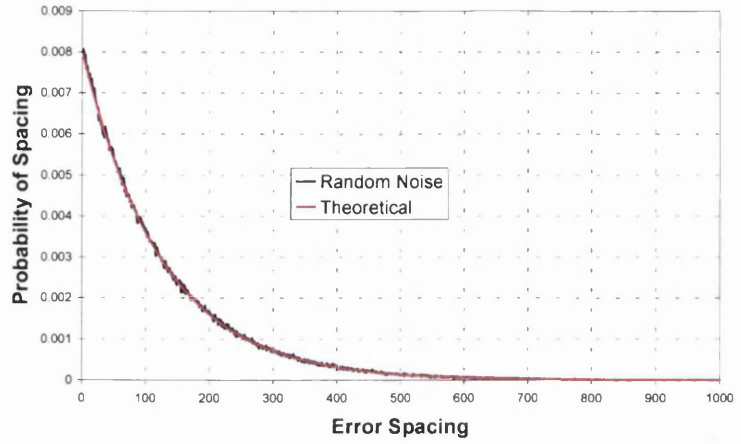
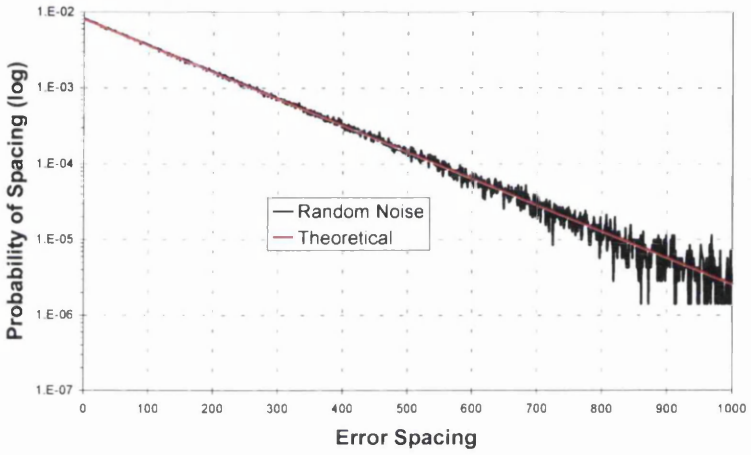


Figure 4-5 Distribution of pulse errors

For random errors (generated with Guassian noise) the spacing between errors varies from error to error. Random errors have an exponential distribution (Figure 4-6(a)). Figure 4-6(b) shows the same data with a vertical logarithmic scale. (4.2) gives the distribution function for random errors.



(a)



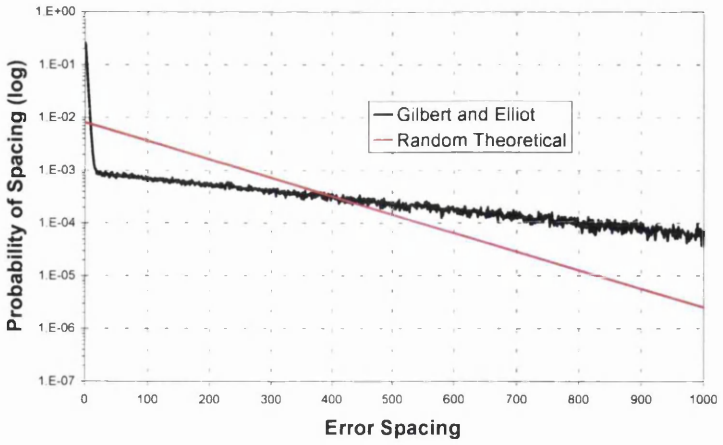
(b)

Figure 4-6 Distribution of random errors

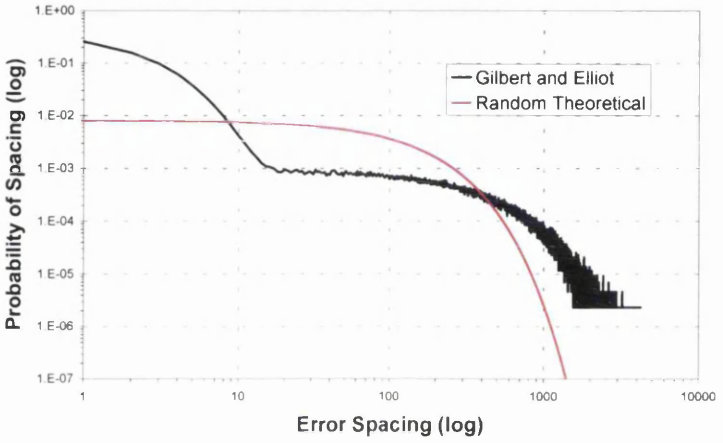
$$pdf(p) = b \cdot (1-b)^p \quad (4.2)$$

where p is the error spacing and b is the bit error probability

One of the standard burst error models is the Gilbert and Elliot model (Section 4.1.3). Figure 4-7(a) shows the distribution of errors where the peak on the left of the graph is caused by the burst. The roll off on the right of the Figure 4-7(b) is caused by the random change from the good to bad state.



(a)



(b)

Figure 4-7 Distribution of errors from Gilbert and Elliot model

The examples of pulse, random and burst errors show that it is possible to distinguish between different types of error distributions. What they do not show is where there are any dependencies between errors. This can be achieved by looking at the conditional probability of two consecutive errors. Figure 4-8 shows that errors from the Gilbert and Elliot model are independent of the previous error. The shape of graph along the error spacing axis is roughly the same for all previous error spacings.

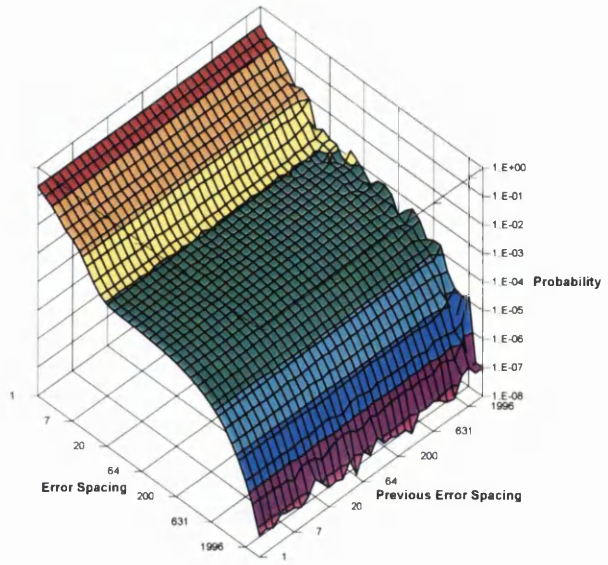


Figure 4-8 Condition probability of Gilbert and Elliot model

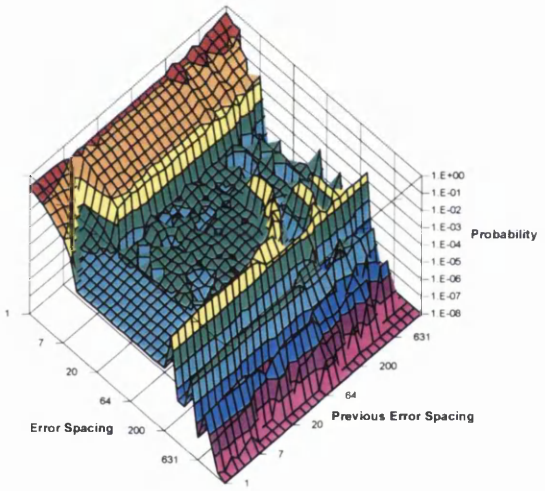


Figure 4-9 Condition probability of pulse mixed with random errors

Figure 4-9 shows that there is some dependence between errors for pulse errors. Random errors between pulses cause a curve in the ridge. The curve in the left most corner is caused by the pulse width. From these known patterns it should be possible to spot any ambiguities in results gathered from real systems.

4.3 Capturing Error Patterns

Most of the research regarding bit error models assumes a particular channel model or is based on results from cable or satellite links. There is a need to capture error patterns of real radio links. This research has access to results that were captured from a link using military radios, and some results from a development radio. The errors are detected by sending a known sequence of bits over the link and comparing the received sequence against the transmitted sequence. The sequence used was a Pseudo Random Binary Sequence (PRBS) of length 2^{15}-1. The stream of errors is then recorded and stored, and the sequence of errors can be replayed or processed to obtain statistics.

Due to the classification and commercial sensitive of the results it is not possible to identify the radios or their operating frequencies and modulation schemes. All that can be given is that there are designed to operate in the same role over roughly the same distance.

4.3.1 Development radio

The development radio was tested on a bench, using an RF attenuator and noise generators to adjust the signal to noise radio (Figure 4-10). This gives controllable test conditions that can be repeated. The radio path is isolated from any natural effects.

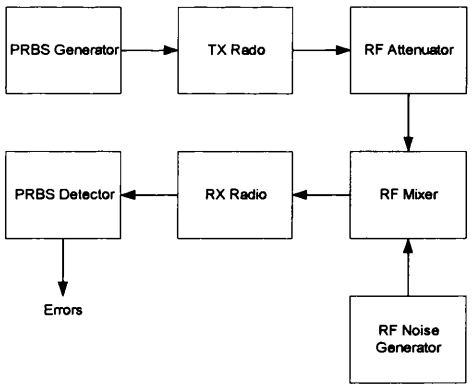


Figure 4-10 Development radio test

4.3.2 Military radio link

The link comprised two duplex radios across an 11.5-km point-to-point link with the received data at one end being looped back (Figure 4-11). This means that the error records are for two consecutive links. The transmit power can be adjusted to change the signal to noise ratio of the received signal. An attenuator between the antenna and the receiver is used to change the signal to noise ratio at the receiver.

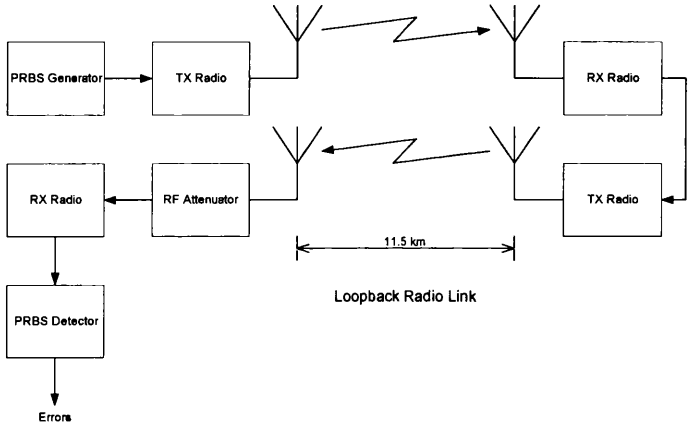
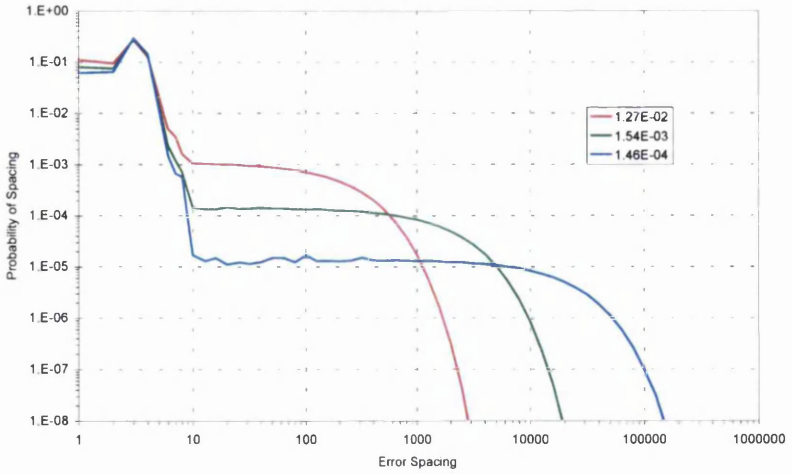


Figure 4-11 Military radio test

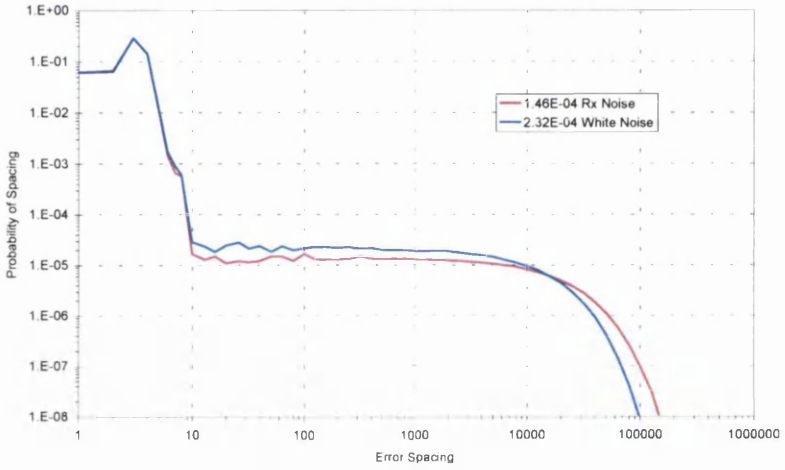
4.4 Results from real radios

4.4.1 Development radio

Figure 4-12 shows the distribution of errors for the development radio. The Figure 4-12(a) illustrates the result when the signal was attenuated into the receiver noise, and the Figure 4-12(b) shows the differences between receiver noise and additive white noise. The shape of the curve is roughly the same for a range of bit error rates and noise. The curves are similar to that of the Gilbert and Elliot model. There is a peak at a spacing of 3 bits that shows a high probability of errors that are separated by 3 bits. This is far too high to be caused solely by the noise. Figure 4-13 shows that there is a high degree of dependency between some of the bit errors. The flat area on Figure 4-13(a) indicates that there is unlikely to be an error with large gaps either side of it. Figure 4-13(b) shows a peak of nearly one, this indicates that a single error is guaranteed to be followed by two other errors with the same spacing almost every time. This effect is most likely to be an effect of the radio rather than the noise.

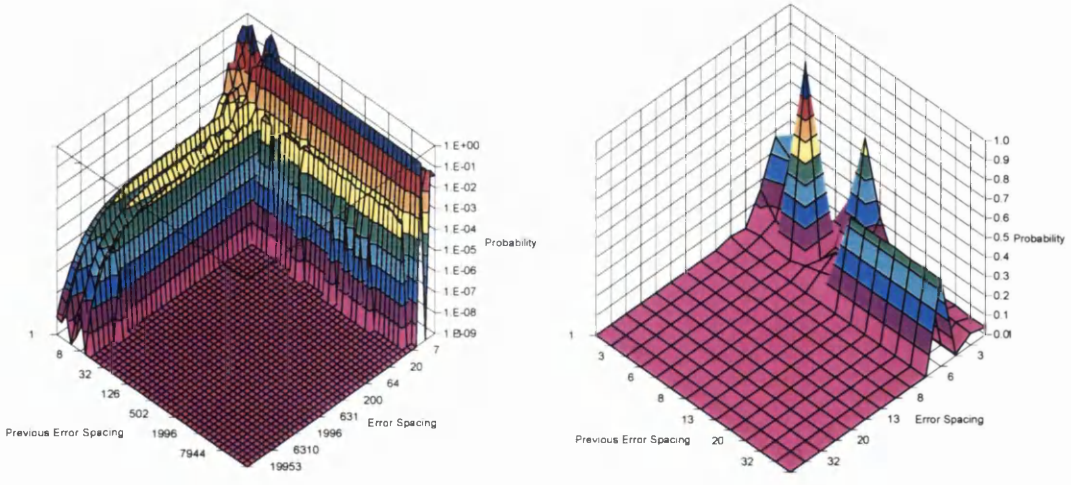


(a)



(b)

Figure 4-12 Error distributions for development radio



(a) (b)

Figure 4-13 Conditional probability of development radio

One function that is needed in any transmission process is to keep bit synchronisation. Bit synchronisation can only be achieved when there are changes between symbols (i.e. impossible to synchronise to a infinitely long stream of ones or zeros). Bit modification techniques such as Manchester encoding can be used to provide edges, but these generally increase the bandwidth requirements. Another approach is to alter the data to prevent large numbers of continuous ones or zeros. Figure 4-14 shows a general approach to generating bit changes. The first shift register is a pseudo random sequence generator that is altered by the incoming data. The second shift register in the receiver de-randomises the bit stream to give the original data stream.

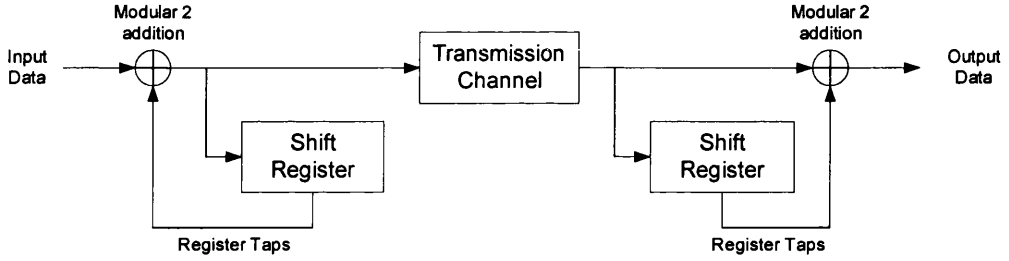
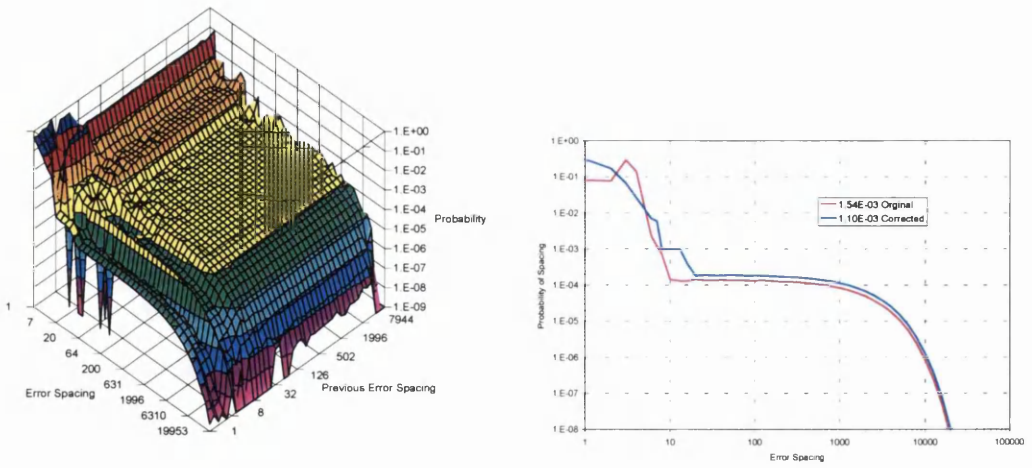


Figure 4-14 Bit change generator

The problem of using a pseudo random bit change generator is that a single error will cause multiple errors at the receiver. This is caused by the error bit passing through the shift register. For every tap position on the shift register there will be an extra error. The distribution of the errors will be the same as the tap positions. In the case of the development radio the tap positions are 4 and 7, giving the peaks at 4 for previous errors greater than 7 and a high peak at 3 for previous error spacing of 4. It is possible to work backwards from the error pattern to the errors that were on the link.



(a) (b)

Figure 4-15 Corrected data

Figure 4-15 shows the results after correcting for pseudo random generator. There is now less dependency between bit errors (Figure 4-15(a)) and there is an improved bit error rate (Figure 4-15(b)). The results are not three fold better as would be expected. The radio obviously has other features that have an effect on the bit stream. The corrected results do show a much better relationship to the Gilbert and Elliot model.

4.4.2 Military radio

Figure 4-16 shows the distribution of errors for several bit error rates from a military radio. There is a mixture of patterns from random at 2.0×10^{-2}, a Gilbert and Elliot type distribution at 8.7×10^{-4} and different pattern at 1.2×10^{-4}. This pattern at 1.2×10^{-4} is 'bursty' as there is a considerable higher probability that errors will be close together. Figure 4-17 shows that there is not a dependency between bit errors.

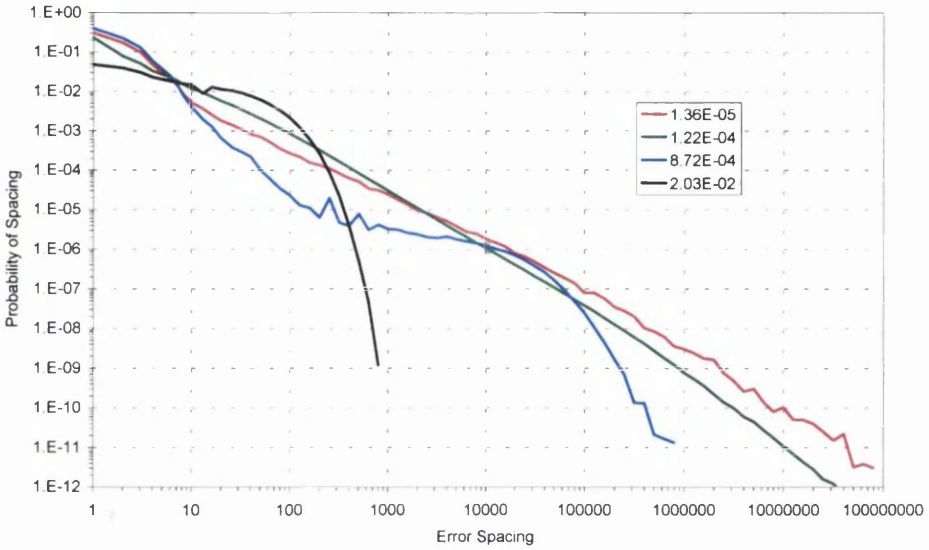


Figure 4-16 Distribution of error from a military radio

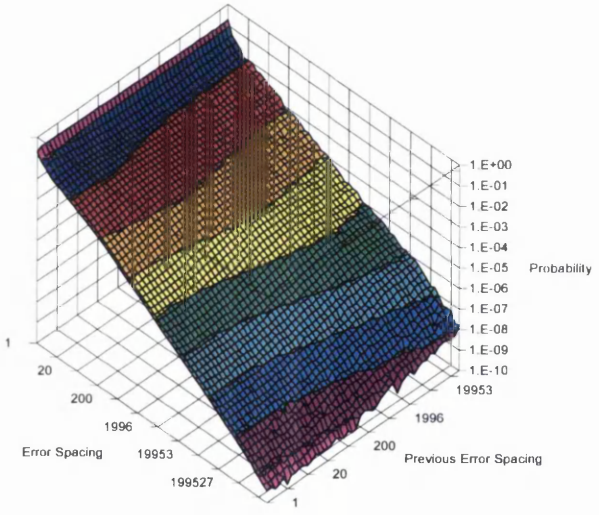


Figure 4-17 Conditional probability for military radio

4.5 Conclusions

The approach of looking at error spacing is an effective way of looking at the distribution of errors. The information gathered from looking at conditional probability can give an indication of potential problems that cannot be seen by just looking at the error distributions.

There is a significant difference between the military and development radios. This could be due to the way the radios operate, or to the way the radios were tested. The development radio was tested on a bench in a controlled environment. The military radio was tested live on air. The background noise that any operating radio is exposed to is unlikely to be Guassian as assumed in basic testing processes. This means there could be a significant difference between the performance of a radio tested in a controlled environment and one that is tested while in its normal operating position. To verify this, the military radio needs to be tested under the same conditions as the development radio. There are some obvious problems with the development radio that need to be fixed before a valid test can be done using this radio.

From the literature, the most accepted model for burst errors over a radio channel is the Gilbert and Elliot model. This does seem to match the results from the development radio, but due to problems with the radio it is difficult to confirm this. For further work with error patterns, it is sensible to use the Gilbert and Elliot model. However, the main backbone of the network is supported with the military radio. The distribution of errors from the military radio is different from either the random, pulse or Gilbert and Elliot. To evaluate the performance of other parts of the system it is necessary to develop a model that matches the characteristics of the military radio.

5. NEW BURST ERROR MODEL

5.1 Introduction

From Chapter 4 it is necessary to develop a burst error model that matches the characteristics of the errors from the military radio. Others have noticed that radio links have a different error pattern to the standard models [Ahrens00].

5.2 Description of Model

The general characteristics of burst errors is that bit error rate varies over time. The Gilbert and Elliot model varies the error rate by randomly switching between two states, this causes the double hump. The humps can be smoothed by using several more states such as a Markov chain [Zorzi97] with slight variation in the error probability for each state. For a flat line, the author extended the model to have infinite states, the state variable becomes a real number rather than an integer. Each state (or state value) would have a different bit error rate, which can be achieved using a linear function to map from the state value to the bit error rate. If all states are equally possible (i.e. there is an equal probability of changing from any state to any other state), a uniform random number generator can be used to select the state. It is only then a matter of mapping the state value onto an appropriate bit error rate to give the correct error distribution.

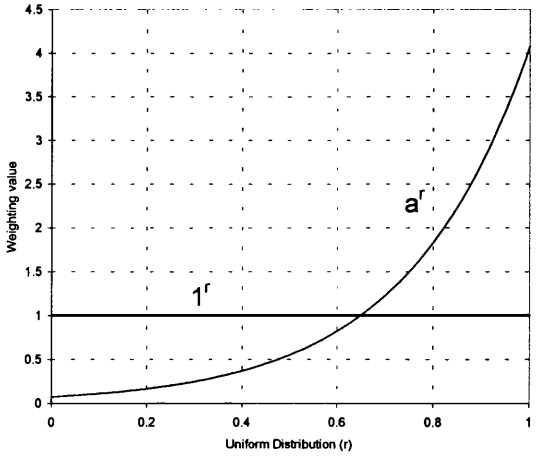


Figure 5-1 Weighting function

From trial and error a suitable weighting function of a^r (Figure 5-1) was found, where r is a uniform number between 0 and 1. When a is equal to 1 all states are given the same weighting. The average bit error rate is then adjusted by a weighting factor to give the variable bit error rate. For a given desired average bit error rate, the distribution of weighting factors should be normalised to give an average of 1 over the given range. This is achieved by using the integral (area under the curve) over the range 0 to 1 (5.1).

$$c \cdot \int_0^1 a^r \cdot dr = 1 = \frac{c}{\ln a}\left[a^r\right]_0^1 \quad (5.1)$$
$$c = \frac{\ln(a)}{a-1}$$

By substituting e^n for a gives the function (*inst_ber*) in (5.2), where n is any value between 0 and an upper limit. n=0 gives a mapping function of unity, hence a purely random distribution. As n becomes large it is possible to get bit error rates of greater than 1. n should not exceed a value that cause the condition in (5.2) to fail.

$$f(r) = \frac{n}{e^n - 1} e^{nr}$$
$$inst\_ber = \frac{BER}{f(r)} \quad (5.2)$$
$$BER \cdot \frac{e^n - 1}{n} < 1$$

The bit error rate is used to decide if each bit is in error. The next stage is to decide when to calculate a new bit error rate. There are several options; change the error rate of every bit, or every N bits. The best approach found is to change when there is an error.

5.3 Results

The model gave the average bit error that was set. Figure 5-2 shows the curves for several bit error rates, the burst factor (n) used was 5. Figure 5-3 shows the curves for different burst factor settings, the bit error rate was set to $1x10^{-3}$. As the burst factor increases the curve becomes straight matching that of the military radio. The condition probability graph (Figure 5-4) shows there are no dependencies between bit errors.

