AN ANALYSIS OF SCTP IN BEST-EFFORT NETWORK WITH COMPETING FLOWS

Elbara Khalid Ali Waleed

UNIVERSITI UTARA MALAYSIA

2011

AN ANALYSIS OF SCTP IN BEST-EFFORT NETWORK WITH COMPETING FLOWS

A project submitted to the Dean of Research and Postgraduate Studies Office in partial

Fulfillment of the requirement for the degree of

Master of Science (Information Technology)

Universiti Utara Malaysia

By

Elbara Khalid Ali Waleed

(804049)

KOLEJ SASTERA DAN SAINS
(College of Arts and Sciences)
Universiti Utara Malaysia

PERAKUAN KERJA KERTAS PROJEK
(Certificate of Project Paper)

Saya, yang bertandatangan, memperakukan bahawa
(I, the undersigned, certifies that)

ELBARA KHALID ALI WALEED
(804049)

calon untuk Ijazah
(candidate for the degree of) **MSc. (Information Technology)**

telah mengemukakan kertas projek yang bertajuk
(has presented his/her project of the following title)

AN ANALYSIS OF SCTP IN BEST-EFFORT NETWORK WITH COMPETING FLOWS

seperti yang tercatat di muka surat tajuk dan kulit kertas projek
(as it appears on the title page and front cover of project)

bahawa kertas projek tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan.
(that this project is in acceptable form and content, and that a satisfactory knowledge of the field is covered by the project).

Nama Penyelia
(Name of Supervisors) : **MR. KHUZAIRI MOHD ZAINI**

Tandatangan
(Signature) : _____ Tarikh (Date) : 20/3/2011

Nama Penyelia
(Name of Supervisors) : **MR. ADIB M. MONZER HABBAL**

Tandatangan
(Signature) : _____ Tarikh (Date) : 20/3/2011

Nama Penilai
(Name of Evaluator) : **MR. ROSHIDI DIN**

Tandatangan
(Signature) : _____ Tarikh (Date) : 21/3/2011

PERMISSION TO USE

In presenting this project of the requirements for a Master of Science in Information and Communication Technology (MSc. IT) from Universit Utara Malaysia, I agree that the University library may make it freely available for inspection. I further agree that permission for copying of this project paper in any manner, in whole or in part, for scholarly purposes may be granted by my supervisor(s) or in their absence, by the Dean of the Graduate School. It is understood that any copying or publication or use of this project or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my project paper.

Request for permission to copy or make other use of materials in this project, in whole or in part, should be addressed to:

Dean of Research and Postgraduate Studies

College of Arts and Sciences

Universiti Utara Malaysia

06010 UUM Sintok

Kedah Darul Aman

Malaysia

ABSTRACT

This study focuses on Stream Control Transmission Protocol (SCTP), which is defined by IETF in RFC 4960 as a new transport protocol. SCTP features such as multi-homing and multi-streaming, has attracted multimedia applications to use it as their transport protocol instead of UDP and TCP. However, the challenge faced by SCTP is in a best-effort network, where the network does not provide any Quality of Service for the upper layer. In this study, a comprehensive performance evaluation of SCTP in the best-effort network in the presence of other traffic flows will be carried out. The objectives of this research are to measure the performance of both TCP and SCTP over a Wired Network in terms of delay, jitter, and throughput in a network environment that has STCP with UDP traffic, and then compare SCTP and TCP performance results in terms of these performance metrics. All experiments conducted in this research were obtained through network simulation tools, i.e. NS 2. It is expected that the results obtained will become useful for future researchers in improving SCTP.

ACKNOWLEDGMENTS

Firstly, praise to Allah SWT for guiding and blessing me with perseverance and strength to complete this project. Secondly, I would like to thank my parents, especially my father, may Allah bless his soul, forgive him and admit him into paradise. Also, my appreciation goes to my mother, siblings and friends for their support throughout my program, may Allah reward all of you abundantly. In addition, my gratitude goes to my Supervisor, En. Khuzairi Mohd Zaini, and my Co-Supervisor, En. Adib M.Monzer Habbal, for guidance, support and constructive criticism towards the completion of this project, may Allah help you all. Lastly, I am indebted if I do not appreciate everybody that took part in the completion of this project, especially the University Utara Malaysian administration, staff and students, May Allah continue to bless all of you, ameen.

Elbara Khalid Ali Waleed

February 2011

TABLE OF CONTENTS

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER STUDY

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

LAN	Local Area Network
WAN	Wide Area Network
HSTCP	High Speed Transmission Control Protocol
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol over Internet Protocol
FTP	File Transfer Protocol
HTTP	Hyper-Text Transfer Protocol
UDP	User Datagram Protocol
TFRC	TCP Friendly Rate Protocol
SCTP	Stream Control Transmission Protocol
MTU	Maximum Transmission Unit
OSI	Open Systems Interconnection
Ns-2	Network Simulator 2
WLAN	Wireless Local Area Network
VOIP	Voice Over IP
Ipdv	IP delay variation
RTEMS	Real-Time Executive for Multiprocessor System
RTO	Recovery Time Objective
MANET	Mobile Ad hoc Network
CWA	Congestion Window Action
CBR	Constant Bit Rate
PC	Personal Computer

CHAPTER ONE

INTRODUCTION

1.1 Introduction

A network is a link between two or more devices such as computers, telephones or anything else for communication to achieve the work that has to do with exchange of files in the case of the sharing of one printer between many computers. Actually, there are two basic types of networks; firstly, Local Area Network (LAN), where many devices link with each other in a limited environment such as a school, building, or even lab. Secondly, a Wide Area Network (WAN), which is also a device link between each other, but in a larger area or environment such as Kedah, Kuala Lumpur or even the world. In addition, there are many ways for arranging networks, which is called network topology. There two types of topology; logical and physical topology. The logical topology describes how the network information flows through the network while the physical topology is concerned with where the access points are placed for the computer layout. The topology classification is regarded as point-to-point, star, link, bus, ring, tree, and mesh. This study will focus on one of the stated classifications.

Indeed, the network has rules and conventions for communicating between devices, which are called network protocols. Any layer from the network standard layers (physical, data link, network, transport, session, presentation and application) has special protocols that are mentioned as the link between two devices and keep the network in good shape. For example, High-Speed Transmission Control Protocol (HSTCP) has

recently been proposed as a mechanism for TCP Congestion Control to achieve high performance at a reasonable speed in the wide area links (De Souza and Agarwal 2003). HSTCP is a crowded protocol control that has to change or modify the mechanism of action protocol congestion in Transmission Control Protocol (TCP), and it assists the protocol TCP to overcome the congestion in the network during the data transferring process. This is one of the more famous protocols among TCP/IP, FTP, HTTP and etc. Those protocols are used in Internet, intranet or WAN as wireless, or through cables or wires, which depends on the kind of network that is needed with its characteristics as mentioned in the HSTCP example above. TCP is the set of rules used with Internet protocol (IP) to send the data in the form of units of messages between computers over the Internet. While the (IP) physically cares about delivery of data and the (TCP) takes care of tracking each unit of data when calling packages that are divided into messages, which is not the fastest protocol for sending and receiving, but it is reliable. On the other hand, there is a protocol that is unreliable, but faster in sending messages. This is known as User Datagram Protocol (UDP). The UDP is a member of the Internet protocols (IP) that uses IP as well as TCP. UDP is a communication protocol that supports the network for sending messages between devices. But, its main function is limited to the protocol as mentioned in carrying the message to the receiver or destination unlike TCP. In addition, there is a protocol related to TCP, which is TCP Friendly Rate Control (TFRC) (Floyd et al. 2008). This is a congestion control mechanism to get a best-effort Internet environment for multicasting flows of operation. In comparing TFRC with TCP, it has lower throughput variation over time, which makes it suitable for applications.

This study will show the performance of one of these protocols, Stream Control Transmission Protocol (SCTP). It is a reliable transporting protocol that facilitates the transfer of data between two points of contact with a connection to the Internet protocol (IP) in the network. SCTP supports carrying data in multiple streams, keeps the message boundaries, and does multi-homing. At the same time, it has many advantages like unduplicated transfer of data without error, fragmentation of data to conform to the maximum transmission unit (MTU) size, and optional bundling of user messages into an SCTP packet (Rane, Kumbhar, and Sovani 2002). Furthermore, SCTP supports congestion control algorithms and error handling (McClellan 2003), and in the case of multimedia, is better than UDP or TCP (Leu, and Ko, 2008). As a result, SCTP has many features that help it become an important protocol. This study will simplify its performance and make it easy for the users.

As mentioned above, the transporting layer is one of the standard OSI network layers. It is a layer for transferring data in a network between computers with other devices of application and have an interest in the protocols.

The analysis will be in a best-effort network; the best-effort network refers to the non-difference between the network activities for dealing with services that come through the network itself. In this study, a proposed protocol is being evaluated as a proposal for improving the evaluation process, which aims to evaluate the performance by using this protocol. The main objective is to measure the performance of SCTP through the network simulation. The idea is to use network simulation NS2 in the application and measure the performance and effectiveness of SCTP in the wired environment.

1.1 Problem Statement

SCTP has been attracting the attention of several researchers due to its appealing features. Many studies on SCTP have been carried out under different environments in order to measure its performance under various conditions. According to Scharf and Kiesel (2006), they studied the performance of SCTP with emphasis on the delay in wireless networks. In the same light, Boussen, Tabbane and Tabbane (2009) carried out experiments measuring the performance of data traffic, video streaming and other application layer protocols over SCTP in a Wireless Local Area Network (WLAN). Further studies in the wireless environment, with special emphasis on throughput, have been carried out (Ma et al. 2005). Islam and Kara (2006) studied the throughput of the SCTP protocol in a multi-homing setup. It was found that a small number of SCTP streams, or SCTP in unordered mode, avoids head-of-line blocking as opposed to the transaction-based signaling applications over TCP (Kaytan, 2010). Irrespective of the large number of studies carried out on the performance of SCTP under various conditions, its performance in a best-effort network have not been explored enough at this time. Hence this study aims to investigate the performance of SCTP in best-effort networks with special emphasis on throughput, delay and delay jitter. The main focus of this study will be to compare the performance of SCTP with that of TCP in a best-effort network under varying network conditions.

1.2 Research Questions

It is envisioned that this research will answer the questions below.

1- What is the performance of delay, jitter, and throughput for SCTP in a Wired Network?
2- What is the performance of delay, jitter, and throughput for TCP in a Wired Network?
3- What is the difference in performance between SCTP and TCP in a Wired Network in terms of delay, jitter, and throughput?

1.3 Objectives

The research objectives must be clear to ensure the track of the project. This study will achieve the following objectives:

- To analyze the performance of delay, jitter, and throughput for SCTP over a Wired Network environment
- To analyze the performance of delay, jitter, and throughput for TCP over a Wired Network environment
- To find the best-effort performance for the SCTP and TCP network.

1.4 Scope of the study

This study is limited to wired network performance in network simulation NS2 by covering the performance of SCTP protocol. Indeed, this study going to experiments the SCTP by creating the network topology through the network simulation NS2 code by

connecting senders and receivers PCs through the network using SCTP protocol linked by wired network, which will be comparatively studied on throughput. Packet delay and jitter will be measured as well.

1.5 Significance of the Study

1- This study shows the performance of SCTP and TCP in the best-effort network with competing flows.

2- The study is helpful for application developers in building their applications network using SCTP or TCP protocol, when the study measured their performance in a best-effort network.

3- This study will be available as reference for further research. Hence the results of this study could be used as a base for conducting further studies in media streaming by using SCTP or TCP connections.

1.6 Summary

This chapter provides the introduction for the study based on the competing performance of SCTP through the best-effort environment for jitter, throughput, and delay as mentioned in the abstract and introduction. The chapter also discusses the network topology and measuring it through network simulation NS2 as was shown in the problem statement of this research. The research questions and research objectives were also tested using Network Simulation tools.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter presents an overview of previous work on related topics for the purpose of providing the necessary background of this study. A literature review is a very important part of research as it involves the study of information found in the research that has been done by other researchers, and which is related to the selected area of study. The literature review is a collection of ideas from the research that has been done previously by different researchers on network simulation. The purpose is to present findings by identifying the gaps or areas of controversy, and to identify the strengths and weaknesses of the currently published works. Various important concepts relating to performance metrics will be reviewed for the purpose of obtaining clear and useful information from previous research in the SCTP and TCP network field.

2.2 SCTP protocol

2.2.1 Definition of SCTP

Stream Control Transmission Protocol (SCTP) is a reliable transporting protocol that facilitates the transfer of data between two points of contact by connecting to the Internet protocol (IP) in the network (Muller, 1999). SCTP supports carrying of data in multiple streams, keeps the message boundaries, and uses multi-homing. At the same time it has many advantages, such as unduplicated transfer of data, without error, fragmentation of

data to conform to the maximum transmission unit (MTU) size; and optional bundling of user messages into an SCTP packet (Rane et al., 2002).

2.2.2 SCTP Architecture

In fact, SCTP is related to IP, which is located between the connectionless packet network service and the SCTP user application. However, SCTP works through groups or associations. It enables each endpoint of the group or an association that provides the other endpoint with multiple IP addresses in combination with SCTP ports. Thus, the endpoints of the group or an association communicate with each other using addresses. In addition, the addresses serve as the point of origination for the SCTP packets as shown in Figure 2.1. The group or association spans across almost entire possible sources or destination combinations, which may be generated from each endpoint's lists.

