Performance Analysis of TCP in a Multiple Reliable Connections environment with UDP Traffic Flows using OPNET Simulator

By

Yakubu Friday

(Msc./Scien/02200/2008-09)

A THESIS SUBMITTED TO POSTGRADUATE SCHOOL,
AHMADU BELLO UNIVERSITY, ZARIA
NIGERIA

IN PARTIAL FULFILMENT FOR THE AWARD
OF MASTER OF SCIENCE IN
COMPUTER SCIENCE

DEPARTMENT OF MATHEMATICS
AHMADU BELLO UNIVERSITY, ZARIA
NIGERIA

MARCH, 2011

DECLARATION

I declare that the work in the thesis entitled "Performance Analysis of TCP in a multiple reliable connections environment with UDP Traffic Flows using OPNET Simulator" has been performed by me in the Department of Mathematics under the supervision of Dr. S.E. Abdullahi

The information derived from the literature has been duly acknowledged in the text and a list of references provided. No part of this thesis was previously presented for another degree or diploma to the best of my knowledge at any University.

Sign:_____ Date:_____

Yakubu Friday

CERTIFICATION

This project titled "Performance Analysis of TCP in a multiple reliable connections environment with UDP Traffic Flows using OPNET Simulator" has been read and approved as meeting the requirements of Postgraduate School, Ahmadu Bello University, Zaria for the award of the degree of Master of Science in Computer Science.

Sign:_____ _____
Dr. S.E. Abdullahi Date
(Major Supervisor)

Sign:_____ _____
Dr. D.N. Choji Date
(Minor Supervisor)

Sign:_____ _____
Date
External Supervisor

Sign:_____ _____
Dr. A.A.Tijani Date
Head of Department

Sign:_____ _____
Prof. A. A. Joshua Date
Dean of Postgraduate School

DEDICATION

To God Almighty.

ACKNOWLEDGEMENT

I give God all the glory, praise and honor for seeing me through my educational pursuit.

I wish to express my sincere appreciation and special thanks to my supervisor Dr. S.E. Abdullahi for his guidance, patience and commitment in the actualization of this thesis.

I am grateful to Dr. M.I. Buhari, and staff members of ICT, IACC for their support in my study. I appreciate the love, support and prayers of all members of my family, course mates and all beloved friends.

Finally, I express my gratitude to my love for understanding. God bless you all.

ABSTRACT

This study is about the performance of TCP in multiple reliable connections environment with UDP traffic flows using OPNET simulator. The performance analysis is subjected to the bottleneck of competing traffics of UDP and TCP, accessed via a limited router buffer size connected to limited wireless link. Using simulation methodology, the analyses of the different TCP enhancements by comparing them with each other on the different test cases made. The research work establishes facts on the effect of data rate, router buffer size and TCP technology on the behavior of the TCP in the emulation environment in the presence of UDP traffic flows. Based on the analyses of the simulation results, different buffer capacity, data rate and TCP technology have effect on the TCP behavior in different ways. TCP connections suffer significantly because of the adaptability behavior and the number of unnecessary retransmissions is usually higher with large router buffer. Using the selected sizes, smallest router buffer increases the probability of having a retransmitted packet dropped and use of medium router buffer size does not cause as low TCP performance as use of the smallest router buffer, because the larger router buffers avoid several packets drop problem which was present when using the smallest router buffer. Similarly, with selected data rate, moderate data rate offers better performance. For this simulation test-bed, moderate data rate, medium router buffer size with NewReno offers a good performance of TCP followed by Reno Technology.

TABLE OF CONTENTS

Declaration……..……ii
Certificate……..……iii
Dedication……..……iv
Acknowledgement……v
Abstract……vi
Table of Contents……vii
List of Tables……..……viii
Lists of Figures……..ix
Glossary……..……x

Chapter one: General Introduction

1.0 Introduction……..……1
1.1.1 Transmission Control Protocol (TCP)……3
1.1.2 TCP Transmission in a multiple reliable connections environment……5
1.1.3 User Datagram Protocol (UDP)……7
1.1.4 Performance Analysis Using Simulation……8
1.3 Motivation……..……9
1.4 Purpose of the Study……10
1.5 Research Questions……10
1.6 Research Goal……11
1.7 Scope of Study……11
1.8 Research hypothesis……12
1.9 Thesis Frame Work……12

Chapter two: Review of Related Literature

2.0 The TCP and its Attributes……13
2.1 TCP Format……14
2.2.0 Synopsis of TCP……15
2.2.1 TCP Connection……16
2.2.2 Packet Loss Detection and Retransmission……17
2.2.3 TCP Time-Out Mechanism……18
2.2.4 Duplicate ACK of Successive Segments……19
2.3.0 TCP Congestion Control……20
2.3.1 Congestion Window……21
2.3.2 Slow Start threshold……22
2.4.0 Congestion Control Algorithms……22
2.4.1 Slow Start……22
2.4.2 Congestion Avoidance……23
2.4.3 Fast Retransmit……24
2.4.4 Fast Recovery……25
2.5.0 TCP Implementation……25
2.5.1 TCP Tahoe……26
2.5.2 TCP Reno……27
2.5.3 TCP NewReno……27
2.5.4 TCP Vegas……28
2.5.5 TCP SACK……29

2.5.6 TCP FAST……30
2.6.0 Multiple TCP Connections……30
2.7.0 Enhance TCP Models……31
2.7.1 Split Connection Model……32
2.7.2 Link Layer Model……33
2.7.3 Explicit Notification……33
2.7.4 End-to-End Models……35
2.8 Wireless Link Issues……35
2.9 Issues of TCP and UDP Flows over Wireless Link……36

Chapter three: Performance Methodology

3.0 Methodology……38
3.1 Simulation Tool……38
3.1.1 Optimum Network Engineering Tools (OPNET)……39
3.2 Area of Interest in the Study……40
3.3 The Targeted Environment……41
3.3.1 Modeling Traffics……42
3.3.2 Configuring Web (HTTP) Applications……43
3.3.3 Configuring e-mail Applications……44
3.3.4 Configuring FTP Applications……45
3.3.5 Configuring Video Stream Application……46
3.4 Metrics of the Performance Analysis……47
3.5 Selected Cases……48
3.6 Simulation Process Configuration……49
3.7 Justification of the Model……50

Chapter four: Analysis of Simulation

4.0 Analysis of the TCP Performance……52
4.1 Multiple Reliable Connections Environment (MRCE)……52
4.2 Multiple Reliable Connections Environment with UDP Traffic (MRCE)……53
4.2.1 Effect of Router Buffer size on TCP behavior using Baseline Configuration (ERTB)……55
4.2.2 Effect of data rate on TCP behavior using Baseline Configuration (EDTB)……56
4.2.3 Effect of data rate on TCP behavior using different Router Buffer size (EDTDR)……58
4.2.4 Effect of TCP Technology on TCP behavior using Baseline Configuration (ETTB)……59
4.2.5 Effect of TCP Technology on TCP behavior using different Router Buffer size (ETTDR)……60
4.2.6 Effect of TCP Technology on TCP behavior using different data rate (ETTDR)……62
4.3 Summary of the various Scenarios……63

Chapter Five: Summary, Conclusion and Recommendation

5.0 Summary……64
5.1 Recommendation……64
5.2 Conclusion……65

LIST OF TABLES

Table 3.0: Application Definitions……43
Table 3.1: Heavy Web Browsing Configuration……44
Table 3.2: Light Web Browsing Configuration ……44
Table 3.3: High Load e-mail Configuration……45
Table 3.4: Low Load e-mail Configuration……45
Table 3.5: Heavy File Transfer Configuration……45
Table 3.6: Light File Transfer Configuration……46
Table 3.7: VIOP Stream Configuration……46
Table 3.8: Video Stream Configuration……47
Table 3.9: Selected Wireless Link Data rates and Buffer sizes ……50
Table 3.10: TCP Attribute Configuration……50
Table 4.1: Simulation Configuration for EDTDR ……58
Table 4.2: Simulation Configuration for ETTDB……60
Table 4.3: Simulation Configuration for ETTDR……64

LIST OF FIGURES

Figure 2.0: TCP Header 14
Figure 2.1: TCP Open Connection, Data Transfer and Close Connection 17
Figure 2.2: TCP Congestion Window Behavior. 24
Figure 2.3: Split Connection Model 32
Figure 2.4: Link Layer Model. 33
Figure 2.5: TCP Header with Additional two bits 34
Figure 3.0: Emulation 41
Figure 4.1: MRCE Dropped Packets 52
Figure 4.2: MRCE Retransmission Count 52
Figure 4.3: MRCEU Dropped Packets 54
Figure 4.4: MRCEU Retransmission Count 54
Figure 4.5: ERTB Dropped Packets 55
Figure 4.6: ERTB Retransmission Count 55
Figure 4.7: EDTB Dropped Packets 57
Figure 4.8: EDTB Retransmission Count 57
Figure 4.9: EDTDR Dropped Packets 58
Figure 4.10: EDTDR Retransmission Count 58
Figure 4.11: ETTB Dropped Packets 59
Figure 4.12: ETTB Retransmission Count 59
Figure 4.13: ETTDR Dropped Packets 61
Figure 4.14: ETTDR Retransmission Count 61
Figure 4.15: ETTDD Dropped Packets 62
Figure 4.16: ETTDD Retransmission Count 62

GLOSSARY OF TERMS

ACK	Acknowledgment
ARQ	Automatic Repeat Request
AIMD	Additive-Increase, Multiplicative-Decrease
BER	bits Error Rate
BS	Based Station
CBR	Constant Bit Rate
CWnd	Congestion Window
CWR	Congestion Window Reduction
ECN	Explicit Congestion Notification
ECT	Explicit Capable Transport Bit
FACK	Forward Acknowledgement
FCE	Forward Error Correction
FIN	Finish
FTP	File Transfer Protocol
HTTP	Hypertext Transport Protocol
IP	Internet Protocol
LAN	Local Area Network
MRC	Multiple Reliable Connections
MSS	Maximum Segment Size
MTU	Maximum Transfer Unit
OPNET	Optimum Network Engineering Tools
PSH	Push
RED	Random Early Detection
RST	Reset

RTT	Round Trip Time
RTO	Retransmission Time Out
SACK	Selective Acknowledgment
SSthresh	Slow Start Threshhold
SYN	Synchronization
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
URG	Urgent
VOIP	Voice over Internet Protocol
WAN	Wide Area Network
Wmax	Maximum Window
WWW	World Wide We

CHAPTER ONE

Introduction

1.0 Background of the Study

The interest for internetworking has steadily experienced an increase over the last few years to global internetworking, what is commonly called the Internet. The large number of computers getting connected everyday to the Internet is becoming more and more seamlessly high with the decrease in the price of computers and network devices.

Originally, the internetworking was developed for low speed networks such as the ARPANET. The growth in network technology has made the internetworking to be flexible enough to incorporate the changing network environments of the past few decades. The Internet technology today runs over a large different link technologies with vastly different characteristics in terms of bandwidth and bit error rate.

With the explosive growth of the Internet, as users need for accessing the Internet increases, in a limited Bandwidth environment, the network traffic congestion grows and available limited Bandwidth shrinks. The performance of the network eventually decline because of traffic congestion among other challenging issues, bringing the need for viable solutions.

Current congestion control mechanisms such as the one used by TCP or equation-based congestion control aim to either reduce or increase the number of packets sent per time interval so as to adjust to the current level of congestion in the network (Allman *et al*,1999; Floyd *et al*, 2000). Many

remote TCP connections experience high loss rates due to gateway congestion. Traffic congestion results from either the packet rate or bandwidth. It follows that is necessary to control the rate at which packets are sent or adjust the packet size. In packet rate limited environment, the concept of reducing the packet rate in situation of congestion is a measure. If the traffic congestion results from low bandwidth capacity, resources are not shared fairly among flows. Instead, they are control by the packet size.

TCP congestion avoidance and control: slow start, congestion avoidance, conservation of packets, and exponential timer back off, was the first concepts initiated by Jacobson (1988). Later in 1990, TCP was integrated with fast retransmission and fast recovery algorithms to avoid inefficiency caused by retransmission timeouts (Jacobson, 1990; Stevens, 2001). The design of these TCP algorithms was based on unproved idea that at the gateway there should be at most one packet per flow when the sender increases the congestion window by one packet per round-trip time. Therefore, TCP's fast retransmission and fast recovery algorithms can quickly recover the loss and resume to fairness scheme of the congestion window.

The deployment of Random Early Detection (RED) principle that causes a proportional number of dropped packets connection to its bandwidth usage at the shared link, was as a result of study that has shown that there was a double packet loss rates on the Internet and the burst dropping was common (Paxson, 1997). Floyd *et al* (2000) also mentioned that RED causes multiple drops for large window. These studies

were carried out at different level of flows, obviously reflected different outcomes. It is evident that the suggestions and schemes may and may not be generalise to all shared link flows environment. Therefore, it becomes necessarily to study the performance of TCP at the individual flows share link. This thesis seeks to carry out a performance analysis of TCP in low bandwidth environment with high TCP's application connections.

1.1 Transmission Control Protocol (TCP)

According to Gilbert (2002), TCP is a reliable connection-oriented protocol that includes a built-in capability to regulate the flow of information, a function referred to as flow control. The flow of information is managed by TCP, by increasing or decreasing the number of segments that can be outstanding at any point in time. For example, under periods of congestion when a station is running out of available buffer space, the receiver may indicate it can only accept one segment at a time and delay its acknowledgment to ensure it can service the next segment without losing data. Conversely, if a receiver has free and available buffer space, it may allow multiple segments to be transmitted to it and quickly acknowledge the segments (Gilbert, 2002). Each segment of memory waits to be encapsulated with an IP header to form a datagram for transmission.

The datagram set for transit are structured in three slides and a "window" is placed over the slides by Gilbert (2002):

1. data transmitted and acknowledged
2. data transmitted but not yet acknowledged, and

3. data waiting to be transmitted.

Because this "window" slides over the three types of data, the window is referred to as a sliding window. Widmer *et al* (2002) pointed out that TCP congestion control mechanism using a smaller packet size than TCP sliding window size, would only achieve a fraction of the throughput justifiable according to its resource usage, if no corrective measure were taken. Another scheme is to make active queue management such as RED drop packets in direct proportion to their size (Floyd and Jacobson, 1993).

On the Internet, the TCP application occupies a higher and higher proportion in the total Internet traffic. However, the current TCP protocol does not work very efficiently; it introduces extra overhead on the network, wasting the network's capacity (Dong and Kung, 1998). More importantly, takes a long time to fetch all the information that the user requires. This is especially obvious in network environment where the bandwidth capacity cannot cater for the total application traffic requirements.

To improve TCP performance, a lot of proposals has been suggested including Persistent Hypertext Transmission Protocol (HTTP), Transaction TCP and Sharing TCP control blocks (Joe *et al.*, 1996). The goal of these proposals is to adapt the resource usage of a flow to the network's available resources, and to do so such that these are shared fairly between competing flows. Current TCP congestion control mechanisms such as adjust the packet rate in order to adapt to network conditions and obtain a throughput not exceeding that of a TCP connection operating under the same conditions, in an environment where the bottleneck resource is packet

processing, the scheme can properly address the behavioural situation (Widmer *et al.*, 2002). However, if the bottleneck resource is low bandwidth capacity, and flows may use packets of different sizes, resource sharing then depends on packet size and is no longer fair.