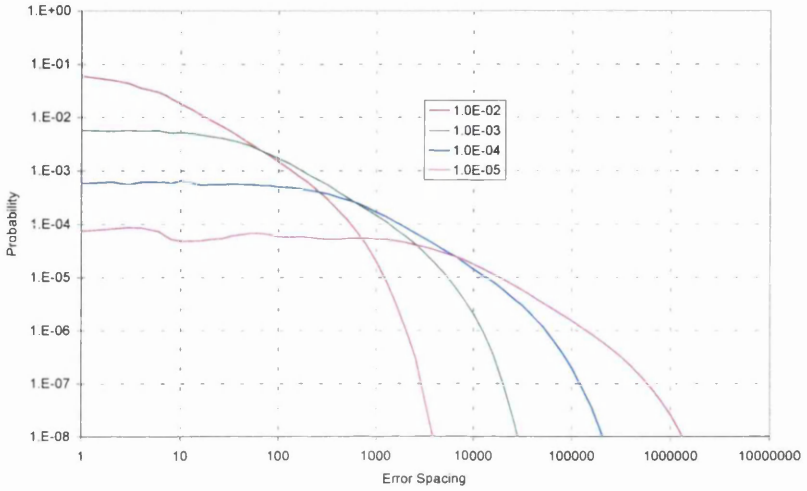


Figure 5-2 Varying the bit error rate

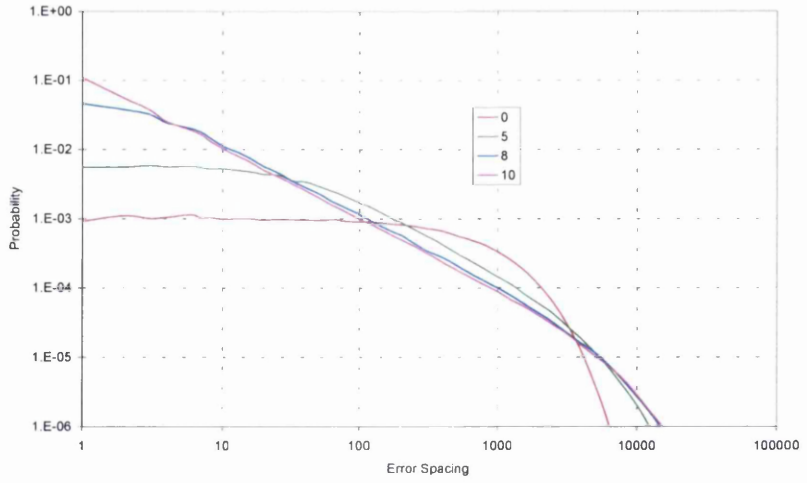


Figure 5-3 Varying the burst factor

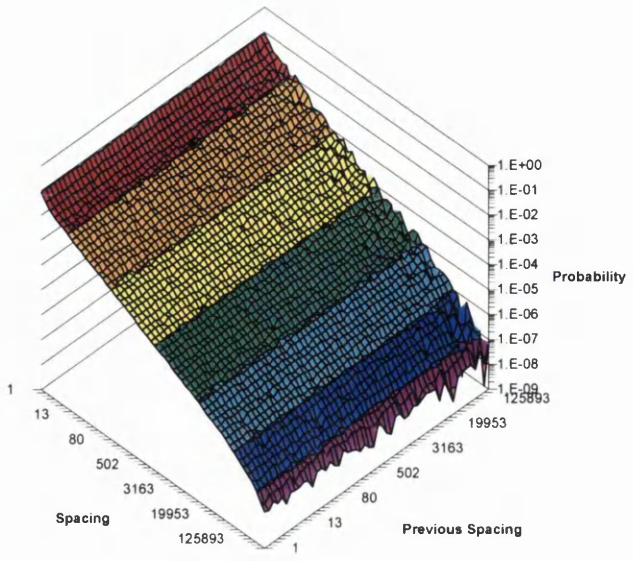


Figure 5-4 Conditional Probability

5.4 Conclusions

This new model produces error patterns that are better matched to the military radio than the existing Gilbert and Elliot model. The model can be tuned to produce 'bursty' or random errors, with no dependence between individual errors.

6. ERROR CORRECTION AND RECOVERY SCHEMES

6.1 Introduction

The link layer handles the transfer of data over a single connection. The link layer protocol on data networks has two main functions, providing data framing and integrity, and flow control. Data integrity checks are normally Cyclic Redundancy Checks (CRC), but could be a check sum. The performance on an error prone link can be improved by using Forward Error Correction (FEC) and Automatic Repeat Request (ARQ). Very few standard protocols have FEC or ARQ (Table 6-1) relying on high level protocol to handle the corrupt data. The present UK military tactical data network uses a Modified X.25 and Bose-Chaudhuri-Hocquenghem (BCH) code for error correction.

Protocols	Error Detection	ARQ	FEC
Ethernet (802.3)	CRC-16	Repeat on Collision	None
HDLC	CRC-16	None	None
PPP	Checksum	None	None
X.25 layer 2	CRC-16	Go Back N	None

Table 6-1 Common Link Layer Protocols

With the high error rates on radio links (Chapter 4) it is important to consider both FEC and ARQ. Integrity check is used to detect errors not corrected by the FEC and used by the ARQ to request retransmission.

6.1.1 Error Detection

Error detection is needed in almost all data networks. This is to prevent corrupted data being accepted by higher level processes. The simplest form of error detection is a single parity bit (used in asynchronous data on RS-232). This is capable of detecting any single error in the data. This works when the probability of more than one error in a given segment is virtually zero. A simple detection done on a processor is a checksum. All the bytes (8 bits) or words (16 bits) are added together. This is capable of detecting more than one error, but there is still a probability that some double errors could be missed. The most common error detection used by physical layer protocols is a CRC [Shay95].

CRC is a polynomial division of the data by the check pattern (generator polynomial). The remainder of the division is sent with the data as check bits. When the division of the data and the remainder by the check pattern is done by the receiver the result should be zero. Anything other than zero means there are errors. CRCs are not perfect as some errors can still go undetected. These error patterns are related to the generator polynomial. The larger the generator polynomial the lower the probability of missing errors. There are four main standard polynomials in use (Table 6-2)[Shay95].

Standard	Polynomial
CRC-12	$x^{12}+x^{11}+x^3+x^2+x+1$
CRC-16	$x^{16}+x^{15}+x^2+1$
CRC-CCITT	$x^{16}+x^{12}+x^5+1$
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}$ $+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

Table 6-2 Standard CRC Polynomials

It is only the remainder of the division that needs to be stored. This allows the division to use shift registers and logic gates. A serial controller can calculate the CRC as the data is being sent to line. The remainder can be shifted out of the register once all the data has been sent. The receiver can calculate the remainder as the data is being received from line. This allows the whole process to be completed in hardware without the need to use complex arithmetic requiring by a processor (USART, 65560 Multi-protocol Communications Controller).

6.1.2 Forward Error Correction

There are two main types of FEC, block codes and convolution codes. Both add extra parity check bits, the extra bits provide redundancy to enable corrupted bits to be located and corrected. Block codes work over a fixed number of bits. Convolution codes are used for bit stream data. Simplistically, convolution codes are overlapped block codes.

Block codes are the most studied set. They vary from Hamming codes that can correct only one or two errors, to large Reed Solomon (RS) codes. There are a few special codes know as perfect codes. These codes have equal separated code words that use the entire code space. The largest known perfect code is the Golay (23,12). Some codes are based on symbols with N possible states. Codes with 2 possible states are known as binary codes. The BCH codes are a large set of binary codes. The advantage of using N states per symbol is that the symbol can be used as single symbol across a multi-level modulation scheme such as QAM.

The problem with block codes is that they only decoded properly if the edges of the code can be found. Adding framing bits adds extra overhead and may be more vulnerable to errors than the block code. Block alignment can be achieved but decoding every bit position possible until the error rate is very low or zero. This process may take a long time to synchronise with large codes or very difficult with a high error rate channel.

Convolution codes do not suffer with the same synchronisation problem. The most common convolution codes are half rate codes. For every bit that goes into the coder two come out. This gives a high level of error correction but the overhead is normally too high for most applications. To reduce the overhead punctured codes are used, here some of the bits from the coder are not sent over the link. If only one of the two parity bits are sent for two data bits this gives a 2/3 rate code.

Some convolution decoders build up a tree of possible bit patterns (trellis). The closest matching bit pattern is selected as the correct stream. The bits used by the decoder consist of both received bits (look forward) and corrected bits (look back); this approach is used by the Veterbi decoder [Ziemer95]. Many modern radios have a convolution coder built into them, so the radio can provide a reasonably low error rate without the need for separate equipment to provide the FEC.

A problem with the Veterbi decoder is that when the error protection limit is reached, there will be error in the output bit stream. The decoder uses this stream to correct further errors. The errors in the output stream could cause further errors. This means that a burst of errors on the radio link could cause more errors in the output stream than there are in received bits stream. Burst errors on the radio link could cause large bursts of errors in the output data stream. It has been shown that the longer the code the longer it takes the decoder to settle back to the correct stream [Heissler99].

An aspect of FEC codes that needs to be considered is the strength of the code compared against its overheads. The stronger the code the more redundant information is needed, hence large overheads. Figure 6-1 is a graph of several BCH codes [Stremler90], it shows the relative error correction against throughput (Transmission rate). It indicates that shorter length codes have better error protection for the same overhead. Figure 6-2 shows the performance of several BCH codes with the same overhead. The shorter length codes are only slightly better. The major difference is the slope of the fall off point, the large codes have a much more sudden change between correcting and failing.

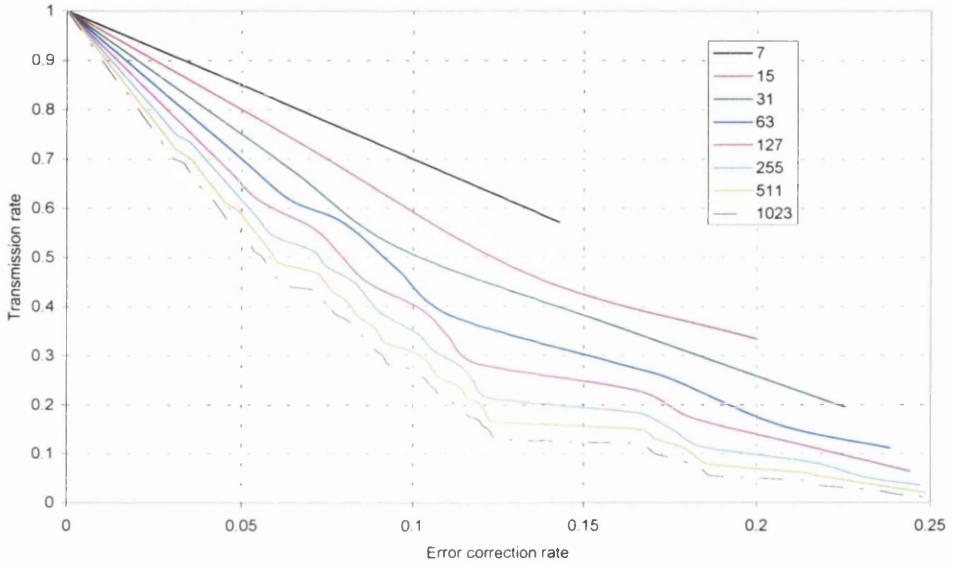


Figure 6-1 Comparison of BCH codes

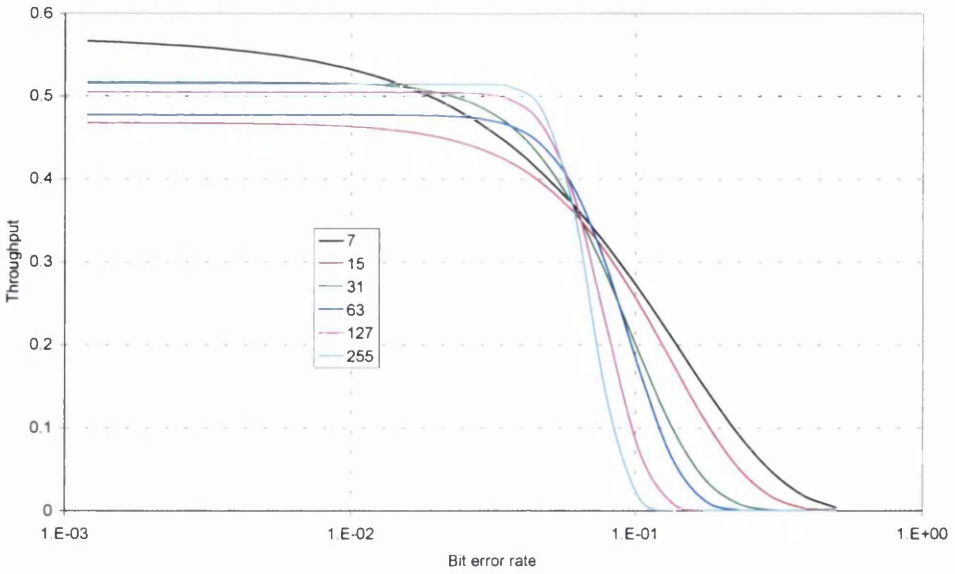


Figure 6-2 Throughput of BCH codes

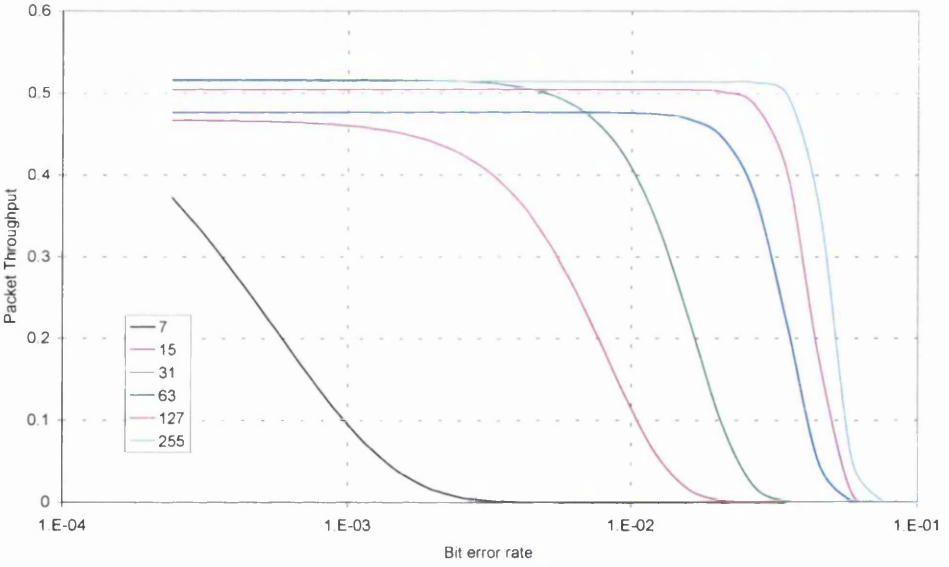


Figure 6-3 Performance protecting packet

Several codes are needed to protect a packet larger than the length of the FEC. The probability of a corrupt packet becomes a function of the error probability of the individual block codes and the number of FEC code blocks required to protect the whole packet. Figure 6-3 shows the performance when different length BCH codes are used to protect a 1 kbit packet. It is clear that the large codes are considerably better at protecting the packet. The ideal size of the block code should be large enough to protect a complete packet.

6.1.3 Automatic Repeat Request

ARQ provides a process for corrupted data to be retransmitted. There are several mechanisms to achieve this. The simplest is 'Stop and Wait' a single packet of data is sent, the sender then waits for the packet to be acknowledged. If the data is not acknowledged within a fixed time the packet is resent, and again waits for an acknowledgement. If the data is acknowledged the next packet can be sent. For this to work the receiver needs to be able to determine if there are any errors in the packet, and if the packet is the correct one. Data corruption is detected with a CRC or checksum and to check the order, each packet needs a sequence number. For 'Stop and Wait' only two numbers are required.

The process can be improved by allowing multiple packets to be sent at once; this is known as 'Go Back N'. When a corrupt packet is retransmitted, all packets following it also need to be retransmitted. The number of packets that can be sent is called the window. A further improvement can be made by only retransmitting the missing packets, this is known as 'Selective Repeat'. The sender can make better decisions on what needs to be sent if it gets more information from the receiver. Selective acknowledgement and negative acknowledgement can be used by the sender when out of order or corrupt data is detected. The number of sequence numbers needed depends on the type of protocol and the size of the window (Table 6-3).

Type	Window	Sequence Numbers
Stop and Wait	1	2
Go Back N	N	N+1
Selective Repeat	N	2N+1

Table 6-3 ARQ Protocols

The performance of an ARQ can be measured by looking at whether the protocol is capable of keeping the link full of data, not only when the link is error free but also when a retransmission is needed (i.e. still send new packet while re-transmitting another). The size of the window can be determined from the bandwidth delay product and the size of the packets (6.1). The bandwidth delay product is simply the amount of data the link can carry in transit. Both the forward and the reverse path need to be considered.

$$pkts = roundup\left(\frac{bitrate \times delay}{packetsize}\right)$$
$$acks = roundup\left(\frac{bitrate \times delay}{ack\_size}\right) \quad (6.1)$$
$$optwin = 2 \times \min(pkts, acks)\ *$$
$$optretwin = 2 \times optwin$$

pkts = Number of packet in transit on the forward path.
acks = Number of acknowledgements in transit on the reverse path.
optwin = Optimum window size with no retransmissions.
optretwin = Optimum window to handle a single retransmission.
* Acknowledgements are generated at the same rate as data packets are received.

6.2 Evaluation of Schemes

There are several mechanisms for improving throughput over error prone links. A lot of these processes are evaluated with random errors. It is not clear how they will perform in burst error conditions. Therefore it is necessary to evaluate the effects of burst errors on the mechanisms.

Burst error is defined as groups of errors combined together. This means some packets will have more errors than others. The increase in errors for some packets means the FEC schemes are going to struggle to correct the errors. The distribution of errors will mean some packets will have fewer errors and may not need protecting, hence an unnecessary cost in overhead. An assumption is that burst errors decrease the performance of FEC codes but increase the performance of unprotected packets. It is then necessary to check if the change in performance of both protected and unprotected packets warrants not using FEC.

Another aspect that needs to be considered is whether burst errors lead to burst packet losses. The effect of multiple errors in a packet may have a smoothing or averaging effect on the packet losses, making the packet losses more random than the burst errors. If this is true then packet network modelling can be done with random packet dropping, there is no need to mimic full effect of burst errors on the packets. This not only helps in modelling but also in network design.

The effect of errors on an ARQ is to cause the mechanism to resend the corrupted packet. This means all data will eventual come through correctly. The effect of the retransmission is to add a delay to the data being received. If the ARQ mechanism is able to keep the link full of un-received data during retransmission the throughput of the link is the same as the throughput of the link if packets were dropped. The difference is that the received data contains all the correct data in the right order.

6.2.1 Modelling of FEC

The study of FEC is a large subject, so for this study forward error correction will be considered as adding extra bits and making the assumption that a certain number of errors can be corrected. There will be no attempt to implant a proper code.

The process of evaluating the FEC was done with simulation software. Two pieces of software were used, one was a modified version of the error analysis software used in Chapter 4, the other a command line program.

The error analysis programme had an extra layer inserted between the error generation code and the statistical process. This layer would count the number of errors in a fixed size block, then only pass the block error information to the statistical process. The block is only considered to be in error if the number of errors in the block is greater than the error correction capability of the code. Setting the block size to one bit, allows the bit error information to be passed to the statistical process. Setting the error correction capability to zero allows the software to analyse unprotected blocks. The analysis software can now look at the distribution of packet errors.

The second piece of software simulates bit errors, inserts the errors into blocks and analyses the throughput of the code. The two added features are that the software accounts for the overhead of using an FEC, and can be set to run a range of test conditions. The software only generates an average block error rate and there is no analysis for distributions. The purpose of this software is to test the FEC with various bit error rates and different level of bursts. The output of the programme is a set of results for a range of test conditions.

The validation of the software was achieved by comparing the results of conditions that can be calculated from mathematical analysis.

6.2.2 Modelling of ARQ

The ARQ model used OPNET as a simulation environment. The ARQ mechanism was a selective repetition that used cumulative acknowledgements (CumAck)(Table 6-4). As in most ARQs, acknowledgements can be piggybacked with data. The ARQ mechanism has a few added features. The header has an independent checksum to that of the payload. This allows the sender and receiver to keep in sync even if the payload is in error, as the receiver is able to identify the sequence number of a corrupt packet. This also enables the checking of the payload to be handled separately to the packet, allowing FEC protect, CRC, and unprotected packets to be used simultaneously (Table 6-4,Figure 6-4).

type	num0	num1	flag	header check	data
NSYNC	X	X	X	Used	X
SYNC	X	X	X	Used	X
ACK	NextSeq	CumAck	X	Used	X
NACK	NegAck	SelAck	Repeat	Used	X
DATAFEC	Seq	CumAck	Last Packet	Used	Data
DATACRC	Seq	CumAck	Last Packet	Used	Data
DATA	Seq	CumAck	Last Packet	Used	Data
DUMMY	Seq	CumAck	X	Used	Used

Table 6-4 Format of ARQ packet

Negative acknowledgements (NACK) carry the sequence number of missing packets (NegAck) (Table 6-4) as well as the sequence number of the packet that caused the sending of the acknowledgement (SelAck). If an out of sequence packet is received, the mechanism is able to inform the sender of the out of sequence packet and acknowledge the packet that was received. For corrupted packets that have an identifiable sequence number a NACK packet can be sent indicating the corrupt packet.

The acknowledgement packets carry the next sequence number to be sent (NextSeq) (Table 6-4). This allows the receiver to determine if the last packet of a burst of packets has been lost. Without this the receiver would have to wait until the next received packet before detecting a packet was missing. If this delay were too long the senders timeout mechanism would be used when no acknowledgement was received.

The ARQ mechanism has been extended to include frame synchronisation. This is done firstly by keeping all ARQ packets to two sizes. The smaller size includes only the ARQ header, this is for ACK and NACK packets. The larger size that should be a multiple of the header size is used to carry data. This means that when frame synchronisation is achieved it is easy to maintain. To avoid the need for excessive frame synchronisation information per packet, the link should always be sending something. In the synchronised idle state, this will be ACK packets. Two other packets are used to achieve synchronisation, a NSYNC packet and a SYNC packet. The NSYNC packet is sent when no valid frames are detected. This NSYNC packet informs the other end of the link that it is unable to receive data. When receiving NSYNC packet the SYNC packet should be sent. This informs the other end that frame synchronisation has been achieved without sending packets. The NSYNC and SYNC packets should be unique enabling the receiver to pattern match and gain synchronisation. Once a packet other than NSYNC is received the mechanism can start sending packets.

The ARQ uses fixed size packets. The data to be sent over the link has to be arranged to fit in the packets. If necessary each block of data is broken into a number of packets. If the block of data is not an exact number of packets, the last packet is padded (Figure 6-4). To separate the blocks of data at the receiver a flag (Table 6-4) is used to identify the last packet of a block of data. The ARQ packets do not carry information about the original size of the data block. Most high level protocols such as IP that will be used for transfer across the link already have a length field.

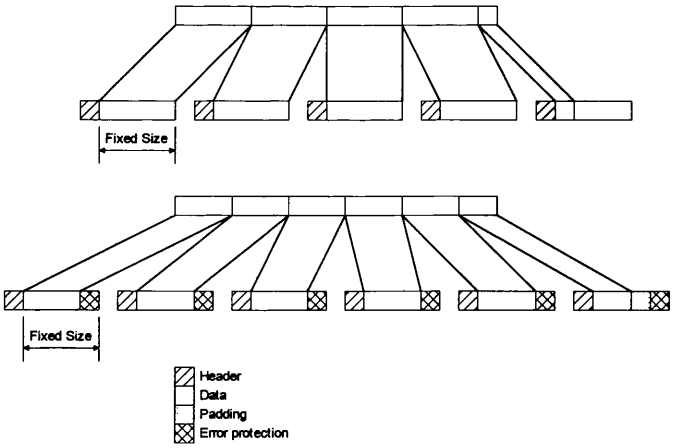


Figure 6-4 Fragmentation into ARQ size packets

This ARQ mechanism is tuned to work across a full duplex point-to-point connection. The frame synchronisation process will not work across simplex or unidirectional links. Due to the continuous sending of packets this mechanism will not be efficient when used over a data network.

The effect of an ARQ is to add a variable delay to the time required to send a block of data over the link. The delay is measured from the time when the first part of the data is first sent, to the time the last part of the data is received. The delays measured are only for blocks of data not the individual ARQ packets.

6.3 Results

6.3.1 Distribution of packet errors

Different error distribution models were evaluated to assess whether the type of distribution affects the distribution of corrupt packets (Figure 6-5). A packet size of 4 kbits was used and the same average bit error rate of $1x10^{-5}$ was used with all models. The random bit errors produced packet corrupts that match the predicted theoretical curve. The Gilbert and Elliot model (G-E) produce something close to a random distribution. This is due to the way the burst errors are generated in close groups with spaces. The groups of errors cause the packet to be corrupted. The spacing of the groups is defined by the probability of changing from the good state to the bad state. The curve in the graph (Figure 6-5) is very close to the distribution generated by a random packet dropping based on the good to bad change probability. The peak at the beginning of the curve is due to two consecutive packets being dropped by the same burst. The new burst model described in Chapter 5 produces packet dropping that is also 'bursty' because the burst errors are distributed more evenly than in the Gilbert and Elliot model.

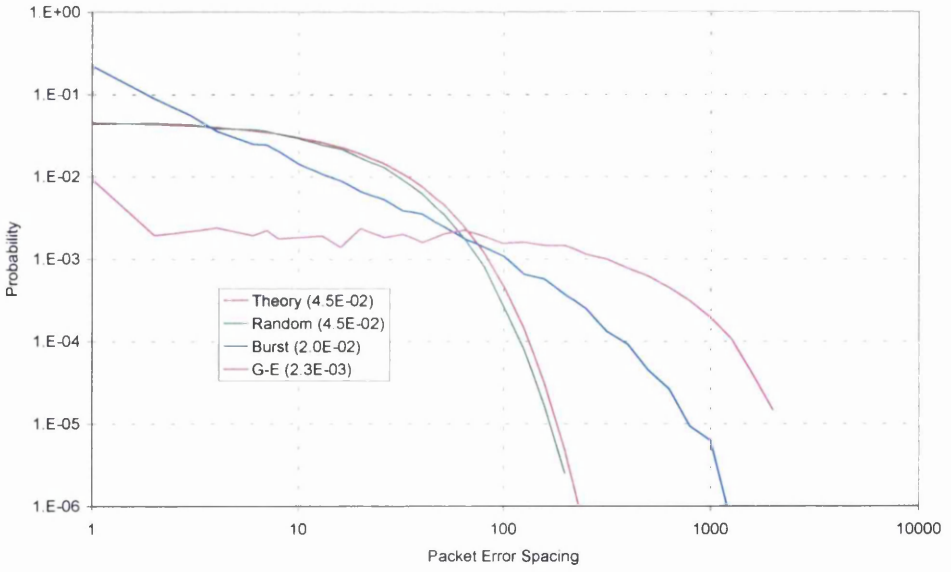


Figure 6-5 Packet Error distributions

The other effect of burst error is to reduce the probability of a packet drop relative to random case (Figure 6-5). The new burst model has the effect of increasing the throughput from 95.5% to 98%. The Gilbert and Elliot model increase the throughput to 99.7%, with the same average bit error rate.

6.3.2 Effect of burst errors

The next set of graphs shows a comparison of the throughput for different conditions (error rate and bursts) against the throughput at the same average random error rate (i.e. not 'bursty'). The vertical scale is the log difference in the throughput (6.2). If the throughput in the burst condition is higher than the throughput in the random error condition, the number will be positive. This equation gives the same value for the same ratio, only the sign changes if the ratio is the other way round (i.e. 1:2 = 1/2 = -3 diff or 2:1 = 2/1 = 3 diff). The FEC scheme used is a BCH(255,131), and the burst error model is the Gilbert and Elliot. The block size used for the unprotected data is the same as the BCH code (255 bits).

$$diff = 10 \cdot \log\left(\frac{th_{burst}}{th_{random}}\right) \quad (6.2)$$

Figure 6-6 shows that the throughput without error protection increases in the presence of burst errors. The effect is far more pronounced at high error rates. This is caused by the burst errors affecting individual packets and leaving more packets to get through unaffected. With FEC the effect is opposite (Figure 6-7). This is because the FEC cannot handle the burst of errors, so the extra redundancy has the effect of reducing the throughput. The high ridge under very high error rates where the FEC is unable to handle the random errors, hence the effect is similar to that of the unprotected packets, when the burst causes more error free packets. It is suggested that large codes deal with burst errors a lot better. This is because a large block can handle more errors even if the errors are grouped together. A group of 10 errors in a small block could cause the FEC to fail whereas a large block could potentially be able to handle them. Figure 6-8 shows the difference in performance when the block is increased from 255 bits to 1023. There is very little difference between the way small and large blocks are able to deal with Gilbert and Elliot burst errors. Figure 6-9 shows the difference between having and not having FEC. The positive values are where unprotected data gives better performance. There is a slight shift due to burst error at the point where the FEC becomes effective. But compared to the range of bit error rates a system is likely to be presented with, there is very little difference.