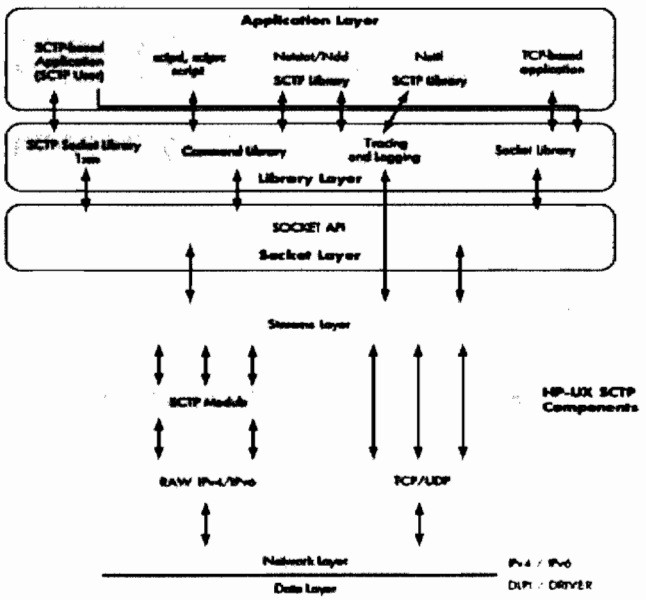

Figure 2.1: SCTP Architecture

2.2.3 SCTP Features

SCTP protocol has many features different than TCP and UDP as shown in the table below. As usual, the comparison between the protocols in the same layer measures the performance between them in order to become clear for using. Actually, the SCTP protocol has become famous for some special features it possesses; multi-streaming and multi-homing. Table 2.1 shows the different features of SCTP, TCP and UDP.

Table 2.1: SCTP features (Amer., and Stewart., 2005)

Services/Features	SCTP	TCP	UDP
Connection-oriented	yes	yes	no
Full duplex	yes	yes	yes
Reliable data transfer	yes	yes	no
Partial-reliable data transfer	optional	no	no
Ordered data delivery	yes	yes	no
Unordered data delivery	yes	no	yes
Flow control	yes	yes	no
Congestion control	yes	yes	no
ECN capable	yes	yes	no
Selective ACKs	yes	optional	no
Preservation of message boundaries	yes	no	yes
Path MTU discovery	yes	yes	no
Application PDU fragmentation	yes	yes	no
Application PDU bundling	yes	yes	no
Multistreaming	yes	no	no
Multihoming	yes	no	no
Protection against SYN flooding attacks	yes	no	n/a
Allows half-closed connections	no	yes	n/a
Reachability check	yes	yes	no
Psuedo-header for checksum	no (uses vtags)	yes	yes
Time wait state	for vtags	for 4-tuple	n/a

2.2.4 Usage of SCTP

SCTP has two important features as mentioned previously, they are multi-homing and multi-streaming. These are distinguished from TCP and UDP as well as its usage features. Multi-homing supports two sets of IP addresses and ports that show that all the IP addresses will be involved in operations of sending and receiving data. The network that has two hosts is regarded as multi-homed (Stewart et al., 2008). On the other side, SCTP can send data by selecting just one address from the Alternate addresses and send; this function will provide a type of flexibility in face outages and network losses. As for multi-streaming, actually SCTP protocol functionality is the same as the TCP protocol, but SCTP is supporting multiple streams in its functionality when it makes SCTP better than TCP for transferring multimedia data. In other words, all the streams are independent within an association, but are related to the association in SCTP. Each stream gives a particular number that is encoded in the SCTP protocol through the association. Therefore, SCTP allows sending and receiving of packets by exchanging these packets among multiple streamings. This study will show the functionality of SCTP (Chaeng et al., 2009).

From Table 2.1, the comparison of SCTP with TCP, and UDP, SCTP possesses more features than another protocol, which means that it is useful in any network environment. This project will adopt the best-effort network by measuring metrics performance of SCTP itself as well as SCTP and TCP in order to get the SCTP behavior in the best-effort network and compare the delay, jitter and throughput results with another traffic protocol.

2.3 TCP protocol

2.3.1 Definition of TCP

Transmission Control Protocol (TCP) is the set of rules used with Internet protocol (IP) to send the data in the form of units of messages between computers over the Internet. While the (IP) physically cares about delivery of data, TCP takes care of tracking each unit of data together with calling packages that are divided into a message. Moreover, it can help sending and receiving through the network. It is slow in sending and receiving data, but it is a reliable protocol.

2.3.2 TCP Architecture

Transmission Control Protocol (TCP) is a transport protocol like UDP and SCTP with different functions and characteristics but is classified a transport layer in OSI models and each of them is working through a specific layer. Therefore, TCP has four layers; they are: Application – Transport – Internet - Network.

TCP layers handle the data by passing the stack from an application layer to the physical network layer as shown in Figure 2.2. Each layer will control information in the stack to ensure delivery data, which is controlled by the information name header (Hunt, 2002). The data will be sent and received from each header to another and the delivery data in this process is called encapsulation. The data is exchanged between the transport layer and the network layer through the internet layer, which is the procedure for processing data, and then is received by the application layer.

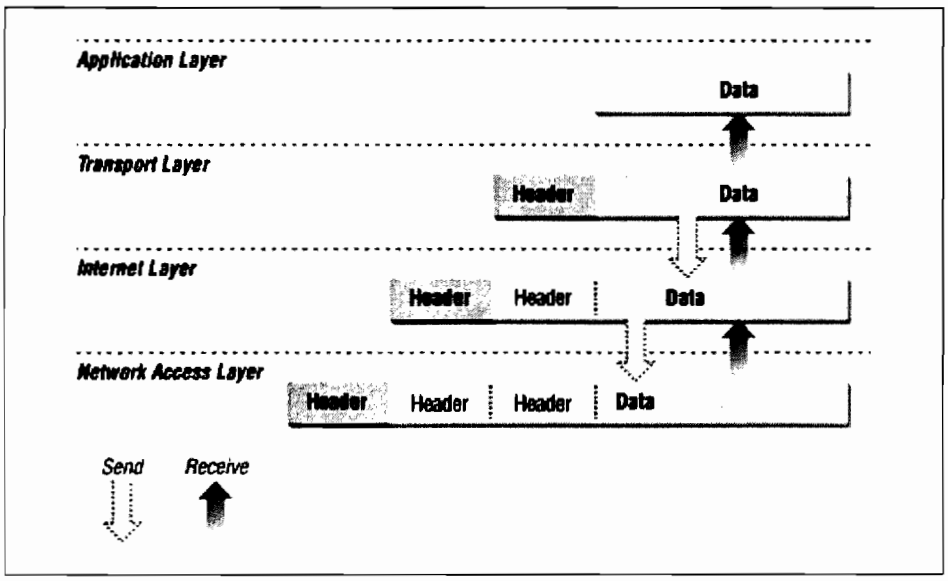

Figure 2.2 TCP Data Encapsulation

In the TCP layers, each of them has its own type of data structure. If comparing this structure between TCP and UDP the difference between them can be seem as shown in Figure 2.3. Each layer has its function within TCP, showing that it is reliable and is a supporting stream as well. In UDP each layer has its usages in terms of streaming and reliability for the transport protocol SCTP, while they have many similar behaviors. Figure 2.3 shows how the data will move through the layers in TCP and UDP as well.

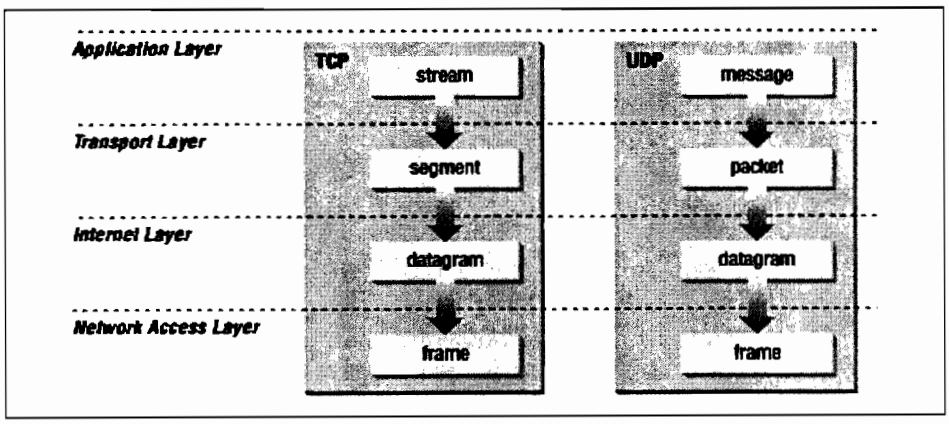

Figure 2.3 TCP Data Structure

2.3.3 TCP Features

TCP has become a famous protocol that is used in the Internet and intranets. In addition, it has some features that work for transferring of data, which are stated below:

1- Reliable Communication

In a TCP connection the stream of the data sent from the sender will be reliably delivered (Del Rey, 1981).

2- Connection Opening

TCP builds the initial fragment to open the connection between the sender and receiver and then the devices will exchange the IP addresses and port number to control the sequencing and data flow.

3- Error Detection

In the receiving operation, the receiver will test the integrity of the data segment that are incoming; if the data is invalid the receiver will return it and send an error message to the sender, so in this case the network will be safe from corrupted data.

4- Connection Closing

In TCP sending and receiving operations, when the sender indicates that the connection will be terminated, the TCP sender will send segments to declare this to the receiver. There will not be more data sent, and the connection will be closed (TinkQuest, 1999).

In addition, TCP has more features as shown in Table 2.1, which compares it with UDP and SCTP, so these features have already been mentioned, with an explanation of their importance, which will help this study to obtain proper results.

2.4. Performance Metrics

The performance metrics are factors that are used to evaluate the performance for something specific, or compare the performance between different systems (Deru and Torcellini, 2005). These metrics will provide information about how the study will evaluate the performance of SCTP over the wired network. In the Best-Effort exchange, the data between a server using SCTP or with TCP, will provide information to compare the results between two or more different systems. In this research, the performance metrics will be used for measuring the results of SCTP and TCP in terms of delay, jitter and throughput. On the other hand, there are other metrics for measuring protocol performance for variables such as fairness. However, this study concentrates only on the metrics that have been mentioned.

2.4.1 Packet loss

Packet loss is an error that can occur in data network when the data in the network is congested. This means that packets of the data are not able to be transmitted as well as in a normal case, or they fail to reach the destination. Therefore, this can cause significant problems in systems that use Voice over Internet Protocol (VOIP) as an application. In addition, packet loss is an important parameter affecting the performance of the networks.

Therefore, from Equation 2.1 we can obtain the percentage of the number of packets lost to the total number of packets sent (Boussen et al., 2009) as is shown below.

$$\text{Packet Loss Rate} = \frac{\text{Number of dropped data packets}}{\text{Total number of data packets sent}} \text{x } 100 \rightarrow (2.1)$$

The packet loss can be caused by many factors; firstly, such as the signal degrading. Secondly, hardware problems, which means faulty network hardware. Thirdly, networks that are in demand more than necessary, and that creates a state of congestion, which makes the network susceptible to packet loss in sending or receiving.

Packet loss is one of the important performance metrics usages for measuring performance. Therefore, there are many of the studies done using packet loss to measure performance protocols. In one of the studies that talk about loss detection for singling traffic in SCTP, the research proposed, evaluated and modified a management algorithm. The modification was put into the algorithm to maintain the timer in a correct state at all times, and provide quick loss detection. In the evaluation the study used Iksctp implementation and showed that the algorithm could reduce loss time significantly, while packet loss fell as much as 43% (Hurtig et al., 2008). This study also compared SCTP and TCP and UDP.

2.4.2. Delay

The packet delay is defined by RFC 3393 as (ipdv) an IP delay variation, which is two kinds of packets surrounded by a stream of packets that is defined for a chosen message. The packets in the stream will be going from measurement point 1 MP to measurement point 2 MP. This (ipdv) is the difference between the one-way-delay of the selection packets (Demichelis and Chimento., 2002). In other words, delay in the network is the time or period for data traveling through the network from the source to the destination or from an endpoint to another, which specifies how long the data will take to arrive (Amer and Stewart, 2005). The packet delay can be seen from inside the packet stream (Mohammed., 2010). More specifically, it can be described as the period for encryption and decryption in sending and receiving processed data between different endpoints.

Packet delay, or delay, is another performance metric that this research will focus on, because measuring of the packet delay of performance of SCTP, or with SCTP working with TCP, appears with the time of data delay sent in a best-effort network. The proportion of importance of packet delay in the network is done in many surveys to measure the delay of sending data in the network. For example, there is a study for improving the efficiency of delay-sensitive transmissions in SCTP in terms of multimedia data such as MPEG-4 traffic (Wang et al., 2009). The evaluation performance in this work in SCTP extension for MPEG-4 video streaming is done by exposing and exploiting the one-way delay, which allows the system to retransmit lost packets selectively, depending on whether the lost packet would arrive at the scheduled time (Wang et al., 2009) .

2.4.3 Jitter

Jitter is the measuring of the variation over time for the packet delay through sending and receiving of the data signal across the network. This can happen unexpectedly, where packets are sent over the network and are expected to be connected to the recipient at a particular time. Sometimes there were problems in the propagation delay of some seconds, and allowing packets to be sent by the first access, which means that there is synchronization in transmission or that defects occurred in the network itself (Almes et al., 1999). This will eventually make the data change the direction of the traffic channel in the delayed access.