1.2 TCP Transmission in a Multiple Reliable Connections (MRC) Environment

It is the explosive growth multiple connections in networks that causes severe traffic congestion problems and is common to see Internet gateways drop a good percentage of the incoming packets because of local buffer overflows. In a multiple reliable connections environment, the traffic congestion bottleneck resource is bandwidth.

Bandwidth in the context of Internet access point is refers to the amount of information that can be passed through a communications channel in a given amount of time; that is, the capacity of the channel (INASP: 2009). It is often expressed in bits per second, or alternatively bytes per second, leading to lots of confusion. Bandwidth figures are used to express the capacity of a network link or interface (e.g., a 100Mbps Ethernet interface) or the rate of information transfer.

However, if the bottleneck resource is bandwidth, and flows may use packets of different sizes, resource sharing then depends on packet size and relies on transport protocol implementations. The amount of Internet Bandwidth that is available to each application traffic requirement on the shared link is therefore a factor that determines the degree of packets drop

and services delivery of the entire network. When there are multiple TCP connections on the shared link, the queuing delay at the link affect the behaviour of the TCP more than it does for a single TCP connection. It is observed that in a network environment with multiple parallel connections, the queuing delay increases and cannot be predictable because the TCP connections tend to interfere with each other (Kojo *et al.*, 1997). When a TCP connection gets its packets in the router, the other TCP connections will have to wait, starving from an unexpectedly long queuing delay.

Another study by Floyd and Jacobson (1993) revealed that when the router buffer is full, packets are dropped, much more from several parallel connections. The number of dropped packets due to buffer overflow is expected to be higher when several parallel TCP connections are in transit, because the last-hop router is likely to be more congested. This causes all the TCP connections that suffered from packet loss, decreasing their congestion window at the same time, which causes a sudden decrease in the amount of traffic outstanding in the network until the congestion window for different TCP connections is increased again.

Whenever one of the parallel TCP connections ceases transmission, the other TCP connections uses the available bandwidth on the link and fill the router buffer with their packets. When the ceased TCP connection starts transmitting, it is possible that the router buffer will be full of packets queue, and the router drops the packet transmitted. Even when the retransmission timer expires, the router buffer may still be occupied by the packets of the other the TCP connections and the retransmission triggered

by the retransmission timeout will be dropped again at the router. Therefore the TCP sender will have to wait for another retransmission timeout in order to retransmit the segment again. According to Braden *et al* (1998), this kind of situation could continue repeating several times, which causes substantial unfairness among the parallel TCP connections.

In the other way round, the nature of congestion control of TCP connections cause unfairness, and whenever one of the TCP connections drops a packet, the dropped TCP congestion window reduces its transmission and causes the other TCP connections which did not have its packet dropped get a larger share of the available bandwidth (Braden *et al.*, 1998). If the packet loss cannot be recovered using fast retransmit, the sender has to wait for the retransmission timeout. This allows the link capacity to be used by the competing TCP connections and might lead to substantial unfairness.

1.3 User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) is a connectionless, best-effort, non-error checking transport protocol (Gilbert, 2002). There is no handshaking between sending and receiving transport layer entities before sending a segment. UDP was developed in recognition of the fact that some applications may require small pieces of information to be transferred. The use of a connection-oriented protocol would result in a significant overhead error checking to the transfer of data. Therefore, UDP transmits a piece of information referred to as a UDP datagram without first establishing a

connection to the receiver. The protocol is also referred to as a best-effort protocol (Gilbert, 2002). To ensure that a series of UDP datagram are not transmitted into a black hole if a receiver is not available, the higher layer in the protocol suite using UDP as a transport protocol will wait for an acknowledgment. If one is not received within a predefined period of time, the application can decide whether to retransmit or cancel the session.

1.4 Performance Analysis using Simulation

Performance evaluation is a critical component of systems research that allows the evaluation of new ideas, identification of problems and bottlenecks and optimization of existing systems. There are three general approaches to performance evaluation:

1. Prototyping: build it (or a scaled down version of it) and see how it works.
2. Analytical modelling: build a mathematical model of it and use it to analyze the system.
3. Simulation: build a software model of the system.

Prototyping is often not feasible or time consuming especially for large scale systems. It also provides limited control and observation. Similarly, analytical modelling cannot capture highly complex systems. Thus, simulation has emerged as an attractive alternative that is heavily used in performance evaluation of computer systems.

In addition, there are many obstacles in the process of performing some studies. The most well know among others is the building real test-bed for

performance analysis. The cost is sometimes not effective or even not feasible. The intensity of the problem increases, as the need for different configuration parameters arises. For these reasons, it becomes necessary to build a simulation model and study it as a surrogate for an actual system. In this thesis, effort is made to build a model for simulating the traffic flows and TCP using one among widely acceptable network simulator, Optimum Network Engineering Tools (OPNET).

1.5 Motivation

TCP/IP is dynamic window based protocol that is used over the Internet. Its widely use is mainly based on its high dynamic nature of adaptability to any kind of network capacity, from a few bits to several gigabits per second. Since the development of TCP as an Internet standard, special attention has been drawn to its behaviour over different network environment. In particular, its TCP connection on a single router connected to an infinite capacity channel with a fixed independent error rate. Many TCP performance studies have been conducted on single TCP connection and multiple TCP connections including those by Bacceli and Hong (2000), and Ott *et al* (1999) through a sequence of finite capacity routers and bandwidths. In most of these studies, the presence of UDP traffics where not considered. Even if attention is given, often these traffic flows are assumed to have negligible effect. However, there has been a little motivation to study the TCP behaviour in a multiple TCP connections environments with UDP traffic flows. The presence of UDP traffic flows in a shared links of multiple reliable connections is worth studying.

1.6 Research Questions

a. What are the problems that are related to implementing real-time applications in a network environment?
b. What is the effect on the TCP performance when buffer size and data rate are considered?
c. Why should OPNET simulator be used as real test-bed for the TCP performance analyses?
d. How are network elements modeled in OPNET simulator?
e. How are generated traffic flows and graphs in OPNET simulator analyzed?

1.7 Research Objectives

There are many challenging issues about TCP performance in a network having many properties to study such as wireless link data rate and router buffer size. The research work focuses on some of these issues with special interest on TCP performance in the presence of UDP traffic flows and requirements for reducing retransmissions that eventually leads to Retransmission Time Out (RTO) and packet loss. The research objectives of this study are as follows.

a. to build a simulation model of the targeted network environment and study it as a surrogate for an actual system.
b. to investigates the effect of data rate and buffer size on TCP performance in the targeted environment.

c. to analyse the TCP performance through simulation.

d. to explore TCP behaviour and propose solutions to the unfairness.

1.8 Organisation of the thesis

The whole work is divided into five chapters. Chapter one gives general introduction of the thesis; chapter two discusses the theoretical background of the internetworking, Internet traffic transmission protocols, TCP technologies and TCP performance. Chapter three explained the methodology adopted for the research work. Chapter four reports the analysis of the study. Finally, chapter five pointed out conclusion, recommendation, and suggestion for further research.

CHAPTER TWO

Literature Review

2.0 The TCP and its Attributes

Transmission Control Protocol (TCP) is the dominant transport protocol in the Internet and supports many of the most popular Internet applications, such as the World Wide Web (WWW), file transfer and e-mail. The TCP has mechanisms that dynamically learn the network bandwidth and delay characteristics, and adapts to changes in traffic behaviour so as to avoid network collapse. Traditionally, TCP is designed to perform well in wired networks with the assumptions that packet dropped are mainly due to network congestion. However, the Internet has experienced explosive growth with the integration of wireless hosts. Wireless hosts are slower in transmission than fixed hosts, and are more vulnerable to natural environment. Therefore, TCP performance in networks with wireless hosts become poor because its design neglected random bit error, treating any packet drop in the network to be a result of network congestion which eventually slows down its transmission rate, or even cause the TCP sender to experience unnecessary timeouts, further reducing its performance. Hence, with high TCP's connection in a low resource link, the queuing delay is expected to affect the behaviour of the TCP more than it does for a single connection. The analysis of TCP performance make it necessary to find ways to improve TCP's efficiency and resource utilization, as well as improve the user's experience and reduce latency times.

To find effective solutions to the effect of multiple TCP's connection, the performance analysis of wireless TCP in a network environment forms a significant part of this research. This chapter gives overview of TCP in further detail and several TCP performance analyses.

2.1 TCP Format

The TCP segment constitutes the TCP header and payload. The information data passed from the above layers for transit is what is referred to as payload. In the TCP header, address information for the segment and all other information used in the implementation of TCP algorithms are included. The TCP header includes an option field that contains specific information for a particular TCP connection. The TCP header size ranges from the default length 20 bytes, to 60 bytes including the option field as shown in Figure 2.0.

<table>
<tr><td colspan="8">Source port (16 bits)</td><td>Destination port (16 bits)</td></tr>
<tr><td colspan="9">Sequence number (32 bits)</td></tr>
<tr><td colspan="9">Acknowledge number (32 bits)</td></tr>
<tr><td>Hlen
(4 bits)</td><td>Reserved
(6 bits)</td><td>U
R
G</td><td>A
C
K</td><td>P
H
S</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td>Receiver Window size
(16 bits)</td></tr>
<tr><td colspan="8">Checksum (16 bits)</td><td>Urgent pointer (16 bits)</td></tr>
<tr><td colspan="9">Options (32 bits)</td></tr>
</table>

Figure 2.0: TCP header

The Source and Destination port fields are each 16 bits in length. Each field denotes a particular process or application. The Header Length (Hlen) field is 4 bits in length. This field, which is also referred to as the Offset field, contains a value that indicates where the TCP header ends and the data field begins. It is required due to the fact that the inclusion of options can result in a variable length header.

TCP is a byte-oriented sequencing protocol. Thus, a Sequence field is necessary to ensure that missing or disordered packets are noted or identified. That field is 32 bits in length and provides the mechanism for ensuring that missing or disordered packets are noted or identified. The Acknowledgment Number field, which is also 32 bits in length, is used to verify the receipt of data. The number in this field also reflects bytes.

The Window field is 16 bits in length and provides TCP with the ability to regulate the flow of data between source and destination. Therefore, this field indirectly performs flow control. The Checksum field is 16 bits or 2 bytes in length. The function of this field is to provide an error detection capability for TCP. To do so, this field is primarily concerned with ensuring that key fields are validated instead of protecting the entire header.

The Urgent Pointer field is 1 byte in length. The value in this field acts as a pointer to the sequence number of the byte following the urgent data. As previously noted, the URG bit position in the Code field must be set for the data in the Urgent Pointer field to be interpreted. The Options field, if present, can be variable in length. The purpose of this field is to

enable TCP to support various options, with Maximum Segment Size (MSS) representing a popular TCP option. Because the header must end on a 32-bit boundary, any option that does not do so is extended via pad characters that in some literature is referred to as a Padding field. The Padding field is optional and is included only when the Options field does not end on a 32-bit boundary. Thus, the purpose of the Padding field is to ensure that the TCP header, when extended, falls on a 32-bit boundary.

2.1.1 Code Bits Fields

There are six individual 1-bit fields within the Code bits field. Each bit position functions as a flag to indicate whether or not a function is enabled or disabled. Gilbert (2002) examined each bit position in the field as follows:

The Urgent (URG) bit: is used to denote an urgent or priority activity. When such a situation occurs, an application will set the URG bit position, which acts as a flag and results in TCP immediately transmitting everything it has for the connection instead of waiting for additional characters. A second meaning resulting from the setting of the urgent bit or flag is that it also indicates the Urgent Pointer field is in use. Here the Urgent Pointer field indicates the offset in bytes from the current sequence number where the urgent data is located.

Acknowledgement bit (ACK): The setting of the ACK bit indicates that the segment contains an acknowledgment to a previously transmitted datagram or series of datagram. Then the value in the acknowledgment

number field indicates the correct receipt of all bytes through byte n by having the byte number n + 1 in the field.

Push (PSH) bit: The third bit position in the Code bit field is the Push (PSH) bit. This one bit field is set to request the receiver to immediately deliver data to the application and flags any buffering. Normally the delivery of urgent information would result in the setting of both the URG and PSH bits in the Code Bit field.

Reset (RST) bit: The fourth bit position in the Code bit field is the reset (RST) bit. This bit position is set to reset a connection. By responding to a connection request with the RST bit set, this bit position can also be used as a mechanism to decline a connection request.

Synchronization (bit): The fifth bit in the Code bit field is the Synchronization (SYN) bit. This bit position is set at start-up during what is referred to as a three-way handshake, which will be covered shortly.

Finish (FIN) bit: The sixth and last bit position in the Code bit field is the finish (FIN) bit. This bit position is set by the sender to indicate that it has no additional data, and the connection should be released.

2.2.0 Synopsis of TCP

The TCP is the most useful protocol among the transport layer protocol in TCP/IP protocol stack. TCP provides a communication channel between two hosts connected to a network. It provides flow-controlled, reliable, ordered, end-to-end delivery of data. This means that if TCP detects a packet that are lost or corrupted in the network, the TCP sender will retransmit the packet so as to allow the receiver to reconstruct the

original data in perfect order. Typically, Packet loss is caused by congestion. Congestion occurs when the demand for transmission of packet is greater than the available resources, such as bandwidths of links, buffer space and processing capacity at the intermediate nodes such as routers. The original TCP specification is described by Postel (1981).

The current TCP is a network-friendly protocol that increases its sending rate during network idle times and reduces its sending rate as the network becomes more congested (Postel, 1981). It uses packet loss as an indicator of network congestion and adjusts its sending rate according to perceived congestion levels in the network. To have a reliable packet delivery, TCP uses packet transmission sequence number to identify packets, and explicit acknowledgements (ACKs). TCP uses cumulative ACKs rather than per-packet ACKs, where an ACK referencing a particular point within the data stream implicitly acknowledges all data with a sequence value less than the ACKed sequence.

2.2.1 TCP connection

TCP is an end-to-end, point-to-point transport protocol used in the Internet. Being point-to-point protocol, it means that there is always a single sender and a single receiver for a TCP session. Being an end-to-end protocol, on the other hand, means that TCP session should cover all parameters and transportations involved from the source host to the destination host (Jamalipour, 2003). TCP provides connection-oriented, reliable byte stream service.

To transfer data from one end to another, TCP need to establish an open connection. The procedure of establishing the open connection is called three-way handshake because TCP open connection is established by exchanging three control messages as follows:

i. The client sends a packet with the SYN bit set and a sequence number N.
ii. The server then sends a packet with an ACK number of N+1, the SYN bit set and a sequence number X.
iii. The client sends a packet with an ACK number X+1 and the connection are established.

When the data transfer is completed, the open connection is closed in a way similar to the three-way handshake. Illustration is shown in figure 1.