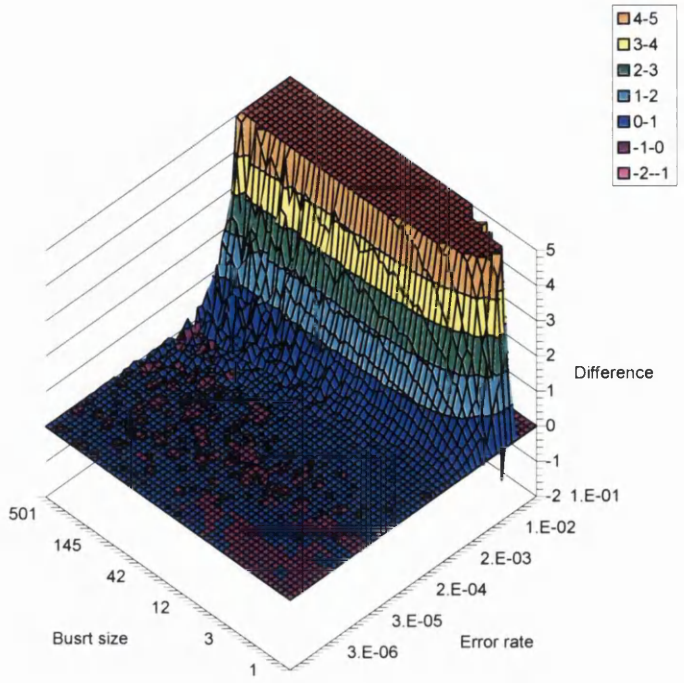


Figure 6-6 Performance without FEC in burst errors

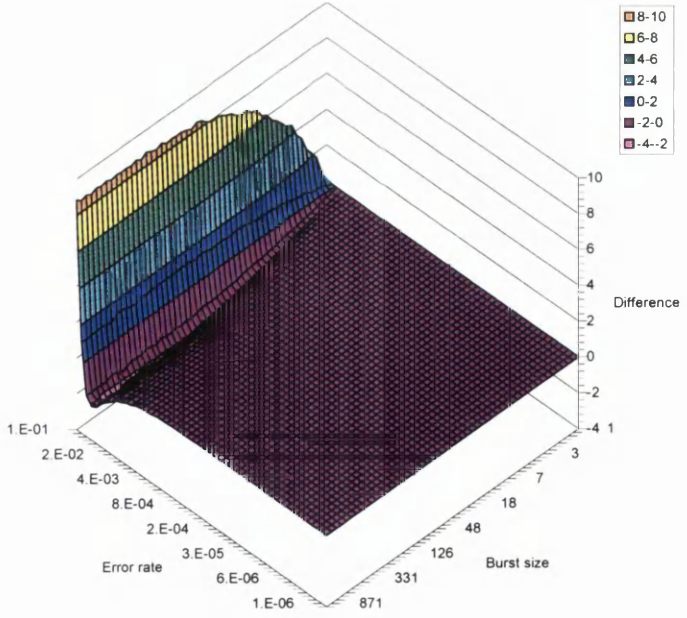


Figure 6-7 Performance of FEC in burst errors

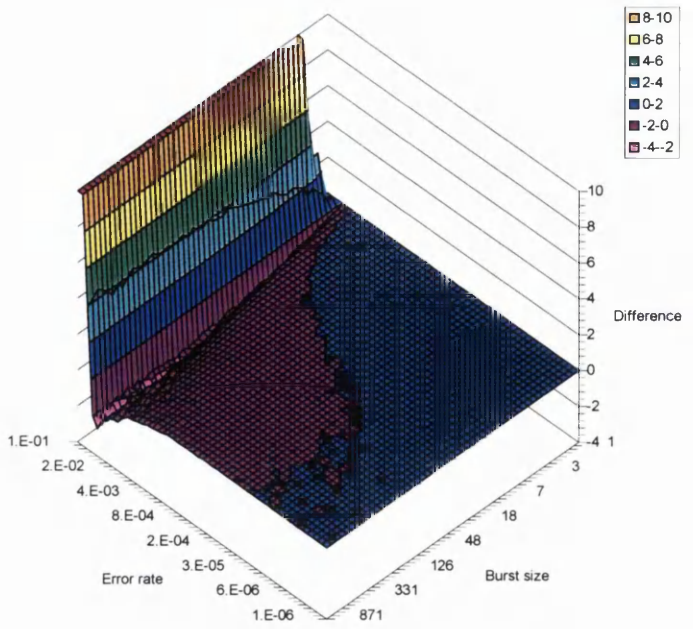


Figure 6-8 Performance of large block code

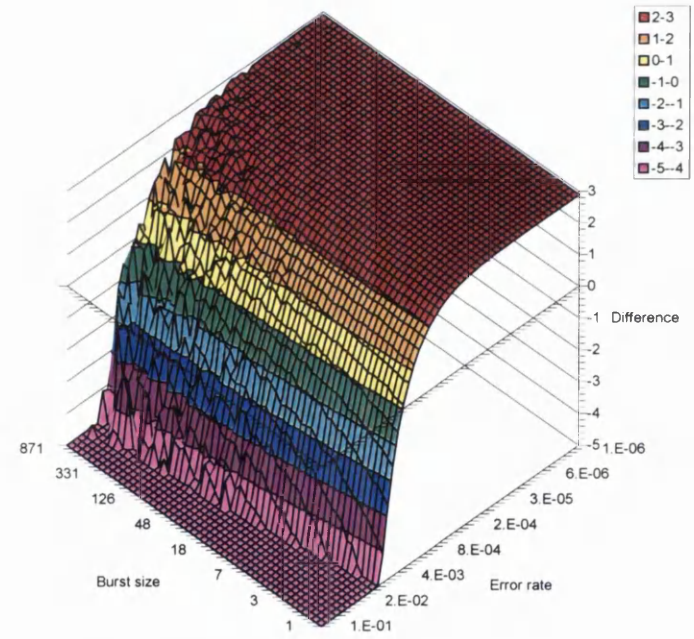
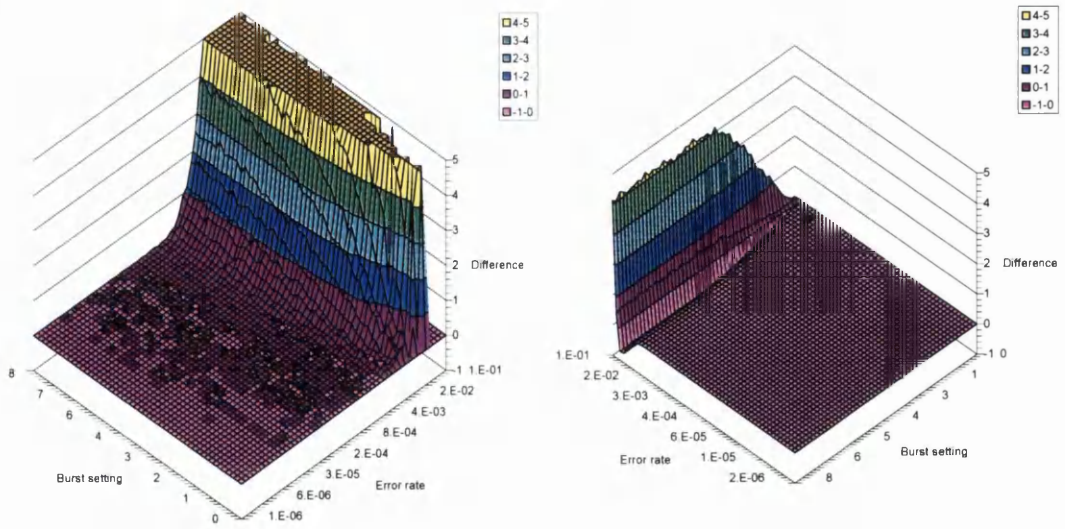


Figure 6-9 Comparsion with and without FEC

The previous set of graphs used the Gilbert and Elliot model. The next set of graphs is for the new burst model (Chapter 5). Figure 6-10 shows that the pattern in the performance is the same for the Gilbert and Elliot model (unprotected Figure 6-10(a), protected Figure 6-10(b)). Figure 6-11 shows that the burst error has virtually no effect on the choice as to when to apply FEC.



(a) (b)

Figure 6-10 Effect of burst with new model

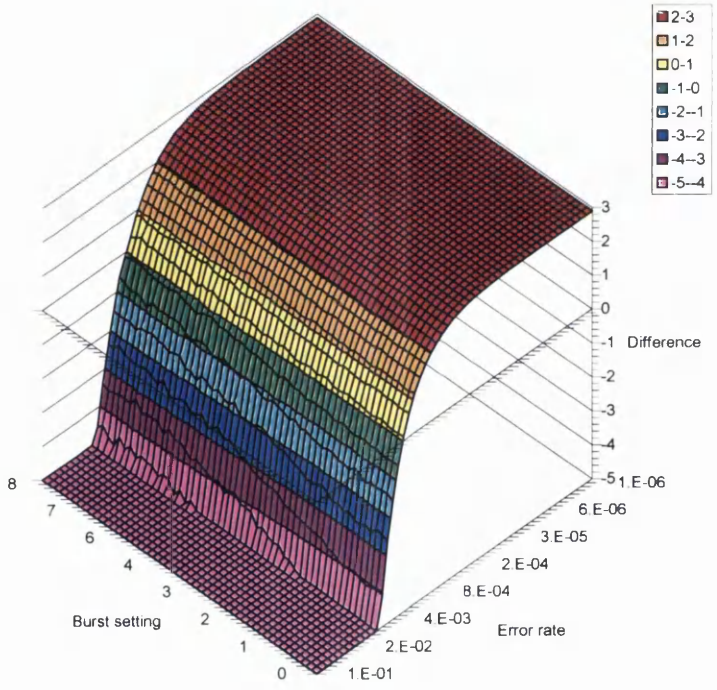
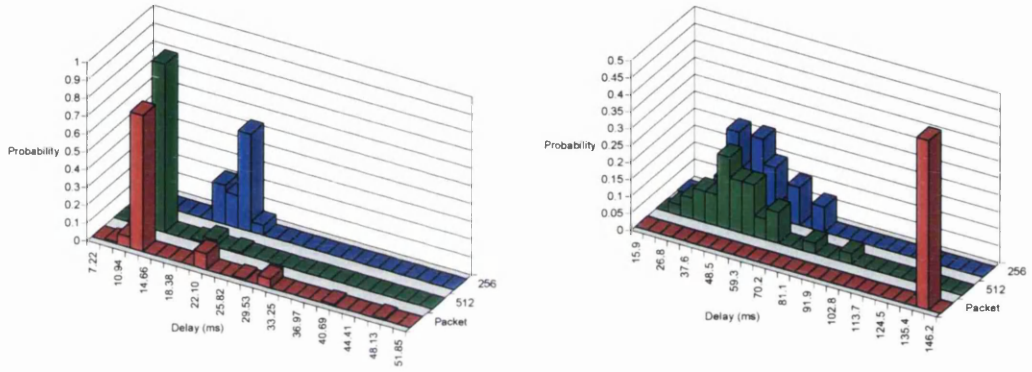


Figure 6-11 Performance difference with and without FEC (new model)

6.3.3 Performance of ARQ

The ARQ was tested with three different packet sizes. 'Packet' (identified in graphs) is where each individual block of data is treated as a separate packet and not fragmented into separate ARQ block for transmission. Two fixed sizes packets were used 512 bits and 256 bits. The size of the block of data was 576 bytes (4608 bits). Figure 6-12(a) represents a bit error rate of 10^{-5} while Figure 6-12(b) represents 10^{-3}. The reason the delay associated with 256 bits is higher is that the sender window is too small to keep the connection fully utilised. The single value for Packet on the Figure 6-12(b) is due to only one packet getting through.



(a) 10^{-5} (b) 10^{-3}

Figure 6-12 Distribution of delays

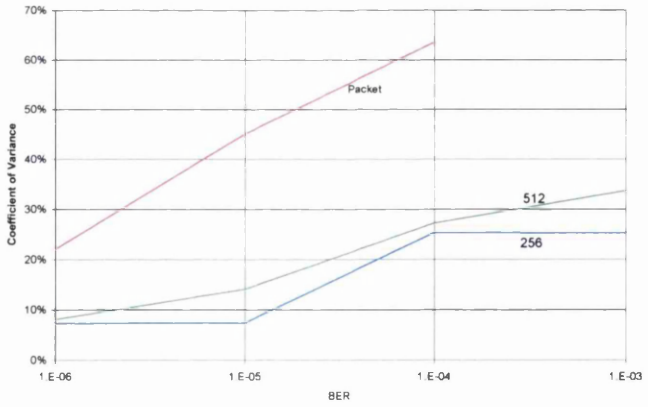


Figure 6-13 Comparison of packet size

Figure 6-13 shows the difference between using different size ARQ packets. The smaller the packets the smaller the jitter. This is because the individual packets are less likely to be corrupt, and less data needs to be retransmitted to recover the corrupt data. The problem with small packets is having enough of them in the window to fully utilise the link. Figure 6-14 shows the throughput of ARQ over 1 and 3 links. The throughput is measured as a percentage of the link capacity. The error rate over three links has been normalised to give the same average error rate as a single link. The unprotected data has the same throughput whether it is over one link or three. There is a major improvement when using ARQ in a high error environment, at a cost of a lower throughput at low bit error rate due to the extra overhead. Using separate ARQ on each link of a multiple link connection improves the throughput even more. This is because throughput of each link is going to be roughly the same. With unprotected data the throughput is also dependent on the amount of data available to send, each consecutive link has less and less data. With ARQ the queue process means delayed data will be available.

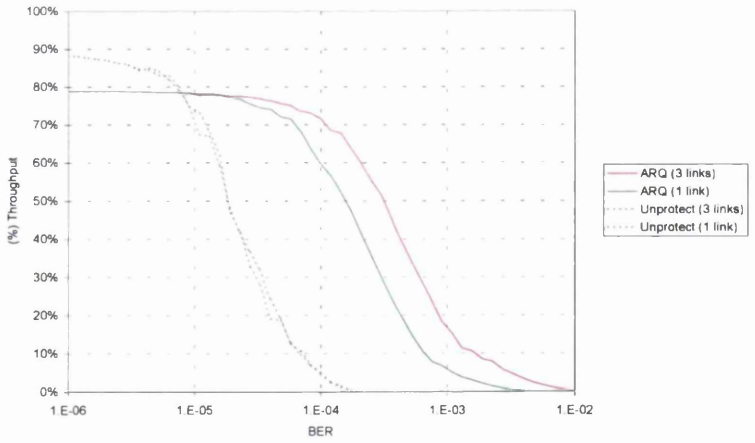


Figure 6-14 Throughput of ARQ

6.4 Conclusions

The Gilbert and Elliot model produces random packet losses. This is due to the random switching between the good and bad states. If the Gilbert and Elliot model is assumed for link errors, it can also be assumed that packet losses will be random. Hence it is only necessary to evaluate random packet losses in higher layers of the ISO stack.

The hypothesis that burst errors have a negative impact on FEC and a positive impact on unprotected data has been proven true. Burst errors improve the throughput of unprotected data but only significantly at very high error rates. At this high error rate it is more efficient to use FEC even though there is performance degradation due to the burst errors.

Burst errors do affect the throughput of a data link but not as significantly as the average bit error rate. For unprotected data the worst case is with random errors on the link. Burst error on FEC protected link can cause a decrease in throughput of up to 20%. This is only at the point where the FEC is on the limit, hence the burst errors are enough to push it over the edge.

FEC code selection should be based on the size of data packets being sent and the expected bit error rate. The use of the FEC could be set by planning for the worst case, a percentage of losses or be adaptive. Planning for the worse case could end up with links that are inefficient due to the extra unnecessary overhead. Allowing for a small amount of losses could improve the efficiency. Both these approaches require knowledge of the characteristics of the link before they can be used. Gathering these characteristics can take time and delays the time before the link can be used. By getting the end equipment to dynamically adjust the level of protection could improve the performance and allow the link to be used quickly. For this to work the link equipment needs to measure and output data about the error rates on the link. The other problem is getting this process to work reliably. The ends of the link need to negotiate the coding process, if the two ends cannot communicate due to errors on the link the whole process fails.

The new burst error model has less affect on the throughput than the Gilbert and Elliot model. The burst errors produced by the model produce 'bursty' packet losses. The type of model selected for further testing will depend on whether the higher level protocols are affected by burst packet losses. If the protocols are unaffected by burst errors then it is possible to use the Gilbert and Elliot model. In this case random errors can be assumed and hence random packet losses. If the protocols are affected by burst errors then the protocols need to be tested with the errors injected at the physical layer. In either model the worst case throughput of unprotected packets is with random errors.

The throughput of FEC protected links can be calculated from the bit error rate, FEC correction rate and the FEC overhead. The effect of FEC is to increase the amount of uncorrupted data but reduces the rate at which it can be delivered across the link. For the purpose of simulation it is possible to mimic the effects of FEC by decreasing both the link bandwidth and error rate.

ARQ provides an effective way of providing reliable data transfer over error prone links. The problem is the variable delay in the delivery of the data. The size of the sender window needs to be tuned to the bandwidth delay product of the link, which is independent of the experienced error conditions. For satellite links the window may need to be very large especial if small packets are used.

FEC has to be tuned to the bit error rate, which could be highly variable even on a single link. ARQ needs to be tuned to bandwidth delay product, which is less variable per link, but can be vastly different between different link technologies.

7. TRANSMISSION CONTROL PROTOCOL (TCP)

The main focus of this study is on TCP/IP. The Internet Protocol (IP) is a network layer protocol that provides a mechanism for routing between different parts of the network. The Transmission Control Protocol (TCP) is a transport layer protocol that provides reliable end-to-end data transfer. The other transport layer protocol in the Internet Protocol suite is the User Datagram Protocol (UDP). UDP has no flow control and does not support reliable transfer.

The purpose of this chapter is look at how TCP behaves in a high error rate radio linked network. The link model is based on the work from the previous chapters. The network architecture chosen is designed to mimic a deployed tactical network.

7.1 Introduction

Most of the Internet traffic consists of TCP connections [Danzig92] because it supports reliable data transfer. Other protocols such as hypertext transport protocol (HTTP) and simple mail transport protocol (SMTP) use TCP as a carrier for their data. The major use of the Internet is web browsing which uses HTTP as its transport protocol hence TCP. The other major contributor to Internet traffic is file sharing and it is predominantly the size of the files and not only the number of that make up this traffic. For example, a Motion Picture 3 audio (MP3) file encoded at near CD quality gives roughly 3.5 Mbytes per track.

In an office environment the major traffic will still be TCP, but the mix of sources will be slightly different. There will be a mix of both HTTP (local web access) and post office protocol (POP3) (email). The sizes of individual emails are also likely to be large with attached documents. There could also be traffic such as database access and file transfers from a central server depending on the structure and function of users using the network.

The traffic on a military network will be similar to an office environment. Some differences are that the users will be dispersed geographically, the system will be more distributed and have higher levels of redundancy.

A major difference between the Internet and military networks is that the Internet has many single users on low bandwidth access connections and a large backbone capacity network, whereas a military tactical network tends to have a number of users on a high capacity local network connected across a low bandwidth backbone network. The other difference is in the distribution of the functions performed by the users. The different levels in the chain of command are not going to be co-located. So any decision or information pertaining to the decision making process will need to be transferred across the backbone network.

Military tactical networks are not static, certainly when they are needed most during insertion or removal of forces. This process of moving parts of the network has an impact on the capacity available in the network. The geographic spread of the network usually means the tactical backbone has to be supported by radio link. As seen already (Chapter 4) these links are prone to errors.

The evaluation of TCP as a transport protocol needs to be assessed to see how well it performs in a tactical military network. The issues of traffic types, network topology and error prone links need to be considered.

7.2 Network Architecture

An important area to consider when studying network protocols is the architecture of the network. The test network has to be representative of the network in which the protocols would normally be used. In military tactical networks one of the major traffic flows is the data exchange between headquarters. The architecture that supports this connectivity is high speed LANs connected with a radio relay connection. There may be a couple of radio links between the two headquarter LANs. Figure 7-1 shows a representative network. The major feature is that the workstations connected to the LAN have no knowledge of the limited 512 kbit/s error prone radio link. There is a large bottleneck from the LAN onto the radio link (19.5:1). The bandwidths are representative of present equipment, though the full 512 kbit/s bandwidth is unlikely to be available for data traffic. The 10 Mbit/s LAN is most likely to be updated to 100 Mbit/s, leading to an even larger bottleneck (200:1). Next generation radio links are likely to support bandwidths of 2 Mbit/s or more, which will help reduce the bottleneck.

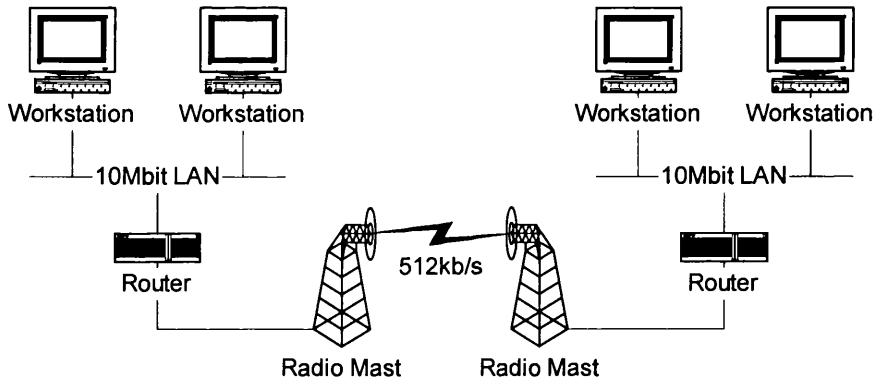


Figure 7-1 Network Architecture

Satellite links are not considered in this study, as their error rates are lower than tactical radio links. There has also been considerable work already done in improving the performance of TCP over satellite with timestamps, SACK and increases in the maximum congestion window size.

7.3 Basic TCP behaviour

TCP is quite a complicated protocol (Chapter 3). It is worth describing the protocol in order to illustrate how the different mechanisms work. A TCP connection always begins with slow start. Slow start isn't that slow with the congestion window increasing exponentially (Figure 7-2). Due to the rapid growth in the window the loading on the network also increases quickly. When TCP reaches the point of congestion, invariably there is so much data that more than one packet is dropped. As more than one packet is dropped a timeout retransmission is needed to recover the data.

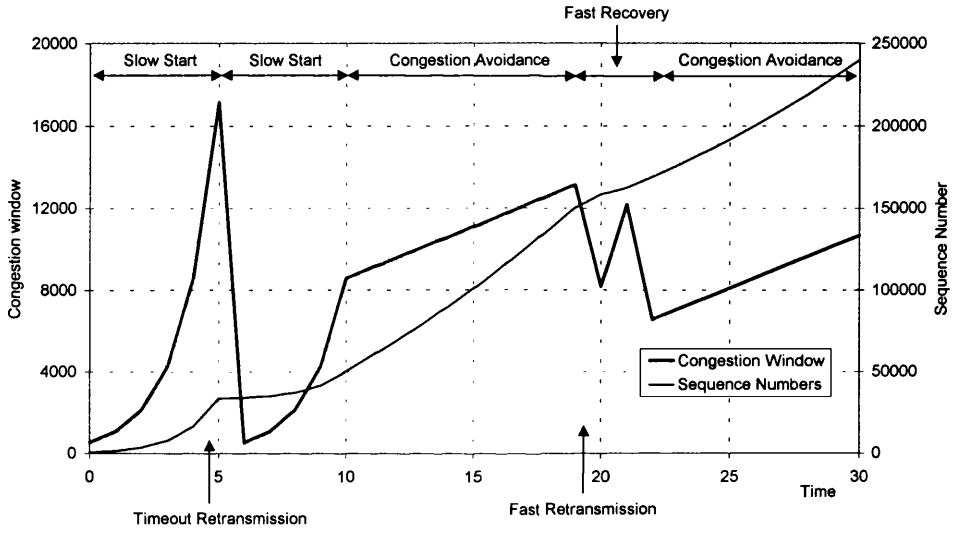


Figure 7-2 TCP Window dynamics

Once there has been a retransmission, TCP keeps track of a threshold above which congestion avoidance is used to control the size of the window. After a timeout retransmission, the window increases exponential up to the threshold then switches to the linear increase of congestion avoidance (Figure 7-2). The congestion avoidance increase is proportional to the round trip time. This prevents the window from growing too quickly on long delay links, where the time to detect congestion is also long.

During the fast recovery stage there is a spike. This large window does not mean more packets in the network. As the lost packet was not acknowledged and one edge of the window is not moving, this prevents more packets from being sent. To allow more packets during fast recovery the window is increased by one packet per acknowledgement. Hence the growth (spike) in the window. On receiving the acknowledgement for the lost packet the window is reset to the threshold and continues with congestion avoidance.

7.3.1 Analytical Models

Several researchers have attempted to derive an analytical model of TCP (7.1) [Floyd99] [Ott96]. The models are only for the steady state condition and low packet loss ratios.

$$T \leq \frac{1.5\sqrt{2/3} \cdot B}{R \cdot \sqrt{p}} \quad (7.1)$$

B = maximum segment size (bytes)
R = round trip time
p = packet drop rate
T = throughput (bytes/sec)

TCP assumes that all packets are lost due to congestion. The response to this is to reduce the amount of traffic offered to the network. If the packet losses are due to corruption not congestion then TCP is unlikely to fully utilise the capacity of the network. On a timeout retransmission the window hence offered load is reduced to a minimum. The fast retransmission only halves the window but requires a window of greater than four packets to be triggered. If there is a large number of corrupt packets the timeout retransmission process could keep the window below the four packets. This would mean all retransmission would be via a timeout.

The timeout period used is dynamically controlled from the measured round trip time. The measurement can only be taken for unambiguous packets. These are packets that are unique and have not been retransmitted. With high error rates on the links the number of retransmissions is likely to be high, hence it is not possible to get an accurate estimate of the optimum timeout period. This non-optimum timeout is also likely to cause an increased degradation in performance.

7.4 TCP Simulation Model

As described above the analytical models only work in a limited set of conditions. The best way to assess performance is with a real network, to give results that will be exactly representative of the way the protocol would normal behave. There are a few problems with using a real network. One is being able to control the environment to give reproducible results and another is that the test may be more dependent on the implementation than on the protocol [Allman97][Comer94][Paxson97]. The implementation on a real network is dependent on the process architecture and the response time of the system.

The aim of this study is to assess the protocol independently of the implementation. This means processor dependent delays need to be removed. The way to do this is with a simulation where the actions of the system do not happen in real time. The time measurement of actions is done relative to the simulation time. The simulation time does not increase linearly with respect to real time, it increases relative to the actions being made. There are several other advantages in using simulations. One is being able to change how the protocol functions and to be able to test different parts of the protocol separately. Another is that it is possible to run tests faster than real time. Hence it is possible to get results quicker than using a real system. It is also possible to try functions that are too processor intensive to run on present equipment. The simulation would run longer than real time but could test functions that could be used with new faster technology.

7.4.1 Measuring Performance

When studying networks it is necessary to measure how well the network is performing. There are many ways to measure the performance of a data network. The measurement used must be appropriate to the protocol being studied.

A useful measure when looking at individual links is link utilisation. This gives a measure of what percentage of the bandwidth is being used, hence whether there is any spare capacity. When a connection flows from a low bandwidth link into a high bandwidth link, it is impossible to fully utilise the high bandwidth. When a connection flows from a high bandwidth link into a low bandwidth link there is going to be congestion, but the utilisation will be high. When looking at links the other useful measure is the size of the queue before the link. The fuller the queue the greater the congestion. The ideal state of the network is where all links are 100% utilised and the queues are kept to a minimum.

Link utilisation and queue length give good indications of how well the network is dealing with the loading. It does not give any indication of how much useful data is crossing the network. It is possible to fill a link with network management data leaving no space capacity for user data. From a users perspective it is better to look at the amount of data being sent or received. Throughput is the rate at which user data is being received, this can be measured as packet by packet or for a complete file. The throughput for a packet is calculated from the size of the data in the packet (P_s) divided by the time delay from the previous packet (T_d) (Figure 7-3). This gives a good instantaneous reading of the received data rate. There are a couple of circumstances where an incorrect or ambiguous measurement could be made. The time delay becomes high when there is a break in the packet flow, such as when one file finishes and the next one starts. The first measurement for a file should be considered invalid. The other condition is when the receiver window buffers an out of order packet. When the correct packet is received the receiver window passes both packets simultaneously, leading to a time delay of zero seconds, hence an infinite packet rate. To get a measure for a file, the packet throughput could be averaged over the number of readings for the file, but this leads to problems. The better approach is to calculate the throughput (G) over the entire file (Figure 7-4). The first packet is only used as a start marker as it is impossible to time the arrival of the first bit of the first packet, due to OPNET only handling packets not bits.

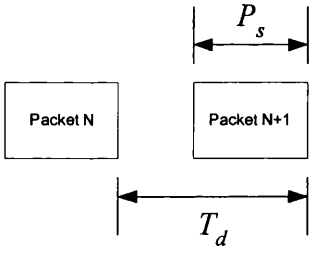
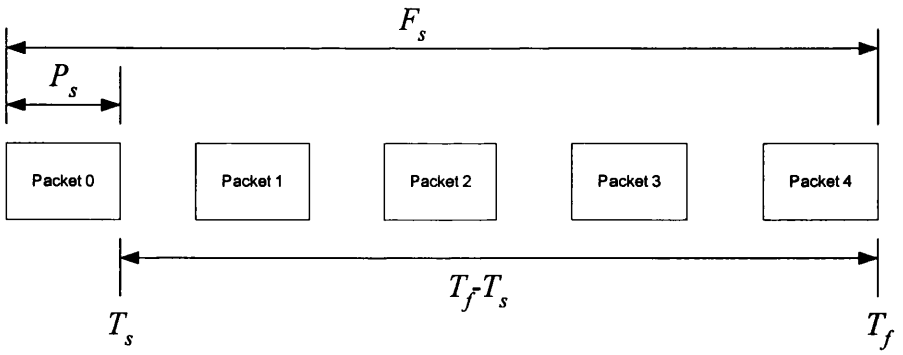


Figure 7-3 Calculating packet throughput



$$G = \frac{F_s - P_s}{T_f - T_s}$$

Figure 7-4 Calculating file throughput

Throughput only gives a measure of how quickly the data is being received not the length of time it took to cross the network. From the user's perspective the important issue is how long it takes before the received data is usable, i.e. the complete file has been received. This delay is simply the time from sending the first packet to receiving the last.