Jitter is one of the performance metrics that this study will measure using SCTP and SCTP with TCP in the best-effort network. There are many studies about jitter as a measurement of Multimedia Streaming in a Wireless Environment through Performance Evaluation of Fast TCP and TCP Westwood. This study will measure the performance of fast TCP and TCP Westwood in terms of fairness, throughput, as well as jitter, while the results that appeared for the fast TCP in all cases of streaming gave a good throughput. In the case of jitter, the TCP Westwood was better than fast TCP in this environment (Rahim and Faisal-Hasan., 2009).

2.4.4 Throughput

Data throughput is the highest quantity of data that can be transported from a sender to a receiver. Nevertheless, the description and measurement of throughput are complicated in defining a satisfactory level of quality (Bradner and McQuaid., 1999). In other words, it is the amount of data that has been transferred from one location to another by sending from the server to the client within a given time frame. Throughput is used for measuring hardware, software, as well as networks. This work will focus on the significance of throughput improvement by simulating a transfer of new data through multiple paths to the destination or receiver (Demichelis and Chimento, 2002).

This study will show the throughput for SCTP connected with SCTP and TCP connected with SCTP in the best-effort environment. This will be done by analyzing the throughput through the handovers in mobile IP that measure the improvement of the effect of SCTP. This study shows that the SCTP compared with TCP-Reno and TCP-Selective Acknowledgment (SACK), and has better performance according to the SCTP, which can support an unlimited number of Selective Acknowledgment (SACK) blocks. Moreover, the study also shows SCTP, in terms of a bottleneck link that is low in bandwidth, and can be used to improve the end-to-end throughput (Fu and Atiquzzaman, 2003).

2.5. Related Work

SCTP and TCP are famous protocols in standard OSI transport layers, which leads us to focus on the measurement of their performance in the best-effort network. Therefore, it is imperative to view the previous studies about them.

2.5.1 Review of SCTP

There are several studies about SCTP which focus on several network environments.

2.5.1.1 SCTP in Different Environments

There is a study that talks about the implementation in an open source real-time operating system. The objective of this study is to describe the challenges and issues when implementing SCTP in a Real-Time Executive for Multiprocessor Systems (RTEMS) operating system (Corsepius et al., 2010). The result of this study describes the results of the progress in implementing SCTP steps in RTEMS when using the UK-DMC satellite environment in a real-time operating system (Rahman et al., 2008).

In the wireless environment, there are many studies that focused on measuring the performance of SCTP. This study proposes a new advantage and smooth handover for data sending and receiving and investigated the recent issues about SCTP mechanisms by identifying communication paths in connection between a source and destination address. Therefore, the study shows a new algorithm in the transport layer handover called SmSCTP. In case the mobile node has one WNIC, this will enhance the performance

connection through the handover. Secondly, it has been shown that the SmSCTP can distinguish between the communication paths of sender and receiver addresses. In case the normal SCTP distinguishes the communication paths based on the receiver's address, the SmSCTP can reply to modify sender's address without delay in order to achieve a smooth handover procedure without suffering unnecessary packet retransmission (Honda et al., 2007). Some wireless networks give two ways to employ CS-SCTP and CS-SCTP, which is a collaborative approach to improve the wireless environment during performance to a secure SCTP. The methods of partial authentication and two-level key encryption are to reduce the performance degradation in order to reduce wireless packet loss (Yang et al., 2007). WLAN showed that the current SCTP implementation behaves in a WLAN environment by an intuitive way by allowing more time for network conditions degrade in the performance switchover (Fallon et al., 2008).

There is another performance of SCTP in an ad-hoc environment that leads this study to propose a dynamic-delayed acknowledgement feature that is quite suitable, and adds a fixed Recovery Time Object (RTO) (Dorion, 2006) as a mechanism for a Mobile Ad hoc Network (MANET) (Valera et al., 2003). The original SCTP enhances the performance of highly reliable end-to-end communication in the ad hoc network environment. This study that confirms the dynamic-delayed acknowledgement features are suitable for MANET and fixed RTO (Takemoto et al., 2007).

The study by Boussen et al. (2009) shows that the SCTP performance in a Wifi network is better than TCP and UDP by reducing the latency and enhancement in the throughput by comparing data traffic and real-time video streaming and VoIP applications. In

addition, the study shows that SCTP has all of the TCP and UDP features, which make the SCTP perform better than TCP and UDP.

In the high-speed environment over the wireless network, it was shown that the proposed mechanism could successfully alleviate the effects of radio channel errors for SCTP performance and improve STCP in wireless environments (Cheng et al., 2009). Moreover, the multi-streaming function shows the advantages of multi-streaming over single streams in some instances. The SCTP single stream has different congestion control and retransmission mechanisms than the TCP protocol (Funasaka et al., 2010). As a result, SCTP is a study that describes the features provided by SCTP and gives a glimpse of the many ways that application can control and configure networks (Stewart et al., 2008).

2.5.1.2 SCTP in performance metrics

The performance metrics mentioned above, delay, jitter and throughput, are supported by other types of performance metrics. Therefore, many works have been done on SCTP in other environments.

The high-speed wide area network shows a throughput of 1.7 Mbps compared to regular SCTP, which has a throughput of 0. 9 Mbps (Nagamalai and Lee., 2004). In the Internet environment, modified SCTP is capable of preventing congestion collapse and improving the fairness of bandwidth allocation. The SCTP retransmission procedure makes better utilization of the network bandwidth by reducing unnecessary multiple retransmissions of any lost packet (Al-Kaisan et al., 2007). Throughput and jitter in heterogeneous

(wired/wireless) wired and wireless environments show new reference frameworks for the development of both throughput, and jitter as well, in SCTP network applications (Emma et al., 2006).

2.5.2 Review of TCP

There are several studies have been done on TCP protocol in different network environments.

2.5.2.1 TCP in Different Environments

Zhao et al (2007) did a study in high-speed networks by evaluating the speed of TCP traffic through high-speed applications to facilitate feeding traffic over fast access links on buffers supplied at mid-point routers. This shows the intention of the speed to be very effective in dropping core buffer supplies. In another type of environment, the study proposes to improve TCP performance through the wireless network with the aim of explicit window adaptation algorithms so the result will enhance the significance of TCP performance over wireless networks with the algorithm that has already been proposed. The results that appeared after the signal figure for the feedback values to TCP sources changed the field of AWND when it was approved by TCP ACKs (Byun and Lim., 2005). Measuring TCP performance over a wireless network environment, which discussed the TCP characteristics of different wireless networks, shows how it can negatively interact with TCP mechanisms. Moreover, evaluation of various TCP performances in the link or transport layers improved TCP protocol over these layers (Xylomenos et al., 2001).

2.5.2.2 TCP in Performance Metrics

In terms of the performance metrics, the study about TCP shows enhancement in average throughput over TCP when the algorithm proposed in this study reduced the congestion size by calculating the number of dropped packets. The new TCP congestion window action (CWA) for transmission error based on delay in a congestion window is a liable loss per window (Alnuem et al., 2008). In another type of performance, there is a study that controlled the size of the link used to communicate with the source and destination; the result shows that there is no packet loss, and the throughput is increasing while the parallel connections are rising to control this amount of connections (Han et al., 2010).

2.6. Summary

This chapter has provided the required background information about SCTP in Section 2.2 and TCP in Section 2.3. The performance metrics were explained in detail in Section 2.4. Furthermore, Section 2.5 presented the related studies performed by earlier researchers on SCTP and TCP.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1. Introduction

This chapter will explain the methodology that was used in this project to conduct network simulations to analyze the simulation data that came from the results. The network topology will be created for measuring the performance of SCTP and TCP in the best-effort wired network environment as mentioned, and to provide sufficient answers to the research questions through the research objectives. Section 3.2 describes the simulation software that was used for this study and explains other types of simulations that are used for the same purpose. Also, Section 3.3 explains NS2 while Section 3.4 discusses the methodological framework for creating the simulation network topology.

3.2 Simulation Description

There are three main types of performance that are used by researchers for doing network performance evaluation; test bed, mathematical analysis and simulation. Firstly, a test bed is the estimation of an environment that has made up for testing purposes (Hunt, 2002). They are experiments for developing large projects that gives room for replicable, rigorous and transparent testing of computational tools, scientific theories, and other new technologies. Secondly, mathematical analysis is an analysis of rigorous formulation that can be described as theory from integration, measure, differentiation, infinite series, limits and analytic functions. These theories are usually studied in the context of complex

numbers, real numbers, and complex functions. Thirdly, simulation is processing many designs by creating a model of a proposed system to do many of the experiments in order to obtain a better understanding of the behavior of the system before designing the real system (Joe and Yan., 2009). The simulation interface is like a real system where you can do all the experiments for measuring performance of a network because the simulation provides a convenient way of estimating results and parameters (Kuhl et al., 2007).

3.3 Network Simulation 2 (NS-2)

NS2 is a discrete event network simulator. It was constructed using C++ and OTCL at the University of California in 1989. NS is used in the simulation of routing protocols, among others, and is heavily used in ad-hoc networking research. NS2 supports popular network protocols, offering simulation results for wired and wireless networks alike. It is popular in research given its open source model and online documentation.

The simulation that will be used in this study is network simulation (NS2) as the methodology (Hassan and Jain., 2004), and the steps are shown in Figure 3.1.

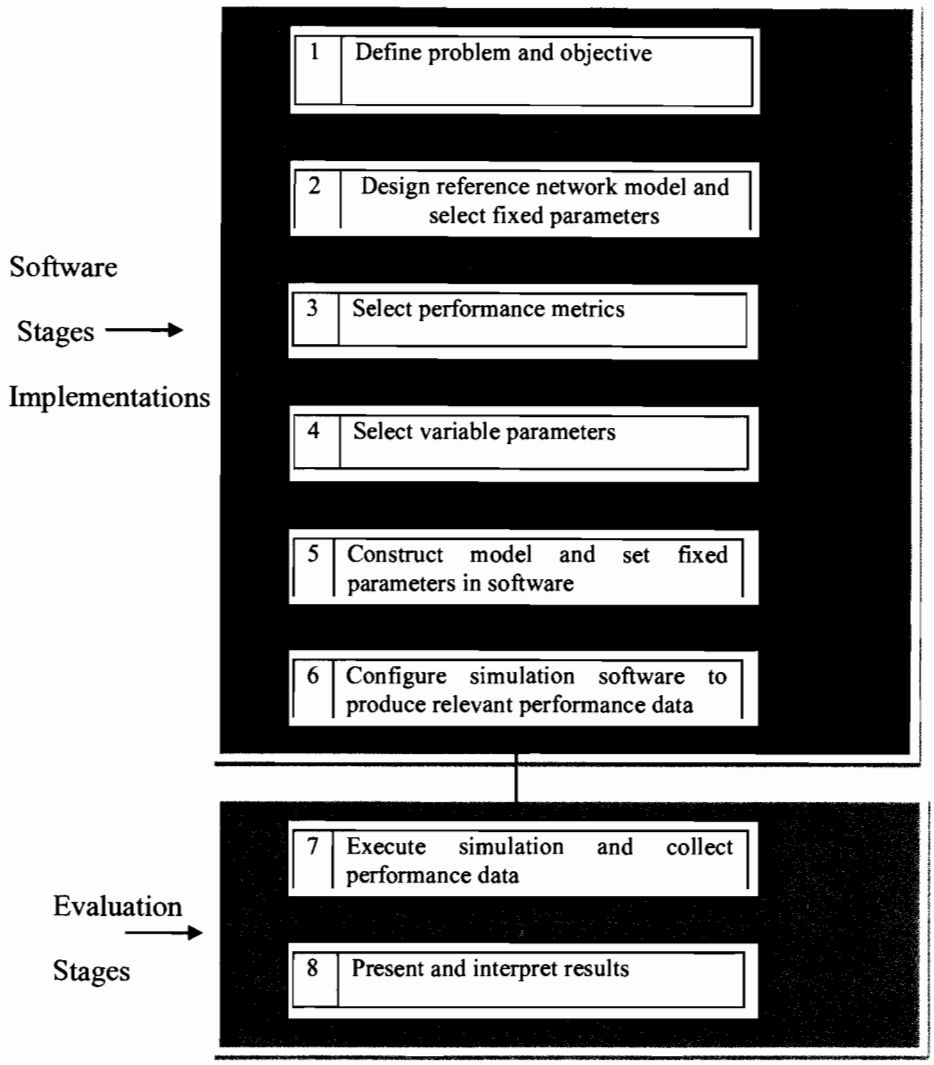

Figure 3.1 Steps for the Simulation Methodology

3.4 The Methodology Steps

(1) Software Implementation Stages

All the steps in this stage will be implemented through NS2 and below is an explanation of the steps:

Firstly, definition of the goals and objects of the study is to be clear and to help keep the track of the study in focus; this is because the problem statement of the study might not be understood by the readers. Secondly, design the topology in this study by selecting the parameters of the network, for example; link bandwidth and selecting the type of the topology itself. These parameters must show the weaknesses and try to fix it for all simulation executions. Thirdly, select the performance metrics, which are jitter, throughput and delay. Fourthly, selection of variable parameters in the SCTP simulation step, the previous studies assessed the performance based on high speed, competing flows and so on. In this study, the environment is a wired network by measuring the competing flows of SCTP. Fifthly, this study will construct the modeling and set fixed parameter steps in the software to build the network model designed in second step as well as installation of the fixed parameters. This step is coding by using the programming language that is used by the NS2 simulation software. Lastly, the study will configure the values of the performance metrics selected in the third step of the software program.