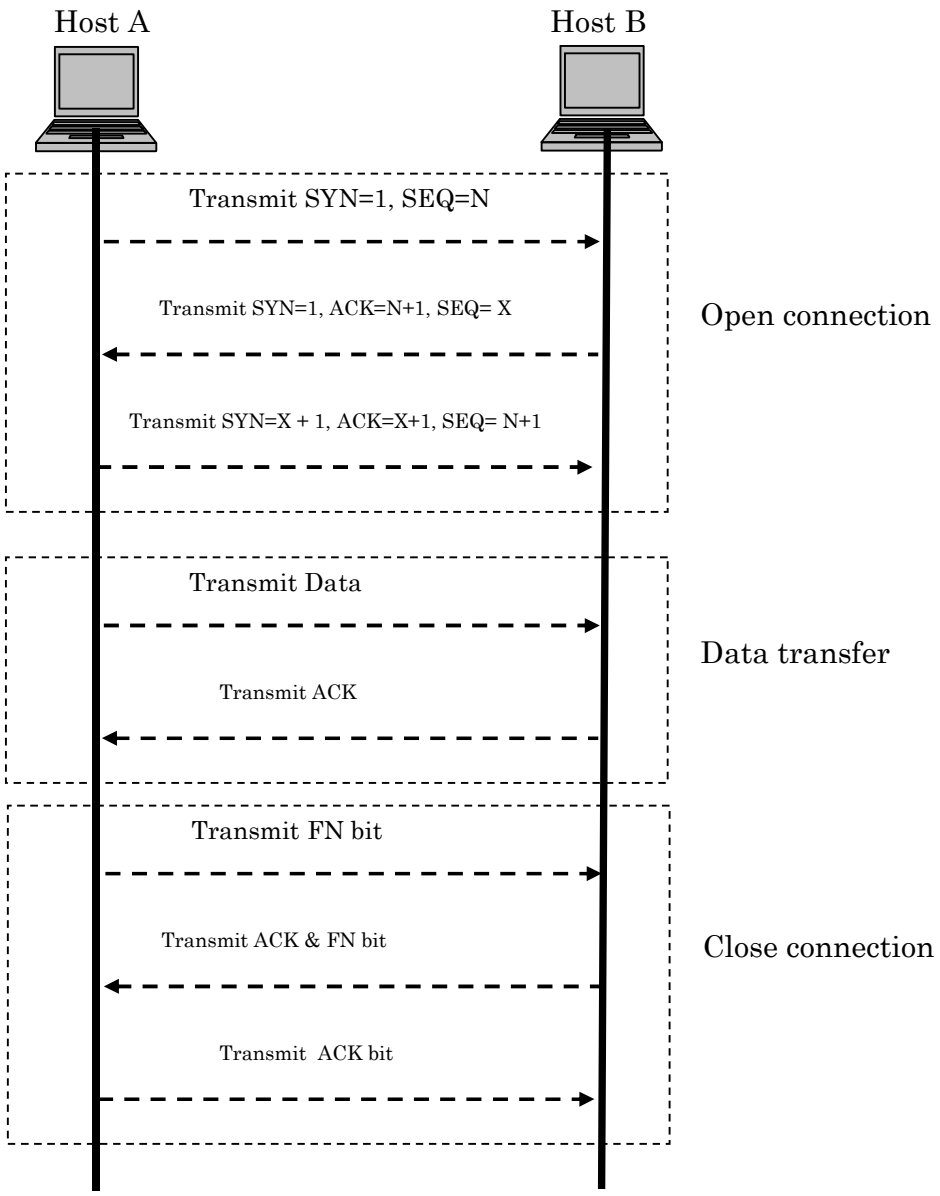


Figure 2.1: TCP open connection, Data transfer and Close connection.

2.2.2 Packet Loss detection and Re-transmission

TCP provides a reliable end-to-end service. During TCP data segment transmission, if there are segment losses, the losses have to be detected by the TCP sender and the missing segments have to be

retransmitted. Segment loss can be detect and re-transmitted by TCP using two schemes:

i. TCP time-out mechanism
ii. Duplicate ACK of successive segments.

2.2.3 TCP TIME-OUT MECHANISM

Time-out mechanism is employed in order to avoid long delays when there is no response from the receiver in a TCP connection. A timer is set, and it starts counting down after each TCP segment transmission by a sender. If the TCP sender does not receive a threshold number of ACKs before the timer expires, it assumes that either the packet or the ACK is lost, and retransmits the same packet again until an ACK is received. The duration of this timer is referred to as retransmission timeout. The details of how to compute the RTO are explained in Computer Communication Review by Jacobson (1988).

The TCP retransmit timeout (RTO) value must be optimized as possible to avoid:

Premature time-out: if the RTO value is set to be too small, that means the time well expires very quickly, giving raise to unnecessary retransmission.

Slowly response to loss segment: if RTO value is set to be too large, that means the TCP will slowly respond to the segment loss, which leads to longer end-to-end delay, and degrade performance.

The RTO value is set as a function of Round Trip time (RTT). When a TCP segment is sent over a TCP connection, the sender notes how long it takes for it to be acknowledged, producing a sequence of RTT samples. Older TCP implementations only time one segment per RTT, whereas newer implementations use the timestamp option to time every segment (Jacobson *et al.*, 1992).

2.2.4 Duplicate ACK of successive segments

When a TCP connection is established, datagram transmission begins. A segment is sent by TCP sender to another TCP receiver. When a segment arrives, the TCP receiver has to send acknowledgement (ACKs) to the TCP sender to indicating successful delivery of the segment. ACKs are classified into three types.

Normal ACK: whenever a new packet is received by the TCP receiver and there is no packet loss during a TCP connection, the ACK received by the TCP sender for such successful delivery is referred as normal ACK.

Partial ACK: if loss packet is detected, the packet has to be retransmitted. ACK received by the TCP sender that indicates the successful retransmitted packet, but not all the data outstanding when loss packet was detected is referred as partial ACK.

Duplicate ACK: refers to TCP receiver ACK, sent to the TCP sender acknowledging the successful delivery of segment following a lost TCP segment. If a segment is lost within a sequence of segments transmitted, the successful delivery packets following the lost packet will cause the

receiver to generate a duplicate ACK for each successive packet. The reception of these duplicate ACKs give a signal of such segment loss. After detection of the segment losses, the sender voluntarily reduces the congestion window, which in its turn decreases the transmission rate avoiding packet congestion. The TCP sender decreases the sending rate in two ways.

i. When a packet loss occurs, the congestion window size is reduced to some threshold, and afterwards it increases slowly.
ii. For every consecutive loss, the retransmission interval is doubled, in order to prevent the network from being crowded by retransmissions.

The TCP has also the implementation of increase the sending rate to it former rate after successful recovery of the lost segment.

2.3.0 TCP Congestion Control

To prevent the network from congestion, the sender's TCP must carefully regulate the amount of data to be transmitted in the network. TCP uses congestion control to smoothen the network congestion condition. Usually, a sender's TCP makes effort to transmit more data into the network when it learns the available of network resource capacity. On the other hand, if congestion is notice in the network, the sender's TCP will reduce its sending rate to allow the network to lessen the network congestion state. TCP's Congestion control uses two state variables, congestion window and slow start threshold.

2.3.1 Congestion Window

Congestion window *(Cwnd)* is a segment flow control mechanism that limits the number of segment in transit. The sender's TCP uses *cwnd* to control the number of packets outstanding in the network (Jacobson, 1988). The availability of the network resources is probed using Additive-Increase, Multiplicative-Decrease (AIMD) mechanism (Allman *et al.*, 1999). The mechanism determines the dynamic adaptation of *cwnd* to the changing network condition. The effectiveness of AIMD is well notice when the TCP connection established requires bulk data transfer continually. The TCP sender starts with initial *cwnd* size of one or two TCP segments and increases its transmission rate exponentially every RTT (cwnd = cwnd * 2) until the congestion window size reaches a *slow start threshold* value and the congestion window size is increased linearly (cwnd = cwnd + 1) after every RTT. When segment acknowledgements are not received, the *slow start threshold* is set to half of the current congestion window size (cwnd), and the algorithm restarts. Congestion window size has a range from one maximum segment size (MSS) to advertised window size by a receiver, *Wmax*.

2.3.2 Slow start threshold

Slow start threshold (*ssthresh)* is another TCP state variable that determines which of the congestion control algorithms to be employed when loss is detected. At the time of setting up TCP connection, *ssthresh* is set to

the window size advertised by the receiver (*Wmax*). When a segment loss is detected the *ssthresh* is then set to the half of the *cwnd*.

2.4.0 Congestion Control Algorithms

Congestion control consists of four algorithms; slow start, congestion avoidance, fast retransmit, and fast recovery (Floyd, 2000). Slow start and congestion avoidance algorithms probe the network condition to allow the sender to transmit new packets. While the other two algorithms, are concerned in loss recovery. The basic TCP congestion control which uses the Slow-Start and Congestion Avoidance algorithms is based on the work initiated by Jacobson (1988).

2.4.1 Slow Start

The slow start algorithm is designed in such a way that congestion window is increased by one for every received ACK that acknowledges successful delivery of a segment at the receiver's TCP. When a TCP connection is established, the sender sends one packet and awaits the corresponding ACK. When the ACK is received, the sender increases its congestion window by one, making the sender's TCP to transmit two segments at a time and the sender's TCP must await the corresponding two ACKs for the transmitted segments. If two ACKs are received, the sender's TCP increases congestion window again by two, the congestion window size gets to four, making the sender's TCP to transmit four. The behavior of the slow-start algorithm continues in that order until it reaches to *ssthresh* or

loss segment is detected by three duplicate ACKs or RTO (Allman *et al*, 1999).

2.4.2 Congestion Avoidance

As the congestion window size increase, if the value is greater than the value of *ssthresh*, the TCP congestion control changes mode from slow start to congestion avoidance (Stevens, 1997). When a new TCP connection is established with a host on another network, the value of *ssthresh* is set to the advertised window size by a receiver, *Wmax*. However, when the congestion window is greater than the value of *ssthresh,* the *ssthresh value* is set to half the current congestion window size to avoid congestion. Figure 2.2 shows the TCP congestion window behaviour in slow start and congestion avoidance. In the figure, the thinner solid line represents the congestion window behaviour of a connection when slow start is used. The dash solid line represents the initial *ssthresh value* and adjusted *ssthresh* value when avoidance is used.

The initial congestion window size is set up to one maximum segment size (Allman *et al.*, 2002), and the slow start congestion window size continues to grow until it gets to initial *ssthresh value* then the congestion window is reduced to one MSS and the *ssthresh value* is updated to half of the previous congestion window size.

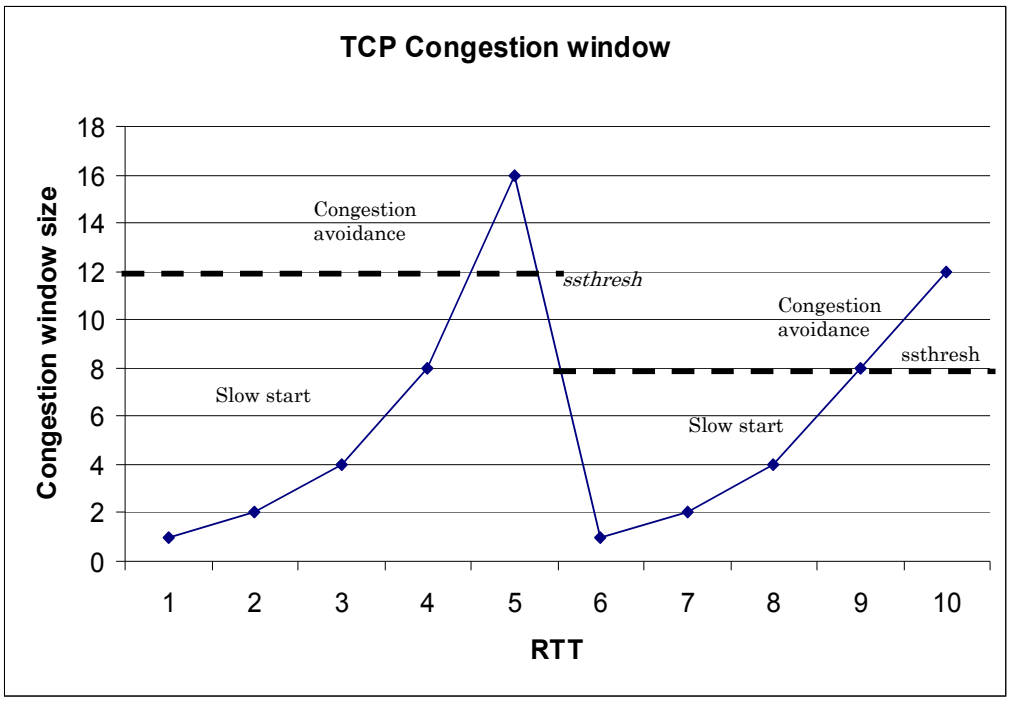


Figure 2.2: The TCP congestion window behaviour

2.4.3 Fast Retransmit

The TCP Slow-Start and Congestion Avoidance algorithms were observed to create new problems. One of the problems relates to the packet loss detection via RTO. As described in section 2.2.2, a packet loss is learned when the RTO has expired. Nevertheless, RTO scheme may lead to significant delays in the data transmissions, so another way to determine packet loss has been added to TCP.

Typically, for every segment received including segment out-of-order, the receiver's TCP must send either an acknowledgment (ACK) or Duplicate ACK. In the case of Fast Retransmission algorithm, the ACK contains the reference to the data, which was received before the packet loss (Floyd, 1994). When a sender's TCP receives three duplicate ACKs (i.e.

ACKs contains reference to the same data) before the RTO expired, the sender assumes a packet lost and retransmission commence immediately not waiting for RTO to be completed. In this scheme, time is optimized.

2.4.4 Fast Recovery

Another noticeable issue associated with the packet loss detection is the problem of great decrease of the congestion window (*cwnd)*. In the initial implementation of TCP, if a segment is lost the value of *cwnd* is set to the value of the minimal *cwnd*. With the Fast Recovery algorithm, the drastic decrease is handled. When a segment loss is learned following the Fast Retransmission algorithm, the new value of the congestion window is set to *ssthresh* + 3*MSS (Floyd and Henderson, 1999). For every duplicate ACK the *cwnd* is incremented by 1, when a new ACK is received the Fast Recovery algorithm terminates, and the congestion avoidance algorithm is continued.

2.5.0 TCP Implementations

The original TCP implementations used window size based flow-control to control the use of buffer space at the receiver and retransmission after a packet drop for reliable delivery. The implementations did not include dynamic adjustment of the flow-control window in response to congestion collapse such as classical congestion collapse (Nagle, 1984) due to unnecessary retransmission of segments, fragmentation congestion collapse (Kent and Mogul, 1987) resulting from

Maximum Transfer Unit (MTU), and undelivered-segments congestion collapse that occurs when networks overloaded with packets that are discarded before they reach the receiver (Floyd and Fall, 1999).

The exponential growth in the Internet usage has caused a great increase in the number of TCP implementations. However, the development of TCP must handle changes that may enforce any network setup into congestion collapse (Floyd, 2002). TCP has experienced number of changes in its primitive design, during its development process. Many versions of TCP exist today and each TCP implementation employs congestion control algorithms as described in section 2.4. The available implementations include TCP Tahoe and TCP Reno.