The advantage of using throughput is that it can be compared against the slowest date rate link the connection uses. The ideal throughput is that of the link date rate. This is unachievable due to network overheads, but gives a good reference condition. The ideal delay is instantaneous transfer, which does not give a good reference condition. The delay is also dependent on the size of the file being transferred.

Another useful measure is the request to response time. When web browsing, the users see the delay between clicking on a link and the page being visible. The delay is the time between the HTTP request being made and when the new page is completely loaded. This could be considered as two independent connections one flowing directly after the other but in opposite directions. Hence the request response time is simply the sum of the delays associated with each connection.

7.4.2 Assumptions

There are a vast number of variables associated with TCP/IP networks. Not only TCP options but also link bandwidths and network architecture all influence the results of the simulations. Realistic values are needed for variables that are not intended to effect the results. The start point for this is selecting a network architecture, which has already been discussed (Chapter 7.2). The link speeds selected (Figure 7-1) are representative of current equipment and the study concentrates on the performance of TCP in an error prone environment. To remove the effects of congestion and bandwidth sharing, only a single connection is considered. The effects of congestion and bandwidth sharing are considered separately. The errors are only injected into the radio link, representative error rates are considered in Chapter 4. The errors are present in both directions across the link affecting both the packets and the acknowledgements. Errors on the wired LAN are negligible compared to those on the radio link. Errors on the LAN can also be minimised with good cabling practices. An issue relative to the network architectures is propagation delay. The transmission delay of a packet is considerably larger than the propagation delay for the links being considered. In the simulation, the propagation delay is assumed to be zero, to avoid measurement of fractions of a packet. The only case where propagation delays are considered is when evaluating the performance over satellite links.

IP packets cannot be carried directly by the physical layer, due to the lack of framing. IP would normally be encapsulated in a Media Access Control (MAC) frame for Ethernet or Point to Point Protocol (PPP) [RFC1661] for a serial connection. To simplify the model all connections use the same physical layer encapsulation process. The chosen framing is High-level Data Link Control (HDLC). HDLC includes framing flags and a CRC (Chapter 6). HDLC is configured to work with bit stuffing, but due to the complexity of doing this in OPNET it has not been implemented. All errors are assumed to be detected by the CRC, however this is not always true (Chapter 4).

Other factors that need to be considered are the size of the files being transferred and the maximum segment size. The chosen file size is 20 kbytes; this is representative of a web page or a reasonable quality image used on a web page. The maximum segment size (MSS) chosen is 536 bytes and is based on an maximum transmitted unit (MTU) of 576 bytes as discussed in Chapter 3.

The effect of process speed is considered negligible and all calculations processed are considered instantaneous. There is one feature of a processor that is implanted and that is the accuracy of timers. Most operating systems need interrupt timers to do timing functions. If the interrupt timer is very fast the operating system becomes clogged with handling the interrupts. There is a balance between loading caused by the interrupt timer and the accuracy of a fast interrupt. Most operating systems default to a timer interrupt of about 100 Hz. In some operating systems the main system timer interrupt is also used to measure system time. This gives an accuracy of 10 ms for both measuring time intervals and generating timeouts. OPNET defaults to a simulation time accuracy of fractions of a picosecond. The TCP simulation software includes functions to reduce the accuracy down to that of a real system.

FEC is not implemented in the simulations. A simple FEC that could be implemented would simply have the effect of lowering the residual bit error rate and a decreasing throughput by predictable amounts.

7.4.3 Error injection

To study the performance of TCP over error prone links it is necessary to corrupt packets travelling across links between nodes. OPNET supports links using a piping process, including transmission delay, propagation delay, error generation and error detection. A special error generator was built to produce Gilbert and Elliot distributed errors and the new burst error model. There is no support to build these models inside OPNET, so they have to be complied outside the OPNET environment. It is then necessary to force OPNET to link in the new error generation during the building of the simulation. OPNET deals only with packets not bits and the error injected has to produce an error for a complete packet at a time. This means storing the state of the error injector between packets.

OPNET does not corrupt the data in the packet but has a separate field to store the number of bit errors injected into the packet. The error detection process of the link model then uses this field to determine whether the packet is in error.

7.4.4 OPNET models

As described in Chapter 2, OPNET models are separated into layers, network, node and process. The node stacks the process (Protocol) into layers that matches that of the ISO stack (Figure 7-5). The traffic generator (gen_flow) produces packets in a defined manner and records when the data is received. This is where the performance measurements are calculated. The TCP source (tcp_flow) takes the data from the generator and encapsulates it into TCP segments. On receiving data the TCP source passes it up to traffic generator where the performance is measured. The TCP flow control algorithms are implemented in the TCP source. Each process is capable of only sending one stream but is able to receive multiple. The process below the TCP source is a simple TCP port multiplexer de-multiplexer (tcp). This combines and splits the separate TCP flows based on the port number defined in the TCP segment. The terminal node is also capable of producing and receiving UDP data. The UDP process (udp_frag) breaks the data from the traffic source into small packets, encapsulates them in a UDP segment and passes them to the UDP multiplexer (udp). Received UDP segments are passed back up the stack to the traffic where again the performance of UDP traffic can be measured.

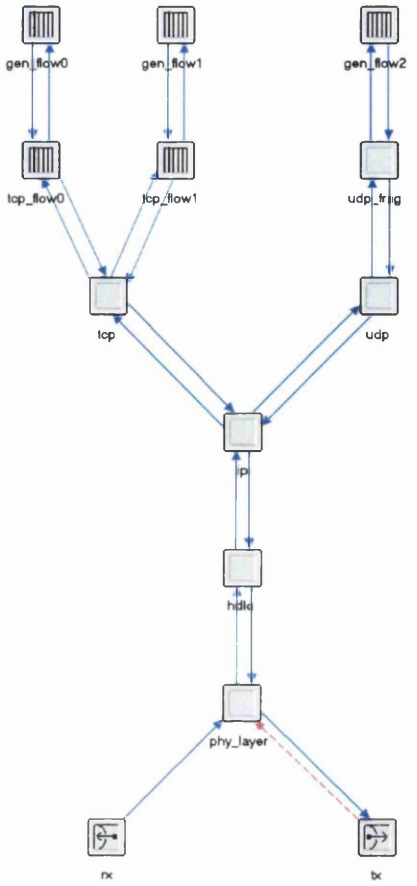


Figure 7-5 Terminal protocol stack

The IP multiplexer (Figure 7-5) (ip) encapsulates the TCP or UDP segments into an IP packet. On receiving packets the protocol field in the IP header is used to separate the UDP and TCP traffic. The HDLC process (hdlc) encapsulates IP packets into HDLC frames. This is to give a more realistic packet size over the links.

The physical layer process (phy_layer) is provided to monitor whether or not packets are dropped due to errors in the payload, packets with errors in the header are dropped by the link model. For the case of IP the header size encompass the whole packet, hence any packets with errors are dropped by the link model. The purpose of this is to allow for the possible development of protocols that allow errors in the payload to be carried through to the end application. The other function of the physical layer is to allow for the possibility of back pressure up the stack. This prevents an application from sending more than one packet at a time, and stops the build of congestion queues in the end terminals. This is achieved in a real operating system by having the application's thread responsible for sending the packet. If the packet cannot be sent the application is blocked (prevented from running) until the packet is sent.

Multiple terminals can be connected together with switches (Figure 7-6) to form a network. The switch removes the HDLC framing and passes the IP packet to the switching process. The switching process routes the packet based on the IP address (Appendix B). The IP packet is then re-encapsulated in a HDLC frame and sent.

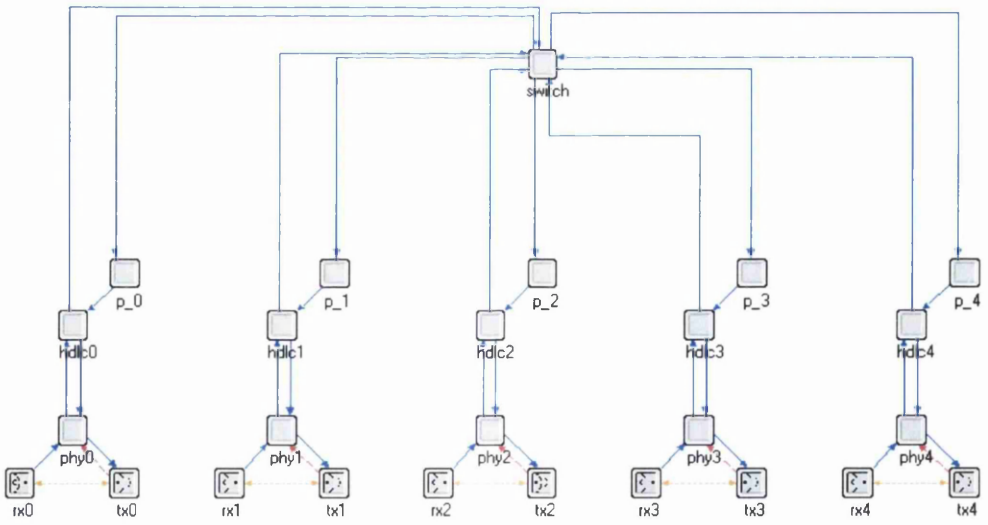


Figure 7-6 Switch Node

It is possible for the total amount of data coming into the node to be greater than a single output port is capable of sending. The standard OPNET transmit process (Figure 7-6 – tx[0-4]) has a queue. The extra process (p_[0-4]) provides monitoring of the queue length. If the length exceeds a given level the packets are dropped.

These terminals and switches can be combined together in different combinations to form an IP routed network. The terminal can send two TCP streams and a UDP stream, and receive any number of streams. The bandwidth of the links can be controlled via parameters in the OPNET transmit process. Errors are injected onto the link by the process describer earlier (Section 7.4.3).

7.4.5 Test conditions

There are a few features of military networks that differ from commercial networks. This is mainly due to the architecture that supports the use of low data rate, error prone links. The effect of low data rates is to increase the latency of packets in the network, the errors cause packets to be corrupted. Another aspect is that there may be malicious attempts to disrupt communications (Electronic Countermeasures). This can be done by forcibly injecting errors into the communication links.

For testing the performance of TCP, an isolated TCP stream is used. This removes any effects caused by the possible interaction of other network traffic. The network used mimics a file transfer between two remote terminals connected across the network (Figure 7-7)(Figure 7-8). The terminals are connected to a local switch via a 10 Mbit/s switched LAN. The switches are connected via a 512 kbit/s point-to-point radio link, or multiple links. Only the 512 kbit/s radio links are prone to errors. The 10 Mbit/s LAN and 512 kbit/s links means that congestion will not occur at the links to the terminal or in the terminal. This means TCP has to be able to detect the link capacity properly.

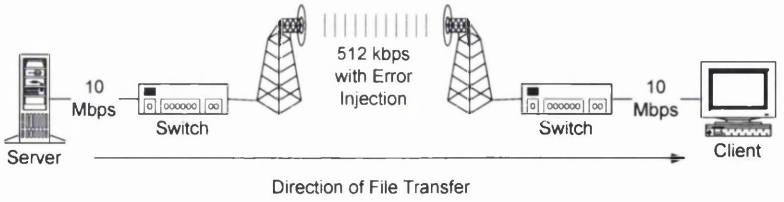
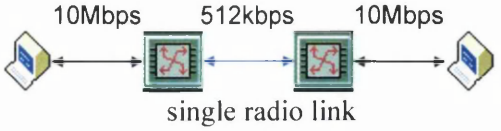


Figure 7-7 Representative Network



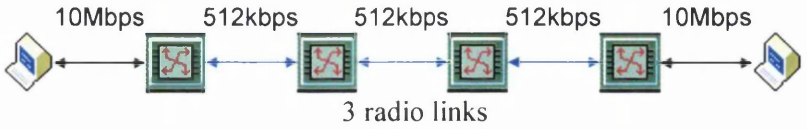


Figure 7-8 Base Test Network (OPNET)

To test the performance of TCP in the presence of congestion, a number of terminals perform a continuous file transfer to a single server (Figure 7-9). The server is connected to the switch via a 512 kbit/s link, all the sending terminals are connected to the switch with 10 Mbit/s links. The point of congestion is from the switch to the server. The reverse path from server to terminal will not be congested due to TCP acknowledgements being considerably smaller than TCP segments and will not be sent at a rate greater than the rate segments are received.

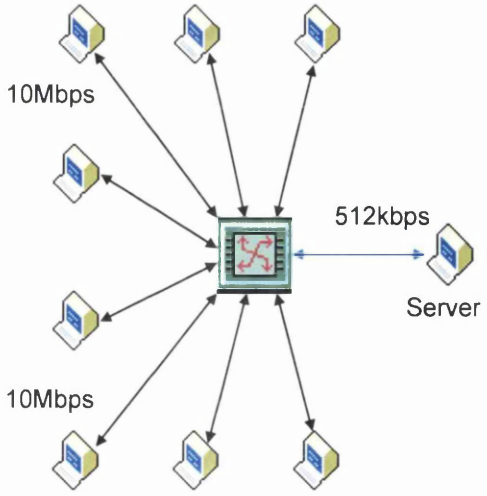


Figure 7-9 Congestion Test Network

The results look at the performance of different TCP mechanisms in the presence of random errors; the effect of burst and pulsed errors. The effects of long delays and shared links are also considered.

The simulations cover the various options that are available in the TCP algorithm. The aim is to find which option produces the best performance in a degraded low capacity network. Another aim is to identify any potential problems that may occur when using TCP in this type of network.

7.5 Results

The results shown here are averaged over a number of files transferred under the same conditions. For a test of the error injection process a UDP flow was used. UDP has no feedback to the sender. The sender was configured to send a continuous stream of packets to keep the link full. The theoretical throughput was calculated for the UDP flow. There is no difference between the simulation and the mathematical analysis (Figure 7-10). This shows that the error injection, network and the measurement process are accurate. This does not validate the TCP model, validation with an analytical model cannot be achieved due to the lack of a representative model (Section 7.3.1).

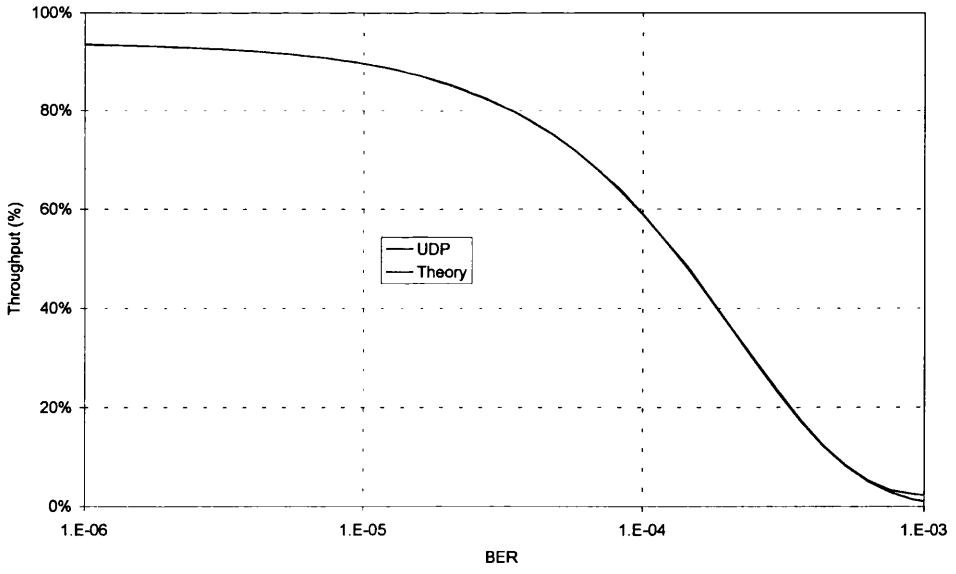


Figure 7-10 Check of Corruption to UDP packets

7.5.1 Base Line Performance

To make any comparisons between different TCP mechanisms it is necessary to select a reference mode (Base-TCP). The reference mode selected is the most common selection of the TCP mechanisms. This includes fast retransmission and fast recovery. One of the default mechanisms used is delayed acknowledgements, the reference model does not use delay acknowledgements but generates an acknowledgement for every packet received.

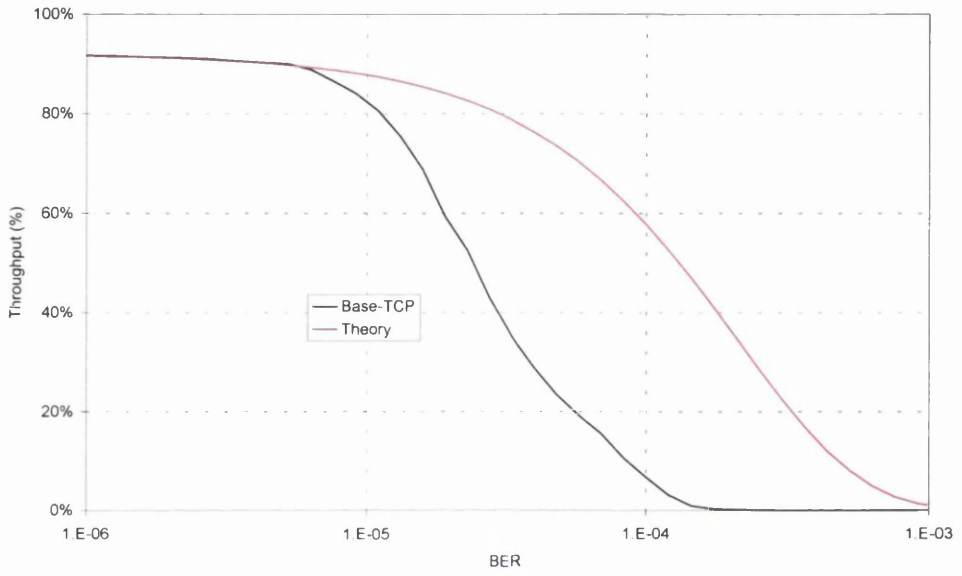


Figure 7-11 Performance of Reference TCP

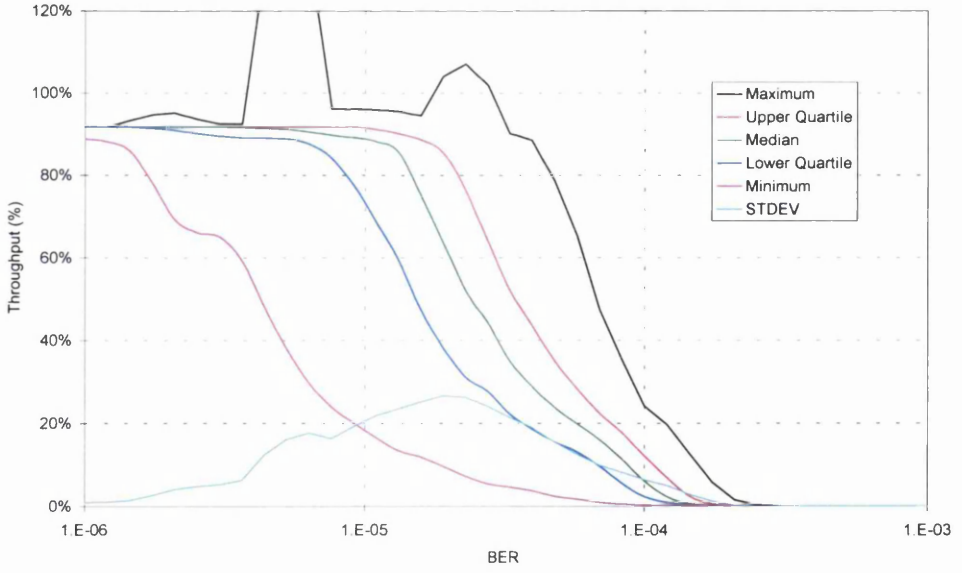


Figure 7-12 Distribution of results

The Base-TCP drastically under performs when compared against the maximum theoretical throughput (Figure 7-11). The theoretical throughput is calculated based on probability of dropping a TCP packet and correcting for the overhead (7.2) (MSS is 536 bytes (Section 3.4.4), TCP/IP header is 40 bytes, HDLC header is 7 bytes, given 536+40+7=583 bytes for total size of packet). The reason for the large difference is due to the way TCP responds to lost packets. TCP assumes all lost packets are due to congestion, it responds by backing off on the transmission rate by decreasing the size of the congestion window or doubling the timeout interval.

$$Throughput = (1 - p_e)^{583\times 8} \cdot \frac{536}{583} \quad (7.2)$$

Figure 7-12 shows how the individual results are distributed. The quartile range is reasonably small, meaning greater than 50% of the results give a good estimate of the throughput. The standard deviation increases at the point where the performance drops off. This is expected, as small variation in the distribution of the random error will cause throughput to change. The large maximum peaks (greater 100%) are caused when the first packet is lost and the whole file is buffered. When the first packet is received the whole file is passed to the measurement process, making it appear as if there is a very high bandwidth. This effect can happen in a real network.

7.5.2 Comparison of TCP mechanisms

There are many different mechanism used by TCP to improve performance. It is necessary to look at the different mechanisms to see whether they are worth using. The first to be considered are fast retransmission and fast recovery. The simple TCP (Figure 7-13) uses only timeouts to recover lost packets. It is clear that fast retransmission does produce a marked increase in throughput in an error prone environment. Fast recovery increases the throughput slightly, this is expected as the window is not reduced to one packet. With very high error rates there is virtually no difference in the performance of the mechanism. This is because TCP has to fall back to timeouts to recover the data. For TCP to do a fast retransmission the congestion window has to be larger than four packets. If a packet is lost before this window size is achieved a timeout is needed and the congestion window is set back to one packet. The fast retransmission and fast recovery is used in the Base-TCP that is being used as a reference implementation of TCP. There is no network overhead associated with using fast recovery and fast retransmission.

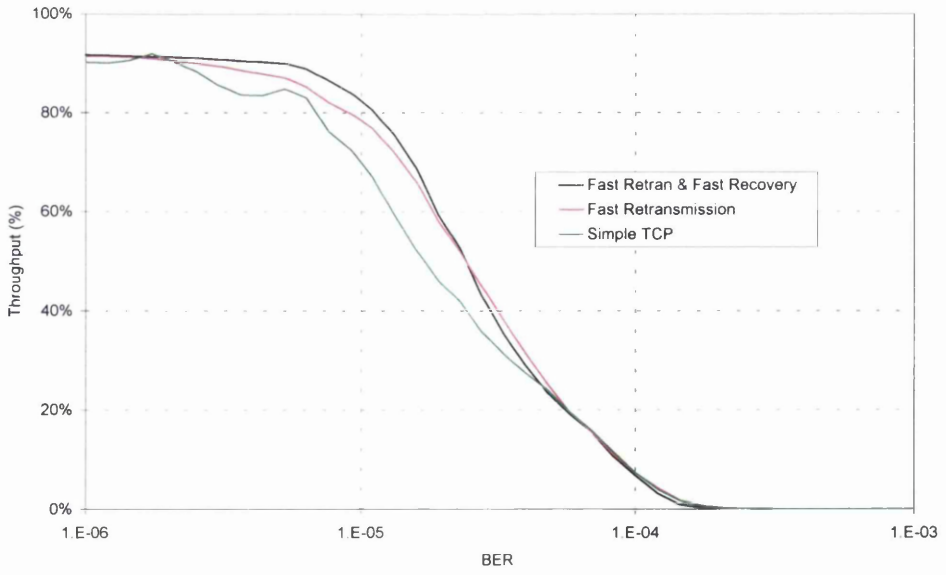


Figure 7-13 Performance with Fast Recovery

One of the default mechanisms used by TCP is delayed acknowledgements. Under very high error conditions, delayed acknowledgements impact on the performance of TCP (Figure 7-14). The effect of delayed acknowledgements is to reduce the amount of information being feed back to the sender. The aim of time stamps is to provide the sender with more accurate information about the round trip time. There is a small cost in throughput due to the extra overhead information carried in each packet (left hand side of Figure 7-14). For the case without delayed acknowledgements there is a very slight increase in throughput due to the better estimation of the round trip time. There is a decrease in throughput when time stamps are used in conjunction with delayed acknowledgements. This is because the round trip time calculated also includes the delay caused by the delayed acknowledgement (Figure 7-15). The increase in round trip time also increases the timeout, hence it takes longer to recover from lost packets.

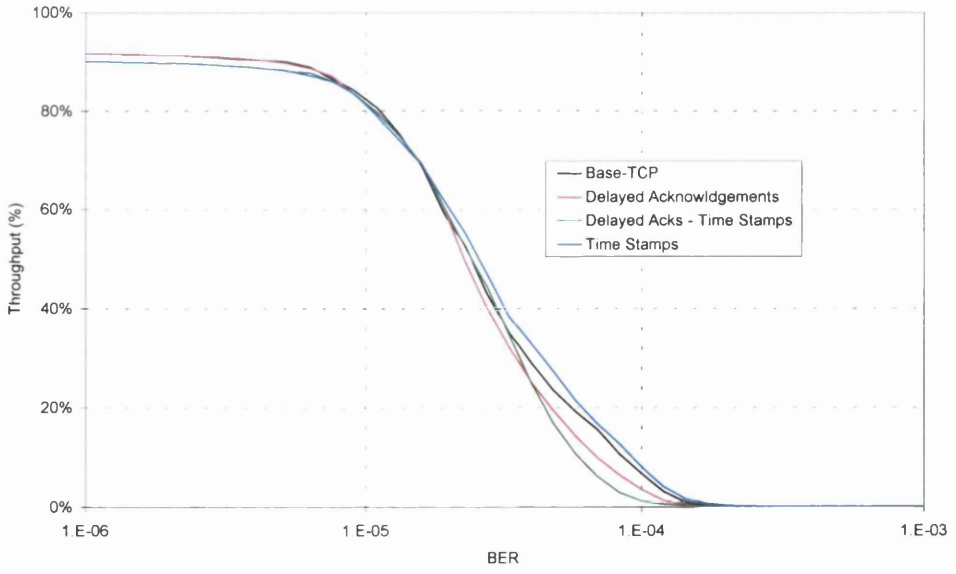


Figure 7-14 Performance with Delayed Acknowledgements and Time stamps

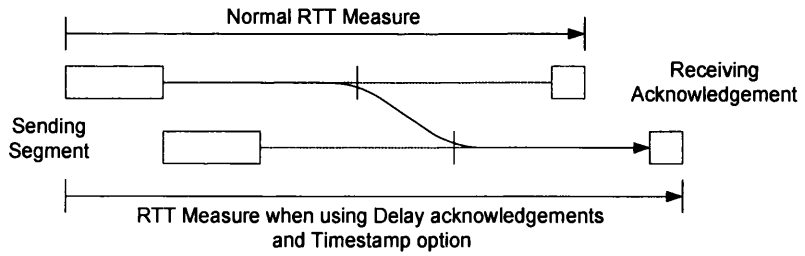


Figure 7-15 Difference in RTT estimation

A TCP mechanism that is expected to increase throughput would be one that employs selective acknowledgements. However, this does not appear to be the case (Figure 7-16). For selective acknowledgements to be used the TCP window has to be greater than six packets, and more than one packet needs to be dropped in a congestion window. A single packet will be recovered with fast recovery. If the error rate is low the window is able to grow to a large enough size but the probability of multiple error is also low. The number of SACK recovered packet is going to be small, and hence does not have a major impact on the performance. With high error rates the window is unable to grow due to timeout retransmissions resetting the congestion window. Figure 7-17 shows the percentage of retransmission types relative to the total number of packets. As stated earlier the majority of packets at low error rates are recovered with fast retransmission and the majority of packets in a high error environment are recovered with timeout retransmissions. Few packets are recovered with selective acknowledgement.