(2) Evaluation Stages:

There are two steps for evaluating the input parameters, which is by executing the NS2 successfully and compare the result with other results to obtain the evaluation performance. Below is an explanation of the steps:

Firstly, execute the simulation software by running the experiment topologies. Once the simulation runs completely, the performance metrics will appear to depict the result of the output of the data. Secondly, when the experiments indicate the data output, the results will present the performance metrics in graph and table format, while the results would be compared for perfect measurement of the performance. The steps may require running the simulation many times by using different values in order to compare the results for measuring the performance.

3.5 Summary

This chapter describes the methods that were used to solve the research problem and achieve the aim of this study. Moreover, the chapter also explains the simulations are being used for the same purpose in order to give an idea about the simulation that will be used for this project.

CHAPTER FOUR

PERFORMANCE EVALUATION OF SCTP

4.1 Introduction

This chapter presents the results of the experiments carried out and a critical analysis of them. Section 4.2 briefly presents the background and the environment that has been setup to run the experiments. Sections 4.3 and 4.4 discuss the results in detail with special reference to the performance of SCTP and TCP protocols in the presence of other traffic.

4.2 Experimental Setup

For the purpose of running the experiment, a simple network has been created in a network simulator. A simulation environment has been selected instead of setting up an actual network with real equipment due to the ease of varying different parameters of the environment and observing the results compared to that in an actual network. NS2 has been selected in the project due to its versatility and ease of use. Also, the fully functional NS2 is an open source tool that can be downloaded at no cost.

The first set of experiments was carried out using a pair of transmitting and receiving nodes connected by using two routers and a dedicated link. The experiments were repeated for both SCTP and TCP traffic several times for the purpose of measuring delay, delay jitter and throughput. A second set of experiments was carried out using five CBR over SCTP and TCP traffic sources. In this setup, the nodes started transmission with predefined delays to simulate the real environment. The results of the first experiment looked artificial compared to the second one, as in the first experiment there were only two nodes connected using a dedicated link. This is far from the real environment that is

encountered on the Internet. Hence, only the results of the second experiment are discussed in this thesis. Figure 4.1 shows the network that has been set up for the second set of experiments to measure the performance of SCTP in the presence of other network traffic. The network consists of ten source nodes, ten destination nodes along with two routers. The five source SCTP nodes would be generating FTP over SCTP traffic destined for the SCTP nodes on the other portion of the network, while the CBR over UDP generated by the rest of the nodes would be used to simulate the actual networking environment. The source nodes are connected to the routers using 1 Mbps point-to-point links with a transmission delay of 10 ms, the destination nodes are connected to the other router using links with a bandwidth of 1 Mbps and 30 ms delay. The routers are connected to each other using a 512 kbps point-to-point links that has a propagation delay of 100 ms.

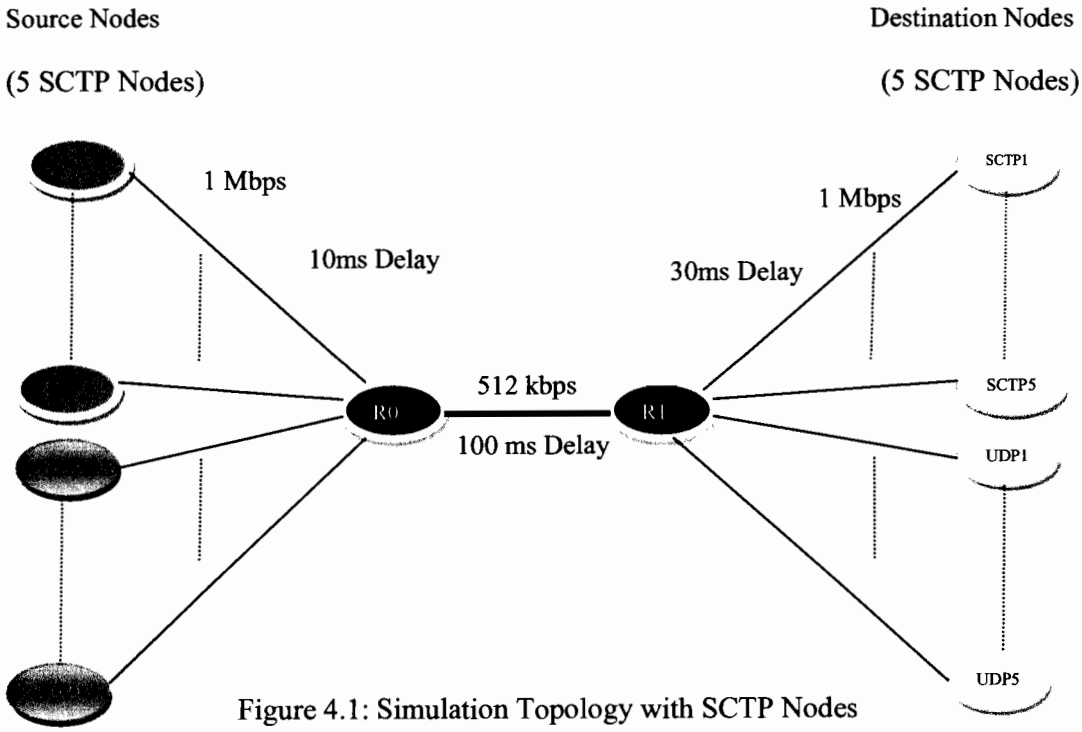

Figure 4.1: Simulation Topology with SCTP Nodes

Figure 4.2 shows the network set-up to measure the performance of the TCP protocol. This network is similar to the network in Figure 4.1 except for the TCP nodes in place of SCTP nodes. In this experiment, the performance of TCP will be measured in the presence of UDP traffic. Similar to the previous experiment, here also FTP and CBR traffic would be carried over TCP and UDP, respectively.

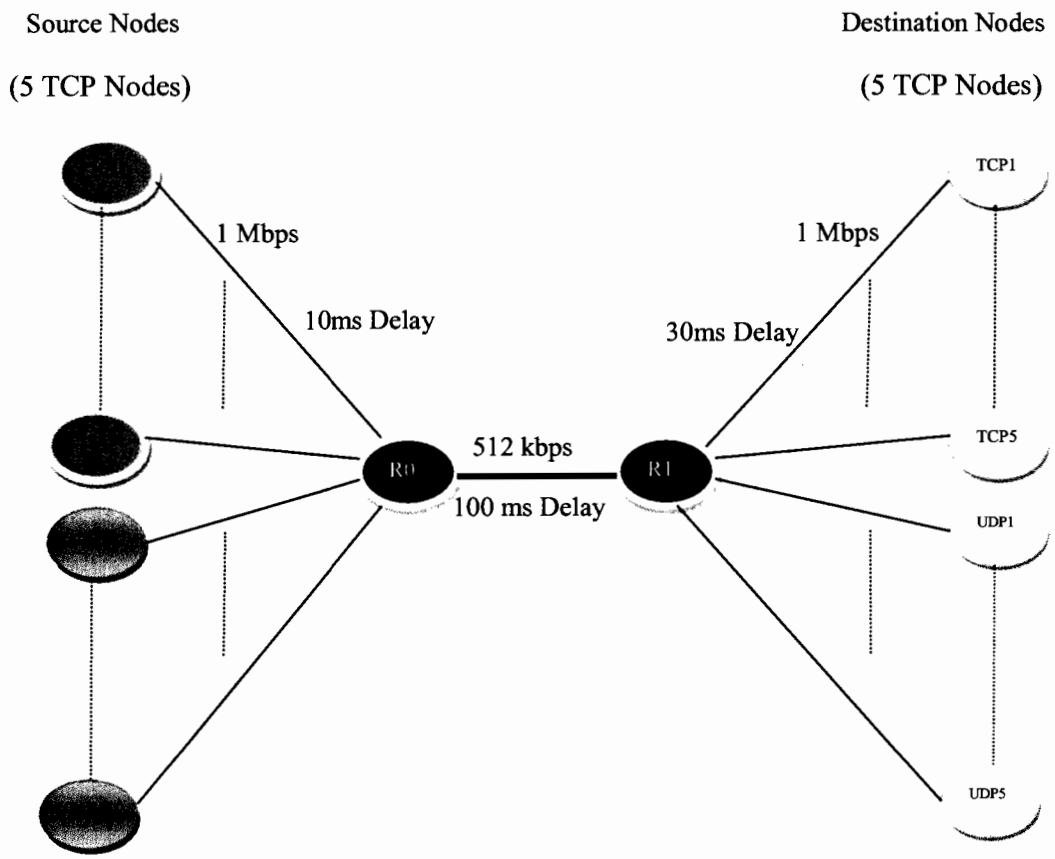

Figure 4.2 Simulation Topology with TCP Nodes

4.3. Simulation

In the simulator, the nodes have been numbered starting from 0 to 21, where the routers were numbered 0 and 1, source nodes from 2 to 11 and the destination nodes from 12 to 21, respectively. Figure 4.3 shows the simulation environment setup created in NS2 for testing the SCTP protocol. The top five nodes on both source and destination sides have been configured to be of SCTP and the rest to be of UDP.

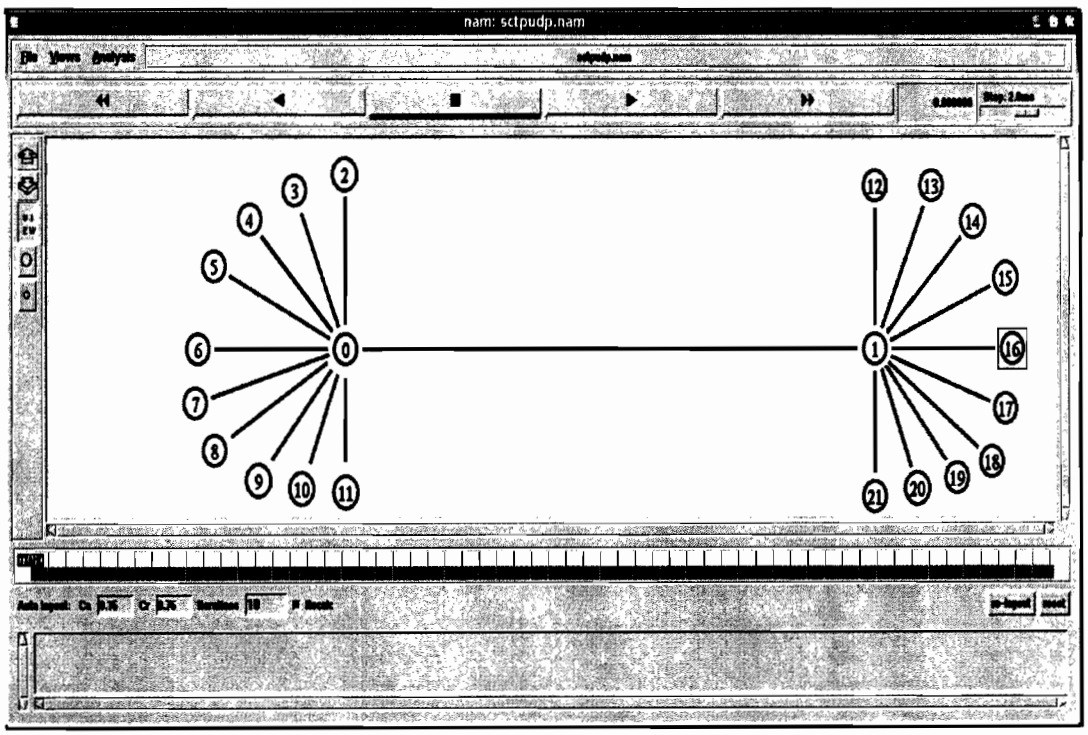

Figure 4.3: SCTP Experimental Environment

Figure 4.4 shows the experimental environment setup for testing the TCP protocol. This configuration is similar to that of the experimental setup for SCTP except for the types of nodes.

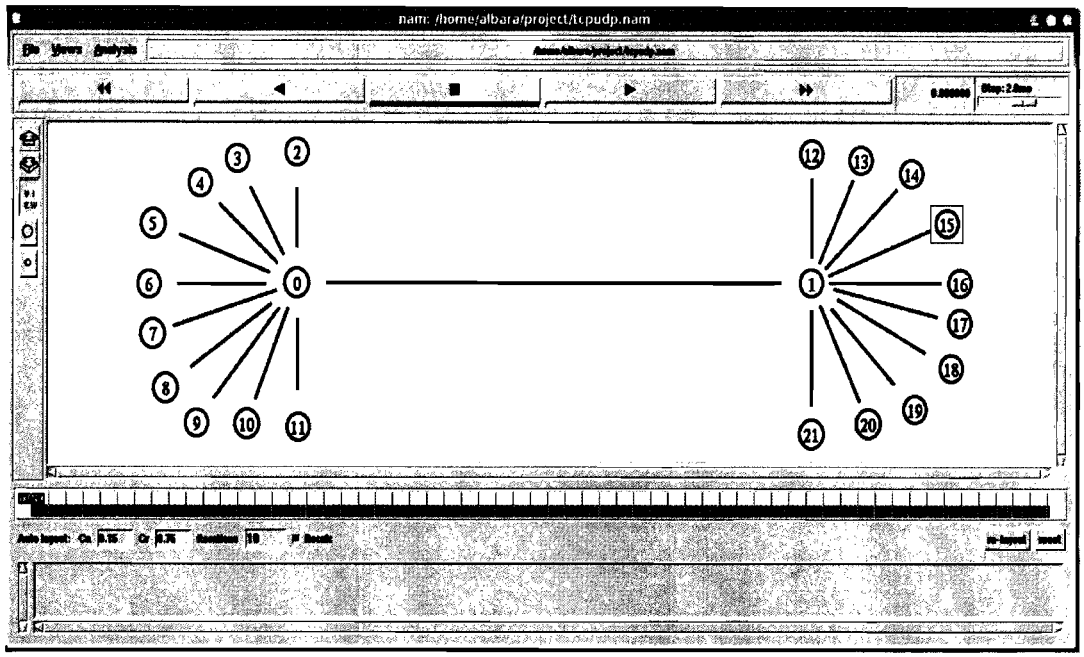

Figure 4.4: TCP Experimental Environment

In both experiments, the connection between the source and destination nodes have been configured through the routers by enabling the communication protocols in the nodes and the routers. The links between the source nodes and the router 1 has been configured to have a bandwidth of 1 Mbps along with a propagation delay of 10ms. The link between the routers was configured to have a bandwidth of 512 kbps with a propagation delay of

100 ms. The parameters of the links connecting the router 2 and the destination nodes are 1 Mbps of bandwidth and 30 ms propagation delay.