2.5.1 TCP Tahoe

The fundamental algorithm for congestion avoidance and control was first introduced by Jacobson (1988). The implementation of the algorithm, called TCP Tahoe, was based on the principle of *conservation of packet* and it has introduced significant improvements for working over a shared network (Floyd and Henderson, 1999). When TCP connection is established at the available resources capacity, a packet is not set in transit into the network until sent packet leaves the network. In other word, a packet can be set in transit only when the sender receives an acknowledgement from the receiver indicating successful delivery of the packet. The acknowledgements clock the outgoing packets because an acknowledgement means that a packet was taken off the transmission medium and resources

(Floyd and Henderson, 1999). TCP Tahoe includes Slow-Start, Congestion Avoidance and Fast Retransmit. The Slow Start algorithm was introduced to control transmission following any detected congestion.

2.5.2 TCP Reno

The modification of how TCP Tahoe response to detection of loss packet through duplicate ACKs is what leads to the implementation of TCP Reno (Floyd and Fall, 1996). The concept is that if packet loss is detected via a duplicate ACK before TCP timeout expired, TCP sender retransmits the loss packet by performing a Fast retransmit algorithm without waiting for RTO and enters into its Fast recovery stage until the non duplicate ACK that acknowledges the entire transmit window of data is received. If it does not receive such an ACK, TCP Reno experiences a timeout and enters the slow-start state as implemented in TCP Tahoe. TCP Reno adds Fast Recovery to TCP Tahoe.

2.5.3 TCP New Reno

TCP New-Reno is a modified version of TCP Reno. In TCP Reno, TCP sender would leave Fast Recovery on the receipt of first ACK that acknowledges new data. This algorithm recovers loss packet efficiently, if there is only one lost packet. However, the algorithm fails from multiple packet recovery within a sliding window (Floyd and Fall, 1996). Therefore, an enhanced version of Fast recovery algorithm was proposed to address the short comings in TCP Reno (Floyd and Fall, 1996). The algorithm

proposed that a smaller value for *ssthresh* causes early termination of the slow start stage and also slow increase of the cwnd. In the other way round, a larger value causes the sender to overwhelm the network with packets, causing congestion.

The idea behind TCP New Reno is that during the Fast Recovery, the TCP sender does not exit its Fast Recovery stage on receiving partial ACK until all the data packets which were outstanding at the time entered Fast Recovery. Thereby recovering from multiple packet loss in a single window of data and exits its Fast Recovery phase either on receiving the ACK that acknowledges entire data within that window or on occurrence of retransmission timeout.

2.5.4 TCP Vegas

TCP Vegas was presented before New Reno, SACK and FACK were developed (Brakmo *et al.*, 1994). Vegas is a TCP implementation which is builds on the fact that proactive measures rather than reactive against packet losses. TCP Vegas is fundamentally different from other TCP variants in that it does not wait for loss to trigger congestion window reductions. It employs an alternative strategy in that it tries to predict when congestion is about to happen and adapts its window to compensate. This is a proactive approach as it attempts to reduce its sending rate before packets start being dropped by the network. TCP Vegas keeps track of the time each segment is sent. When an ACK arrives, it estimates RTT as the difference between the current time and the recorded timestamp for the

relevant segment. TCP Vegas has not been widely implemented and is not universally accepted by the Internet community and is still a subject of much controversy.

2.5.5 TCP SACK

The default TCP acknowledgment behavior is to acknowledge the highest sequence number of in- order bytes. This default behavior is prone to cause unnecessary retransmission of data, which can exacerbate a congestion condition that may have been the cause of the original packet loss. A proposed modification to TCP, selective acknowledgement (SACK) allows a TCP receiver to acknowledge out-of-order segments selectively rather than just cumulatively acknowledging the last correctly received in-order segment (Mathias *et al.*, 1996). SACK option is used by TCP receiver to inform the TCP sender that a non contiguous segment of data has been received and it is queued. So the sender need retransmit only the packets that have actually been lost (Floyd and Fall, 1996), (Blanton *et al.*, 2003). When a retransmitted packet is itself dropped, the SACK implementation detects the drop with a retransmit timeout, retransmitting the dropped packet and then slow-starting. If all of the outstanding data is ACKed (received ACK sequence is greater than *recover*), the sender exits fast recovery and continues in congestion avoidance. To use SACK, both the TCP sender and receiver must support the feature and must enable it by negotiating the SACK-Permitted option during the connection establishment. Adding the SACK option to the TCP flavours such as TCP

Tahoe or Reno, does not change their basic underlying congestion control algorithms. The information about missing sequence numbers is transmitted to TCP sender using three SACK blocks with each ACK, using the rules outlined by Mathias *et al* (1996).

2.5.6 TCP FACK

The Forward Acknowledgment (FACK) algorithm proposed, aims at better recovery from multiple losses (Mathias *et al.*, 1996). In FACK, TCP maintains two additional variables: the forward-most segment that has been acknowledged by the receiver through the SACK option and retransmitted data that reflects the amount of outstanding retransmitted data in the network. Using these variables, the sender can estimate the actual quantity of outstanding data in the network and can inject new data if allowed by receiver's window. TCP FACK regulates the amount of outstanding data in the network to be within one segment of CWND, which remains constant during the Fast Recovery phase.

2.6.0 Multiple TCP connections

Multiple TCP connections give raise to a new situation to TCP performance. When there is a group of multiple TCP connections over a link, the queuing delay is expected to affect the behaviour of TCP, aggregating congestion window more aggressively than a single TCP connection and is more likely to cause congestion losses. Since the connections are independently of each other, the connections that detect

losses reduce their congestion windows while the others still put data onto the network increasing rates. This problem becomes more serious in a low bandwidth network environment where the highest traffic through the link is TCP application traffics. Another problem of multiple connections as pointed out by Stadler and Gelman (1998) is that when one or more of the concurrent connections close, the amount of time it takes the remaining connections to absorb the additional bandwidth can be significant.

2.7.0 Enhance TCP models

The incorporation of wireless networks into the typical Internetworking is becoming increasingly popular in the world of telecommunication. As a result, traffic behaviour changes over the available link resources. To ensure reliable data delivery, efforts have been devoted to provide TCP enhancement model over different network infrastructures. TCP supports the most popular suite of applications on the Internet today and it has been enhanced in recent years to improve robustness and performance over network of varying capacities and quality. With the explosion of the Internetworking infrastructure, TCP behaviour outlined by Postel (1981) has not change. Many TCP optimization models have been proposed and can be categorized into four groups; split connection, link-layer, explicit notification and end-to-end protocols.

2.7.1 Split connection model

There is significant difference in characteristics between wireless and wired networks. The wired part of the heterogeneous network is more reliable than the wireless part in terms of link capacity and error rate; however, with the explosive growth of the network infrastructure transmission would be bottlenecked at low link resources. TCP splitting is an enhancement that attempts to shield the wireless part from the fixed network by separating the flow control at the intermediate node. The intermediate node behaves as a terminal in both the fixed and wireless parts. Both end hosts communicate with the intermediate node independently without knowledge of the other end. The intermediate node is reinforced with functionalities to coordinate the transaction between the two network parts (Postel, 1981). Therefore, split mode divides a TCP connection into wireless and wired parts, and ACKs are generated for both parts separately as shown in the figure 2.3. The intermediate node becomes responsible for the recovery of loss packets.

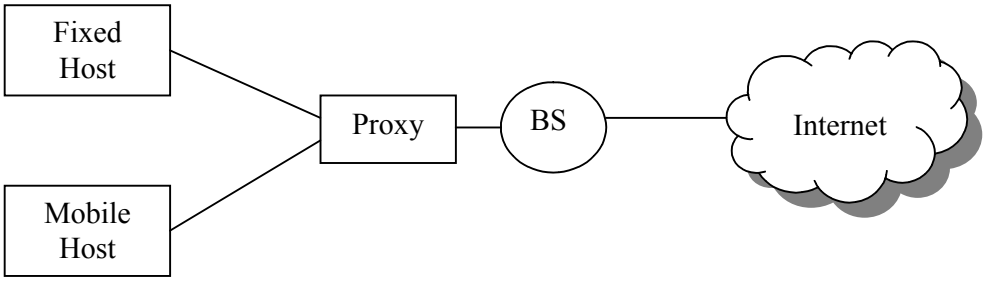


Figure 2.3: Split connection model

2.7.2 Link-Layer model

The Link-layer models are designed to improve the poor performance of TCP over wireless. The models are not TCP specific and they do not use any knowledge of TCP in enhancement at the transport

layer (Postel, 1981). They perform link-level functions such as Automatic Repeat Request (ARQ) or Forward Error Correction (FEC) to improve TCP performance, thereby making the wireless link layer look similar to the wired case for TCP by recovering the wireless error locally. There have been several proposals for reliable link-layer protocols. The main techniques employed by these models are forward error correction (FEC) and ARQ (Postel, 1981). This method is illustrated in Figure 2.4. ARQ When a frame is detected to contain errors after decoding, it is discarded and an ACK is sent back to the sender requesting a retransmission of the frame.

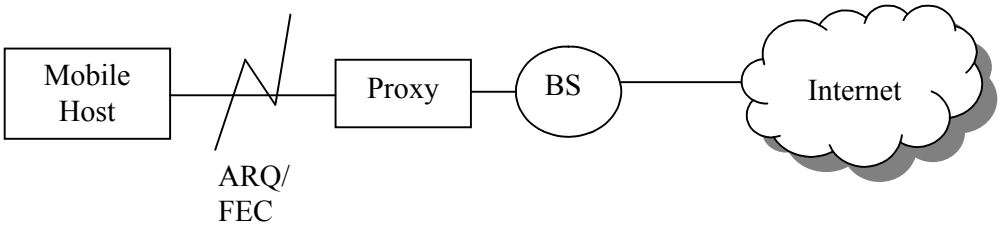


Figure 2.4: Link-layer model

2.7.3 Explicit Notification

There are many explicit notification models that have been proposed to enable the TCP sender to differentiate between different types of packet losses due to congestion from wireless error. Among such proposal are Explicit Congestion Notification (ECN) and Explicit Loss Notification (ELN) (Ramani and Karandikar, 2000), (Balakrishnan *et al.*, 1997). The idea behind this approach is to enable the TCP sender to distinguish packet losses due to congestion from wireless error.

For instance, in explicit congestion notification, the TCP receiver uses one of the reserved bits called the ECN-echo (ECE) flag in the TCP header to explicitly notify the TCP sender of the network congestion. ECE flag requires support from both the routers as well as the end hosts. It requires the routers to be able to identify packets that are ECE capable and to mark only such packets from ECE capable hosts. Two bits in the IP header field are utilized to achieve this; the ECE Capable Transport Bit (ECT) is set by the sender during the connection establishment process if both the end systems are ECE capable. Therefore, the model uses two new flags, namely the ECE flag and Congestion Window Reduction (CWR) flag, in the reserved field of TCP header as shown in Figure 2.5

<table>
<tr><td colspan="9">Source port (16 bits)</td><td>Destination port (16 bits)</td></tr>
<tr><td colspan="10">Sequence number (32 bits)</td></tr>
<tr><td colspan="10">Acknowledge number (32 bits)</td></tr>
<tr><td>Header length (4 bits)</td><td>Reserved (4 bits)</td><td>CWR</td><td>ECE</td><td>URG</td><td>ACK</td><td>PHS</td><td>RST</td><td>SYN</td><td>FIN</td><td>Receiver Window size (16 bits)</td></tr>
<tr><td colspan="9">Checksum (16 bits)</td><td>Urgent pointer (16 bits)</td></tr>
<tr><td colspan="10">Options (32 bits)</td></tr>
</table>

Figure 2.5: TCP header with additional two bits

To inform the TCP sender of the network congestion, the TCP receiver sets the ECE flag. When the sender receives an ACK with the ECE flag, the CWR flag is set to notify the receiver that the action has been taken against the congestion notification.

2.7.4 End-to-End models

Typical TCP assumed that packet loss is cause by network congestion. But there are other parameters that contribute a lot to packet loss such as noisy links. Overlapping radio channels, signal attenuation and additional noises they have a significant impact on packet losses. This means that these parameters can cause the TCP to reacts with great reduction of congestion window, therefore degrading the performance of TCP. End-to-end proposals make the TCP sender handle packet losses caused by both congestion and random wireless errors and requires minimal or no processing at the base station (Jacobson, 1990). Some of the end-to-end schemes proposed are SMART, WESTWOOD, and FREEZE-TCP among others.

2.8 Wireless link issues

Wireless link characteristics have significant impact on TCP performance. TCP was not designed with wireless link characteristics in mind. Hence, TCP is not expected to perform well in a wireless network. The main issue of a wireless link is the high Bits Error Rate (BER). High BER in wireless networks causes packet loss regardless of network congestion (Chandra *et al.*, 2003). Eventually, this causes TCP to unnecessarily reduce its transmission rate, degrading the performance of

the TCP. Another important issue of the wireless link that was not considered in the design of TCP is link delay variations. Long delay is also among the causes of packet losses.

Long sudden delays during data transfers are common properties in wireless WANs (Ludwig and Katz, 2000), (Luo and Fapojuwa, 2003). Experiments by Ludwig and Katz (2000), Luo and Fapojuwa (2003) indicated that data transmission over wireless link increases packets delay and delay variation degrades the throughput of TCP connections. The results of the studies by Ludwig and Katz (2000) reveal that among the major adverse effects on TCP performance are spurious fast retransmit and spurious timeouts.

2.9 Issues of TCP and UDP flows over Wireless link

A wireless link in a network could be viewed as an abstract notion defined by the connectivity between two nodes. Data sent by a node might be overheard by many other nodes or even be perceived as only interference. The presence of a wireless link in a network might be intermittent. Unlike wired networks, where congestion is in general caused by overfull queues, there may also be contention for the spatial reuse of the ether, causing congestion in the network. The TCP feedback loop is responsible for adapting the sender data rate in response to, e.g., congestion. However, this end-to-end congestion control mechanism has reduced efficiency in wireless networks because transmission is inherently broadcast (Jacobson, 1990).

Furthermore, there are different ranges for uncast radio transmission, broadcast radio transmission and interference. Even if the offered data rate is adapted to the bottleneck in a network path, transmissions might still contend and interfere with other. TCP has been designed to be fair between competing flows whereas Constant Bit Rate (CBR) UDP flows are not rate adaptable and lacks congestion control. In the presence of UDP flows, TCP should probe the data path by increasing its congestion window until loss occurs. In that case it should back off and not use more bandwidth than what is left after the UDP flow. The work in this study examines the behaviour of the TCP adaptation in the presence of UDP flows.

CHAPTER THREE

Evaluation of TCP Behaviour in a Network

3.0 Methodology

This section describes the methodology adapted for the TCP performance measurements. The environment and arrangements used in the performance measurements, the exact test parameters, workload models and the TCP features used are described. Finally, the validity and possible inaccuracies in the selected model were discussed

The purpose of this study is to evaluate the performance of the current state of TCP behaviour of network generating heterogonous traffic flows with high TCP traffic over router and wireless link, to compare the effect of the link data rate and buffer size on the TCP performance. The performance tests are observed, the results analysed and reasons for the observed behaviour explained.

3.1 Simulation Tool

The use of network simulator to emulate an existing network environment has many merits. With the network simulator, the wireless link and buffer properties, and the emulated network architecture under TCP can be controlled. The predefined test configurations may also run automatically, which makes the execution of the performance measurements more convenient.