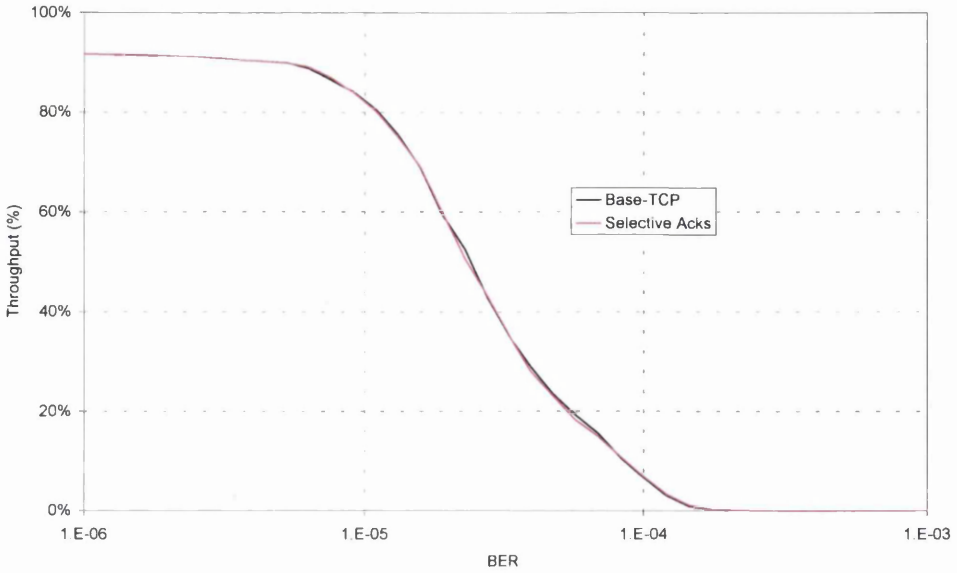


Figure 7-16 Performance of Selective Acknowledgement

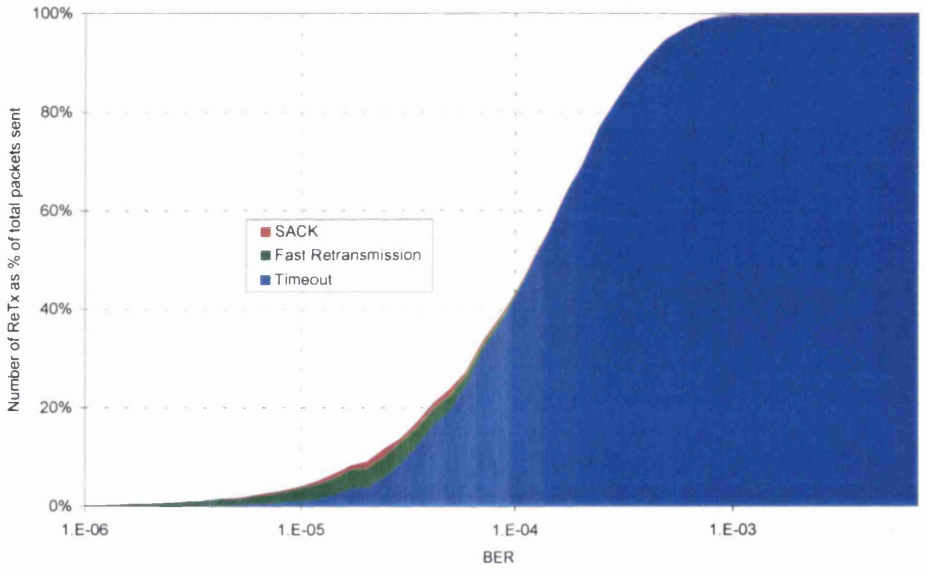


Figure 7-17 Ratio of Retransmission types

One feature of TCP that needs to be considered is the size of the packets. The value that controls this is the maximum segment size (MSS). There is an upper limit defined by the network but there is no reason why a packet cannot be smaller. The size selected for most of the simulations is 536 bytes (Chapter 7.4.2). The MSS with Ethernet packets is 1460 bytes and the other value tested is 248 [((536+40)/2)-40] (i.e. half the size of the transmitted packet). The cost of smaller packets is an increase in the network overhead (left side of Figure 7-18). The advantage is a higher probability of getting through uncorrupted and hence a higher throughput with a higher error rate.

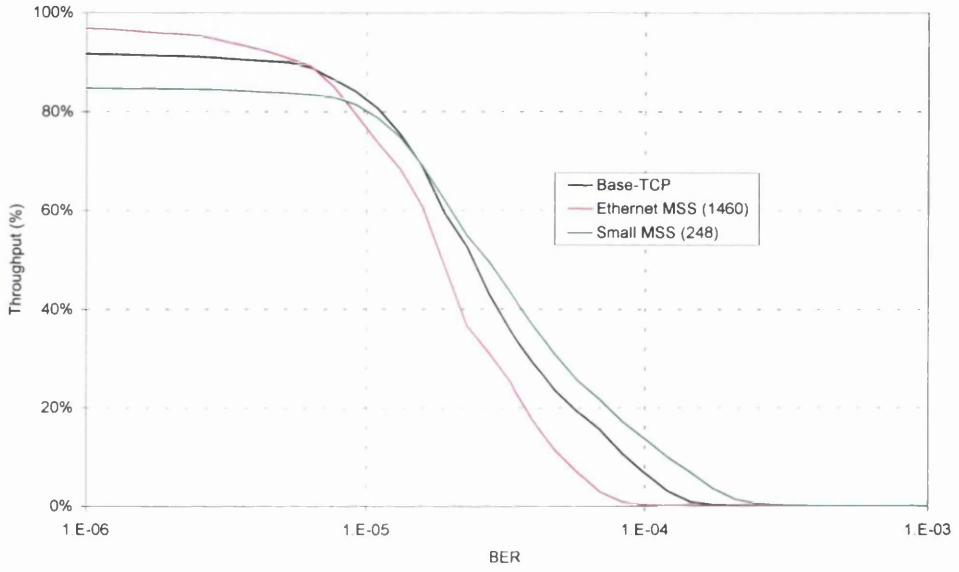


Figure 7-18 Performance with different size packets

So far the results have only considered the throughput. The other measure of performance is file transfer delay (Figure 7-19). The delay in all cases increases very rapidly at the point of collapse. Again altering the MSS has a greater effect on the delay than other TCP mechanisms. With very little difference between the mechanisms, the only obvious difference being that of using delayed acknowledgements and timestamps. The file transfer delay can be transformed into a graph that has the same characteristics as the throughput graphs. This is achieved by simply dividing the file size by the file transfer delay.

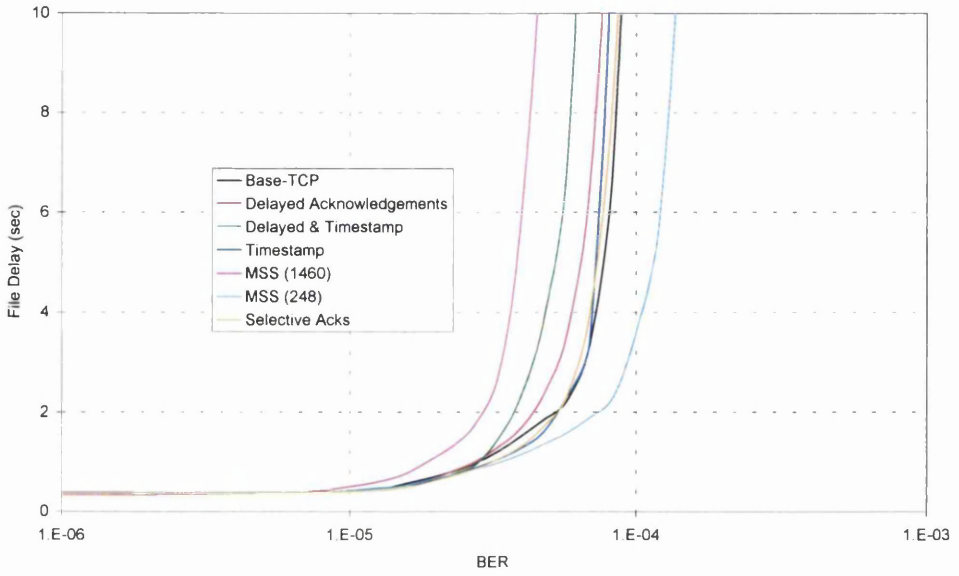


Figure 7-19 File Transfer delay

7.5.3 Link Error Patterns

In Chapter 4 it is shown that the errors on radio links tend to be 'bursty' in nature. Figure 7-20 shows the throughput of TCP in the presence of different length bursts. The bursts are generated with the Gilbert and Elliot model (Chapter 4). The difference in performance due to burst errors is very slight. This is the same difference generated on a raw block data stream (Chapter 5). There does not seem to be any strange effects caused by burst errors.

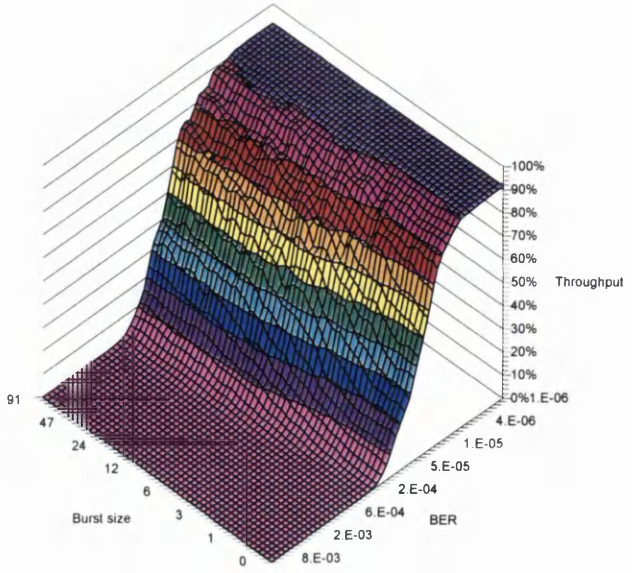


Figure 7-20 TCP with burst errors

The effect of pulse errors is different (Figure 7-21). The bit error rate is the same across the whole graph. The flat area is caused by the pulse width being greater than the pulse period, hence all have the same random bit error rate (10^{-5}). For most cases the pulse errors have the effect of an increase in throughput compared with the purely random case. There is a large hole in the middle of the graph where the interaction of the pulse errors and TCP cause the throughput drop. The hole is at point where the pulse width is large enough to cause errors in two consecutive packets (4608 bits / 512 kbps = 0.009 sec). The pulse period is roughly that of the estimated round trip time (0.2s). The double packet loss causes TCP to use a timeout retransmission and the pulse period means that the retransmitted packet will also be corrupted, causing yet another retransmission. As the timeout retransmission is doubled the timeout period is still a multiple of the pulse period, hence further retransmission could also be affected. These consecutive retransmissions have the effect of considerably reducing the throughput.

The selective acknowledgement mechanism should be able to handle consecutive corrupt packets without requiring a timeout. Figure 7-22 shows the throughput using selective acknowledgements. Figure 7-23 shows the difference in throughput between using and not using selective acknowledgements. There is a major improvement at the point where TCP collapses due to pulse errors.

The pulse error used so far has been a single long pulse with an adjusted error rate to give the same average bit error for different pulse patterns. With long pulses the error rate during the burst is reduced, possibly allowing packets to get though even during the error pulse. If a double pulse is used then each pulse could more reliably corrupt a packet. The double pulse giving a higher probability that two consecutive packets will be corrupted with the same error rate. When this double pulse is used the hole in the throughput is increased (Figure 7-24).

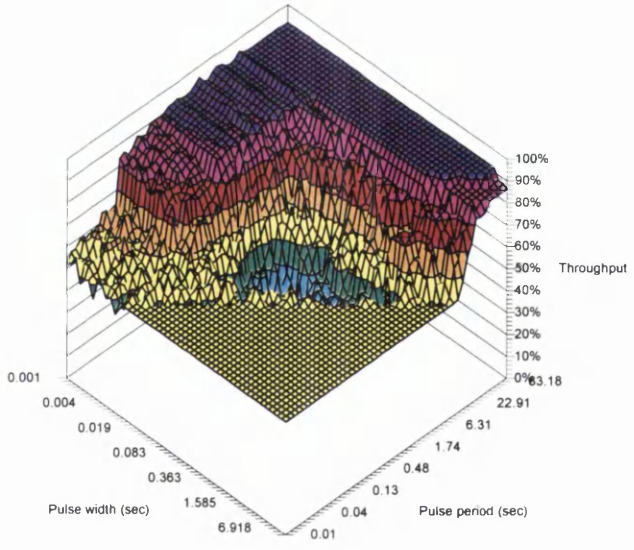


Figure 7-21 Effect of pulse errors on Base-TCP

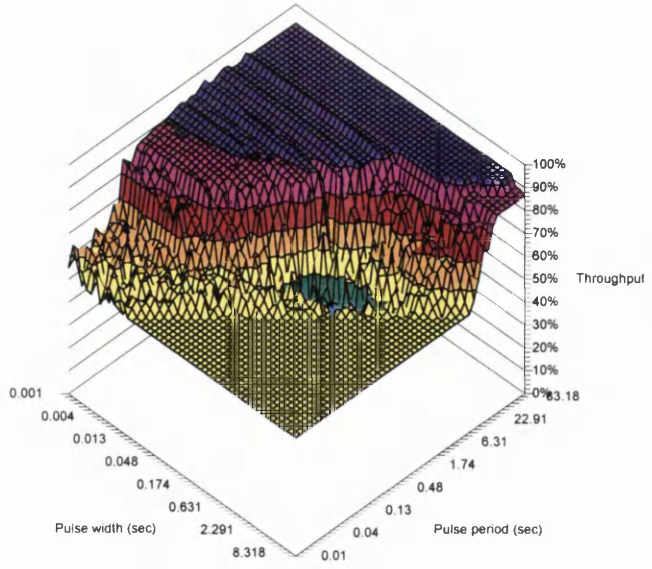


Figure 7-22 Effect of pulses errors on SACK

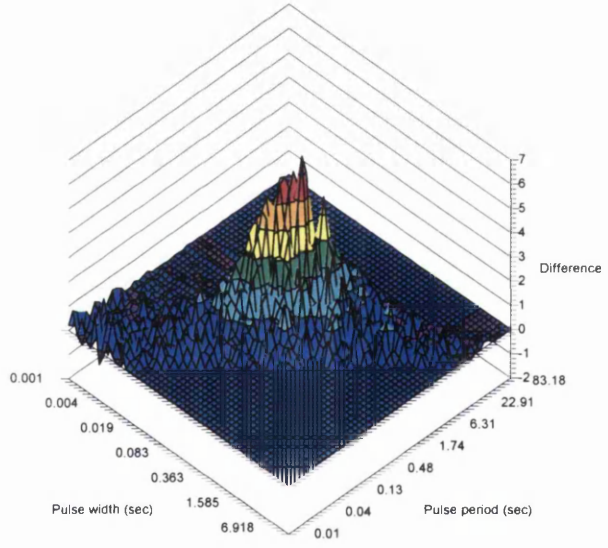


Figure 7-23 Difference in performance with SACK

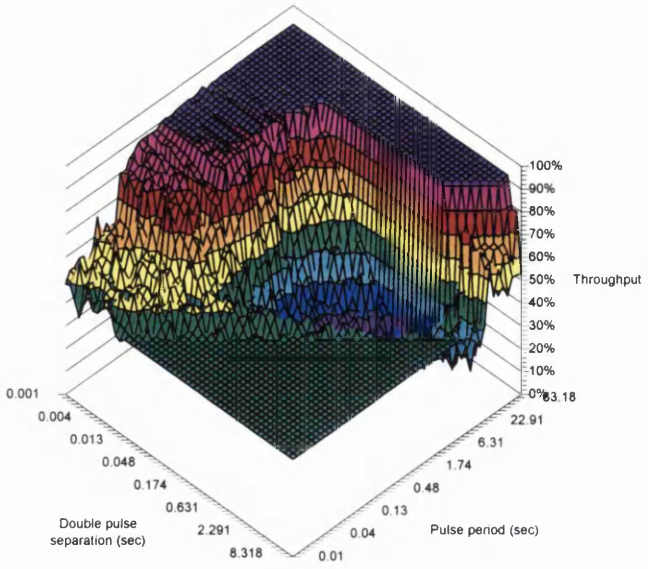


Figure 7-24 Effect of double pulse errors

7.5.4 Effect of Link Delays

The only network variable considered so far is the link bit error rate. Another important variable is the total end-to-end delay. This is simulated by increasing the propagation delay of a link. The range chosen covers short-range point-to-point radio link (3 μs) through to a double hop satellite link (500 ms). TCP's start up process, begins with a window of one packet and increases it by one for every acknowledgement. If the delay between sending a packet and receiving an acknowledgement is high then the rate at which the window increases will be slow. The slower the growth of the window the longer TCP takes to reach the capacity of the link. This leads to considerably lower throughput for long delay links (Figure 7-25). The longer the connection is used the more data is sent at the higher rate (window has already grown). Increasing the size of the file increases the throughput (Figure 7-25). The ridges are caused by the interaction of the size of the window and the size of the file. If the last packet has to wait for the next window of data, the throughput will be lower than if it fits exactly into the number of packets in the window. The size of the congestion window doubles per burst of packets. This is why the ridges are spaced with doubling distance between them.

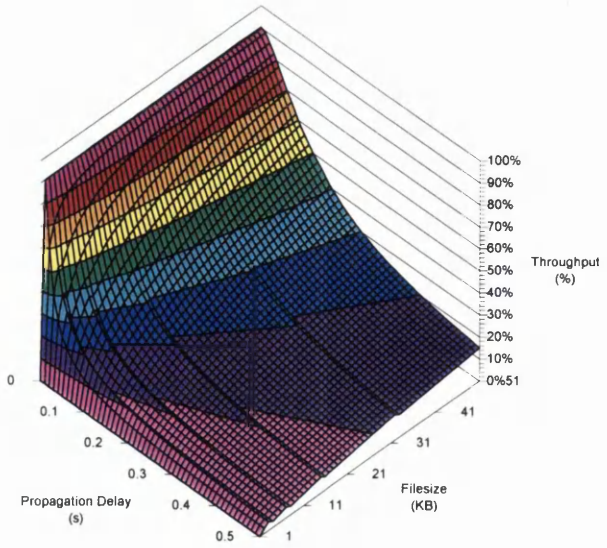


Figure 7-25 Effect of Propagation Delay and File size

An excessively high round trip time can have a major impact on the performance of TCP. This happens when the delay is greater than the initial timeout value. This means a timeout occurs before there is enough time for the acknowledgement to be returned. Due to the retransmission the slow acknowledgement cannot be used to estimate a new round trip time. At just over the default timeout of 3 seconds the first packet is retransmitted (Figure 7-26). The timeout is doubled to 6 seconds for the retransmission. The second packet has time for the acknowledgement to return before the timeout expires, hence it can be used to estimate a round trip time. This means that TCP recovers quickly and carries on normally after the first retransmission. When the delay is greater than 6 seconds delayed acknowledgements suffer due to the acknowledgement being delayed beyond the timeout period. With a delay of greater than 9 second TCP fails to recover. This is because the acknowledgement for the first packet is delayed by two timeout periods, first being 3s the second being 6s. On receiving the acknowledgement the timeout period is set back to 3s, but without being able to estimate a new round trip time. TCP ends up permanently in a retransmit state without being able to estimate the round trip time (Figure 7-26). This will have a major impact on the throughput.

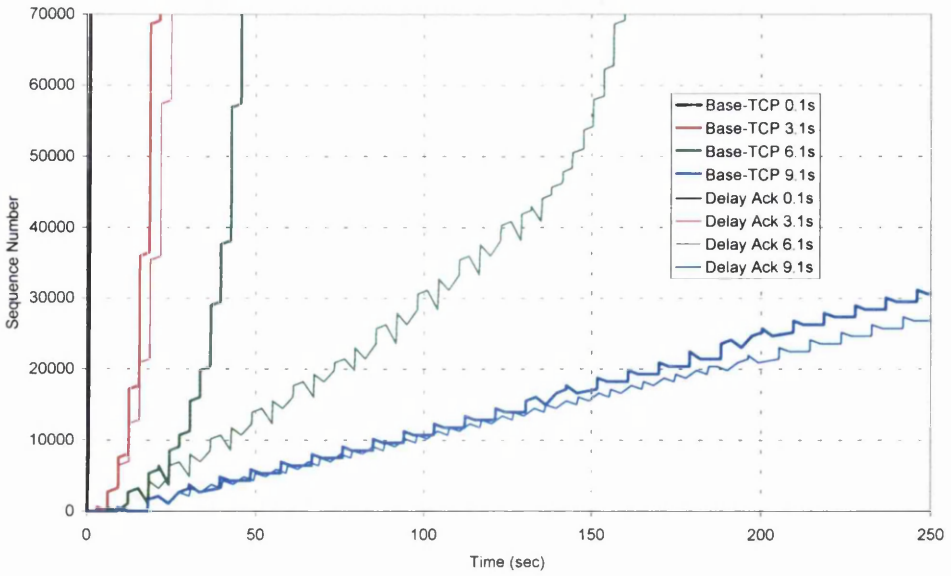


Figure 7-26 Effect of very long round trip time

7.5.5 Congestion

TCP is able to adjust the rate at which it sends data to match the available bandwidth, even if the available bandwidth is dynamic such as in the case where several TCP flows share the same link. Figure 7-27 shows the progression of the sequence numbers for eight TCP flows sharing the same link. The buffer preceding the link is large enough to hold 50 packets. The rate at which each flow proceeds changes over time and can be seen as a non-linear attribute of each line. This means TCP is able to adjust to the bandwidth but is unable to reach a stable state where all flows proceed at the same rate.

TCP detects the limit of the bandwidth by detecting lost packets. To create lost packets TCP first needs to push the packet rate to a point where packets are dropped from the queue. This means the queue preceding the congested link will usually be full (Figure 7-28). The effect of continuously overloading the buffer means packets are going to be dropped. In this simulation the packet drop ratio was 2.22%, equivalent to a bit error rate of 5×10^{-6}.

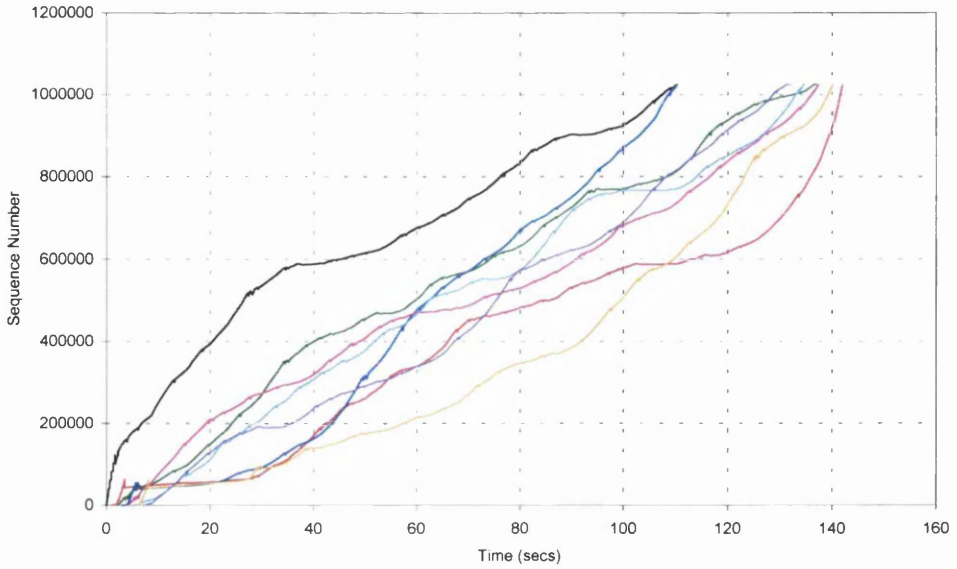


Figure 7-27 Eight TCP flows sharing single link

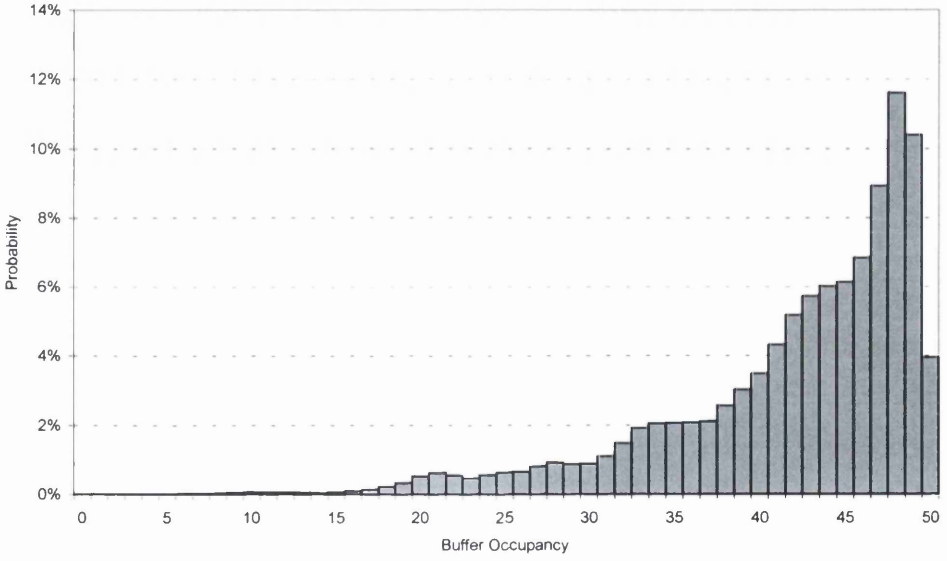


Figure 7-28 Occupancy of congested buffer

7.6 Conclusions

There is very little difference in the performance of the different TCP mechanisms. The selection of mechanisms to produce the best throughput is fast retransmission and recovery, time stamps and disable delayed acknowledgement (default is to use it in most systems). Selective acknowledgement does not produce enough improvement to warrant the process and memory requirements needed to implement it. The major controversial suggestion is the disabling of delayed acknowledgements. The reason for using delayed acknowledgements is two fold, firstly to reduce the problems of silly window syndrome (SWS) [RFC813] and secondly to reduce loading on the network and processing requirements of the end terminals [RFC896]. Over the years since delayed acknowledgements were introduced the processing power of computers has out paced that of the growth in link speed technologies. This is certainly the case for the wireless technologies that are considered in this study. In higher error environments the cost of extra loading by the acknowledgements is offset by the increased throughput. Therefore there is no real justification for using delayed acknowledgements in a radio link network.

A parameter that has a large impact on the throughput is the maximum transmitted unit (MTU), or in TCP terms maximum segment size (MSS). The smaller the MTU the smaller the packet. Small packets have a lower probability of being dropped than larger ones. Hence reducing the size of the packet increases the throughput on error prone links. The MTU on a system can be changed by adjusting the MTU defined for an interface. The cost of decreasing the MTU is to increase the overhead, especially if the packets have to be fragmented. It may be possible to change the MTU on an interface (link) dynamically. It is necessary to check that all the protocols on the network are able to handle dynamically changing MTUs.

TCP is unaffected by short term burst errors. If anything the burst error slightly increase the performance. From Figure 7-20 and the work in Chapter 6, the worst case for unprotected links is random errors. The effect of long term burst errors or fading is different, it is more like switching the link on and off rather than injecting errors. The effect of pulse is quite different. If the pulse can be tuned to the TCP connection, it is possible to greatly reduce the performance. For this to be effective consecutive packets need to be corrupted, and being able to also corrupt the retransmitted packets. These results only consider a single TCP connection. In a real network there will be multiple flows with different round trip times and packet spacings. This means packets from the same TCP connection may not be transmitted over a link consecutively. The dynamics in the network will mean that the timeout period will change over time. Tuning a pulse to corrupt the correct packets from a single flow could be impossible without detailed knowledge of the system being used.

There will be a major impact on throughput if the round trip time is greater than 9 seconds. This is easy to achieve in networks with low data rate links. One of the present systems only has 32 kbit/s allocated for packet data. Assuming a port buffer of 512 kbits (this is small for a router) the delay with 32 kbit/s link will be 16 s when the buffer is full. This is in excess of the 9 s already without considering the return path. This one fact will have a major impact on the performance of TCP. The problem of not being able to estimate a round trip time can be resolved by using timestamps, where an estimate can be made even if there is a retransmission. As seen in the results (Figure 7-14) the mix of timestamps and delayed acknowledges has an impact on the performance. As it seems advisable to use timestamps and it again reinforces the need to disable delayed acknowledgements.

Even with best selection of mechanisms, TCP still far under achieves what is theoretically possible (Figure 7-11). The TCP mechanism as it stands suffers badly in a high error prone environment. This is due to the way TCP responses to losses, it assumes all losses are due to congestion. An approach is needed that can distinguish between congestion and corruption. The congestion window should only be adjusted to congestion, ideally before it starts to causes losses. Keeping the window as large as possible will make it easier to handle losses due to corruption. There are two ways to approach this problem, adapt TCP to handle corruption or adapt the environment to suit TCP. Changing TCP may be a better solution but could make it incompatible with other TCP implementation. The major reason for considering TCP is for interoperability issues. Adapting the environment to reduce corruption is possible, but a considerable amount of the bandwidth could be taken up with the FEC redundancy. A half rate convolution code would reduce a 512 kbit/s link to 256 kbit/s.

The major problem with TCP is the dynamics of the congestion window. The slow growth of the window causes poor performance for long delay links. The decrease in window size and slow growth after a corruption loss causes poor performance. One way to improve performance for web access is to re-use the same connection for different parts of the web page. This means the window is not reset for each individual component. This is known as persistent connections [Nielsen97][Padman95][RFC2616]. The window growth process could be changed to give a faster increase. One approach that has been looked at is starting with an initial window greater than one segment [RFC2414][RFC2415]. The problem with increasing the window growth rate is the ability to control congestion. The faster the window changes the faster the congestion changes. If the congestion changes faster than the window can respond then the system becomes unstable [Jacobson88].

TCP is able to adjust to the available bandwidth, but there is still room for improvement in congestion management. A better solution for congestion management would be achieved without forcing and detecting lost packets. If this can be achieved then all loss must then be due to corruption, so the normal back off process of TCP is not needed. The lack of lost packets in error free environments would mean not having to send packets more than once and will help reduce the amount of redundant packets in the network.