In both experiments, the source nodes generated FTP and CBR packets of 1000 kB at the rate of 1000 packets per second. The nodes generated the packets with a delayed start in order to simulate a controlled real environment. Node 2 generated the first packet at 10ms and the nodes 3 and 4 generated the first packet at 15ms while nodes 5 and 6 generated their first packet at 20ms. The packet generation of the UDP nodes is as follows: nodes 7 and 9 generate the first packet at 30ms, node 8 at 35 ms, node 10 at 40ms and node 11 at 50ms.

4.4 Results

This section presents the result of the experiments based on throughput, delay and delay jitter as performance metrics. The results of both experiments will be compared with each other in order to evaluate the relative performance of both protocols.

4.4.1 Delay

Delay has been used as a metric in this project to compare the performance of SCTP against TCP. Delay in the network is the time or period for data traveling through the network from the source to the destination or between the endpoints specifying how long the data will take to arrive at the destination (Amer and Stewart, 2005). The delay was measured by computing the time between the start and receive times. Table 4.1 presents the average delay of SCTP traffic between selected nodes. The average delay between the nodes 2 and 12, nodes 3 and 13, nodes 4 and 14, nodes 5 and 15 and nodes 6 and 16 were

1122.54 ms, 1134.04 ms, 1136.28 ms 1137.93 ms and 1138.76 ms, respectively. Figure 4.5 shows the delay between the nodes in a graphical fashion.

Table 4.1: SCTP Average Delay

Nodes	Average Delay
Nodes 2 and 12	1122.54 ms
Nodes 3 and 13	1134.04 ms
Nodes 4 and 14	1136.28 ms
Nodes 5 and 15	1137.93 ms
Nodes 6 and 16	1138.76 ms

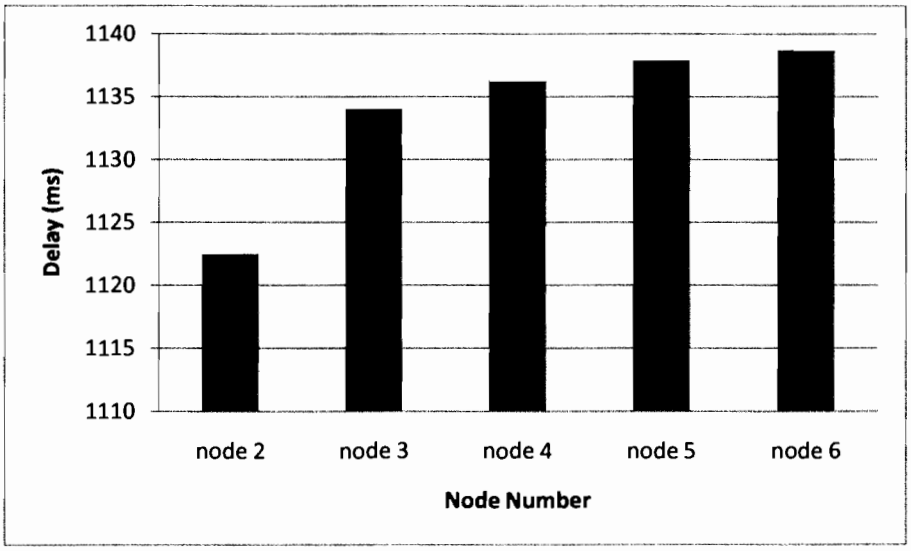

Figure 4.5 Average of Delay of SCTP Nodes

From the results shown in Table 4.1 and Figure 4.5, it can be seen that the delay increases as the number of packets in the network increases. The average delay between the nodes 2 and 12 is the lowest as when these nodes start communication, the network was idle

and the entire bandwidth is available for this communication. When other nodes start their communication the network has already been used by the nodes that had already commenced their communication. Hence, the packets need to share the link bandwidth, router buffer (queue) and other resources. This increases the average delay of communication between nodes that delay starting the generation of packets.

Table 4.2 presents the average delay of TCP traffic between selected nodes. The average delay between the nodes 2 and 12, nodes 3 and 13, nodes 4 and 14, nodes 5 and 15 and nodes 6 and 16 were 780.416 ms, 793.06 ms, 795.888 ms, 803.97 ms and 805.11 ms, respectively. Figure 4.6 shows the delay between the nodes in a graphical fashion.

Table 4.2 TCP Average Delay

Nodes	**Average Delay**
Nodes 2 and 12	780.416 ms
Nodes 3 and 13	793.06 ms
Nodes 4 and 14	795.888 ms
Nodes 5 and 15	803.97 ms
Nodes 6 and 16	805.11 ms

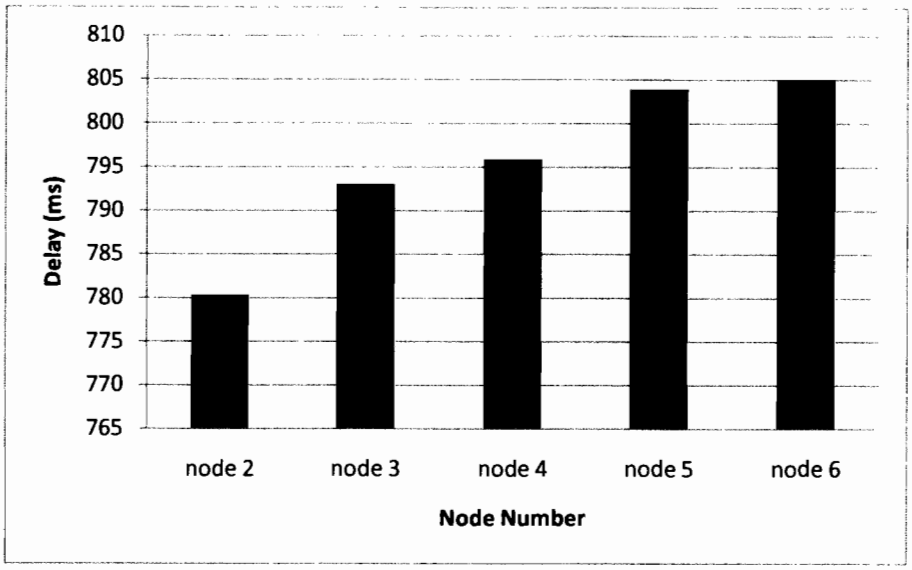

Figure 4.6 Average of Delay of TCP Nodes

In the results shown in Table 4.2 and Figure 4.6, it can be seen that the delay increases as the number of packets in the network increases. The average delay between the nodes 2 and 12 is the lowest as when these nodes start communication, the network was idle and the entire bandwidth is available for this communication. When other nodes start their communication the network has already been used by the nodes that had already commenced their communication. Hence, the packets need to share the link bandwidth, router buffer (queue) and other resources. This increases the average delay of communication between nodes that delay starting the generation of packets.

Figures 4.7, 4.8, 4.9, 4.10 and 4.11 show the comparison of mean delay experienced by each pair of traffic for both SCTP and TCP protocols. It can be seen that the SCTP traffic had undergone a delay of more than 1 Sec and the delay experienced by the TCP traffic was less than 1 Sec on the whole. Figure 4.7 depicts the average delay of the SCTP traffic being between the nodes 2 and 12 was around 1.5 seconds than that of the TCP traffic.

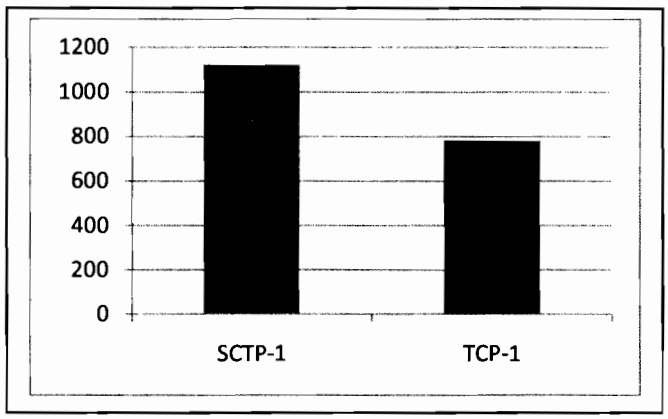

Figure 4.7 Comparison of Nodes 2 Average Delay of SCTP and TCP

From Figure 4.8, it can be seen that the delay experienced by the SCTP traffic between the nodes 3 and 13 was approximately 1.4 seconds more than that of the TCP traffic.

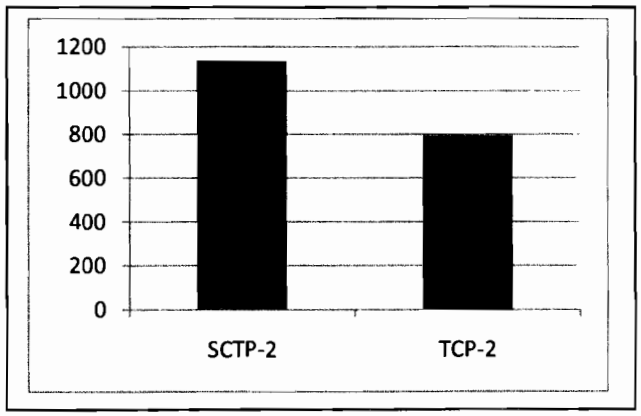

Figure 4.8 Comparison of Node 3 Average Delay of SCTP and TCP

The Figure 4.9 illustrates the delay performance of SCTP traffic between nodes 4 and 14 was also in the region of 1.3 seconds more compared with the TCP traffic.

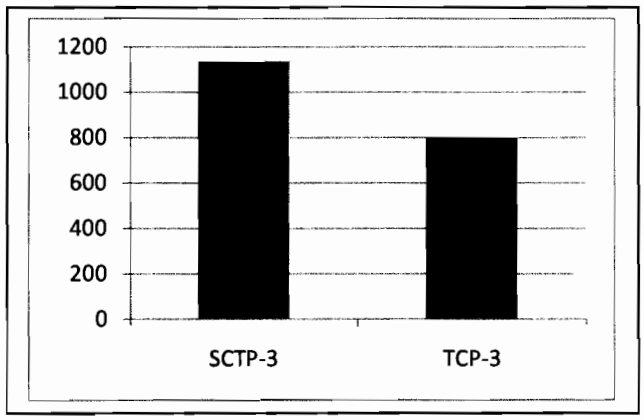

Figure 4.9: Comparison of Nodes 4 Average Delay of SCTP and TCP

Figure 4.10 represents delay of the SCTP traffic, and between nodes 5 and 15 was roughly 1.5 seconds more than that of TCP traffic.

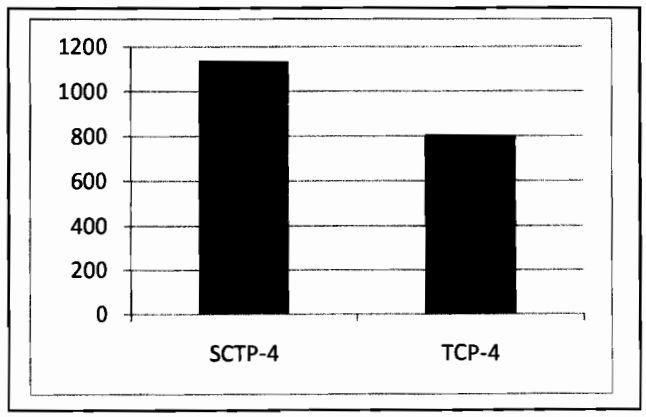

Figure 4.10 Comparison of Average Delay of SCTP and TCP Nodes 4 and 5

Figure 4.11 explains average delay of SCTP traffic between nodes 6 and 16, which was also around 1.4 seconds more than TCP traffic performance.