There are many simulators that have been created. However, a lot of them were written for special purposes, for testing just one network component or protocol. It is obvious that those simulators cannot be widely used and this work does not deal with them. On the other hand, the goal of other simulators is to be extensible and to allow others to add their own modules for currently unsupported or new protocols, create new network devices. The most widely known simulator of this kind is OPNET. For emulation of the targeted environment, OPNET tool is selected for this research work.

3.1.1 OPNET

OPNET is a graphic network simulator that provides a set of tools for network modelling, displaying statistics. Many predefined types of nodes are present and almost all widely used protocols and technologies are supported. The supported operating system is Windows.

OPNET provides network modelling, simulation, and analysis features. It provides the user with the ability to choose network devices, such as switches, routers, and workstations; connect them together with various types of links, such as Ethernet 100BaseT, FDDI, and ATM; and define network traffic patterns. OPNET can be used to simulate the behaviour of the modelled network, to collect statistics, such as application response time or link utilization, and to display graphs of the collected statistics. It may also be used to design computer networks from scratch, to

validate or troubleshoot an existing configuration, or to evaluate a proposed upgrade.

OPNET provides "global" objects for defining profiles and applications. The advantage of using a global object is that once you have defined the profiles and applications, you can re-use them across the entire topology. OPNET ships with pre-defined profiles and applications that may suit the behaviour you wish to describe. You may, however, wish to modify the existing definitions to suit your needs or even create new application and profile definitions.

3.2 Area of Interest in the Study

There are many topics of interest in the performance tests of a network last element, and wireless link. In this study, one of the areas of interest is the TCP performance implications of the data rate and propagation delay at the wireless. To have the idea of what happens at the TCP level, the data rate that causes spurious retransmission and packets drop have been studied. By observing the effect of data rate and competing traffic flows on the TCP performance, the successful data rate which do cause less spurious TCP retransmission and packets drop were identified.

Another area of interest is the effect of congestion at the router when the buffer size is low compare to the rate at which traffics flow, which does cause packet drop at the link layer. Congestion at the link does affect the TCP behaviour more severe when there is high traffic flows. To observe its effects, heterogonous traffic flows were analysed. The congestion at the link

with small, medium and large router buffer sizes was observed to see how the selected router buffer size affects the TCP behaviour. Finally, the effects of various TCP technologies on the TCP behaviour on the link were evaluated.

3.3 The Targeted Environment

The target environment of study is a LAN network generating UDP and TCP traffic flows. The network environment consists of ten client hosts each communicating with one of the three different server hosts via a wireless link and a last network element. The study concentrated on the behaviour of a client TCP in the LAN, in the presence of multiple TCP and UDP traffic flows. The LAN has bandwidth of 100 megabits, wireless link data rate of 64,000 bps and router buffer of 2MB. The network is assumed to provide traffic flows significantly faster than what the wireless link can transmit. Figure 3.0 illustrates the elements which are emulated from the target network environment.

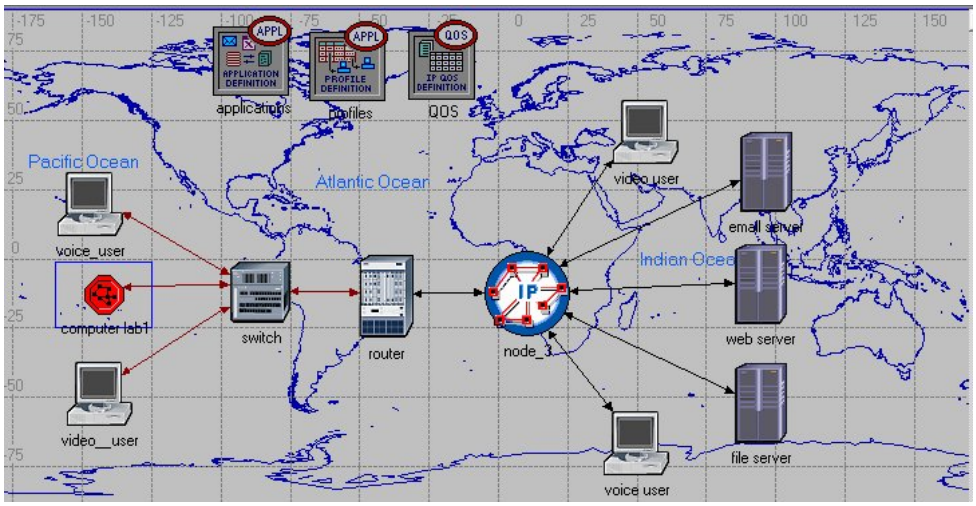


Figure 3.0: Emulation environment.

OPNET software was used to emulate the target network environment. In the emulation environment, performance tests were made using ten client hosts on the network and two different hosts were set to generate voice and video streams. Each of the ten hosts generates TCP traffic. Traffic shaping was not imposed in the network set up. The traffic flows are forwarded base on Best-effort scheme.

The network hosts were assumed to be running the Windows operating system. Each of the client hosts and the server hosts are the TCP endpoints communicating with each other using TCP connection. The TCP enhancements that are used in the performance tests are implemented only in the endpoint hosts.

The wireless link is emulated by issuing appropriate data transmission rate and propagation delay. Link layer retransmissions are emulated by causing an error delay for a packet. During the error delay no packets are released from the link receive buffer. The last-hop router queue is emulated using the input queue. The traditional first-in-first-out scheduling is used on the input queue. If there is no room for incoming packet in the input queue, emulator router discards the packet.

3.3.1 Modelling traffics

The most commonly used applications in the target network environment are those that require Internet access, which we call *the Internet applications* and Internet application traffic must traverse via the last router, the IP32 cloud, and finally via the Internet Servers gateway to

reach the Internet Servers. The Internet applications are web browsing, instant messaging, video conference, FTP, e-mail, and online games.

OPNET Modeler provides standard built-in models for software applications such as *web (HTTP)*, *e-mail* and *FTP* which can be easily configured to simulate applications used in the subnets. Thus, these applications are implemented and configured via OPNET's Custom Application feature. Table 3.0 summarizes the application definitions.

Table 3.0: Application Definitions

Application	Meaning
e-Mail (light)	Indicate light activities of e-mail application
e-Mail (heavy)	Indicate heavy activities of e-mail
FTP (light)	Indicate light file transfer activities
FTP (heavy)	Indicate heavy file transfer activities
Web browsing (light)	Indicate light browsing activities
Web browsing (heavy)	Indicate heavy browsing activities

To set-up and configure any application in OPNET, the Application Configuration and Profile Configuration modules was added to the emulated target network environment. The Application Configuration module contains the application definitions, while the Profile Configuration module contains the profiles of user behavior, e.g. describes how the users employ the applications defined in the Application Configuration module.

3.3.2 Configuring Web (HTTP) Applications

To configure a web browsing application, the application named HTTP was selected from the list of built-in models. OPNET provides pre-set configurations such as: *Light Browsing, Heavy Browsing, Searching*, or

Image Browsing. For this study, light and heavy browsing were selected. In addition, table 3.1 and 3.2 provide the pre-set configurations parameters of the selected web applications. *HTTP Specification* which defines the version of HTTP protocol, *Page Interarrival Time* which specifies time in seconds between consecutive webpage downloads, and type of service specifies traffic shaping policy adapted.

Table 3.1: Heavy Web Browsing Configuration

Attribute	**Value**
HTTP specification	HTTP 1.1
Page interarrival time (second)	Exponential (60)
Type of service	Best effort

Table 3.2: Light Web Browsing Configuration

Attribute	**Value**
HTTP specification	HTTP 1.1
Page interarrival time (second)	Exponential (720)
Type of service	Best effort

3.3.3 Configuring e-mail Applications

To configure an e-mail application, the application named e-Mail was selected from the list of built-in models. OPNET provides preset configurations for e-mail application as well. The preset E-mail configurations include *Low Load*, *Medium Load*, and *High Load*. The model also allows custom configuration via parameters such as *Send(Receive) Interarrival Time* which specify the amount of time in seconds between consecutive sent (receive) operations, *Send(Receive) Group Size* which determine the number of e-mails messages per single sent(receive)

operation, and *E-Mail Size* which defines the size of e-mail message in bytes. The e-mail configurations for selected loads are shown in Table 3.3 and 3.4.

Table 3.3: High Load e-mail Configuration

Attribute	Value
Send interarrival time (second)	Exponential (360)
Send group size	Constant (3)
Receive interarrival time (second)	Exponential (360)
e-mail size (bytes)	Constant (2000)
Type of service	Best effort

Table 3.4: Low Load e-mail Configuration

Attribute	Value
Send interarrival time (second)	Exponential (3600)
Send group size	Constant (3)
Receive interarrival time (second)	Exponential (3600)
e-mail size (bytes)	Constant (500)
Type of service	Best effort

3.3.4 Configuring FTP Applications

To configure ftp application, the application named ftp was selected from the list of built-in models. OPNET also provides preset configurations for FTP application. The preset FTP configurations include *Low Load*, *Medium Load*, and *High Load.* The *Inter-Request Time* parameter is the time in seconds between consecutive FTP requests. The *File Size* defines the size in bytes of the FTP file to be transferred. The e-mail configurations are shown in Table 3.5 and 3.6.

Table 3.5: Heavy File Transfer Configuration

Attribute	Value
InterRequest time (seconds)	Exponential (360)
File size (bytes)	Constant (50000)
Type of service	Best effort

Table 3.6: light file transfer configuration

Attribute	Value
InterRequest time (seconds)	Exponential (3600)
File size (bytes)	Constant (1000)
Type of service	Best effort

3.3.4 Configuring VOIP Stream Application

In order to model VOIP traffic in OPNET, an object called application definition attribute which consist predefined applications was modified to support voice streaming.

Table 3.7: VOIP Stream Configuration

Attribute	Value
Silence length (seconds)	Exponential (0.65)
Talk spurt length (seconds)	Exponential (0.352)
Encoder scheme	G.711 (silence)
Voice frame per packet	1
Type of service	Interactive (6)

Table 3.7 shows the VOIP configuration used in the simulation model. The Voice application was modeled by configuring the *(Voice) Table* parameters. The VoIP application uses G.711 encoder scheme and Interactive Voice (6) as the type of service for establishing the VoIP calls.

3.3.4 Configuring Video Stream Application

An object called application definition attribute which consist predefined applications was used to model the video stream application by modifying the predefined parameters.

Table 3.8: Video Stream Configuration

Attribute	Value
Frame interval time	10 frames/sec
Frame size	128x120 pixels
Type of service	Best effort
Traffic mix (%)	All discrete

Table 3.8 shows the Video configuration parameters used in the simulation model. The Video application was modeled by configuring the *(Video) Table* parameters.

3.4 Metrics of the performance analysis

The selected metrics for this performance evaluation and why they are of interest are discussed below:

Retransmissions: another interesting metric is the number of retransmission which does strongly affects the throughput of a TCP connection. When the number of retransmission is reduced, the throughput of the TCP connection most significantly improves. In this research study, retransmission is triggered by packet drop at the last-hop router or by a retransmission timeout caused by excessive delay.

Dropped packets: the number of dropped packets has direct affects on the number of retransmissions. This metric gives additional information needed to derive the number of unnecessary retransmissions. Because packets drop take place only at the last-hop router as a result of buffer overflow, this metric measures the severity of congestion at the last-hop router.

It is difficult to automatically measure these parameters with the current tools we have. However, the metrics are relatively easy to observe from the traces manually. Thus, the parameters were analytically measured.

3.5 Selected Test Cases

Two kinds of analysis of TCP behaviour were selected for the performance evaluation. Firstly, the selected metrics are measured using set of test cases and the observed performance are summarised based on the measured metrics. Secondly, the most interested test cases are selected and analysed to explain the reasons for the observed performance. The selected test cases are then defined by considering the three different factors: the application traffic model used in the test, the emulated network and the link model, and the TCP implementation used.

Targeted Environment Model: two application traffics models are selected for the purpose of this study. The application traffic models are described below.

Multiple TCP connections environment model: in this model, multiple TCP connections workload for transfer of only TCP traffic is setup. Each TCP traffic connection is transferring a bulk data from the fixed host on the Internet to a client host on the subnet of the fixed network.

Multiple TCP connections environment with UDP traffic flows model: in this model, multiple TCP and UDP connections are established. The TCP traffic connections are engaged in transfer of bulk data from a fixed host on the Internet to a client on the subnet of the fixed network. While UDP traffics are flows of point-to-point interactions. These flows could be video conference or voice over IP (VOIP) stream.

Number of tests using these workload models were selected and tested. Having multiple-parallel application connections transmitting over a bottleneck link has remarkable implications on TCP. However, this research work is concern with TCP behaviour in the model, in the presence of UDP traffic flows.

3.6 Simulation Process Configuration

As described earlier, the target network path contains a wireless link and last-hop router. The main objective is to inspect the performance of TCP in a multiple reliable connections with the presence of UDP traffic flows over wireless link and router of a limited buffer memory, and to inspect certain details of TCP behaviour more closely.

Buffer size of 8mb and data rate of 64,000 kb were used as baseline for the performance analysis. However, three different router buffer sizes are combined with three different data rates scenarios were also analysed. The primary interest of the study is not to find exactly the optimal buffer size and data rate for the different scenarios. Hence the selection of the three different buffer sizes and data rates are to limit the number of test runs. Table 3.9 shows the data rates and buffer sizes selected for the performance tests.

Table 3.9: Selected Wireless Link Data Rates and Router Buffer sizes

Data rate (bps)	Buffer size (mb)
128,000	16
256,000	32
512,000	64

SACK scheme was used as the baseline technology for establishing a reliable connection. Using the same modelled environment, a number of TCP implementations were tested and compared to see which among give better performance. The selected TCP technologies are Reno and New Reno. The TCP attribute configuration for client and server hosts are summarised in table 3.10.

Table 3.10: TCP Attribute Configuration

Attribute	Value
Maximum segment size (bytes)	512
Receive Buffer (bytes)	8760
Delay ACK mechanism	Segment/clock based
Maximum ACK delay (sec)	0.2
Slow start initial count (MSS)	1
Duplicate ACK threshold	3

3.7 Justification of the model

The justifications of why the test results are relevant for analysis of TCP behaviour over the targeted environment is that by using parallel TCP connections we may easily inspect the theoretical TCP behaviour, but the TCP behaviour in the actual network with competing traffic is likely to be different. Therefore, it is important to perform tests with parallel competing UDP and TCP traffics in addition to the tests with parallel TCP connections. Although it is very difficult to accurately model the competing traffic in the Internet, the target network environment is easier to model because the focus of the study is only to inspect traffic originating from the local network. Moreover, because the bottleneck link and router are local network elements, other elements in the Internet connections are relatively of limited concern for the analysis. We use several parallel Internet application connections to inspect the effect of competition, which is likely to cause synchronization effects that might not be present in an open network.

The model resembles the behaviour of real network environment. However, because we are inspecting TCP performance, the study is not interested in the reasons behind the modelled environment in detail, but the consequences of UDP traffic flows in a multiple reliable connections environment.

CHAPTER FOUR

Analysis of the TCP Performance

4.0 Introduction

The performance of the TCP was analysed with various data rates and different router buffer sizes scenarios as figured out in section 3.6. The goal of the study was to evaluate the performance of TCP and to understand how the behaviour should be changed in order to improve the TCP performance. Another motivation for the detailed analysis is to validate the experimental results.