7.6.1 Recommendations

From the results it is possible to make recommendations on what options to use when operating TCP over a radio linked network (Table 7-1). These options can be used in all conditions as it improved performance in certain conditions (high bit error rate), but does not significantly reduce performance in others. Some of the options are implemented in the sender, some in the receiver. Some of the options in Table 7-1 are dependent on both the sender and receiver, if either end does not support the algorithm the option is not used. This does not prevent the TCP connection from working. When inter-operating with other TCP/IP systems that are not tuned to the wireless environment there may be degradation in performance.

Algorithm	Use	Where
Fast Retransmission	Yes	Sender
Fast Recovery	Yes	Sender
Delay Acknowledgements	Disable	Receiver
Time Stamps	Yes	Sender/Receiver
Selective Acknowledgement	Yes	Sender/Receiver
Maximum Transmitted Unit	<=576 Bytes	Sender

Table 7-1 Recommended TCP Options

To keep TCP throughput acceptable, the residual bit error rate (after any error correction) should be better than 10^{-5} (Figure 7-14). This represents about a 5% packet drop ratio, and the round trip time of the connections should ideally be kept below 9 s.

8. IMPROVING TCP PERFORMANCE

8.1 Introduction

From Chapter 7, it is clear that TCP does not perform well in high error environments. This is due to the way TCP does its congestion control and how this responds to corrupt packets. The aim of this chapter is to look at alternative ways of doing congestion control without causing buffer overflows. The modification to TCP should not break compatibility with existing implementations. The other option to consider is reducing the corruption and leaving TCP untouched, allowing TCP to deal with the congestion it causes.

The starting point for adjusting the window management process is to look at the dynamics of a congested link (Figure 8-1). There are two points of interest, the knee and the cliff. The knee is where the loading offered to the link is approximately the same as the link, where the difference in the loading is taken up by the queue. Between the knee and the cliff, the queue is starting to fill. The cliff is the point where the offered load is too high and overfills the queue. This is the point where the queue starts dropping packets. TCP operates at the cliff. If TCP can be adjusted to operate at the knee this could greatly reduce congestion. TCP Vegas aims to move the operating point [Brakmo94].

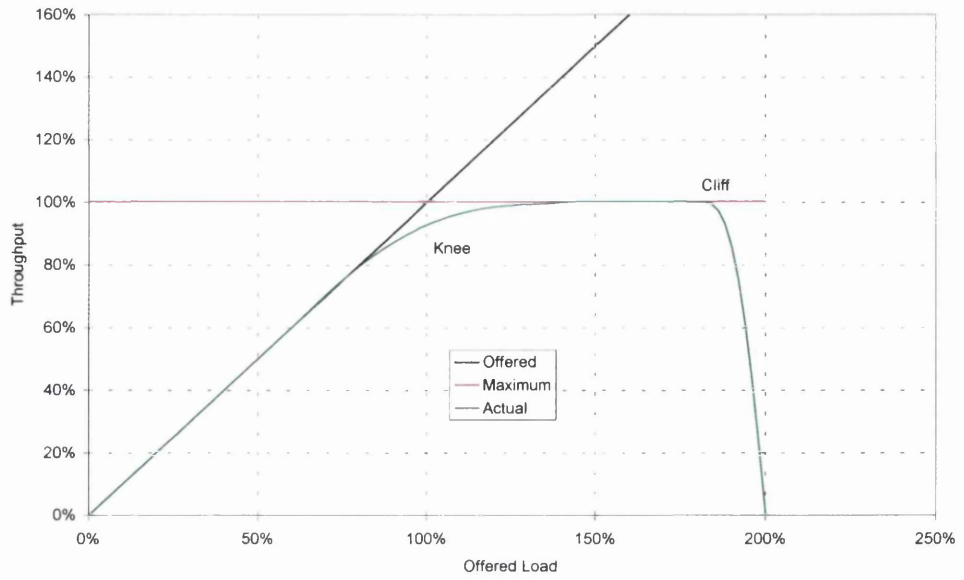


Figure 8-1 Congestion Dynamics

The other way to consider congestion is to look at the link as a whole. The optimum window size is set so that there are enough packets to just keep the connection full. The packet transmission rate required is that of the slowest link. The total amount of data the link can carry is the bandwidth delay product. The delay is simply the round trip time and the bandwidth is the minimum available along the path the connection follows. This will not always be the forward path in an asymmetrical connection as the return path of the acknowledgements could be the limited link. The available bandwidth is not the link bandwidth but the percentage available to the flow. The allocation of bandwidth depends on the queue process, the most common is first come first serve. This leads to a problem in determining the available bandwidth. One way to achieve this is with feedback from each queue indicating the bandwidth, but as most queue processes handle each packet individually it is impossible to determine the bandwidth for an individual flow. There needs to be a process independent of the queuing processes. One way to measure the available bandwidth is to use the data flow as stimulus and measure the response of the network. Using packet pairs is a way of measuring the bandwidth [Bolot93][Keshav][Lai99].

8.1.1 Vegas

L. Brakmo [Brakmo94] and J. Mo [Mo99] suggested a way of modifying TCP to operate at the knee of the congestion curve (Figure 8-1). The method operates by measuring the difference between the offered load and the measured throughput (Figure 8-2). The aim is to control the difference by adjusting the size of the congestion window. If the difference is too high (i.e. congesting the link) the congestion window is reduced. If the difference is too small, the congestion window is increased to use any spare bandwidth. To calculate the offered and actual loading, some standard TCP parameters plus one extra parameter are used (8.1). The values used are the congestion window (*cwnd*) and the individual round trip time (*irtt*). The extra value is the minimum round trip time (*mrtt*). This represents the condition of the network when there is no congestion. The *offered_load* is the amount of data being sent into the network, in the case of TCP a whole window of data is sent per round trip time. The actual load can be measured in a similar way. The difference is measured in bytes per second. This can be converted to bytes by multiplying by the minimum round trip time. The *wnd* measure of the difference is now a measure of the difference in the window size. The aim is to keep a small amount of data in the queue before the congested link. In the simulation, the difference in the window size is between 2 and 3 packets. If *wnd* is less than twice the maximum segment size (MSS) the congestion window is increased by one MSS. If *wnd* is greater than three MSSs the congestion window is decreased by one MSS.

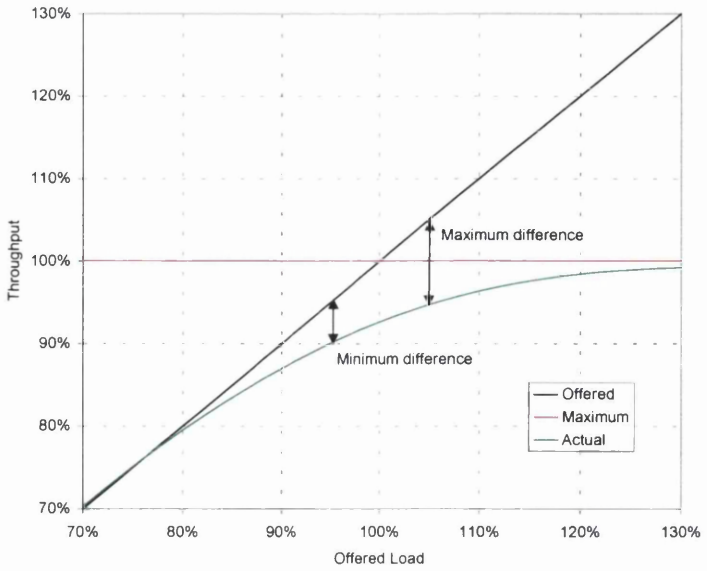


Figure 8-2 Detecting the knee

$$
\begin{aligned}
offered\_load &= \frac{cwnd}{mrtt} \\
actual\_load &= \frac{cwnd}{irtt} \\
diff &= \frac{cwnd}{mrtt} - \frac{cwnd}{irtt} \\
wnd &= diff \cdot mrtt = cwnd \cdot \left(1 - \frac{mrtt}{irtt}\right)
\end{aligned}
\qquad (8.1)
$$

The change to TCP is to the slow start and congestion avoidance that becomes one mechanism (Vegas). The normal TCP timeout and fast retransmission are still used. If a packet is lost and recovered the window will be reset according to the appropriate mechanism.

8.1.2 Modified Vegas

To avoid the retransmission, resetting the congestion window, TCP Vegas was modified. The modifications listed here are by the author, similar work as been done by others at the same time [Hengart00][Vendict02]. Timeouts and fast recovery process do not reset the size of the congestion window. The other changes are to the estimation of the minimum round trip time and the window control process. The minimum round trip time is allowed to increase (8.2) slowly to allow for changes in the path of the connection. If a round trip time measurement is lower than the present minimum round trip time the value is reset to this value. The congestion window control has been modified to give a smoother change (8.3). When there is no congestion, *wnd* is approximately zero and the congestion window is increased by one MSS per acknowledgement. As congestion builds *wnd* increases reduce the amount the congestion window is increased by. Under heavy congestion when *wnd* is greater than MSS the congestion window is decreased.

$$mrtt_{i+1} = mrtt_i + (irtt - mrtt_i) \cdot \alpha \quad (8.2)$$

$$cwnd_{i+1} = cwnd_i + mss - wnd \quad (8.3)$$

8.1.3 Packet pair measurement

Packet pair measurement is a mechanism for measuring the bandwidth of a link [Keshav91][Lai99]. A pair of packets are sent simultaneously. During the path through the network the packets can get separated. This could be due to the speed of links or congestion. The bandwidth can be calculated from packet spacing (t) and the size of the second packets (s) hence (s/t).

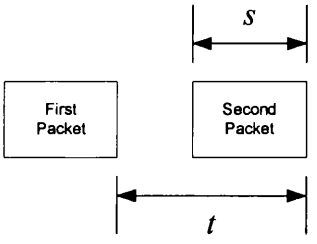


Figure 8-3 Packet pair measurement

8.1.4 Packet pair window control

The aim of this mechanism is to adjust the congestion window to match the link. The window is adjusted to the bandwidth delay product. This means making estimates of both the bandwidth and the delay. The delay can simply be measured from the round trip time. The bandwidth is measured using the packet pairs (Section 8.1.3). (8.4) shows the calculation needed to estimate the congestion window (*win*). The transmission delay (t_s) and the packet separation (t_d) cannot be measured independent only as a single value, packet spacing. The round trip time for the second packet (*rtt*) is a combination of the packet spacing and the propagation time (t_p). Figure 8-5 shows how the window size changes as the dynamic parameters change. As propagation delay (t_p) increases the window increases linearly providing more data to fill the connection. As congestion delay (t_d) increases the window decreases exponentially, reducing the amount of data offered to the network. This exponential decrease means the mechanism should be stable under congestion.

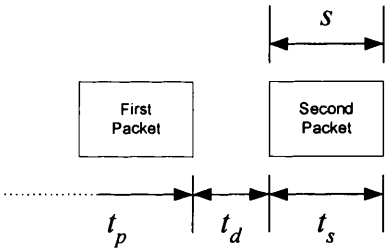


Figure 8-4 Packet pair values

$$rate = \frac{s}{t_d + t_s}$$

$$rtt = t_d + t_s + t_p \qquad (8.4)$$

$$win = rate \cdot rtt = \frac{s \cdot (t_d + t_s + t_p)}{t_d + t_s}$$

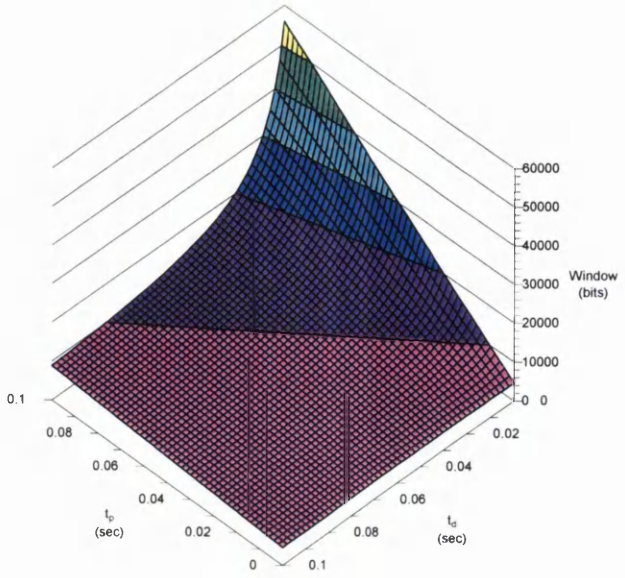


Figure 8-5 Theoretical window dynamics

It is possible to make an estimate of the bandwidth delay product for the first two packets. This gives a major advantage over the standard slow increment, as the window can be expanded quickly to this estimate. To avoid over expanding the window the value can be smoothed to slow the growth rate (8.5). The alpha (α) value can then be tuned to control the growth rate. Figure 8-6 shows the comparison of the new window control against TCP. The new window control grows faster than TCP and does not overshoot.

$$bytewin_{i+1} = bytewin_i + (win - bytewin_i) \cdot \alpha \quad (8.5)$$

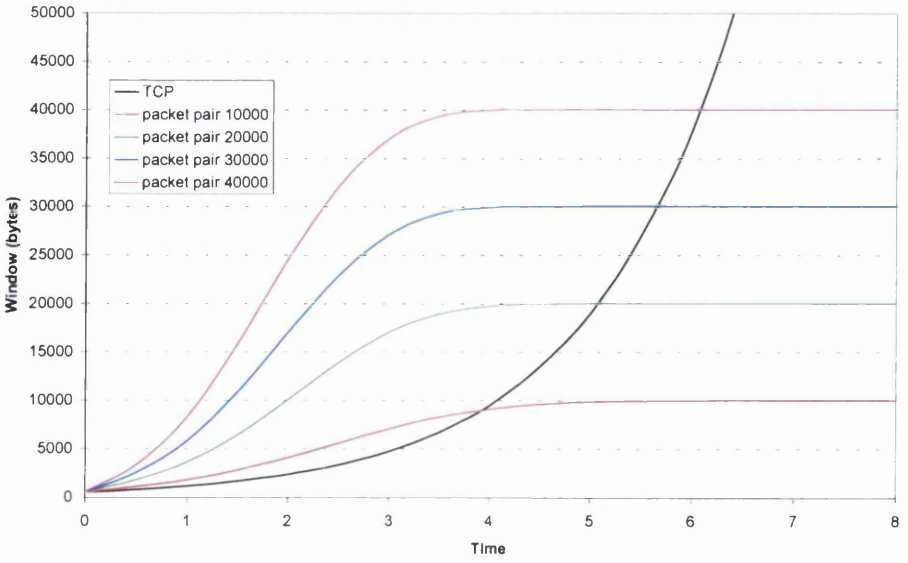


Figure 8-6 Comparison of window growth rates

The alpha value can also be tuned to give the same growth rate for connection with different round trip times. To calculate the alpha value the growth of the window is matched against the exponential growth equation (T is the time constant, n is number of packets, t is time and p is the packet rate). With some assumptions and simplifications, the alpha value comes out as a simple calculation based on some standard TCP parameters (8.6). Alpha (α) can be tuned to give a maximum window within T seconds, which is independent of the round trip time.

$$w_n = x \cdot \left[1 - (1-\alpha)^n\right] \quad v = 1 - e^{-t/T}$$
$$w_n = \tfrac{1}{2}x \qquad v = \tfrac{1}{2}$$
$$\tfrac{1}{2} = (1-\alpha)^n \qquad \tfrac{1}{2} = e^{-t/T}$$
x = target window size

$$n = t \cdot p$$
$$\ln\tfrac{1}{2} = t \cdot p \cdot \ln(1-\alpha) = -t/T$$
$$T = -1/\left(p \cdot \ln(1-\alpha)\right)$$
$$\ln(1-\alpha) \approx -\alpha \quad \alpha \to 0$$
$$T = 1/(p \cdot \alpha)$$
$$p = window / rtt$$
$$\alpha = \frac{rtt}{window \cdot T} \qquad (8.6)$$

8.1.5 Automatic Repeat Request

The other approach to solving the mismatch between corruption and congestion is to minimise or eliminate corruption. Two main approaches exist, forward error correction (FEC) and automatic repeat request (ARQ). FEC is a simpler process, but can be difficult to implement especially if large real time block codes are needed. In the case of TCP the optimum size of an FEC block would be in the order of several kilo bits. To implement these large codes in real time is very difficult. FEC does not guarantee correct data, it only reduces the probability of having uncorrectable data.

A mixture of a good cyclic redundancy check (CRC) and ARQ can give error free data. The throughput is still limited by the error rate on the link. If data can be transmitted across, it will eventually get through as the ARQ keeps retrying. ARQ requires bi-directional connection to allow feedback to the sender. FEC does not require feedback and hence can be used with a unidirectional link, unlike ARQ.

In the network architecture chosen for this study, all links are bi-directional. The effect of FEC would simply be to reduce the error rate and throughput due to the overhead (Chapter 6). ARQ has the effect of introducing extra jitter into the connection. There is the issue of how TCP will interact with this variable delay. It seems more appropriate to study the effects of ARQ rather than FEC.

8.2 Simulations

The simulations used here are similar to those in Chapter 7. For the tests of different TCP algorithms the same terminal node (Figure 7-5) is used but replacing the TCP process with one of the modified TCP processes. The only major change to the architecture of the network is with the introduction of ARQ.

8.2.1 OPNET models

Some of the simulations require changing the physical layer framing from HDLC to the ARQ process. This requires a bridge node Figure 8-7. The bridge is similar to the switch (Figure 7-6), but with no switching process and one of the HDLC processes is replaced with an ARQ process. This node sends packets from the HDLC side onto the ARQ side and visa versa. The ARQ scheme used is that described in Chapter 6.

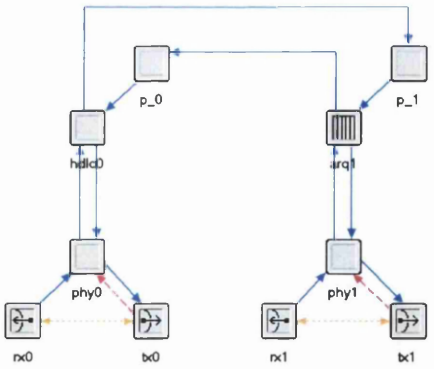


Figure 8-7 ARQ Bridge

8.2.2 Network Architecture

The networks used for tests with the ARQ links uses a bridge to link between the terminal and degraded network link. Multiple ARQ links can be connected together with modified switches (Figure 8-8). The switches have the HDLC process replaced by the ARQ process.

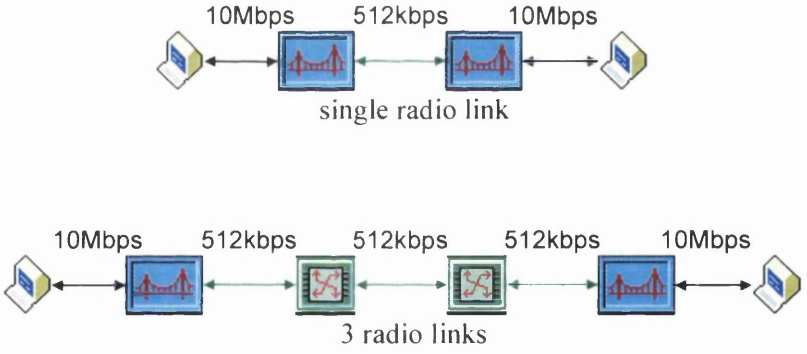


Figure 8-8 ARQ Test Networks

8.3 Results

8.3.1 Throughput

There is a marked difference between the different schemes in the presence of errors (Figure 8-9). Vegas suffers quite considerably, whereas Modified Vegas and Packet Pair Control performs much better in the presence of errors. Base-TCP still performs better at error rates before the point of collapse. The improved performance of Modified Vegas and Packet Pair Control is due to the mechanism not backing off in the presence of packet losses.

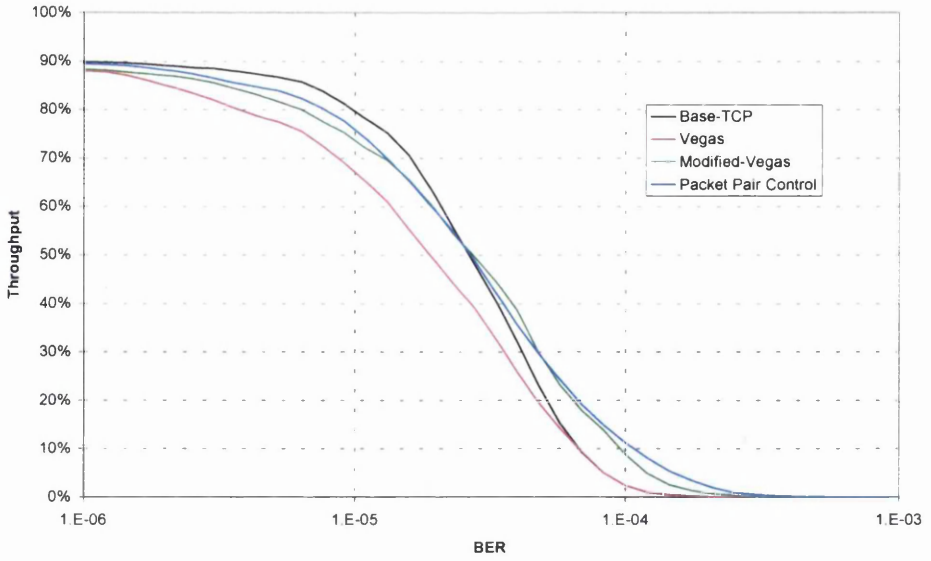


Figure 8-9 Comparsion of throughput

8.3.2 Window Dynamics

The study of TCP behaviour is achieved simply by expanding the window until a packet is dropped. The rate of increase in the window is dependent on the rate at which acknowledgements are received. Hence the longer the round trip time, the longer it takes for the window to expand. Figure 8-10 shows the increase in the window for propagation delays varying from 0 to 0.5 seconds. The Packet Pair Control adjusts the size of the window based on estimates of the capacity of the connection. This means the increase in the window can be controled so that the window expands at roughly the same rate with different round trip times (Figure 8-11). This removes bias towards connections with short delays on shared links [Floyd91][Floyd92].

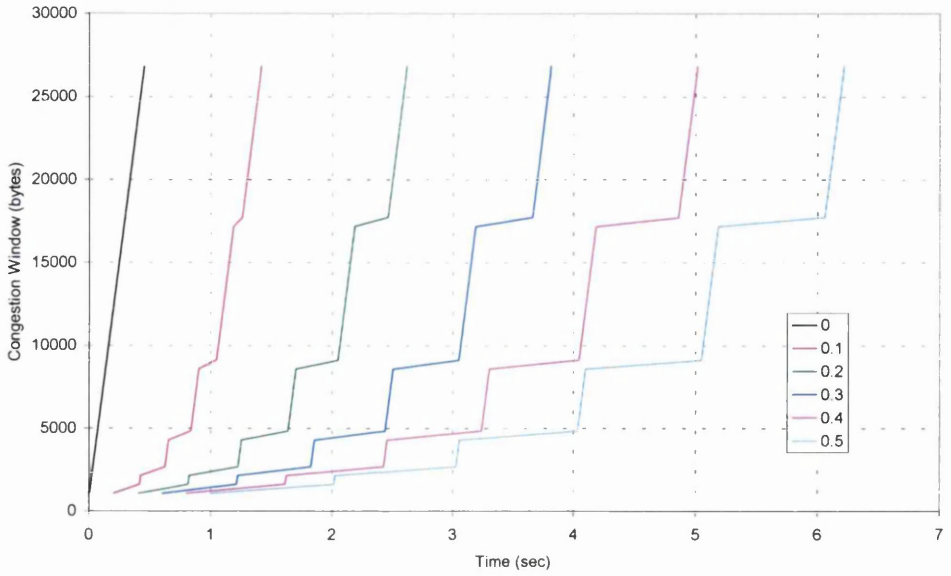


Figure 8-10 TCP Congestion window dynamics

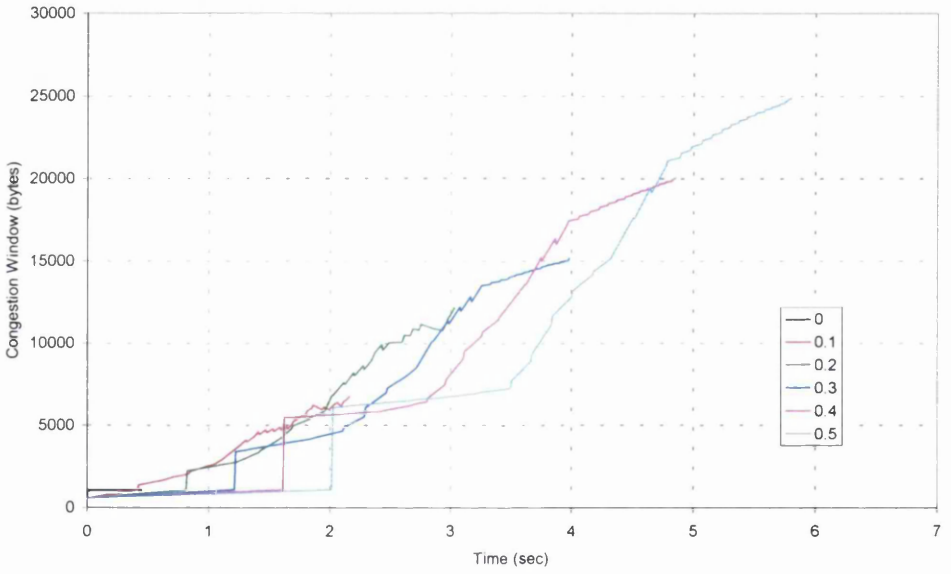


Figure 8-11 Packet Pair Control, congestion window dynamics

The changes to the window control mechanism of Packet Pair Control produces a much flatter graph for throughput with varying delay and file size compared to TCP (Figure 8-12). The consequence was not to improve the performance over long delay connections but to degrade the performance over short delay connections.

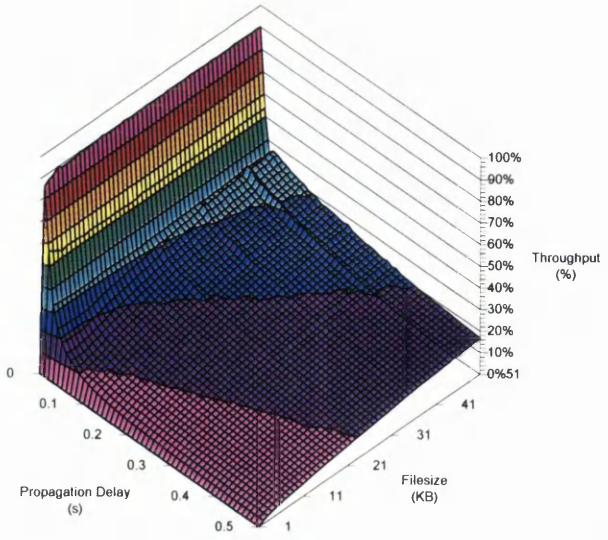


Figure 8-12 Effect of delay and file size on Packet Pair Control

8.3.3 Congestion

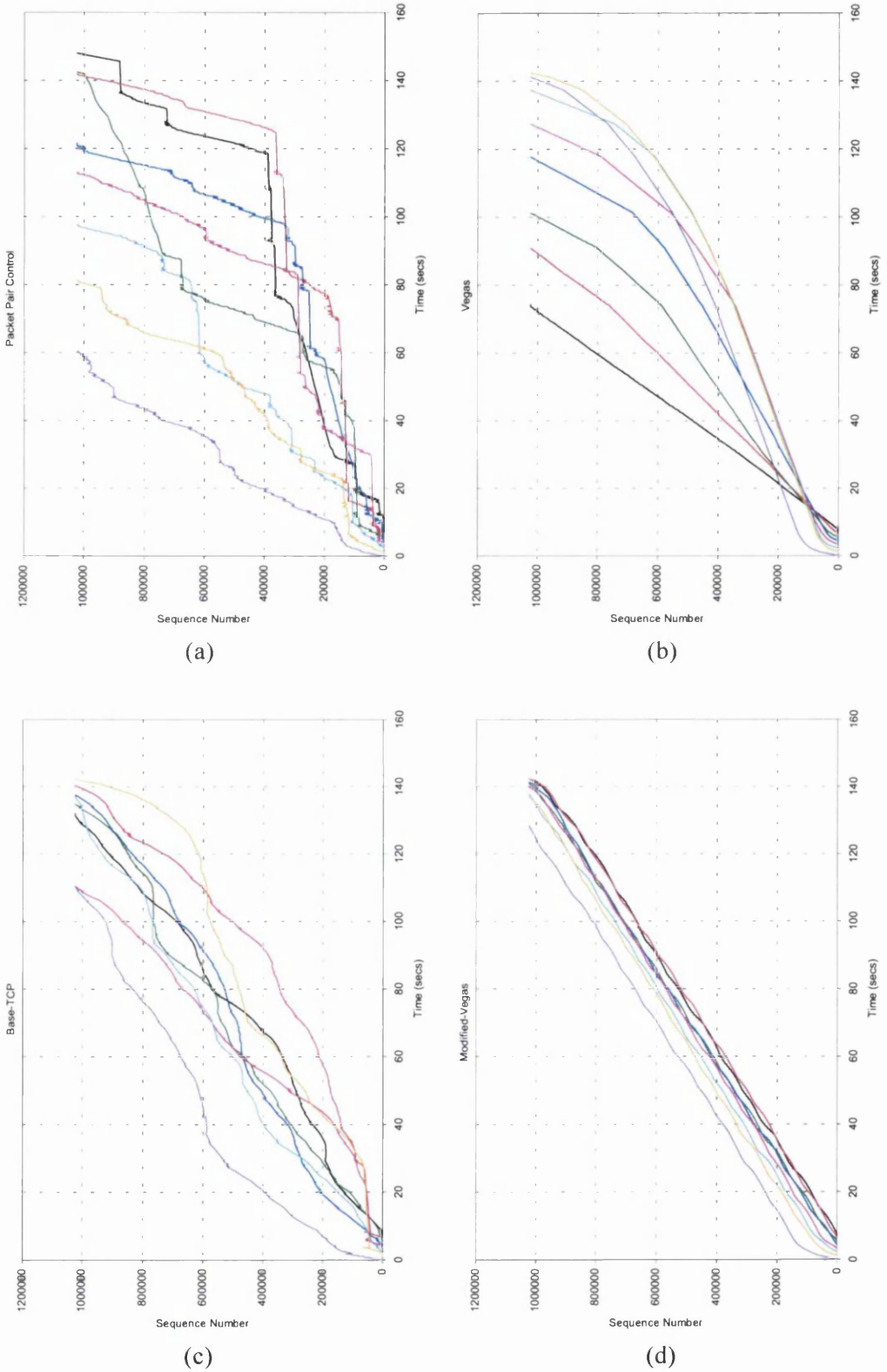


Figure 8-13 Comparison of schemes under congestion

TCP performance suffers with high levels of corruption due to the response of the congestion mechanism. The modified mechanisms try to break the link between lost packets and congestion control. For the mechanisms to be considered useful, they need to be stable under congestion. The ideal congestion mechanism is one where all flows proceed with an equal share of the bandwidth. One way to look at this is to look at the progress of the sequence numbers. If they are straight and parallel then there is equally shared bandwidth between the flows. Figure 8-13 shows sequence progression for the different mechanisms. There is room for improvement in the TCP algorithm (Figure 8-13(c)). The lines are all vaguely on the same gradient, but the actual rate changes considerably. The Packet Pair Control is very unstable (Figure 8-13(a)) and produces a large number of retransmission (Table 8-1). Vegas has straight lines but at different gradients (Figure 8-13(b)). This means the flows are stable but do not equally share the bandwidth. This is due to the use of a minimum round trip time. New flows will have a larger minimum round trip time hence have a larger share of the bandwidth. The Modified Vegas adjusts the minimum round trip time during the flow, increasing it slowly to match the network conditions. This improves the sharing (Figure 8-13(d)), but with an increase in the number of retransmissions (Table 8-1).