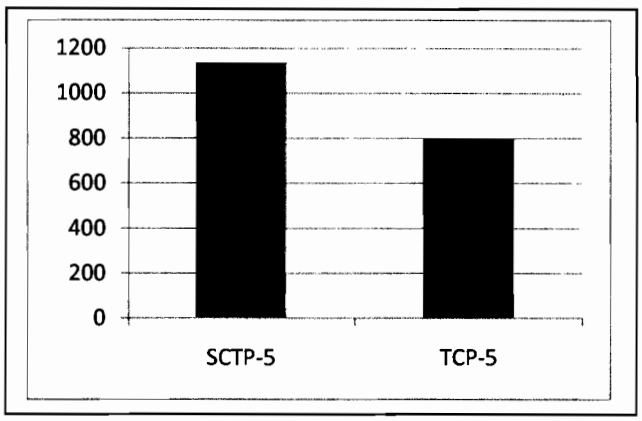

Figure 4.11 Comparison of Node 6 Average Delay of SCTP and TCP

Figures 4.12, 4.13, 4.14, 4.15 and 4.16 show the delay undergone by different data flows of SCTP and TCP. In the figures, delay undergone by SCTP is shown green in color and the TCP in red color. Most of the time, the delay values have been found to line up between 0 and 1 second, except for few instances where the SCTP stream shows a delay more than 1 second on the whole. Figure 4.12 shows the average delay of the SCTP traffic between nodes 2 and 12, and was more than 1.2 seconds compared with the TCP traffic, which was less than 1 sec.

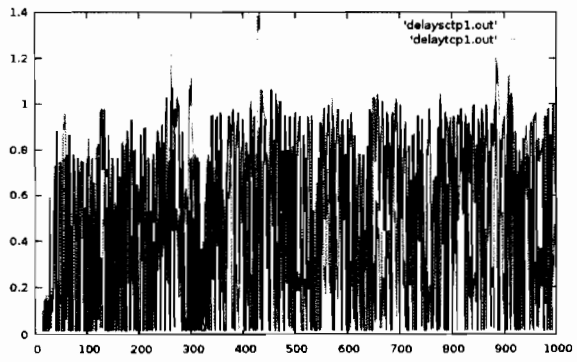

Figure 4.12 Comparison of Nodes 2 Delay of SCTP and TCP Streams

Figure 4.13 illustrates the SCTP traffic performance between nodes 3 and 13, where it registered a delay time of 1.4 sec more than the 1.2 sec for TCP traffic.

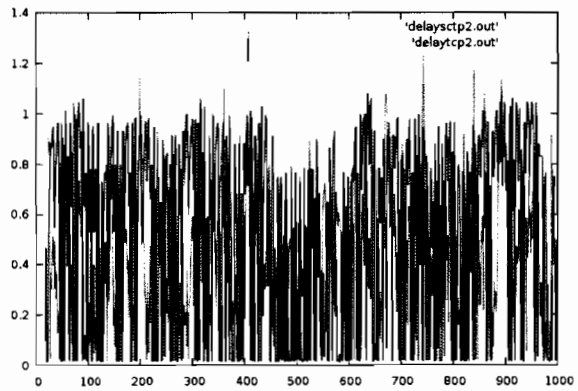

Figure 4.13 Comparison of Node 3 Delay of SCTP and TCP Streams

The Figure 4.14 depicts SCTP traffic delay between nodes 4 and 14 was more than 1.2 Sec compared with that less than 1 Sec of TCP traffic.

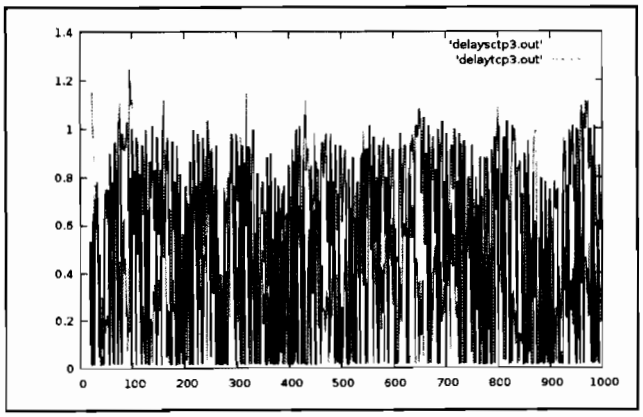

Figure 4.14: Comparison of Nodes 4 Delay of SCTP and TCP Streams

Figure 4.15 represents the delay of the SCTP traffic between nodes 5 and 15, and was in the region of 1.5 Sec more than that of TCP; however, TCP has one line around 1.8 Sec, which means that at this time TCP was affected by UDP.

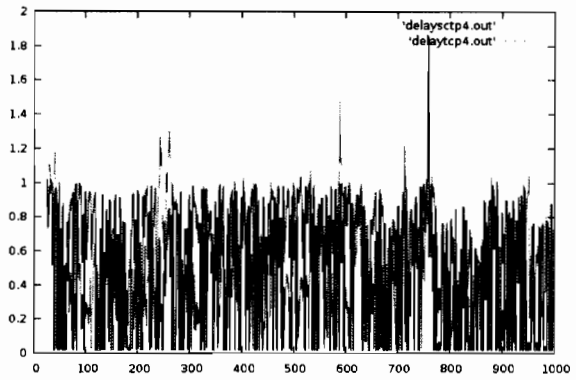

Figure 4.15 Comparison of node 5 Delay of SCTP and TCP Streams

Figure 4.16 explains the average delay of SCTP traffic between nodes 6 and 16, and was in the range between 1 and 1.4 Sec more than range of 1 and 1.2 Sec of TCP. But TCP showed one line at more than 1.2 Sec, which means that TCP affection increases when most of the nodes are involved in the network.

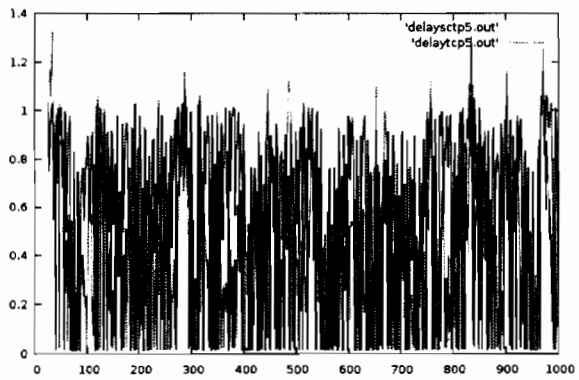

Figure 4.16 Comparison of node 6 Delay of SCTP and TCP Streams

Figure 4.17 shows the delays undergone by the SCTP and TCP streams as a combined figure. From this figure, it can be seen that the SCTP stream undergoes a longer delay than the TCP stream under similar conditions. All the nodes of SCTP have delay times of more than 1 Sec, however average delay time is less than 800 ms for TCP.

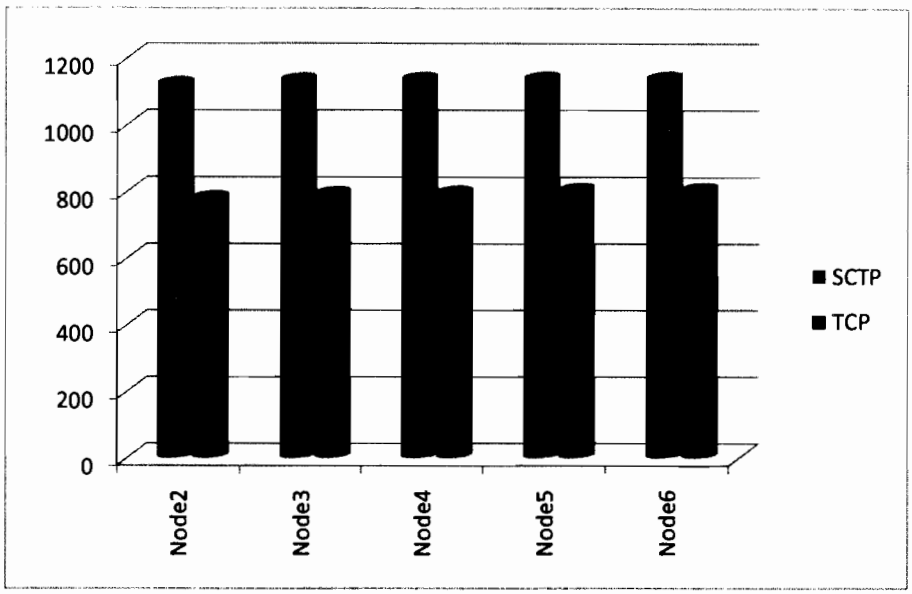

Figure 4.17 Comparison of Pair-wise Delay of SCTP and TCP Streams

4.4.2 Jitter

Jitter is the time variation of delay a periodic signal undergoes during transmission, often measured in relation to a reference clock source (Wolaver, 1991). Jitter is one of the performance metrics commonly used in communication research. This study also uses jitter as a metric to compare the performance of both TCP and SCTP. Average jitter has been used in this study for the purpose of evaluation of the relative performance of these two protocols. The jitter was computed by measuring the variation of time between the

send and receive times. Table 4.3 shows the average jitter of SCTP traffic between selected nodes. The average delay jitter between nodes 2 and 12, nodes 3 and 13, nodes 4 and 14, nodes 5 and 15 and nodes 6 and 16 were 32.9559 ms, 33.5674 ms, 33.2597 ms, 33.6277 ms and 33.1688 ms, respectively. Figure 4.18 presents the same in a graphical fashion.

Table 4.3 SCTP Average Jitter

Nodes	**Average for Jitter**
Nodes 2 and 12	32.9559 ms
Nodes 3 and 13	33.5674 ms
Nodes 4 and 14	33.2597 ms
Nodes 5 and 15	33.6277 ms
Nodes 6 and 16	33.1688 ms

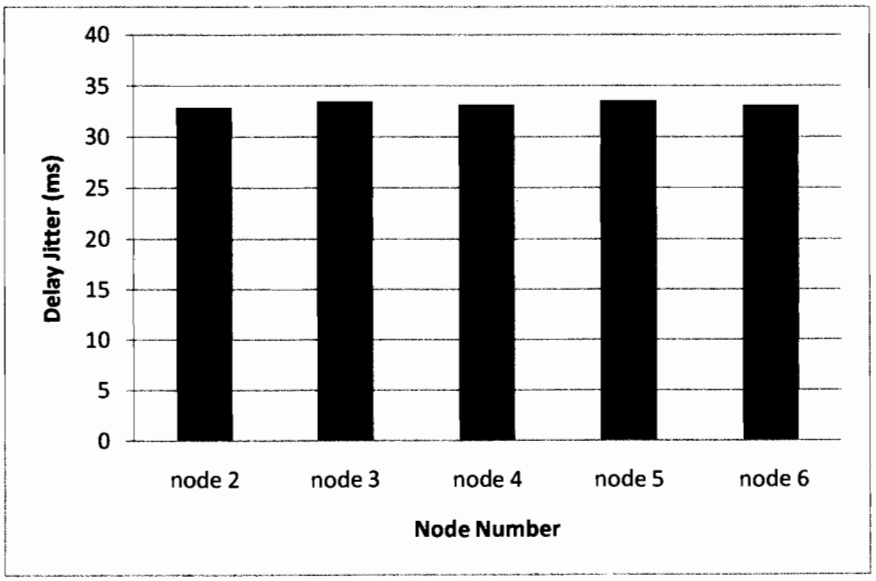

Figure 4.18 Average Jitter of SCTP Nodes

From Table 4.3 and Figure 4.18 it can be seen that average jitter varies from one node to another. The average jitter between nodes 2 and 12 is the lowest. This is due to the reason that when these nodes start communicating the network is idle and the entire bandwidth is available for this communication. The highest jitter has been observed between nodes 5 and 15. The variation in delay jitter is due to the different loading of the network as different streams were started at different times.

Table 4.4 shows the average jitter of TCP traffic between the selected nodes. The average delay between nodes 2 and 12, nodes 3 and 13, nodes 4 and 14, nodes 5 and 15 and nodes 6 and 16 were 12.8424 ms, 13.754 ms, 14.0515 ms, 13.2253 ms and 14.1221 ms, respectively, and Figure 4.15 shows the same in a graphical fashion.

Table 4.4: TCP Average Jitter

Nodes	**Average for Jitter**
Nodes 2 and 12	12.8424 ms
Nodes 3 and 13	13.754 ms
Nodes 4 and 14	14.0515 ms
Nodes 5 and 15	13.2253 ms
Nodes 6 and 16	14.1221 ms

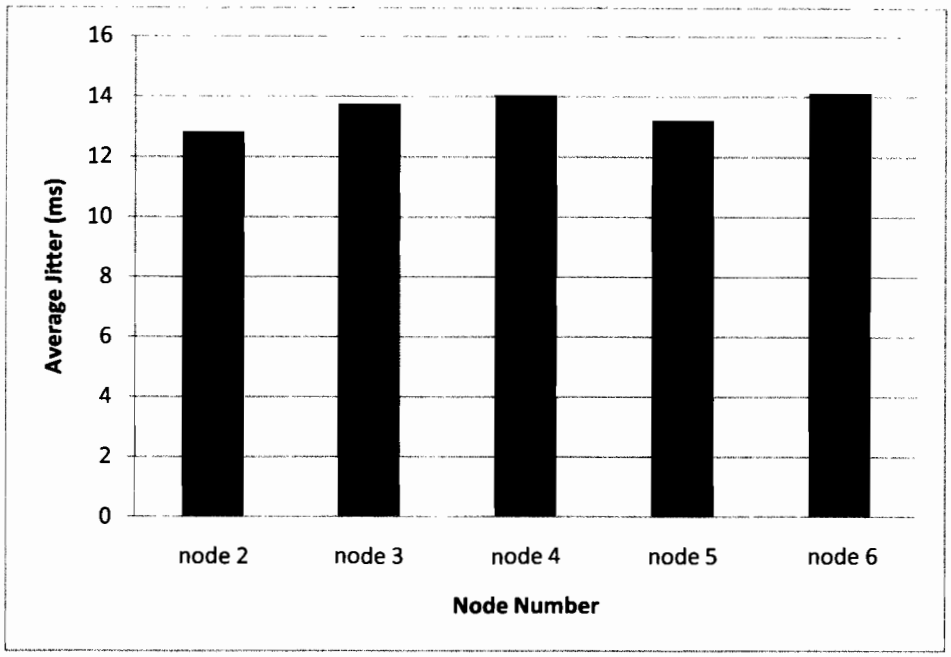

Figure 4.19 Average Jitter for TCP Nodes

From Table 4.4 and Figure 4.19, it can be seen that the jitter varies randomly between pairs of nodes. The average jitter between nodes 2 and 12 is the lowest and between nodes 6 and 16 is the highest. This is due to the reason that when nodes 2 and 12 commenced their communication the network was idle and the network was occupied with traffic when nodes 6 and 16 commenced communication.