4.1 Multiple Reliable Connections Environment (MRCE)

This section analysed the results of tests with multiple reliable connections workload. In this model, each host established TCP connection for traffic flow of different modelled Internet application traffic described in section 3.3. Figures 4.1 and 4.2 showed the results of baseline configurations.

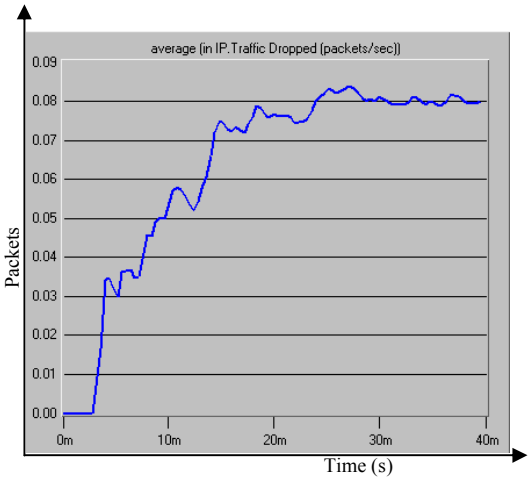


Figure 4.1: MRCE Dropped Packets

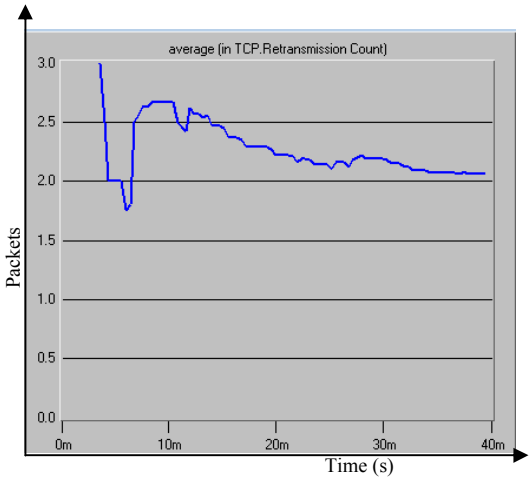


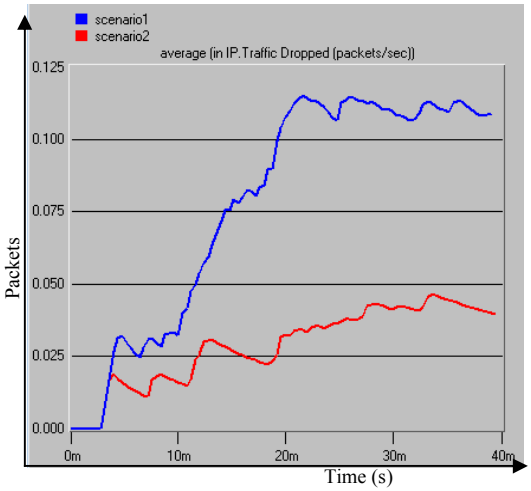
Figure4.2: MRCE Retransmission Count

The results pointed out the performance of TCP in a multiple TCP connections environment when no UDP traffic flows are involved.

It is observed from figures that the first 6 minutes of simulation, TCP experience slight packets drop, but packets drop increases gradually and reached up to 8.6%. The reasons for gradual packets drop could be link noise, packet processing rate, link capacity, and router buffer overflow when the congestion window size exceeds the pipe capacity. After 28 minutes of simulation, gradually increase curves of packets dropped maintained it state because of the SACK acknowledgements algorithm. The re-transmission count as shown in figure 4.2 is caused by the packet dropped and delay.

4.2 Multiple Reliable Connections Environment with UDP Traffic flows

The results of tests for multiple reliable connections environment workload with UDP (MRCEU) traffic flows are analysed as follows. In addition to the model discussed and analysed in the above section 4.1, two client host each established UDP connection. One of the client hosts is actively involved in video conference transaction and the other involved in VOIP communication as described in section 3.3. Figure 4.3 and 4.4 showed the results of baseline configuration.



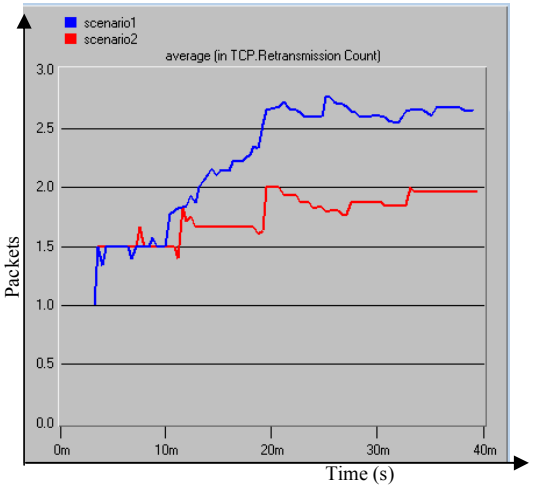


Figure 4.3: MRCEU Dropped Packets **Figure 4.4:** MRCEU Retransmission Count

The results pointed out the performance of TCP in a multiple TCP connections with UDP traffic connections involved.

Observing the graph pattern of how the packets are dropped at the router, it is obvious that the presence of UDP traffic interfered with TCP behaviour, inducing jitter and packet loss. The presence of UDP traffic flows in the multiple reliable connections environment caused an average packet loss difference of up to 8.56% and the packet dropped reached up to 11.9% compared to the behaviour reflected in figure 4.1 section 4.1. The results indicated that TCP and UDP contend for both buffers and bandwidth.

The pattern of the retransmission shows that UDP traffic is favoured and significantly lower TCP performance. This is because in a shared link, TCP suffers from significant loss. The number of packets that are processed by the forwarding processor and the forwarding rate are the parameters that significantly results to the retransmission behaviour.

4.2.1 Effect of the router buffer size on TCP behaviour using baseline configuration (ERTB)

Using the selected buffer sizes in table 3.9 in section 3.6, three simulation scenarios were tested. The modelled scenarios used the same test-bed environment. Each scenario used a different router buffer size. In scenario 1, 16mb of router buffer size was configured, 32mb configured in scenario2, and 64mb configured in scenario 3. The modelled scenarios used the same test-bed environment over the same data rate of 64,000bps. The objective of these simulations is to see the effect of buffer size on the performance of TCP and find the possible causes of the change in behaviour. The graph pattern of the simulation result in figure 4.5 showed variation in TCP behaviour.

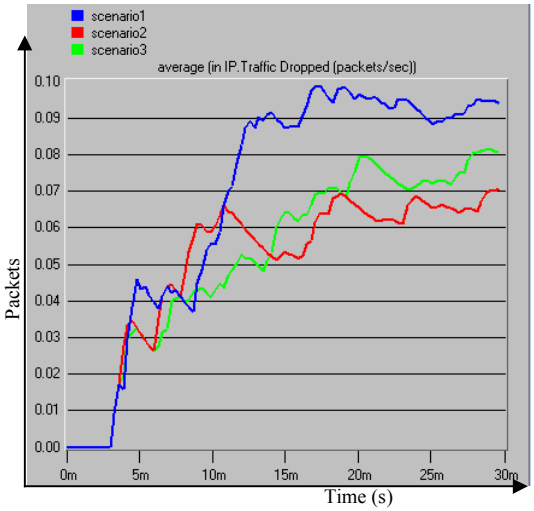


Figure 4.5: ERTB Dropped Packets

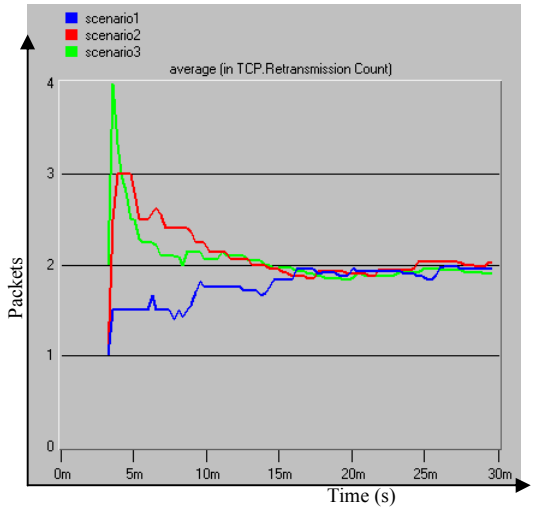


Figure 4.6: ERTB Retransmission Count

Owning to the fact that packets are dropped because of router buffer overflow and different router buffer were used on the assumption that it will reduce the rate of packet drop, notwithstanding the graph pattern in figure 4.5 shows only significant different of 3.5% compare to previous TCP

packet drop behaviour reflected in figure 4.3 in section 4.2. The possible causes of still having gradual increase of packet dropped are likely links capacity, the number of packets that are processed per second by the forwarding processor, and the forwarding rate per second. Because the presence of these parameters induces jitter, causing spurious retransmission when connection time for the retransmission expired.

Observing the pattern of the retransmission graph, within 1 to 10 minutes of simulation there was high retransmission in scenario 3 and 2. One of the possible causes of having high retransmission within that time is link noise. Since the packet dropped graph does not show high packets dropped, the mean that sent packets get loss along the path before reaching the router. When there is no ACK response from the, TCP sender assumed that the packet or the ACK is lost, and therefore retransmitted the same packet again until an ACK is received. The steady pattern of retransmission count graph after 10 minutes of simulation indicates the TCP adaptability, reacting to lack of ACK response from the receiver. However, it is observed there is change in the TCP behaviour when the three buffer sizes were tested. Hence, router buffer size significantly has effect on performance of TCP.

4.2.2 Effect of data rate on TCP behaviour using baseline configuration (EDTB)

Since link capacity is mentioned among problems causing poor performance of TCP, using the selected data rate in table 3.9 four

simulation scenarios were tested. Each scenario used a different wireless link data rate. In scenario 2, data rate of 64,000bps was configured, 128,000bps configured in scenario 3, 256,000bps configured in scenario 4, and 512,000bps configured in scenario 5. The modelled scenarios used the same test-bed environment over a router buffer size of 16mb. The objective of these simulations is to see the effect of data rate on the performance of TCP over the same router buffer size and find other possible causes of the change in behaviour. The graph pattern of the simulation result in figures 4.7 and 4.8 showed that different data rate significantly yield different packet dropped and retransmission behaviour.

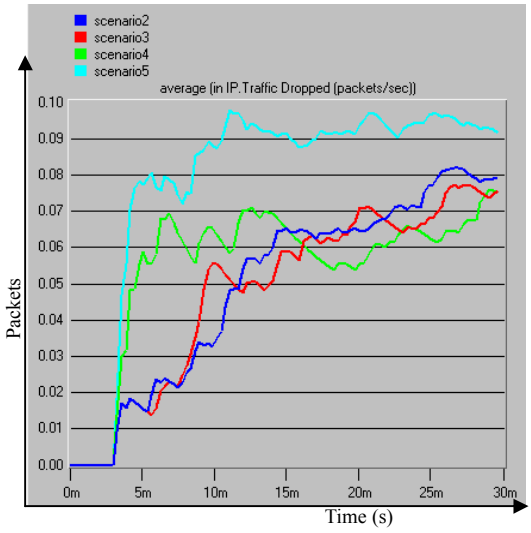


Figure 4.7: EDTB Dropped Packets

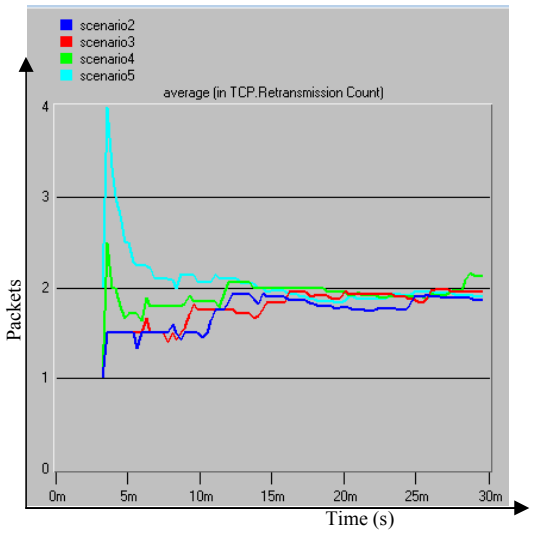


Figure4.8: EDTB Retransmission Count

With the router buffer size of 16mb, 64,000kps and 128,000kps offered better TCP performance. This is because the pattern of the graph in figure 4.7 indicated low packet drop for in 3-to-8 minutes of simulation. Within the same period of simulation, scenario 4 with data rate of 256,000 experiences significant packets dropped difference of 5.5% compared to scenario 2 and 3. But scenario 5 with data rate of 512,000bps experiences

6.5%. This means that router buffer size or data rate only cannot optimise the performance of TCP. Therefore, in the performance of TCP data rate and router buffer size are strongly related. After 10 minutes of simulation, the graph pattern of packet dropped gradually increased because of the TCP window size adjustment, adapting to available link resources. As explained in section 4.2.1 the graph pattern of retransmission count in figure 4.8 shows the presence of link noise. However, it is observed there is change in the TCP behaviour when the four different data rates were tested. Hence, data rate significantly has effect on performance of TCP.

4.2.3 Effect of data rate on TCP behaviour using different router buffer size (EDTDR)

The selected data rates and router buffer sizes shown in table 3.9 were tested the same emulated environment. For each of the router buffer size, the selected data rates were tested to observe among the simulation scenarios which give the best TCP performance. Table 4.1 summarises the data rate and router buffer size configurations for the simulations.

Table 4.1: Simulation configuration for **EDTDR**

Simulation1: over 16mb		Simulation2: over 32mb		Simulation3: over 64mb	
Scenarios	Data rate	Scenarios	Data rate	Scenarios	Data rate
1	64,000bps	1	64,000bps	1	64,000bps
2	128,000bps	2	128,000bps	2	128,000bps
3	256,000bps	3	256,000bps	3	256,000bps
4	512,000bps	4	512,000bps	4	512,000bps

After running the three simulations, the best scenario from each simulation were selected and simulated again. The best scenarios are scenarios1 from simualtion1, scenario2 from simulation2, and scenario3 from simulation3. The best scenarios were tested together to identify the degree of effect on the performance of TCP. Scenario2 with data rate of 128,000bps over 32mb of router buffer size reflected low packets dropped followed by scenario1 with data rate of 64,000 bps over 16mb with slight fluctuating packets dropped difference of between 0.1% to 1.0%.