Mechanism	Packet drop ratio
TCP	2.22%
Vegas	0.01%
Modified-Vegas	0.09%
Packet Pair Control	9.53%

Table 8-1 Retransmission due to congestion

The other way to look at congestion is with the buffer occupancy (Figure 8-14). It is clear that TCP does operate at the cliff as the major buffer occupancy is at the capacity of the buffer (50 packets). Both Vegas and Modified Vegas keep the buffer occupancy below the limit but above zero. This means the link is fully utilised without dropping packets. The instability in the Packet Pair Control produces a wider spread of buffer occupancy as the rate of the flow changes, sometimes filling the buffer sometimes allowing the buffer to empty. Figure 8-15 shows the distribution of buffer occupancy for the different mechanisms and two different buffer sizes, the left graph with a maximum buffer of 50 packets, the right graph with 100 packets. TCP adjusts its rate to keep the buffer full, whereas Vegas and Modified Vegas adjust their rates just enough to fully utilise the link. The instability in Packet Pair Control can be seen with the spread in the buffer (long bar indicate the standard deviation) (Figure 8-15).

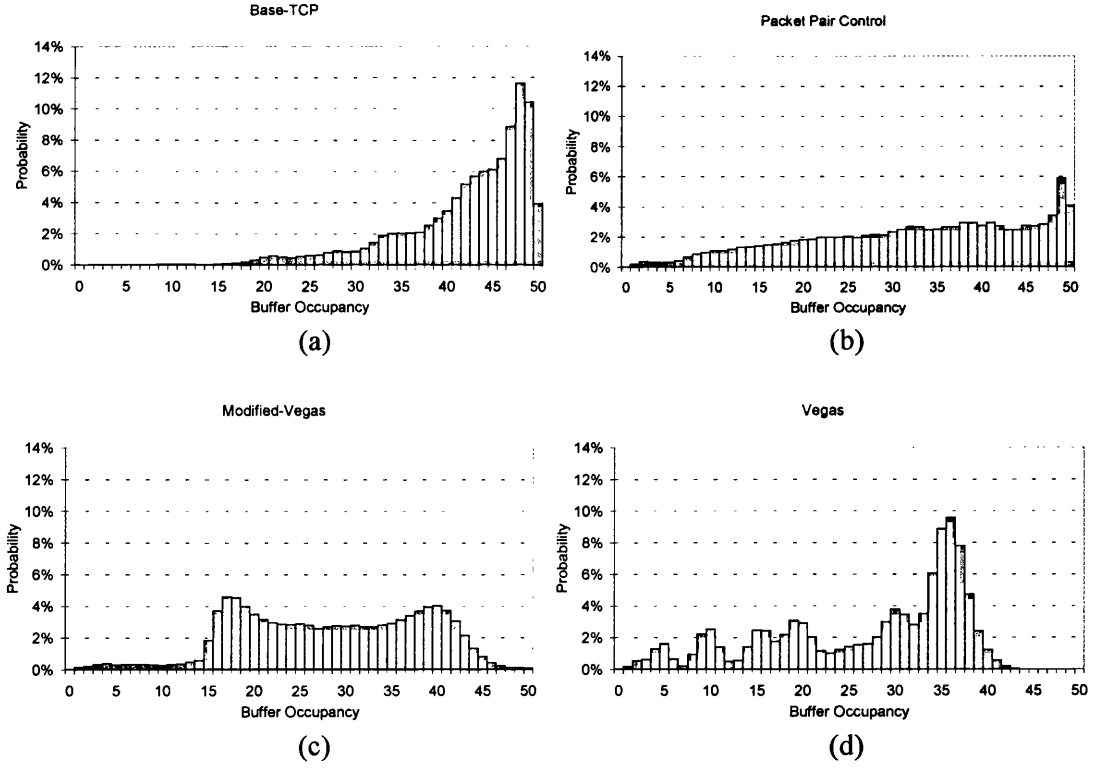


Figure 8-14 Occupancy of congested queue

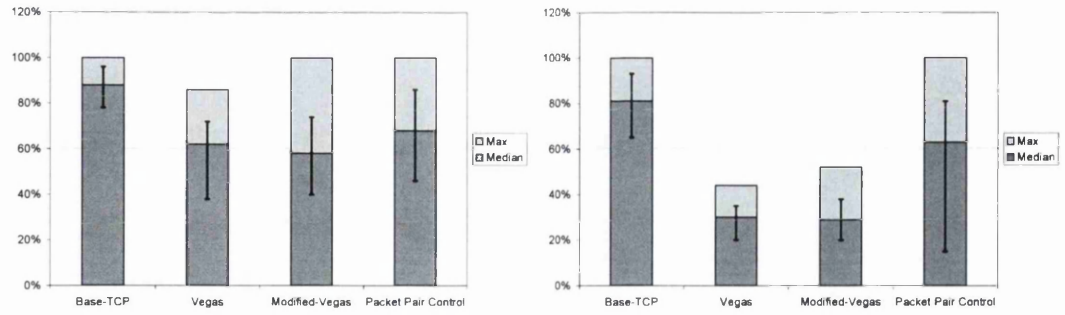


Figure 8-15 Comparsion of buffer sizes

8.3.4 Automatic Repeat request

The purpose of the ARQ is to eliminate the lost packets that cause TCP to do retransmissions. This prevents TCP backing off and hence increasing the throughput. Figure 8-16 compares TCP with and without ARQ against the theoretical throughput. TCP has already been shown to under achieve (Chapter 7). The ARQ greatly helps to improve performance with a cost at low error rates of extra overhead, but there is still a considerable difference between the theoretical throughput and what is achieved. The margin between theoretical and actual is reduced slightly with ARQ, this is more noticeable at high error rates.

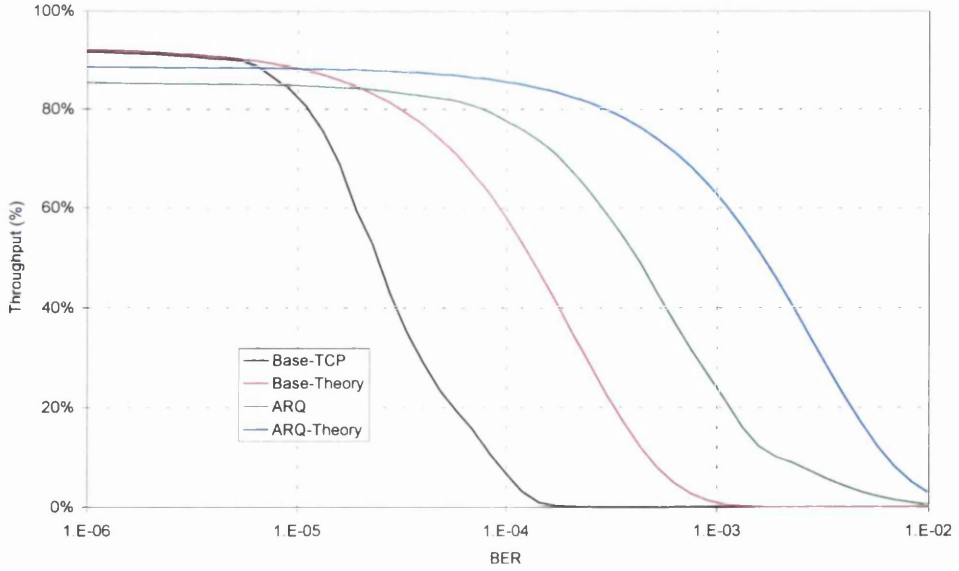


Figure 8-16 TCP throughput with and without ARQ

The effect of ARQ on the other mechanisms is far more noticeable (Figure 8-17). The jitter caused by the ARQ has a large effect on Vegas and Modified Vegas. These mechanisms use the timing information, the variation in delay introduced by the ARQ is interpreted by the mechanism as congestion and reduces the data rate, hence the poor performance. None of the mechanisms perform as well as TCP when ARQ is used.

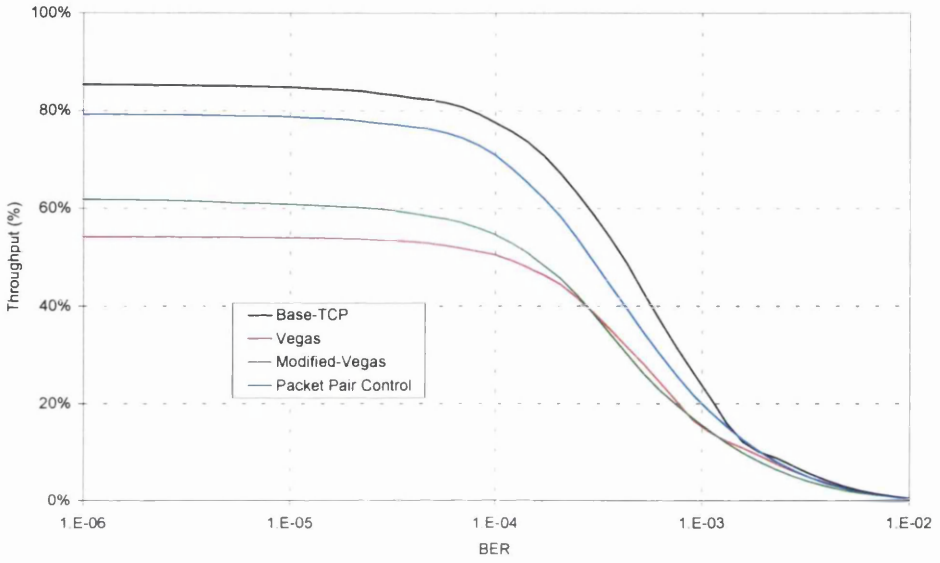


Figure 8-17 Comparsion of scheme with ARQ

8.4 Conclusions

8.4.1 Throughput

Modified Vegas and Packet Pair Control mechanisms perform better than TCP in the presence of high error rates. The unmodified Vegas struggles due to reduction in the congestion window, hence there is less data in transit to detect losses. As Vegas still uses the normal TCP back-off process it suffers quite badly. The removal of the back-off process from the Modified Vegas produces a marked improvement in performance.

There is little gain from Packet Pair Control for handling different length delays. Rather than improving the performance of long delay links it seems to hinder short delay links. The flatter response of the Packet Pair Control should mean that sharing between connections with different round trip times should be better, but there seems to be a stability problem when used with shared connections.

8.4.2 Congestion

Modified Vegas is far more stable in congestion than the other mechanisms. The Packet Pair Control mechanism is highly unstable under congestion. The Vegas mechanisms make better use of the buffers than TCP. The lower utilisation will help in reducing the overall connection delay.

The Packet Pair Control looked promising in theory. It provided a quicker window increase than TCP but without the problem of a large overshoot. It allowed for the distinction between corruption and congestion, hence performed well with link errors. If the issue of stability can be addressed it may produce an algorithm better than the standard TCP. The problem is being able to estimate the bandwidth accurately and quickly enough to respond to congestion. Two consecutive packets in a queue will appear to have the full bandwidth even if the queue is heavily congested. This problem can be alleviated by using round robin polling in the routers.

8.4.3 Automatic Repeat Request

Link by link ARQ increases the performance of TCP far more than any other mechanism. ARQ drastically affects the performance of the Vegas algorithms due to the sensitivity to timing information.

The main choice is between adjusting TCP to the environment or the environment to TCP. Adjusting TCP does not create leaps in improvement. The potential of the Modified Vegas mechanism could greatly improve the performance over networks with limited capacity such as military networks. The performance is better in high error environments than TCP, therefore the Modified Vegas mechanism is useful in military networks. Adjusting the environment with mechanisms such as ARQ and FEC could produce much greater improvements. This would also allow the use of standard TCP stacks delivered with most operating systems.

Packet Pair Control has a lot of potential for solving many of the problems with TCP. The sensitivity to the round trip time makes it unstable under congestion.

The mechanisms considered using a TCP packet are implemented simply by changing the way information is gathered and processed by the end terminal. Most of the modifications are only to the sender. This means that these mechanisms can inter-operate with existing implementation of TCP. It may be possible to improve on TCP with other algorithms but this could potentially break backward compatibility of TCP, rendering the algorithm useless for multinational military operations.

8.4.4 Recommendations

The most effective method for improving the performance of TCP over error prone links is with ARQ. For unprotected links it is possible to improve the performance and bandwidth sharing by using Modified Vegas. It is recommended that Modified Vegas and ARQ are not used on the same network, due to the interaction between them. The recommendation is to use either ARQ or Modified Vegas. A possible option that was not tested is to use Modified Vegas with FEC. The effect of the FEC is to reduce the error rate without the introduction of the variable delay associated with ARQ.

9. CONCLUSIONS

To get a good appreciation of how a network will perform it is necessary to look at all the layers of a communication system. This allows for consideration of artefacts in one layer affecting the performance at a different layer. An example is the burst error models. Gilbert and Elliot models produce random packet losses whereas the model derived from real link errors produces burst packet dropping.

9.1 Physical Layer

The physical layer is responsible for transferring bits from one end of a link to the other. The characteristics of the channel depend on the technology being used. This study concentrated on point-to-point radio links, as this is the main backbone of the tactical network. The tactical network provides connectivity between the soldiers on the front line and the various levels of command. The radio links provide an effective way of deploying a network quickly. It was necessary to study the characteristics of these links to determine what simplification can be made to the models of higher layers in the ISO stack.

Two radios were tested, one an in-service radio the other a future development radio. The results from the two radios were considerably different. It was not possible to determine if this was due to the testing process or the radios. The development radio had some effects, due to the bit synchronisation process, that needed to be corrected. The corrected results appeared to match the characteristics of the Gilbert and Elliot model, which has been used by many researchers. The military radio on the other hand had characteristics that did not match any of the analysed models.

The main approach to analysing the error pattern characteristics was by looking at the distribution of the gaps between errors. This was extended to see if there was any dependency between errors by looking at the distribution of condition probabilities. This technique can be extended further to look at dependency between several errors. The difficulty with this is gathering enough raw information. The error gap distributions were effective at distinguishing different types of error patterns. The conditional probabilities were useful in identify artefacts of the modulation process.

It was discovered that the military radio did not produce one of the standard error distributions. A new burst error model was developed to match these characteristics. The model is relatively simple requiring two parameters, the bit error rate and the burst factor. The burst factor is used to alter the shape of a weighting function. The weighting function alters the ratio of closely packed errors to sparsely spaced errors, producing burst errors.

From the research material, the Gilbert and Elliot model appears to be the typical model. The corrected results from the development radio seem to be consistent with this. Simulations of higher level protocols were run with both the Gilbert and Elliot model and the new burst error model, there was greater amount of interaction with the Gilbert and Elliot model. The new burst error model was used as a comparison to see if there were any major differences.

A recommendation for further work is that the military radio is tested under the same test bench conditions as the development radio. This will confirm if there is any significant difference in the error pattern between the two test processes. If there is, then this effectively proves that bench tests with attenuators and white noise sources cannot be used as acceptance testing of radios. An isolated test platform is still need during development to avoid interfering with other systems in case there is a fault with the radio.

9.2 Link Layer

The link layer provides the mechanisms for carring the packets of data over the link. This includes framing (packet delimitation), error detection and possible error correction and recovery. The framing used in the simulation was High-level Data Link Control (HDLC) and the Automatic Repeat Request (ARQ) frames. Most link layer error detection is Cyclic Redundancy Check (CRC) based. In the simulation it was assumed that any error would be detected by the CRC. The packet throughput was tested in the presence of burst errors, with and without Forward Error Correction (FEC).

Burst errors do have an effect on the packet throughput. In the case of unprotected packets (no FEC, only CRC) the throughput increases. This is because the burst errors bunch together in a small number of packets, leaving more packets free of errors. Hence, the worst case for unprotected packets is random errors, where the errors are more widely distributed. The opposite effect happens in the case of FEC protected packets. The groups of errors take the FEC over the limit causing the packet to still be corrupted. The packets that do get through do not require the level of redundancy (overhead). The extra overhead has the effect of reducing the throughput. These effects are only noticeable at very high error rates at the point where the throughput is about to collapse. Hence, the burst errors do not have a major effect on the point where FEC becomes more effective than unprotected packets.

The Gilbert and Elliot model produces a larger change in the performance of error protected packets and non-error protected packets compared with the new burst model. The major difference between the two is that the Gilbert and Elliot model produces random packet losses, whereas the new model produces burst packet losses. The Gilbert and Elliot model also produces a higher throughput than the new burst model, for the same bit error rate. If modern radios produce Gilbert and Elliot type error distribution it can be assumed that packet loss will be randomly distributed and have a lower corruption rate than calculated from the random bit error rate (assuming unprotected packets). The worst case for further testing can be assumed to be random errors.

The testing with ARQ shows that it is possible to improve the throughput with very little overhead. The main gains in the ARQ are achieved by breaking large packets into smaller chunks and hence reducing the probability of an individual chunk being in error. The other end of the link can then rebuild the packet to send it on. The side effect of ARQ is the variably delay (jitter) in rebuilding the packet when chunks are lost. This jitter can be decreased by sending smaller chunks over the link. The cost of doing this is to increase the size of the window required and to increase the overhead.

The effect of using FEC is to reduce the probability of packet loss, but also decrease the available bandwidth. For FEC to be efficient it needs to be tuned to the error rate of the link. The normal practice is to over engineer the link, accounting for the worst case. If block FEC is used, the optimum block size is that of a packet. On an IP network, packets can have highly variable sizes making this very difficult. To continuously optimise the link requires regularly changing the FEC to match the conditions of the link. Changing the FEC process while the link is running has to be done carefully to avoid making the link completely unusable. With a fixed FEC the link can still be characterised with bandwidth and residual error rate. This is not true for ARQ. ARQ has no residual error rate, all data received will be correct and in order. The effect of ARQ it to add jitter. This jitter may affect higher level protocols differently to that of corruption.

9.3 Network Layer

The network layer protocol provides a means of getting data from the source to the destination. The IP network considered in this study has no intelligence in the network layer. The connection between two terminals from the network layer can be considered as a single pipe. The best effort approach means the higher network layers will see all the effects caused by the lower layers.

9.4 Transport Layer

The transport layer is responsible for the end-to-end delivery of data. This can be unreliable as in the user datagram protocol (UDP) or reliable as in the Transport Control Protocol (TCP), which has been the focus of this study.

9.4.1 Standard TCP

The term standard TCP does not really apply. The description of the TCP algorithm is split over a considerable number of Internet Engineering Task Force (IETF) documents (Request For Comments (RFC)), which are a collection of suggests. At times the IETF will ratify an RFC and it will become a standard. Even in the description of the protocols that are standard there are parts that are defined as MAY or SHOULD. This means that these parts of the algorithm do not necessarily need to be implemented. The implementers of the algorithms do not have a strict set of guidelines to follow. Some choose only to implement the standards, others will also incorporate some of the RFC that are not standard. This leads to a great variety in implementation of the TCP (Table 9-1).

	TCP Tahoe (BSD 4.3)	TCP Reno (BSD 4.3)	Linux (2.4.18)	MS Windows NT (v4.0)	MS Windows 2000
Version/release date	June 1988	1990		V2.0	Beta 3
Congestion window					
Advertised window	4KB	16KB	32KB	4*MSS	4*MSS
Initial size	4 segment	1 segment	1 segment	1 segment	1 segment
Slow start	No	Yes	Yes	Yes	Yes
Congestion avoidance	No	Yes	Yes	Yes	Yes
Fast recovery	No	Yes	Yes	No	No
Window scaling	No	No	Yes	No	Yes
Retransmission					
Initial timeout (secs)	3	3	3	3	3
RTT estimation	smoothed	variance	variance	smoothed	smoothed
Timer accuracy	500ms	500ms	10ms	10ms	10ms
Backoff	No	Binary Exp	Binary Exp	Binary Exp	Binary Exp
Fast retransmission	No	Yes	Yes	No	Yes
Num Dup acks	3	3	3	2	2
Max retx per segment	4	4	15	5	5
Max retx per syn	4	4	10	3	2
Acknowledgements					
Delayed ack	No	Yes	Yes	Yes	Yes
Timeout	-	500ms	dynamic	200ms	200ms
Selective ack	No	No	Yes	No	Yes
Time stamps	No	No	Yes	No	Yes

Table 9-1 Differences in TCP implementations

Chapter 7 looked at a selection of commonly used algorithms, that are defined in the RFCs. The algorithms that produce a noticeable difference are fast retransmission and recovery, and delayed acknowledgements. The use of fast retransmission increases the throughput by avoiding the need to wait for a timeout. The delayed acknowledgements decrease throughput at high error rates (10^{-4}). This is due to the lack of information about the state of the connection being fed back to the sender. The inclusion of timestamps with delayed acknowledgements decreases the throughput further, by increasing the timeout period. It is recommended that delayed acknowledgements be disabled when using TCP over error prone links. Fast retransmission and recovery is standard in modern implementations of TCP. With the appropriate algorithms being used the residual bit error rate needs to be kept to better than 10^{-5}, or 5% packet loss ratio.

Selective acknowledgements (SACK) have virtually no effect on the throughput. This is because at least two packets need to be dropped from the same window. The window has to be large enough to allow at least four packets through without corruption. With high rates of packet corruption the large window required cannot be achieved, due to timeout retransmission causing the window to reset to one packet. The SACK option is useful for dealing with non-random corruption conditions, such as pulse errors. A burst of packets into a congested queue can also cause multiple packet losses [Floyd91][Floyd92]. It is recommend that the SACK option is used as there is no degradation in throughput when the option is triggered.

One modification that can be made to improve the performance is to reduce the maximum transmitted unit (MTU) over the radio links. The small packets are less prone to errors, hence have a higher throughput in the presence of errors. The cost of this is an increase in the overhead causing a drop in throughput for error free links. One way to use this information is to dynamically change the MTU for a link based on the bit error rate. It would require studying how all the TCP/IP protocols respond when using small packets.

An important issue arises with TCP when the round trip time is excessively high. If the RTT is greater than 9 seconds, TCP fails to achieve a stable state. This is due to unnecessary retransmission, making it difficult to get an estimate of the round trip time. Failing to get an estimate causes TCP to stay in a retransmit state. The 9 seconds comes from 3 times the initial timeout (3 seconds). One is for the first timeout and another two for the second timeout retransmission. There are several approaches to the problem, design the system so as the round trip times are less than 9 seconds, increase the initial timeout to handle the worst case, or use the timestamps option. The first two are possible but require detailed knowledge of the system. Using timestamps is the best options, this will not stop unnecessary timeouts at the start of the connection, but will settle into a stable state once the first timestamp is returned. This requires that delay acknowledgements are disabled due to the degradation in performance when these two are used together.

The cause of most TCP problems lies in the window control mechanism. When a connection is started the window opens too slowly to make full use of the network capacity for transferring small files. The window opens too quickly to avoid causing some congestion. This stems from having to work over a stateless network. An IP network does not control the amount of data entering the network. It requires the end terminal to manage their data rates to avoid congesting themselves and others. This leads to a vulnerability of IP where a terminal can congest the network by flooding it with packets.

9.4.2 Improved TCP

The problem with TCP in a high error environment is that it cannot distinguish between congestion and corruption. TCP uses the packets dropped by congestion to determine the network capacity. This means modifying TCP to use some other mechanism to determine congestion. Two approaches are taken, the first is based on TCP Vegas, the other on Packet Pair measurements.

When the offered load to a network increases the connection will eventual become congested. At the point where this congestion starts the RTT starts to increase. TCP Vegas uses this information to control the size of the window. TCP Vegas was modified to disable the normal TCP back off mechanism, such as adjusting the congestion window on a retransmission.

Packet pair measurement is a mechanism for measuring the bandwidth of a congested link. With the use of the RTT and congestion bandwidth, it is possible to calculate the bandwidth delay product for the connection. This product is the amount of data the connection can support in transit across the network, hence the size of the required window. Again the normal TCP back off mechanisms can be disabled.

Both the modified Vegas and Packet Pair Control still use the same TCP/IP packet format and will interoperate with other variants of TCP. Both perform better than TCP in the presence of high corruption rates. The modified TCP still needs to work under congestion as well as corruption. The Packet Pair Control becomes unstable in the presence of congestion. Modified Vegas performs better than normal Vegas and unmodified TCP, in the presence of both congestion and corruption.

The other way of dealing with TCP's response to corruption is to reduce the corruption. This can be done with forward error correction (FEC) and/or automatic repeat request (ARQ). It is possible to map the existing data to the expected performance of TCP when using FEC. This is done by using the residual error rate of the FEC in the known bit error rate and adjusting the throughput to account for the increase in overhead. The ARQ process produces an error free link, but introduces a variable delay (time taken to correctly transport the data across the link). The ARQ greatly increases the throughput achieved by TCP. With respect to TCP the ARQ process causes a variable congestion. TCP is able to handle this dynamic congestion. Vegas and the modified Vegas handle the variable congestion by reducing their data rate to avoid congestion. The throughput is considerably lower than that of TCP.

The recommendations are that error prone links should be protected with ARQ or FEC. The problem with ARQ is that it only works on bi-directional links. It cannot be used on broadcast technologies such as broadcast satellite or adhoc networks such as combat net radios. It is still possible to apply FEC to reduce the error rate. The MTU for error prone links should be kept small, but this has to be balanced against the reduction in throughput due to the increased overhead.

TCP is probably the best it can be. To get any vast improvement requires completely changing the protocol and approach to networking. IP networking is based around best effort delivery, TCP is tuned to make use of this. Changing to a back-pressure / compelled system such as X.25 [X25] or credit based flow control [Kung95], could produce a more efficient network over the low bandwidth high error environment of military networks. The problem is then providing interoperability with other networks and integration of applications.

If TCP/IP network is chosen as the way forward, the requirement is then placed on the link process to keep the residual error rate better than 10^{-5} or ideally 10^{-6} (Chapter 7). This can be achieved with FEC and or ARQ. The problem with FEC is the amount of overhead (redundancy) when the error rate is low (Chapter 6). The jitter caused by ARQ affects variances of TCP that are timing based (Chapter 8). These timing based variances do have the advantage of better congestion control (Chapter 8), which could help in the management of time critical data.