Figures 4.20, 4.21, 4.22, 4.23 and 4.24 show the pair-wise relative performance of SCTP and TCP protocols in terms of delay jitter. Figure 4.20 represents the average jitter of the SCTP traffic between the nodes 2 and 12, and was around 2.6 seconds more variation time compared to that of the TCP traffic.

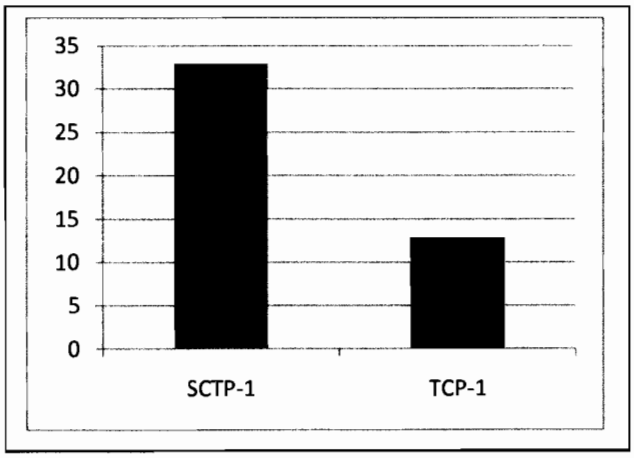

Figure 4.20 Comparison of Node 2 Average Jitter of SCTP and TCP

From Figure 4.21, it can be seen that the variation time experienced by the SCTP traffic between nodes 3 and 13 was approximately 2.4 seconds more than that of the TCP traffic.

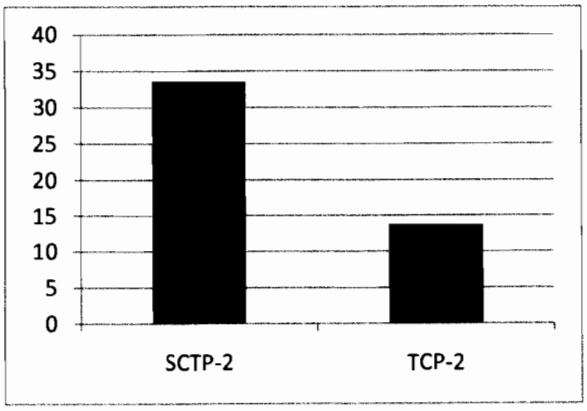

Figure 4.21 Comparison of Node 3 Average Jitter of SCTP and TCP

Figure 4.12 illustrates the jitter performance of SCTP traffic between nodes 4 and 14, which was also in the region of 2.4 seconds more compared with the TCP traffic.

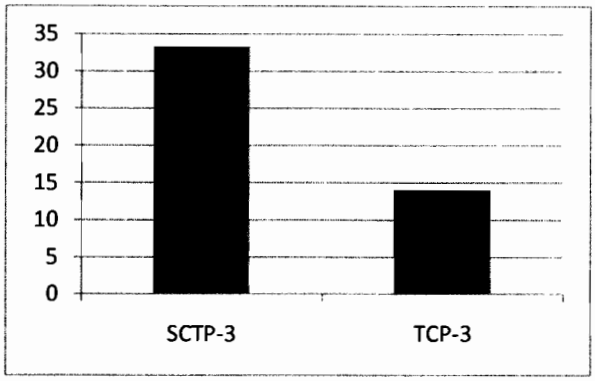

Figure 4.22 Comparison of Node 4 Average Jitter of SCTP and TCP

Figure 4.23 represents jitter of the SCTP traffic between nodes 5 and 15, and was roughly 2.5 seconds more than that of TCP traffic.

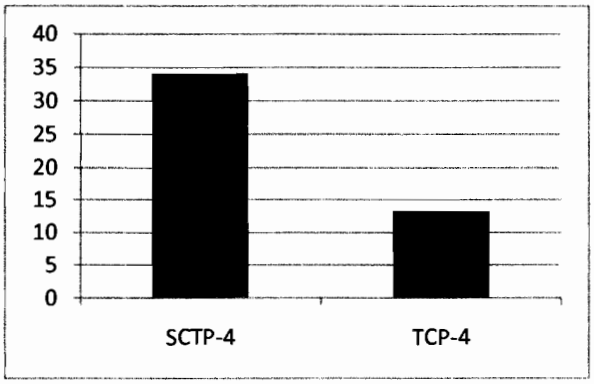

Figure 4.23 Comparison of Node 5 Average Jitter of SCTP and TCP

Figure 4.24 explains average jitter of SCTP traffic between nodes 6 and 16, and was also around 2.3 sec more than TCP traffic performance. The nodes started with high jitter and then showed a variation time between increases and decreases, because the nodes have different start times. Comparatively, SCTP undergoes more delay jitter than TCP; this has been observed to be true for all cases of the experiment as shown in Figure 4.12.

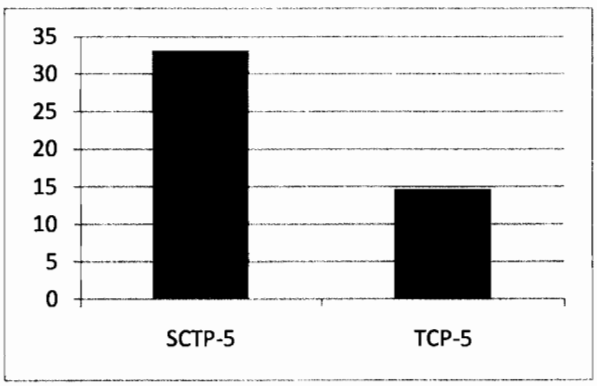

Figure 4.24 Comparison of Node 6 Average Jitter of SCTP and TCP

Figures 4.25, 4.26, 4.27, 4.28 and 4.29 show the jitter undergone by different data flows of SCTP and TCP. The graph in green shows the delay jitter undergone by SCTP and the blue graph is the delay jitter experienced by the TCP stream. It can be seen that the jitter experienced was confined to a maximum of 100 ms most of the time except for few instances where the SCTP stream shows a jitter excess of 100 ms. Also, the jitter of the TCP stream has at certain instances reached zero.

Figure 4.25 shows the average delay jitter of the SCTP traffic between nodes 2 and 12, more than 0.8 sec compared with the TCP traffic, which was less than 0.4 seconds.

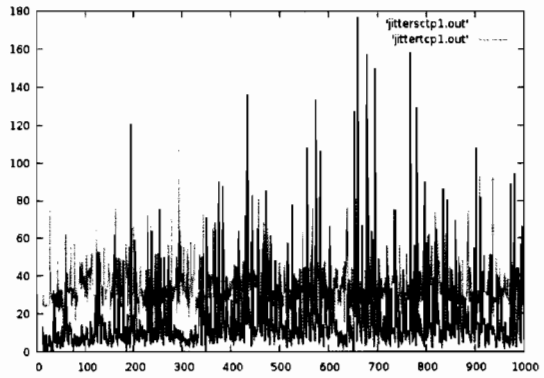

Figure 4.25 Comparison of Node 2 Jitter of SCTP and TCP Streams

Figure 4.26 illustrates the SCTP traffic performance between nodes 3 and 13, where it registered delay jitter around 0.6 seconds more than 0.2 seconds for TCP traffic.

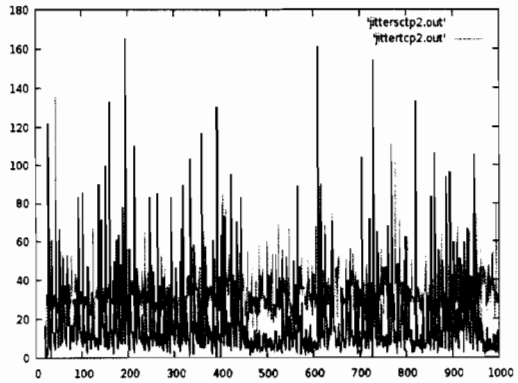

Figure 4.26 Comparison of Node 3 Jitter of SCTP and TCP Streams

Figure 4.27 depicts SCTP traffic delay jitter between nodes 4 and 14, which was more than 0.8 seconds compared with less than 0.4 seconds for TCP traffic.

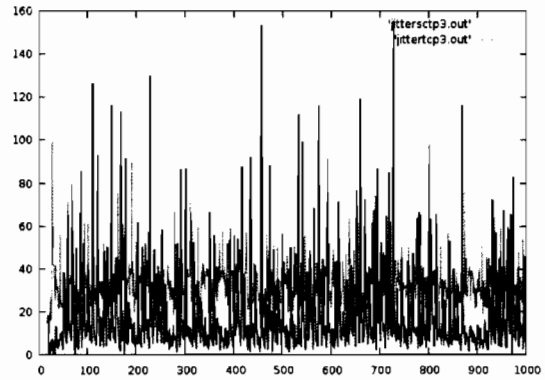

Figure 4.27 Comparison of Node 4 Jitter of SCTP and TCP Streams

Figure 4.28 represents the delay jitter of the SCTP traffic between nodes 5 and 15, which was in the region of 1 second more than that of TCP, which was around 0.5 seconds, however, SCTP has one line around 2 seconds, which means that within this time SCTP was affected by UDP.

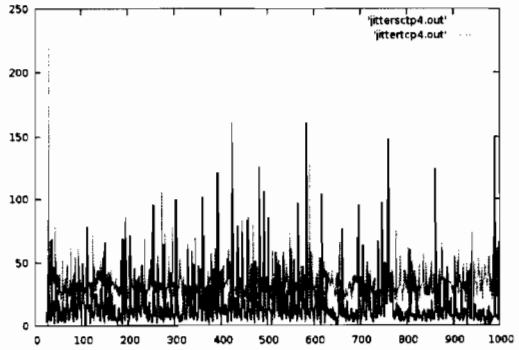

Figure 4.28 Comparison of Node 5 Jitter of SCTP and TCP Streams

Figure 4.29 explains the average delay jitter of SCTP traffic between nodes 6 and 16, which was in the range between 0.3 and 1 second, more than the range between 0 and 0.5 seconds for TCP. TCP has one line more than 3.0 Sec, which means that TCP was affected by UDP.

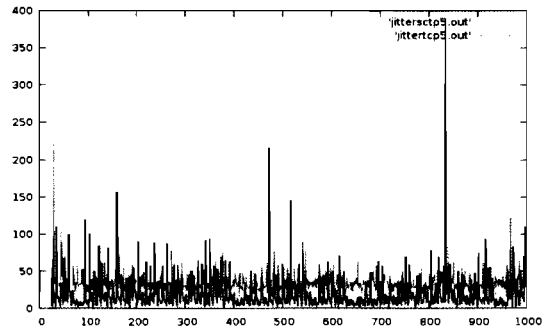

Figure 4.29 Comparison of Node 6 Jitter of SCTP and TCP Streams

Figure 4.30 shows the average jitter undergone by the SCTP and TCP streams in a combined figure. On average, the SCTP stream underwent three times more jitter than the TCP stream under similar conditions. This indicated that all SCTP nodes have delay jitter more than 3 Sec compared with 1.4 Sec for TCP.

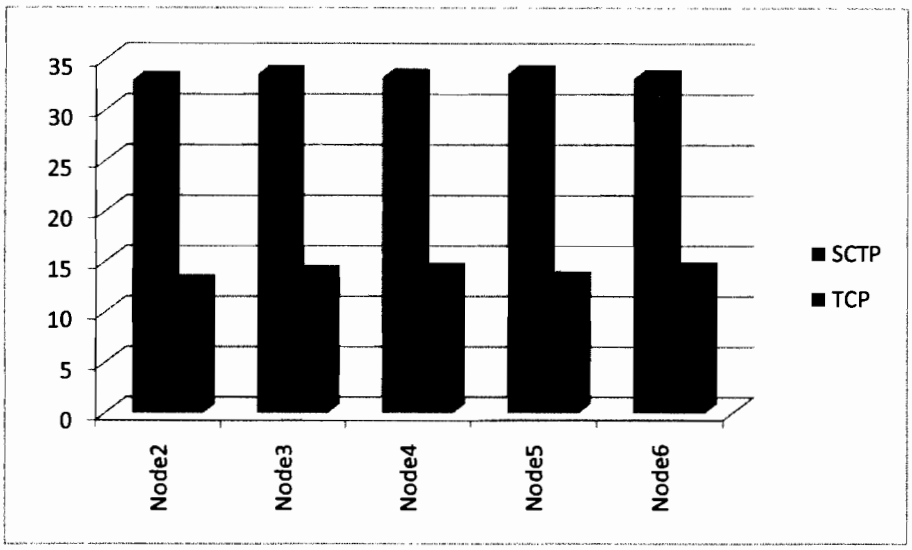

Figure 4.30 Comparison of Pair-wise Jitter of SCTP and TCP Streams

4.4.3 Throughput

Throughput refers to how much data is transferred between two locations. It is used to measure the performance of hard drives, memory, as well as the Internet and other computer networks (Demichelis and Chimento., 2002). Throughput has also been used in this project as a metric to compare the performance of the SCTP and TCP streams. Throughput was measured by computing the amount of data transferred between the nodes. Table 4.5 presents the average throughput of SCTP traffic between selected nodes. The average throughput between nodes 2 and 12, nodes 3 and 13, nodes 4 and 14, nodes 5 and 15 and nodes 6 and 16 was 64.2493 Kbps, 59.492 Kbps, 62.9314 Kbps, 55.6609 Kbps and 61.9075 Kbps, respectively. And Figure 4.31 shows the same in a graphical fashion.