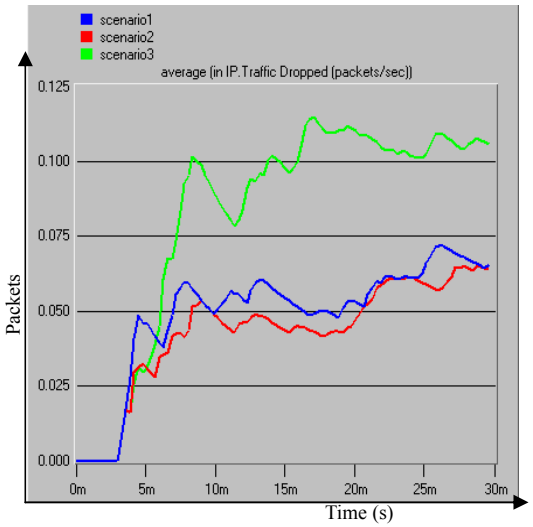


Figure 4.9: EDTDR Dropped Packets

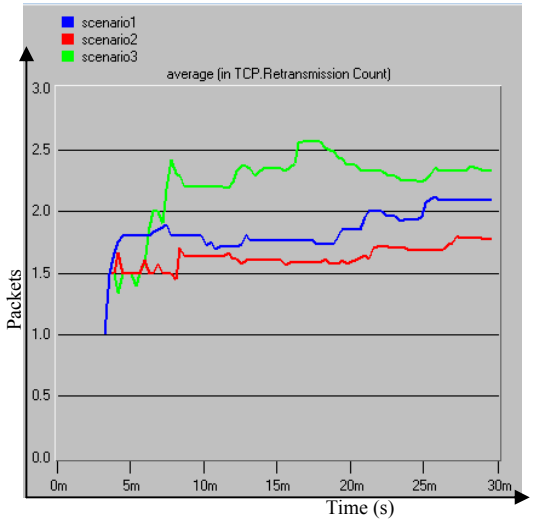


Figure 4.10: EDTDR Retransmission Count

The TCP retransmission behaviour appeared almost to have the same degree of behaviour with packets dropped as shown in figure 4.10. Scenario3 with data rate of 256,000bps over 64mb yielded worse performance.

The simulation for finding out the effect of selected different data rate and router buffer size on the performance of TCP in a multiple reliable in the presence of UDP traffic connection agrees with the finding in section 4.2.1 and 4.2.2. Hence, router buffer size and data rate are strongly

correlated in determining requirements for optimum performance of TCP in any king of network environment.

4.2.4 Effect of the TCP Technologies on TCP behaviour using baseline configuration (ETTB)

The selected TCP technologies discussed in section 2.4 were tested to see the effect of TCP technology on the performance and to find possible problems in the behaviour. In these simulation scenarios, the interest is to see how the TCP implementation affects the behaviour of the TCP performance in presence of UDP traffic connections.

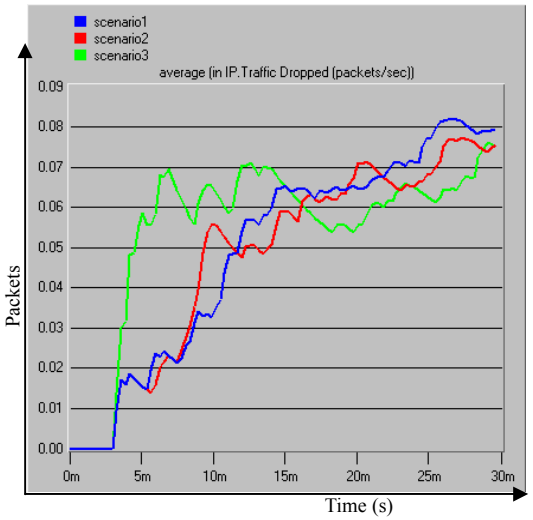


Figure 4.11: ETTB Dropped Packets

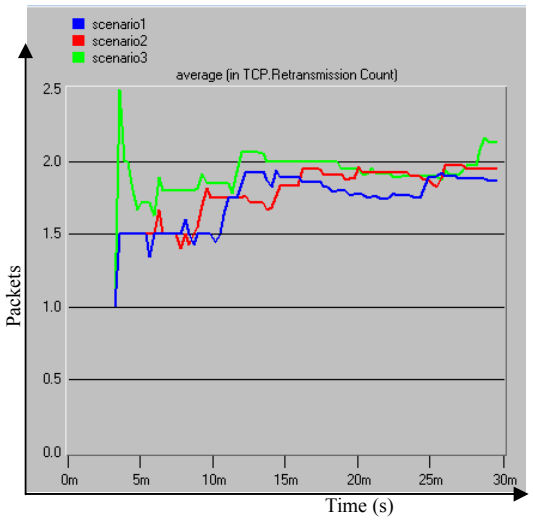


Figure 4.12: ETTB Retransmission Count

SACK, NewReno and Reno TCP technologies were configured on the same test-bed environment to see their effect on TCP behaviour. Four simulation scenarios were tested. Each scenario used a different TCP implementation. In scenario 1, TCP technology is SACK, NewReno configured in scenario 2, and Reno configured in scenario 3. The modelled scenarios used the same test-bed environment over a baseline router buffer size of 16mb and data rate of 64,000kps. The graph pattern of the packets dropped in figure 4.11

showed that NewReno and SACK technologies offered better TCP performance. There were slight packets drop within 3-to-10 minutes of simulation, after which the graph pattern of packets dropped gradually continued increasing. The possible causes of such behaviour could be strongly related to noisy links as observed in section 4.2.3 and 4.2.4. Overlapping radio channels, signal attenuation and additional noises they have significant impact on packet losses. This means that these parameters are enough to cause the TCP to degrade its performance. The results of the simulation reveal that TCP implementation over router buffer size of 8mb and data rate of 64,000bps strongly affect the performance of TCP in the emulated network environment.

4.2.5 Effect of the TCP Technologies on TCP behaviour using different router buffer size (ETTDB)

In theory, it is expected that when buffer size is increased, the expected result would be a significant increase on the TCP performance, because buffer overflow that leads to packet drop will decrease. The selected TCP technologies discussed in section 2.4 were tested on the same test-bed to see the effect of TCP technology on the performance using different router buffer size and to find possible causes of change in the behaviour. After running the three simulations with same data rate of 64,000bps, the best scenario from each simulation were selected and simulated again. Table 4.2 summarises the TCP implementation and router buffer size configurations for the simulations.

Table 4.2: Simulation configuration for **ETTDB**

Simulation1: for SACK		Simulation2: NewReno		Simulation3: Reno	
Scenarios	Router buffer size	Scenarios	Router buffer size	Scenarios	Router buffer size
1	8mb	1	8mb	1	8mb
2	16mb	2	16mb	2	16mb
3	32mb	3	32mb	3	32mb
4	64mb	4	64mb	4	64mb

The best scenarios are scenarios2 from simualtion1, scenario3 from simulation2, and scenario3 from simulation3. In these simulation scenarios, the interest is to see how the technology implementation adapts to buffer overflows and noise link in the presence of UDP traffic connection.

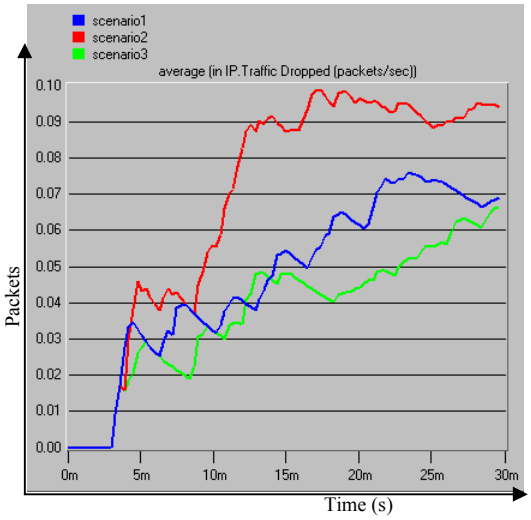


Figure 4.13: ETTDB Dropped Packets

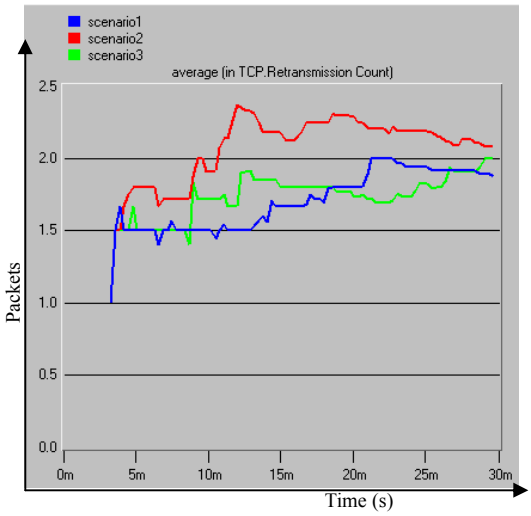


Figure 4.14: ETTDB Retransmission Count

The scenarios of the TCP technologies that offered best performance were selected, scenarios renamed and simulated again. The best selected scenario2 renamed as scenario1, scenario3 renamed as scenario2, and scenarios3 was not renamed. The graph pattern of the packets dropped in figure 4.13 showed that scenario3 using Reno implementation with 32mb offered better TCP performance followed by scenario1 using SACK

technology with 16mb. The gradual increase in packets dropped and retransmission as shown in figures 4.13 and 4.14 indicate the effect of overlapping radio channels, signal attenuation and additional noises. They have significant impact on packet losses and retransmission.

The graph pattern of retransmission count in figure 4.14 indicates that using SACK technology configured in scenario1, the TCP experienced the highest retransmission count throughout the simulation period. Within simulation time interval of 8-to-18 minutes and 18-to-27 minutes, scenarios 1 and 2 reflected switching states mode of retransmission count. But their retransmission count is better compared to scenario1 that uses SACK. Among the possible reasons for the switching behaviour is that whenever any of the TCP connections adjust its sending window size to avoid congestion that can lead to packets drop and retransmission. On the other hand, when the TCP proves the available path and discover is available to transmit more packets, more packets are then sent. Another reason is because it can use the whole bandwidth of the link while the sender of the other connection waits for the retransmission timer to expire. Finally, the results of the simulations reveal that TCP Implementation using different router buffer size and different data rate yielded different performance of the TCP in the emulated network environment. Hence, TCP implementations in relation to data rate and router buffer size strongly affect the performance of the TCP.

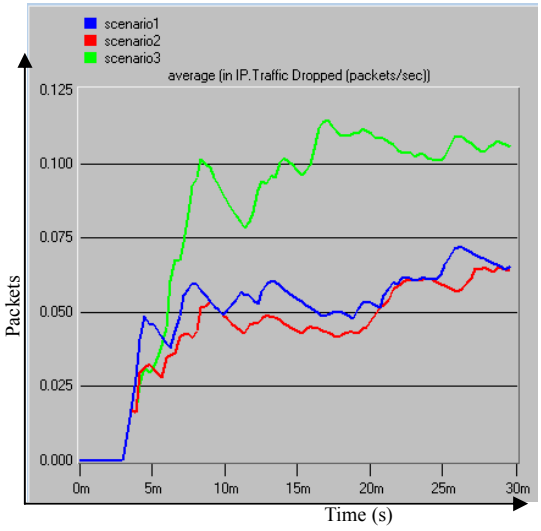
4.2.6 Effect of the TCP Technologies on TCP behaviour using different data rate (ETTDR)

The selected TCP technologies discussed in section 2.4 were tested on the same test-bed to see the effect of TCP technology on the performance of the TCP using different data rate over fixed router buffer size of 16mb, and to find possible causes of change in the behaviour. Table 4.3 summarises the TCP implementation and data rate configurations for the simulations.

Table 4.3: Simulation Configuration for **ETTDR**

Simulation1: for SACK		Simulation2: NewReno		Simulation3: Reno	
Scenarios	Data rate	Scenarios	Data rate	Scenarios	Data rate
1	64,000bps	1	64,000bps	1	64,000bps
2	128,000bps	2	128,000bps	2	128,000bps
3	256,000bps	3	256,000bps	3	256,000bps
4	512,000bps	4	512,000bps	4	512,000bps

After running the three simulations with same router buffer size of 16mb, the best scenario from each simulation were selected and simulated again. The best scenarios are scenarios2 using SACK implementation with data rate of 128,000bps from simulation1, scenario3 using NewReno implementation with 256,000bps from simulation2, and scenario3 using Reno implementation from simulation3. When simulating the best scenarios selected, scenario2 from simulation1 was renamed as scenario1, scenario3 from simulation2 renamed as scenario2, and scenarios3 from simulation3 left as scenario3. Figures 4.15 and 4.16 below show the results of the simulation of the best scenarios selected.



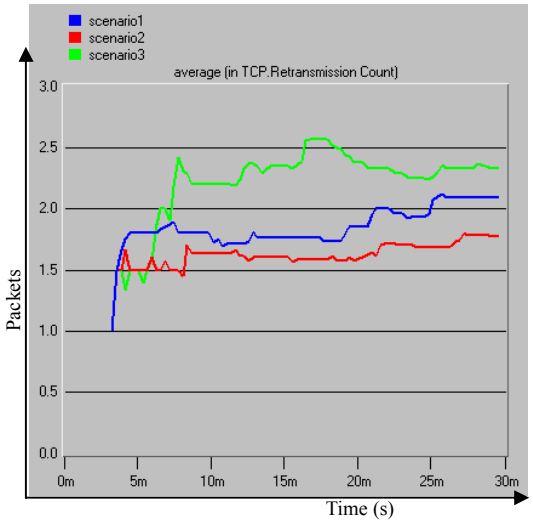


Figure 4.15: ETTDD Dropped Packets **Figure 4.16:** ETTDD Retransmission Count

When the best scenarios were simulated, NewReno technology (scenario2) with wireless link data rate of 258,000bps over a router buffer size of 16mb offered the best TCP performance followed by SACK (scenario1) technology with wireless link data rate of 128,000bps over the same router as shown in figure 9. The Reno technology with wireless link rate of 258,000 bps over the same router buffer size yielded the worse performance of the TCP.

The graph pattern of retransmission count in figure 4.16 indicates that using Reno technology configured in scenario1, the TCP experienced the highest retransmission count throughout the simulation period and SACK technology configured in scenario1 offered better performance of TCP followed by NewReno technology configured in scenario2. Among the possible reasons for having such patterns of retransmission is due to packets dropped at the router as a result of buffer overflow and probably less effect of link noisy. Finally, the results of the simulations reveal that TCP Implementation using the selected different data rate over the same

router buffer size of 16mb yielded different performance of the TCP in the emulated network environment.

4.3 Summary of the Various Scenarios

After analysing the behaviour of the baseline TCP in various scenarios certain problems were encountered in the TCP behaviour. In this section, the brief summary of the noticed problems during the analysis are discussed.

The variant of the TCP technology configuration in the test-bed environment does not prevent the retransmission after a spurious retransmission timeout and causes more unnecessary retransmissions. The variants can also cause an extra retransmission timeout if a retransmission following the spurious timeouts is lost, for example, due to router buffer overflow.

In the simulation discussed in section 4.2 that determines the performance of TCP in a multiple TCP connections environment when no UDP traffic flows are involved, it is obvious that the presence of UDP traffic interfered with TCP behaviour, inducing jitter and packet loss. The presence of UDP traffic flows in the multiple reliable connections environment caused an average increase packet loss difference of up to 8.56% on the TCP behaviour and the packet dropped reached up to 11.9%. The results indicated that TCP and UDP contend for both buffers and bandwidth. The pattern of the retransmission shows that UDP traffic is favoured and significantly lower TCP performance.