9.5 Further Work

9.5.1 Identified Issues

During the studies in this thesis a number of issues were identified that could be studied further. With the study on the radios (Section 4.5) there is a difference in the results that could be due to the testing process. If results from an isolated bench test are significantly different from real use then final performance test results should be based on the results from real on air tests. Further work is required to confirm whether the difference in the results is due to test process or the radios.

The effect of pulse errors on TCP is significant if the pulses are tuned to the TCP connection (Section 7.5.3). This is mainly due to the doubling effect of the timeout on a retransmission. If the increase in the timeout is made non-linear if may be possible to limit the effects of pules errors. Another option is to add a small amount of jitter to the timeout used, meaning the retransmission would occur at an unpredictable time.

In the study on packet pair congestion control (Section 8.3.3) there is major instability under congestion. Further work could be done to look at the problem in more detail. If packet pair could be made stable it has the potential to provide better performance than TCP in almost all conditions.

The Modified Vegas performs better than TCP under high corruption rates and congestion. The only problem is its performance in the presence of jitter introduced by ARQ (Section 8.3.4). If the algorithm could be adapted to handle the jitter it has the potential to be used over any network technology and deliver a better performance than any of the other algorithms.

9.5.2 Extending Study

The new burst error model uses a weighting function (Section 5.2) to adjust the distribution of errors. There is the potential to extend the basic model to produce a number of different distributions by using different weighting functions. It may be possible to produce Gilbert and Elliot error distribution by using non-linear functions such as a step.

The use of FEC is to protect data. To get optimum performance from FEC the code has to be tuned to the error rate of the link. A study of the reliability of an auto adjusting mechanism could lead to a way of optimising the throughput. Part of this study should be to look at how a continuously changing bandwidth would effect higher level protocols such as TCP.

9.5.3 Split Connections

An approach that has been studied to improve the performance of TCP over wireless links is split connections [Balak98][Keshav97]. This is where the TCP connection is broken into two parts and the data is temporally stored between the two connections. There are two ways of doing this, with an application layer proxy such as a web proxy, or a TCP proxy that acts a stream buffer. There is an independent TCP connection on either side of the proxy. This has been looked at as a means of improving the performance over the last hop of a connection to a mobile device, where the data is temporally stored in the base station. To implement this on a military network would require a proxy at either end of each backbone link. This would mean that the data for a single connection would be forwarded along a single path through the network, making dynamic route changes difficult. If a network node is lost, then all the data in that node would also be lost, as the source of the data would assume the data had been receive and there would be no way to recover the data. It would require very carefully consideration as to what is happening at different parts of the network to make a split connection process work reliably. The TCP connection would only run over a single link. This would mean that TCP would not be doing congestion management and another mechanism would be needed. If this could be made to work it may be possible to replace the intermediate connection with a protocol that is more suited to working in a high error environment.

REFERENCES

[ACTS] "Advanced Communications Technology Satellite (ACTS)", http://www.ee.byu.edu/ee/comm/amt/actsstuff.html

[Ahrens00] A. Ahrens, "A New Digital Radio-channel Model Suitable for the Evaluation and Simulation of Channel Effects", IEE Electronic and Communications, April 2000, http://www.informatik.uni-roctock.de/(en)/studium/gradkolleg/mitarb/ahrens.html

[Allman97] Mark Allman, "Fixing Two BSD TCP Bugs", http://gigahertz.lerc.nasa.gov/~mallman/papers/bug.ps

[Allman98] Mark Allman, "On the Generation and Use of TCP Acknowledgements", ACM Computer Communications, October 1998, http://www.aciri.org/floyd/tcp_small.html

[Ambrose99] Ambrose Au, David Ranch, "Linux IP Masquerade mini HOWTO", Feb 1999. http://en.tldp.org/HOWTO/IP-Masquerade-HOWTO/index.html

[Balak97] H. Balakrishnan, V Padmanabhan, S. Seshan, R. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", IEEE/ACM Transactions on Networking, December 1997, http://http.cs.berkeley.edu/~hari/papers/papers.html

[Balak98] H. Balakrishnan, R. Katz, "Explicit Loss Notification and Wireless Web Performance", Proc. IEEE Globecom 1998. http://nms.lcs.mit.edu/papers

[Brakmo94] Lawrence S. Brakmo, "TCP Vegas: New Techniques for Congestion Detection and Avoidance", Proc SIGCOMM symposium, August 1994, http://www.cs.arizona.edu/xkernel/www/people/brakmo.html

[Bolot93] J C. Bolot, "End-to-End Packet Delay and Loss Behaviour in the Internet", Proc. SIGCOMM, Sept 1993.

[Cerf00] Vinton Cref, "Interview with", http://www.wcom.com/about_the_company/cerf_up/internet_history/q_and_a.p html

[Cerf93] Vinton Cref, "The birth of the ARPANET", http://www.geocities.com/siliconvally/2260/cerf1.html

[CerfKahn74] V.G. Cerf, R.E. Kahn, "A protocol for packet network interconnection", IEEE Trans. Comm. Tech., vol. COM-22, V 5, 637-648, May 1974.

[Chen99] R. Chen, K.C. Chua, B.T. Tan, C.S. Ng, "Adaptive error coding using channel prediction", Wireless Networks Vel 5 pp 23-32, 1999, http://www.baltzer.nl/winet/contents/1999/1-5.html

[Comer94] Douglas E. Comer, John C. Lin, "Probing TCP Implementations", Proc. USENIX 1994, http://www.cs.purdue.edu/homes/lin/probe.tcp.html

[Danzig92] P. Danzig, S. Jamin, R. Caceres, D. Mitzel, D. Estrin, "An Empirical Workload Model for Driving Wide-Area TCP/IP Network Simulations", Wiley Journal of Internetworking Reaseach and Experience Vol 3 No 1, March 1992, http://www.kiskeya.net/ramon/work/pubs

[Elliot63] E.O. Elliot, "Estimates of error rate for codes on burst-noise channel", Bell Systems Technical Journal Vol 42 pp 1977-1997, Sept 1963.

[Fall96] K. Fall, S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", Computer Communication Review pp. 5-21, July 1996. http://www-nrg.ee.lbl.gov/floyd/papers.html

[Floyd91] S. Floyd, "Connections with Multiple Congested Gateways in Packet-Switched Networks", Computer Communication Review, Oct 1991. http://www.icir.org/floyd/papers.html

[Floyd92] S. Floyd, V. Jacobson, "On the Traffic Phase Effects in Packet Switched Gateways", Internetworking: Research and Experience, September 1992. http://www.icir.org/floyd/papers.html

[Floyd99] S. Floyd, K. Fall, "Promoting the use of end-to-end congestion control in the Internet", IEEE/ACM Transaction on Networking, May 1999, http://www.aciri.org/floyd/papers/collapse.may99.pdf

[Gilbert60] E.N. Gilbert, "Capacity of a burst-noise channel", Bell Systems Technical Journal Vol 39 pp 1253-1265, Sept 1960.

[Heissler99] J.R. Heissler, Y.A. Barsoum, R. Condello, "An Analysis of the Viterbi Decoder Error Statistics for ATM and TCP/IP over Satellite Communication", IEEE MILCOM 99. http://www.argreenhouse.com/society/TacCom/Papers99

[Hengart00] Hengartner, U., Bolliger, J., and Gross, T., TCP Vegas Revisited. Proc. of IEEE Infocom 2000, Tel Aviv, Israel, March 2000, http://www-2.cs.cmu.edu/~uhengart/infocom00.pdf

[ISO8473] "Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol for use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473)", ISO DP 10589, February 1990.

[Jacobson88] Van Jacobson, "Congestion Avoidance and Control", Proceedings ACM SIGCOMM, August 1988, http://www.w3.org/Protocols/HTTP-NG/ReadingList.html

[Jacobson90] Van Jacobson, "Modified TCP Congestion Avoidance Algorithm", e-mail end2end-interest@ISI.EDU, April 1990, ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt

[Kahn72] R. Kahn, "Communications Principles for Operating Systems", Internal BBN memorandum, Jan 1972

[Keshav91] S. Keshav, "A Control-Theoretic Approach to Flow Control", Proc. SIGCOMM, Sept 91.

[Keshav] S. Keshav, "Packet-Pair Flow Control", To appear in IEEE/ACM Transactions on Networking, http://www.cs.cornell.edu/skeshav/papers.html

[Keshav97] S. Keshav, "SMART Retransmission: Performance with Overload and Random Losses", Proc Infocom 1997, http://www.cs.cornell.edu/skeshav/papers.html

[Kumar96] A. Kumar, "Comparative Performance Analysis of Versions of TCP in a Local Network with Lossy Link", IEEE/ACM Transactions on Networking, December 1996, http://www.ensc.sfu.ca/people/grad/zjiang/our_group/index1.html

[Kung93] H.T. Kung, "The FCVC (Flow-Controlled Virtual Channels) Proposal for ATM Networks", International Conference on Network Protocols, Proc pp. 116-127, October 1993.
ftp://virtual.harvard.edu/pub/htk/atm-forom/fcvc.ps

[Kung95] H.T. Kung, R. Morris, "Credit-Based Flow Control for ATM Networks", IEEE Network Magazine, March 1995,
http://www.eecs.havard.edu/htk

[Kuo95] F. F. Kuo, "The ALOHA System", ACM SIGCOMM Computer Communication Review, Volume 25 Issue 1, January 1995
http://www.acm.org/sigcomm/ccr/archive/1995/jan95/ccr-9501-kuo.pdf

[Lai99] K. Lai, M. Baker, "Measuring Bandwidth", INFOCOM99, March 1999,
http://mosquitonet.stanford.edu/~laik

[Licklider60] J.C.R. Licklider, "Man-Computer Symbiosis", 1960
http://memex.org/licklider.html

[Leiner00] Barry M. Leiner, "A Brief History of the Internet",
http://www.isoc.org/internet/history/brief.html

[McKusick99] M. K. McKusick, "Open Sorces: Voice from the Open Source Revolution", published by O'Reilly, January 1999,
http://www.oreilly.com/catalog/opensource/book/kirkmck.html

[McQuillan80] John M. McQuillan, "The New Routing Algorithm for the ARPANET", IEEE Transactions on Communications Vol 28 No 5 page 711-719, May 1980.

[Mo99] J. Mo, R. La, V. Anantharam, J. Walrand, "Analysis and Comparsion of TCP Reno and Vegas", Proc. IEEE Infocom, March 1999,
http://divine.eecs.berkeley.edu/~ananth/

[Nakagami60] M. Nakagami, "The m-distribution, a general formula of intensity distribution of rapid fading," in Statistical Methods in Radio Wave Propagation, W. G. Hoffman, Ed. Oxford, England: Pergamon, 1960.

[Nielsen97] H.F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hummeaux, E. Lie, C. Lilley, "Network Performance Effects of HTTP/1.1, CSS1 and PNG", Proc. ACM SIGCOMM, Sept 97,
http://www.acm.org/sigcomm/sigcomm97/papers/p102.pdf

[OpenBSD] OpenBSD,
http://www.openbsd.org

[Ott96] T. Ott, J. Kemperman, M. Mathis, "The stationary distribution of ideal TCP congestion avoidance", Aug 1996,
http://networks.ecse.rpi.edu/natun/papers/tcp-equn.ps

[Padman95] V.N. Padmanabhan, J. Mogul, "Improving HTTP Latency", Computer Network and ISDN Systems, Dec 1995,
http://research.microsoft.com/~padmanab/

[Paxson97] Vern Paxson, "Automated Packet Trace Analysis of TCP implementations", Proc. SIGCOMM 97,
http://ftp.ee.lbl.gov/papers/vp-tcpanaly-sigcomm97.ps.Z

[Peterson72] William W. Peterson, E.J. Weldon, "Error-correcting codes", 2nd edition published by MIT Press, 1972.

[RFC60] R.B. Kalin, "Simplified NCP Protocol", Jul-15-1970.

[RFC761] J. Postel, "DoD standard Transmission Control Protocol", Jan-01-1980.

[RFC768] J. Postel, "User Datagram Protocol", Aug-28-1980.

[RFC790] J. Postel, "Assigned numbers", Sep-01-1981.

[RFC791] J. Postel, "Internet Protocol", Sep-01-1981.

[RFC792] J. Postel, "Internet Control Message Protocol", Sep-01-1981.

[RFC793] J. Postel, "Transmission Control Protocol", Sep-01-1981.

[RFC801] J. Postel, "NCP/TCP transition plan", Nov-01-1981.

[RFC813] D.D. Clark, "Window and Acknowledgement Strategy in TCP", Jul-01-1982.

[RFC821] J. Postel, "Simple Mail Transfer Protocol", August 1982.

[RFC879] J. Postel, "TCP maximum segment size and related topics", Nov-01-1983.

[RFC881] J. Postel, "Domain names plan and schedule", Nov-01-1983.

[RFC890] J. Postel, "Exterior Gateway Protocol implementation schedule", Feb-01-1984.

[RFC896] J. Nagle, "Congestion control in IP/TCP internetworks", Jan-06-1984.

[RFC1058] C.L. Hedrick, "Routing Information Protocol", Jun-01-1988.

[RFC1067] J.D. Case M. Fedor M.L. Schoffstall J. Davin, "Simple Network Management Protocol", Aug-01-1988.

[RFC1072] V. Jacobson, R.T. Braden, "TCP extensions for long-delay paths", Oct-01-1988.

[RFC1105] K. Lougheed Y. Rekhter, "Border Gateway Protocol (BGP)", Jun-01-1989.

[RFC1106] R. Fox, "TCP big window and NAK options", Jun-01-1989.

[RFC1117] S. Romano, M.K. Stahl, M. Recker, "Internet numbers", Aug-01-1989.

[RFC1122] R. Braden, "Requirements for Internet Hosts -- Communication Layers", Oct-01-1989

[RFC1131] J. Moy, "OSPF specification", Oct-01-1989.

[RFC1185] V. Jacobson, R.T. Braden, L. Zhang, "TCP Extension for High-Speed Paths", Oct-01-1990.

[RFC1195] R.W. Callon, "Use of OSI IS-IS for routing in TCP/IP and dual environments", Dec-01-1990.

[RFC1256] S. Deering, "ICMP Router Discovery Messages", Sep-01-1991.

[RFC1323] V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance", May 1992.

[RFC1371] P. Gross, "Choosing a Common IGP for the IP Internet", October 1992.

[RFC1519] V. Fuller, T. Li, J. Yu, K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", September 1993.

[RFC1597] Y. Rekhter B. Moskowitz D. Karrenberg G. de Groot, "Address Allocation for Private Internets", March 1994.

[RFC1661] W. Simpson, "The Point-to-Point Protocol (PPP)", July 1994.

[RFC1700] J. Reynolds, J. Postel, "Assigned Numbers", October 1994.

[RFC1883] S. Deering R. Hinden, "Internet Protocol Version 6 (IPv6) Specification", December 1995.

[RFC1918] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear, "Address Allocation for Private Internets", February 1996.

[RFC1939] J. Myers & M. Rose, "Post Office Protocol - Version 3", May 1996.

[RFC2001] W. Stevens, "TCP Slow Start Congestion Avoidance Fast Retransmit and Fast Recovery Algorithms", January 1997.

[RFC2018] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "RFC2018: TCP Selective Acknowledgement Options", October 1996.

[RFC2136] P. Vixie Ed. S. Thomson Y. Rekhter J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", April 1997.

[RFC2414] M. Allman S. Floyd C. Partridge, "Increasing TCP's Initial Window", September 1998.

[RFC2415] K. Poduri K. Nichols, "Simulation Studies of Increased Initial TCP Window Size", September 1998.

[RFC2416] T. Shepard C. Partridge, "When TCP Starts Up With Four Packets Into Only Three Buffers", September 1998.

[RFC2525] V. Paxson M Allman S. Dawson W. Fenner J. Griner I. Heavens K. Lahey J. Semke B. Volz, "Known TCP Implementation Problems", March 1999.

[RFC2581] M. Allman V. Paxson W. Stevens, "TCP Congestion Control", April 1999.

[RFC2582] S. Floyd, T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm", April 1999.

[RFC2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", June 1999.

[RFC2663] P. Srisuresh M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", August 1999.

[RFC2883] S. Floyd, J. Mahdavi, M. Mathis, M. Podolsky, "An Extension to the Selective Acknowledgement (SACK) Option for TCP", July 2000.

[RFC2923] K. Lahey, "TCP Problems with Path MTU Discovery", September 2000.

[Roberts67] L. Roberts, "Multiple Computer Networks and Internet Communication", ACM Gatlinburg Conf, October 1967

[Roberts70] L. Roberts, "Computer Network Development to Achieve Resource Sharing", pp 543-549 Proceedings SJCC 1970

[Roberts00] L. Roberts, "History of the Internet" http://www.caspiannetworks.com/internethistrian/

[Seeley88] Donn Seeley, "A Tour of the Worm", University of Utah, http://world.std.com/~franl/worm.html

[Shannon48] C.E. Shannon, "A Mathematical Theory of Communication", The Bell System Technical Journal, Vol 27, pp 623-656, October 1948, http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf

[Shay95] William A. Shay, "Understanding Data Communications and Networks", published by PWS Publishing Company, 1995, ISBN 0534202446.

[Speth98] M. Speth, S. Fechtel, G. Fock, H. Meyr, "Broadband Transmission Using OFDM: System Performance and Receiver Complexity", IZS98, http://www.iss.rwth-aachen.de/Personen/speth.html

[Stevens96] W. Richard Stevens, "TCP/IP Illustrated Volume 1, The Protocols", published by Addison Wesley, December 1996, ISBN 0201633469.

[Stremler90] Ferrel G. Stremler, "Introduction to communication systems", 3rd edition published by Addison Wesley, 1990, ISBN 0201184982.

[Taurus] http://www.taurus2.plus.com/ukip/

[Vendict02] Andrea De Vendictis, Michela Bonacci, Andrea Baiocchi, "TCP NewVegas: Providing Good TCP Performance in both Homogeneous and Heterogeneous Environments", in Proc. of 15th ITC Specialist Seminar, Wuerzburg (Germany), July 2002, http://net.infocom.uniroma1.it/papers/ITC-SS15-33-devendictis.pdf

[X25] ITU-T X.25, "Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit", 1996

[Zakon00a] Robert Hobbes Zakon, "Hobbes' Internet Timeline v5.1", http://www.isoc.org/guest/zakon/Internet/History/HIT.html

[Zakoo00b] R. Zakon, "A Brief History of the Internet and Related Networks", http://info.isoc.org/guest/zakon/Internet/History/Brief_History_of_the_Internet

[Ziemer95] R.E. Ziemer, "Principles of Communications: systems, modulation, and noise", 5th edition published by John Wiley & Sons, June 1995, ISBN 0471124966.

[Zorzi97] M. Zorzi, R. Rao, "ARQ Error Control for Delay-Constrained Communications on Short-Range Burst-Error Channels", VTC 97, May 1997, http://it.ucsd.edu/~rao/papers.html

[Zorzi99] M. Zorzi, R. Rao, "Perspective on the Impact of Error Statistics on Protocols for Wireless Networks", IEEE Personal Communication, October 1999, http://www-cwc.ucsd.edu/~zorzi/publ1.html

Appendix A: Collecting Error Statistics

A.1 Introduction

Software was developed to analyse the statistics of different types of error patterns. The software contained a number of error models and a process for reading in error files. The error files were generated from data captured from real equipment. The software was developed as a Windows application allowing the collected statistical data to be displayed as it is processed (Figure A-1).

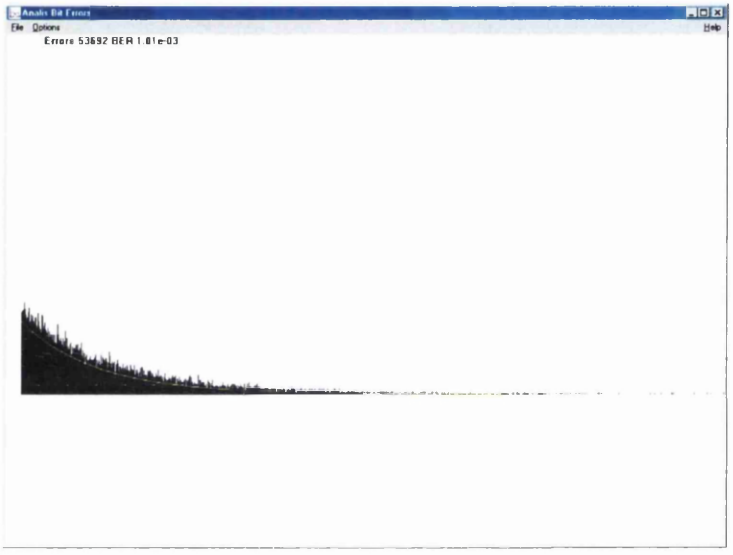


Figure A-1 Analysis Software

A.2 Structure of Software

The main process part of the software contains an error statistic collection function that calls a block code function (Figure A-2). The block code function then calls a selected error model that returns a single bit. The single bit can be in error. Each error model has state information that is used to hold information between calls. The block code function can then process the single error to produce block errors. This is to mimic the effects of a block FEC code. The block code can be configured with the size of the block and the number errors that can be corrected. The block can be set to one bit and zero errors that effectively passes the single errors to the statistics collection process.

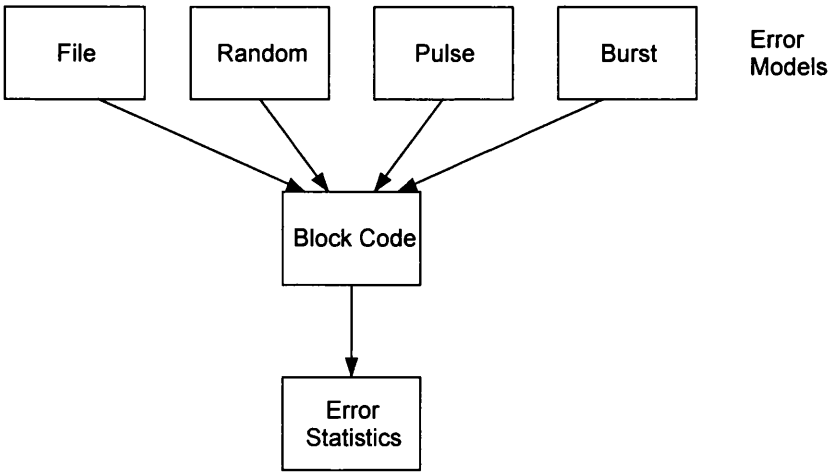


Figure A-2 Structure of code

A.3 Collecting Statistics

The statistical collection process consists of counting the number of bits between consecutive errors. This spacing between errors gives an indication of the distribution of the errors. The software collects the spacings into a collection of bins. The bins can be either linear or logarithmically scaled, selectable in the software. The program handles each bin as a node in an ordered binary tree. This means the software has some degree of self organisation, where a fixed size array may not be large enough for the data being processed or not enough granularity in the results.

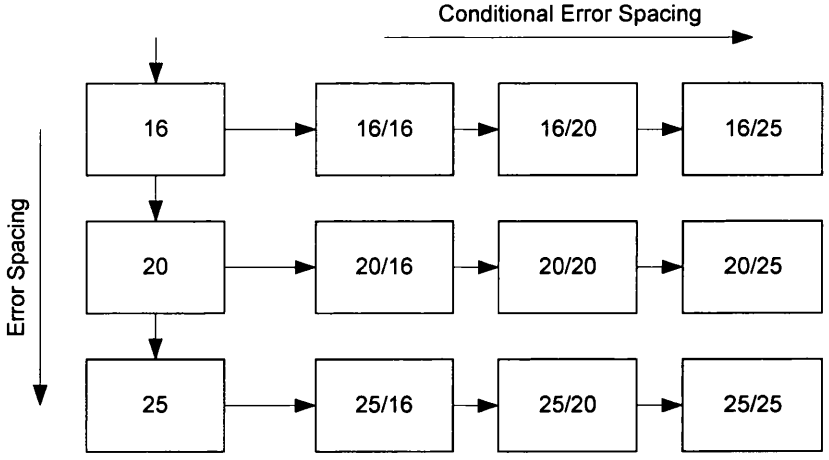


Figure A-3 Structure of stored data

To collect the condition probability data each node has a conditional branch. The condition data is added to the sub-branches of the main list (Figure A-3). The structure of the data can be visualised as a two dimensional link list. The main list being the distribution of error spacings the sub-lists as condition error spacings. The structure can be extended to hold any level of condition probabilities.

A.4 Processing of the Data

The collected data can be displayed as the errors are being collected. This includes the mean bit error rate and a histogram graph of the data. The data can also be saved to a text file to be read by any other software such as MS Excel for further processing. The only process done on the collected results is to normalise the data so the value for each bin representing the probability of that particular error spacing. This includes taking into account the width of the bin. For the logarithmic sized bins the size of each bin is different, hence each bin is divided by a different normalising value.

A.5 Conclusions

This software enables the processing of error statistics from different sources and allows the comparison for various models against errors collected from real equipment. The statistics collected can be saved and imported into other applications.

Appendix B: ROUTING MECHANISM

B.1 Introduction

The routing mechanism used in the Internet is based on an IP address against a routing table. This requires each node to know where to send each address packet. The routing table is either entered manually or via a route discovery process (RIP, OSPF). The aim in the simulation was to plug an arbitrary network architecture together with minimal manual network configuration. With normal IP routing this would require running a routing protocol which would over complicate the simulation. What is described here is a simple mechanism for routing data in the network that does not require the switching nodes to know the topology of the network. This avoids the need to run routing protocols or manually entering a routing table.

B.2 Basic Mechanism

The mechanism is based on following a described route (i.e. next junction take second left, following junction take first right, etc.). At each switching node the route taken by the packet is based on N^{th} port after the port the packet arrived on (Figure B-1)(B.1).

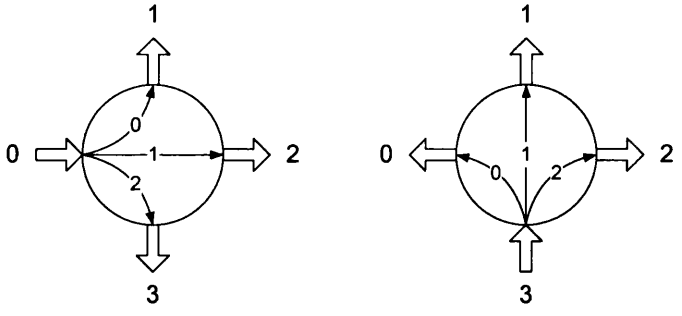


Figure B-1 Selecting outgoing port

$$out\_port = \mathrm{mod}(in\_port + 1 + route, num\_ports) \quad \text{(B.1)}$$

This means the route taken by a packet is determined by the direction it takes at each turn. The sequence of turns is based on a sequence of numbers predetermined by the source (source routing) (Figure B-2).

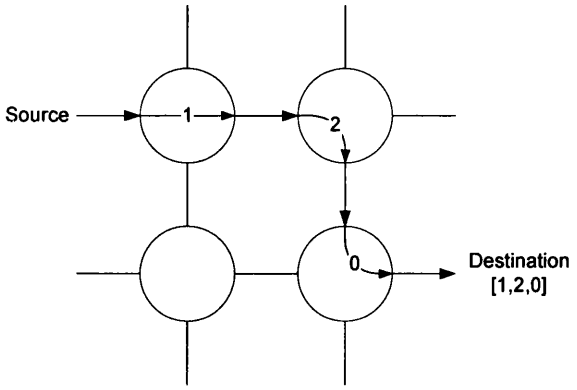


Figure B-2 Simple route

The consequence of this process is that a destination has different address depending on the relative position of the sources. The requirement that is missing is for the destination to be able to determine the route required to send a packet back to the source. It is possible for the switch nodes to calculate the return path and store it in the destination field. Each switching node on the path adds extra information to the source field, recording the return route. The return path ($route_r$) can be calculated from only the forward path and the number of ports (B.2).

$$\begin{aligned} out &= in + 1 + route \\ out_r &= in_r + 1 + route_r \\ in_r &= out = in + 1 + route \\ out_r &= in \\ in &= in + 1 + route + 1 + route_r \\ route_r &= \mathrm{mod}(-2 - route, num\_ports) \end{aligned} \qquad \text{(B.2)}$$

A simple approach to implementing this routing mechanism is for each switching node to remove the forward path data from the destination field and calculate the out going port and return path. The return path is added to the source field. If a binary number is used to hold each part of the route information binary manipulation can be used. The binary manipulation for adding or removing route information can be done with left or right bit shifts (right for removing from destination, left for adding to source). The calculation for the return path becomes a simple binary inversion of the forward path. This fixes the number of ports on each switch node to 2^n+1 (3,5,9,17...). The network does not have to be made up of one type of switch it is possible to use a combination.

B.3 Conclusions

This routing mechanism is simple to implement and does not require any routing table in the switching nodes. It only requires the source to have the destination pre-configured. The destination is able to determine the route back to the source from the source field in the packet, which is calculated as the packet traverses the network.

This mechanism could not be used on a real network as the source would need to determine the route to the destination. This could only be implemented as flood search and the path to the destination recorded. It also assumes point to point routing so is unlikely to work on ad-hoc networks.