The performance of TCP behaviour with selected data rate in section 3.9 show that the router buffer size results in several packets dropped. The simulation discussed in section 4.2.3 reflected that Scenario2 with data rate of 128,000bps over 64mb of router buffer size reflected low packets dropped followed by scenario1 with data rate of 64,000 bps over 32mb with slight fluctuating packets dropped difference of between 0.1% to 1.0%. This increases the rate of retransmission when the data rate is high. A larger router buffer of allows a larger average congestion window, which results in a larger number of unnecessary retransmissions. Using the small router buffer increases the probability of having a retransmitted packet dropped when retransmitting after a retransmission timeout, which results in another retransmission timeout. Use of the medium router buffer size does not cause as low TCP performance as use of the small router buffer, because the larger router buffers avoid several packets drop problem which was present when using the small router buffer.

The graph pattern of the packets dropped in figure 4.13 in section 4.2 shows that scenario3 using Reno implementation with 32mb offered better TCP performance followed by scenario1 using SACK technology with 16mb. The gradual increase in packets dropped and retransmission indicated the effect of overlapping radio channels, signal attenuation and additional noises. They have significant impact on packet losses and retransmission. The graph pattern of retransmission count in figure 4.14 in section 4.2 indicates that using SACK technology configured in scenario1, the TCP experienced the highest retransmission count throughout the simulation

period. Within simulation time interval of 8-to-18 minutes and 18-to-27 minutes, scenarios 1 and 2 experience retransmission count switching states. Among the possible reasons for the switching behaviour is that whenever any of the TCP connections adjust its sending window size to avoid congestion that can lead to packets drop and retransmission. The results of the simulations run in section 4.2.3 reveal that TCP Implementation using different router buffer size and different data rate yielded different performance of the TCP in the emulated network environment.

CHAPTER FIVE

Summary, Conclusion and Recommendations

5.0 Introduction

This chapter provides an overall review of the results, discussions, implications, and limitations. It will begin with a analyzing and discussion on the research findings, followed by a discussion on the implications of the study.

5.1 Summary of the finding

The purpose of this research was to investigate the performance of TCP in a multiple reliable connections environment with UDP traffic flows. In addition, the study also aimed to help researchers who are interested in improving TCP or UDP performance by giving them an accurate insight about TCP behavior in a multiple TCP connections environment with UDP traffic flows. The research focused on studying TCP behavior in the emulated network environment, using simulation tool OPNET. Six kinds of traffic, three different router buffer sizes, three data rates, and three TCP implementation as discussed in chapter three were configured, simulated on the same emulation environment.

Based on the research work, the performance of the TCP in the emulation environment are not the same in the presence of UDP traffic flows using the selected test cases. SACK implementation with data rate of 128,000bps offered best performance of TCP followed by NewReno

technology with data rate of 258,000bps using the same router buffer size of 16mb. Reno platform with data rate of 258,000bps using the same buffer size offered worst performance.

From the analysis of results of the simulation, having then known that UDP flows do not have any congestion avoidance mechanisms and they do not slow down when their packets are dropped at the routers, the current TCP implementations in the presence of UDP is at a disadvantage in shared link. Unlike TCP, UDP protocol does not provide flow and congestion control, its only offer best effort policy. On the other hand, UDP can send data faster than TCP because there is no connection establishment, no connection state and unregulated send rate. Furthermore, in the presence of UDP flows, TCP probes the data path by increasing its congestion window until loss occurs. In the case of the emulated environment, the TCP connections back off and not use more bandwidth than what is left after the UDP has fetch enough bandwidth smooth flow of its traffics. In such network environment, multiple reliable connections are left with competition for the left over bandwidth. Hence, the performance of TCP becomes poor.

5.3 Contributions of the study

This study has attempted to contribute to the body of work on simulation, modeling, analyzing and improving TCP and UDP protocol. This research also help towards modeling a student's computer lab or internet café to an idea of what happen to applications under TCP transport

when any user of UDP application is allowed, because the user satisfaction is important in term of economic.

This study contributes in the development of a network services, and may form a useful simulation for utilization by other researchers in order to assess, study, and test the TCP behavior in presence of UDP. This research also provides guide to TCP and UDP researchers whom thinking about improving TCP or UDP performance. Since, the findings reveal that TCP behaves better under competing reliable connections with the presence of UDP

5.4 Conclusion

Accurate network bandwidth and resources are important to a variety of network applications. In spite of the recent sudden increase in accessibility of broadband Internet access, the best part of business organization, computer laboratory in learning environment, and public organization have fairly small-bandwidth links in comparison with the sites hosting beloved content. It is quite common for multiple users of the Internet to get connected at the same time, and running multiple networking applications including video and VOIP applications. Consequently, there are many circumstances in which the last hop link becomes a bottleneck resulting in congestion among user's applications. Based on the research work, different type of TCP connections can give different result depending on the network environment. In this of this work, using the selected test cases, TCP SACK under 128,000bps of data rate,

router buffer size of 16mb seems to be the best followed by NewReno under 258,000bps of data rate, and the same router buffer size. Besides, it is observed that the rate of UDP has the possibility to affect TCP send rates, depends on the UDP's rate. More higher the UDP's rate, more suppressed the TCP traffic will be at the shared bandwidth. Hence, the performance of the TCP becomes poor.

5.5 Recommendations

Future work for this thesis could be to have an insight into other TCP platforms protocols performance, other router buffer size, and data rate. Yet many factors such as impact of processing speed, forwarding rate, multi speed network, implementation of multiple profiles with multiple applications on different nodes, and impact of defective nodes on the network performance need to be considered to improve the performance of TCP.

References

Allman M., Paxson V., and Stevens W. (1999). *TCP Congestion Control*. Internet RFC 2581. Available: http://www.ietf.org/rfc/rfc2581.txt, accessed on 25/10/2010.

Allman M., Floyd S., and Partridge C. (2002). *Increasing TCP's Initial Window*. Internet RFC 3390. Available: http://www.ietf.org/rfc/rfc3390.txt, accessed on 29/10/2010.

Bacceli F. And Hong D. (2000). *TCP is Max-plus Linear and What it tells us on its Throughput*. Proceeding of SIGCOMM held at Stockholm, Sweden. August, pp. 219-224.
Available:http://conferences.sigcomm.org/sigcomm/2000/conf/paper/sigcomm2000-7-1.pdf, accessed on 10/09/2010.

Balakrishnan H, Padmanabhan N.V, Seshan S., and Katz H.R. (1997). *A Comparison of Mechanisms for Improving TCP Performance over Wireless Links*. Proceeding of IEEE/ACM Transactions on Networking held at San Diego, California. December, pp 756-769.
Available:www.cs.rutgers.edu/~badri/552dir/papers/TCP/Hari97.pdf, accessed on 29/08/2010.

Braden B., Clark D., Crowcroft J., Davie B., Deering S., Estrin D., Floyd S., Jacobson V., Minshall G., Partridge C., Peterson L., Ramakrishnan K., Shenker S., and Zhang J.L. (1998). *Recommendation on Queue Management and Congestion Avoidance in the Internet*, IETF RFC 2309. Available: http://tools.ietf.org/html/rfc2309, accessed on 29/10/2010.

Blanton E., Allman M., Fall K., and Wang L. (2003). *A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP*. RFC 3517. Available: http://www.ietf.org/rfc/rfc3517.txt, accessed on 29/08/2010.

Brakmo S.L, O'Malley W.S., and Peterson L.L.(1994). TCP Vegas: *New Techniques for Congestion Detection and Avoidance*. Proceeding of ACM SIGCOMM '94 held at Stanford, California. August, Pp. 158-181.
Available:http://www.cs.umd.edu/class/spring2010/cmsc711/vegas.pdf, accessed on 25/08/2010.

Chandra D., Harris R.J., and Shenoy N. (2003). *TCP Performance for Future IP-based Wireless Networks*. Proceeding of 3rd IASTED International Conference on Wireless and Optical Communications (WOC2003) held at Banff, Canada. July, Pp. 1-18.
Available:http://www.nssa.rit.edu/~ns/professional/pubs/2003/IASTED_383_52.pdf, accessed on 25/10/2010.

Dong Lin and Kung H.T (1998). *TCP Fast Recovery Strategies: Analysis and Improvement*. Proceeding of IEEE INFORCOM'98 held at San Francisco, California. April, Pp. 263-271.
Available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.7372pdf, accessed on 1/10/2010.

Floyd and Sally. (1994). *TCP and Successive Fast Retransmits, Technical Report*.Available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.44.7085.pdf, accessed on 1/10/2010.

Floyd S. and Fall K. (1999). *Promoting the Use of End-to-end Congestion Control in the Internet*. IEEE/ACM Transactions on Networking. Vol. 7, pp. 458-472.
Available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.6774.pdf, accessed on 1/10/2010.

Floyd S. (2000). *Congestion Control Principles*. RFC 2914.
Available:http://www.ietf.org/rfc/rfc2914.txt, accessed on 20/10/2010.

Floyd S., Handley M., Padhye J., and Widmer J. (2000). *Equation-based Congestion Control for Unicast Applications*. Proceeding of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication held at Stockholm, Sweden. October, pp. 43-56.
Available: http://web.cs.wpi.edu/~cs529/f04/slides/FHPW00.pdf, accessed on 01/09/2010.

Floyd S. and Henderson T. (1999). *The NewReno Modification to TCP's Fast Recovery Algorithm*. RFC 2582.
Available: http://www.ietf.org/rfc/rfc3390.txt, accessed on 20/10/2010.

Floyd S. and Fall K. (1996). *Simulation-based Comparisons of Tahoe, Reno, and SACK TCP*. Computer Communication Review, July, Vol. 26, No. 3, pp. 5-21.
Available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.3327.pdf, accessed on 20/10/2010.

Floyd S. and Jacobson V. (1993). *Random Early Detection Gateways for Congestion Avoidance*. IEEE/ACM Transactions on Networking, August, vol.1, No. 4, pp.397–413.
Available:http://lion.cs.uiuc.edu/courses/cs497hou/RED_jchang2.pdf, accessed on 15/10/2010.

Gurtov A. (2001). *Effect of Delays on TCP Performance*. Proceeding of IFIP Personal Wireless Communications (PWC'01) held at B.V. Deventer, Netherlands. August, pp 8-27.
Available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18pdf, accessed on 10/09/ 2010.

Gilbert H. (2002). *The ABC of IP Addressing*. AUERBACH, Baca Raton London, New York Washington, D.C.

International Network for the Availability of Scientific Publications (INASP: 2009), *Executive Summary: Optimising Internet Bandwidth in Developing Country Higher Education*.
Available:www.inasp.inf/uploaded/documents/BMO-chap2.pdf, accessed on 5/09/2010.

Jacobson V. (1988). *Congestion Avoidance and Control*, Proceeding of ACM SIGCOMM '88 held at Palo Alto, California. August, pp. 133-147. Available: http://ee.lbl.gov/papers/congavoid.pdf, accessed on 10/09/2010.

Jacobson V. (1990). *Modified TCP Congestion Avoidance Algorithm*. Proceeding of ACM SIGCOMM '90 held at Stanford, California. April, pp. 314– 329.
Available:http://eventos.spc.org.pe/jpc2007/MyReview/FILES/p37.pdf, accessed on 25/10/2010.

Jacobson V., Braden R., and Borman D. (1992). *TCP Extensions for High Performance*. Internet RFC 1323.
Available: http://www.ietf.org/rfc/rfc1323.txt, accessed on 10/09/2010.

Jamalipour A. (2003). The Wireless Mobile Internet: Architectures, Protocols and Services, Publisher: John Wiley & Sons.

Joe T., John H., and Katia O. (1996). *Analysis of HTTP Performance, USC/Information Sciences Institute*. Research Report. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.150.8953pdf, accessed on 10/09/2010.

Jorg W., Catherine B., and Jean-Yves L. B. (2002). *End-to-End Congestion Control for Flows with Variable Packets Size*. Technical Report.
Available:http://infoscience.epfl.ch/record/485/files/IC_TECH_REPORT_200282.pdf, accessed on 10/08/2010.

Kent C. and Mogul J. (1987). *Fragmentation Considered Harmful*. Proceeding of the ACM workshop on Frontiers in Computer Communications Technology held at New York, NY, USA. November, pp. 390-401.
Available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.5308.pdf, accessed on 25/10/2010.

Kojo M., Liljeberg K., Kiiskinen M., and Alanko J. (1997). *An Efficient Transport Service for Slow Wireless Links*. IEEE journal on Selected Areas in Communications, Vol. 15, No. 7., September.
Available:www.ieeexplore.ieee.org/iel3/49/13557/00622916.pdf?arnumber=622916, accessed on 20/09/2010.

Ludwig R. and Katz R.H. (2000). *The Eifel Algorithm: Making TCP Robust Against Spurious Retransmission*. Proceeding of SIGCOMM Computer Communication Review held at New York, NY, USA. January, pp. 51–64.
Available:http://iceberg.cs.berkeley.edu/papers/LudwigEifelAlg/eifel_algorithm.pdf, accessed on 28/11/2010

Luo K. and Fapojuwa A.O. (2003). *Impact of Terminal Mobility on TCP Congestion Control Performance.* Proceeding of 3rd IASTED International Conference on Wireless and Optical communications (WOC'03) held at Banff, Canada. July, pp. 360 – 365. Available:

Mathis M., Mahdavi J., Floyd S., and Romanow A. (1996). *TCP Selective Acknowledgment Options*. RFC 2018.
Available:http://www.ietf.org/rfc/rfc2018.txt, accessed on 10/08/2010.

Nagle J. (1984). *Congestion Control in IP/TCP Internetworks*. ACM *SIGCOMM* Computer Communication Review held at Palo Alto, California. June, pp. 3-5.
Available:http://www.sigcomm.org/ccr/archive/1995/jan95/ccr-9501-nagle84.pdf, accessed on 15/09/2010.

Ott T.J., Kemperman J.H.B., and Mathis M. (1999). *The Stationary Behavior of Idea TCP Congestion Avoidance*. Proceedings of IEEE INFOCOM'99 held at New York,US. August, pp. 65-71.
Available:http://www.teunisott.com/Papers/TCP_Paradigm/TCPwindow.pdf, accessed on 25/08/2010.

Paxson V. (1997). *End-to-End Internet Packet Dynamics*.
Proceedings of ACM SIGCOMM '97 held at Cannes, France. September, pp. 277-292.
Available: www.acm.org/sigcomm/sigcomm97/papers/p086.pdf, accessed on 01/08/2010.

Postel J (1981). *Transmission Control Protocol.* Internet RFC 793.
Available: http://tools.ietf.org/html/rfc793, accessed on 25/08/2010.

Ramani R. and Karandikar A. (2000). *Explicit congestion notification (ECN) in TCP over wireless network.* Proceeding of IEEE, ICPWC'2000 held at Bangalore, India. December, pp.495-499.
Available:http://dspace.library.iitb.ac.in/jspui/bitstream/10054/433/1/19560.pdf, accessed on 25/08/2010.

Stadler J.S and Gelman J. (1998). *Performance Enhancement for TCP/IP on a Satellite Channel*. Proceeding of IEEE MILCOM' 98 held at San Francisco, USA. December, pp. 19-25
Available:http://www.research.telcordia.com/society/TacCom/papers98/08_02i.pdf, accessed on 5/09/2010.

Stevens W. (1997). *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recover Algorithms*. RFC 2001.
Available:http://www.ietf.org/rfc/rfc2001.txt, accessed on 25/08/2010.