Table 4.5: Average Throughput of SCTP Traffic

Nodes	Average Throughput (kbps)
Nodes 2 and 12	64.2493
Nodes 3 and 13	59.492
Nodes 4 and 14	62.9314
Nodes 5 and 15	55.6609
Nodes 6 and 16	61.9075

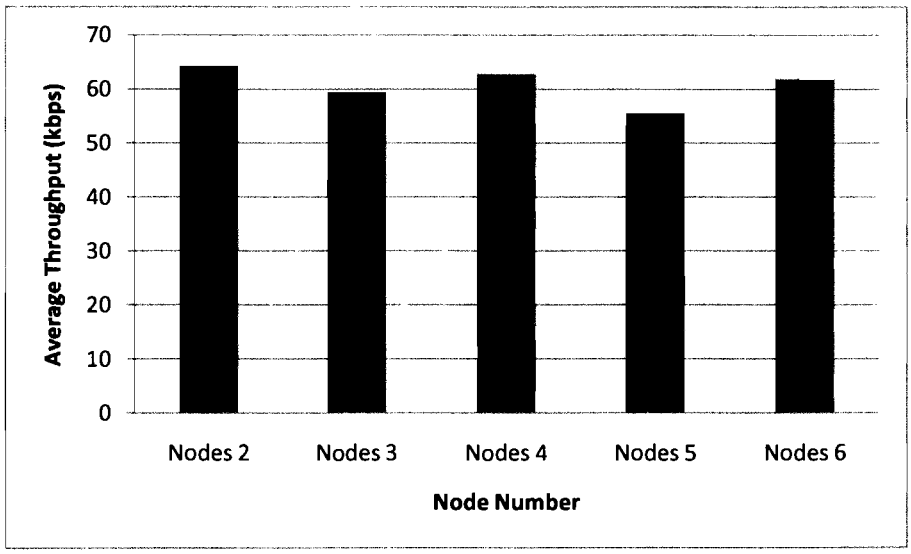

Figure 4.31 Average Throughput of SCTP Nodes

In Figure 4.31, it can be seen that SCTP has a varying average throughput. It can also be seen that the throughput was high at the beginning between nodes 2 and 12 and then goes down when transmission between nodes 3 and 13 started. This is due to the reason that with the increase of traffic in the network, the throughput of each stream goes down. Again the throughput goes up between nodes 4 and 14 as the transmission between these

nodes was started after a long time, which was sufficient for the network to become idle. Moreover, it is observed that the SCTP throughput data flows affected this type of network environment.

Table 4.6 presents the average throughput of TCP traffic between selected nodes. The average delay between nodes 2 and 12, nodes 3 and 13, nodes 4 and 14, nodes 5 and 15 and nodes 6 and 16 were 94.3721 Kbps, 86.8597 Kbps, 85.1145 Kbps, 82.7298 Kbps and 81.0108 Kbps, respectively. And Figure 4.32 shows the same information in a graphical fashion.

Table 4.6 Average Throughput of TCP Traffic

Nodes	**Average Throughput (Kbps)**
Nodes 2 and 12	94.3721
Nodes 3 and 13	86.8597
Nodes 4 and 14	85.1145
Nodes 5 and 15	82.7298
Nodes 6 and 16	81.0108

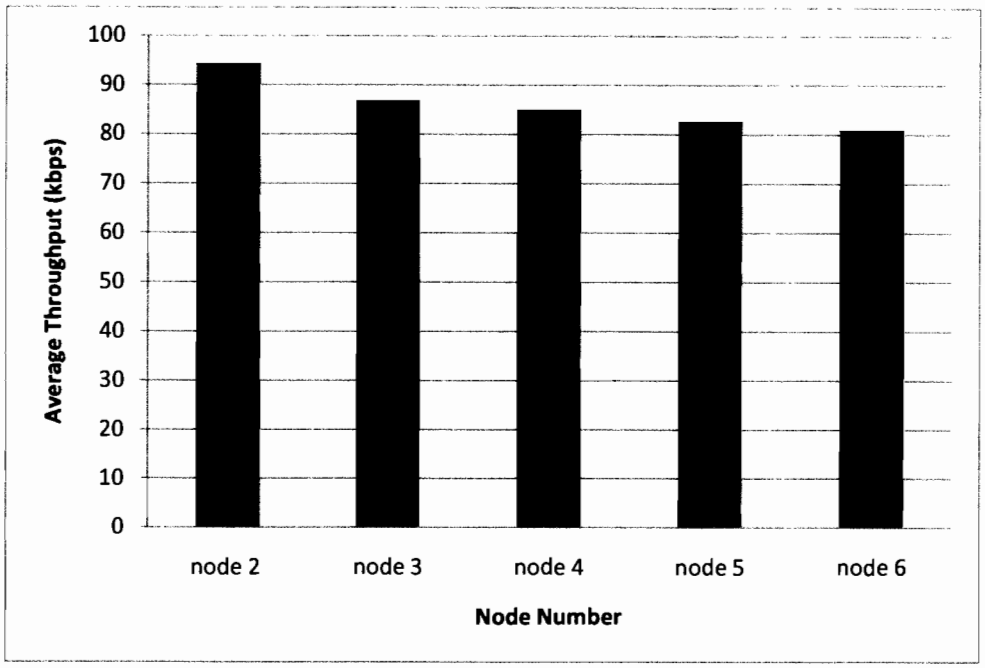

Figure 4.32 Average Throughput of TCP Nodes

From Figure 4.32, it can be seen that TCP traffic in the network has a different average throughput between different pairs of nodes. Similar to the SCTP stream, TCP also had higher throughput initially between nodes 2 and 12 and then went down somewhat steadily, until it registered the lowest between nodes 6 and 16. This is due to the reason that the network was initially idle and then became busy with each node starting transmission one after the other in a delayed fashion.

Figures 4.33, 4.34, 4.35, 4.36 and 4.37, show the comparison of mean throughput in a pair-wise comparison between corresponding nodes. From this figure, it can be seen that TCP registers better throughput compared to SCTP. Figure 4.33 represents the average of throughput of the SCTP traffic between nodes 2 and 12, which was around 0.7 seconds of throughput time, which was less compared to that of the TCP traffic.

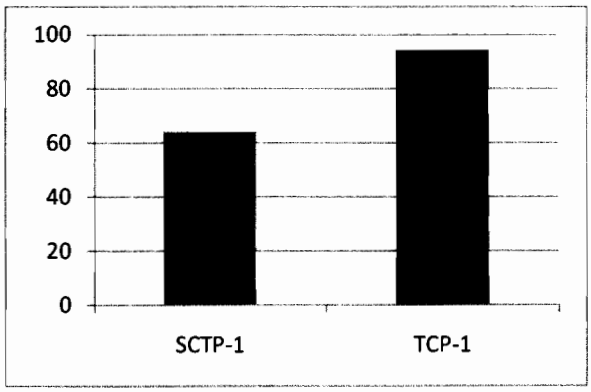

Figure 4.33 Comparison of Node 2 Average Throughput for SCTP and TCP

From Figure 4.34, it can be seen that the throughput time experienced by the SCTP traffic between nodes 3 and 13 was approximately 0.7 sec less than that of the TCP traffic.

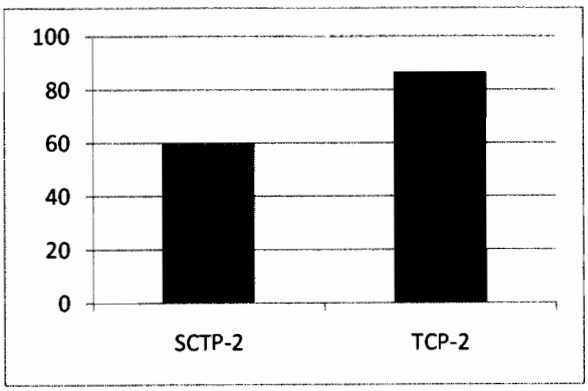

Figure 4.34 Comparison of Node 3 Average Throughput of SCTP and TCP

Figure 4.35 illustrates the throughput performance of SCTP traffic between nodes 4 and 14, and was also in the region of 0.8 seconds less compared with the TCP traffic.

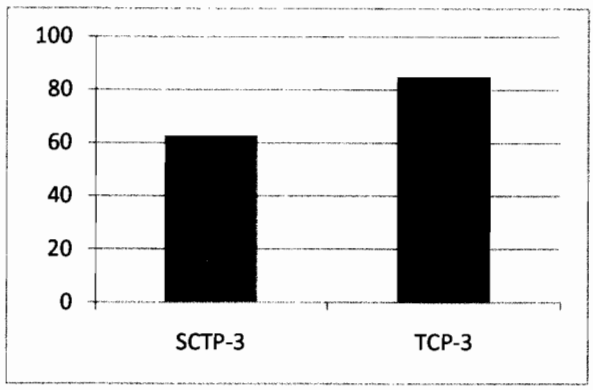

Figure 4.35 Comparison of Node 4 Average Throughput of SCTP and TCP

Figure 4.36 represents throughput of the SCTP traffic between nodes 5 and 15, which was roughly 0.6 seconds less than that of TCP traffic.

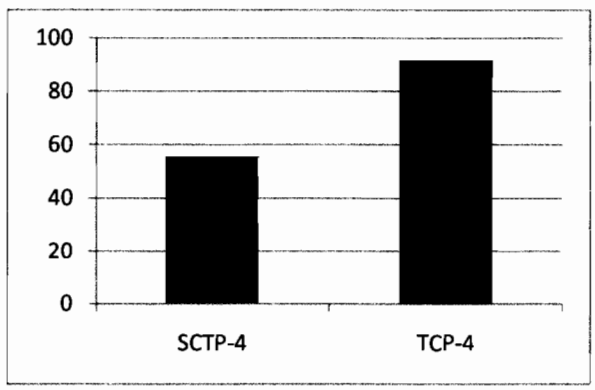

Figure 4.36 Comparison of Node 5 Average Throughput of SCTP and TCP

Figure 4.37 explains the average throughput of SCTP traffic between nodes 6 and 16, which was also around 0.8 seconds less than TCP traffic throughput performance.

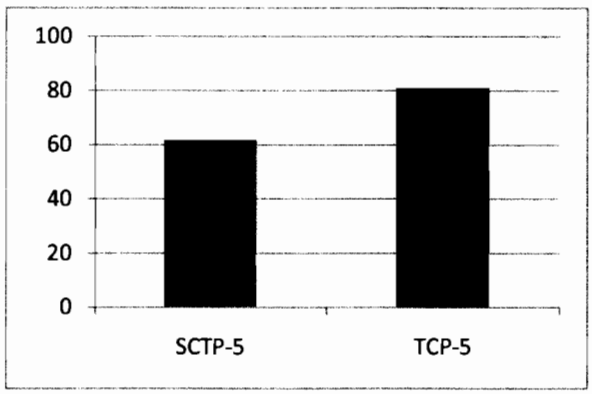

Figure 4.37 Comparison of Node 6 Average Throughput of SCTP and TCP

Figures 4.38, 4.39, 4.40, 4.41 and 4.42 show the throughput of different data streams of data carried over SCTP and TCP protocols. In this figure, throughput of the SCTP streams is shown in green and that of the TCP streams in gray. It can be seen that the throughput of the streams is between 0 and 250 kbps, except for few instances where the TCP stream shows higher throughput (more than 250 kbps). It can also be seen that sometimes the throughput of the SCTP stream becomes zero.

Figure 4.38 shows the average throughput of the SCTP traffic between nodes 2 and 12, which was less than 1.5 seconds compared with the TCP traffic at less than 2.5 seconds.

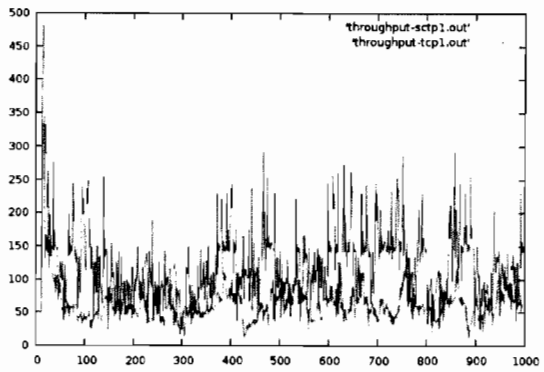

Figure 4.38: Comparison of Nodes 2 Throughput of SCTP and TCP Streams

Figure 4.39 illustrates the SCTP traffic performance between nodes 3 and 13, where it registered throughput around 1.4 seconds, which was less than the 2.4 seconds for TCP traffic.

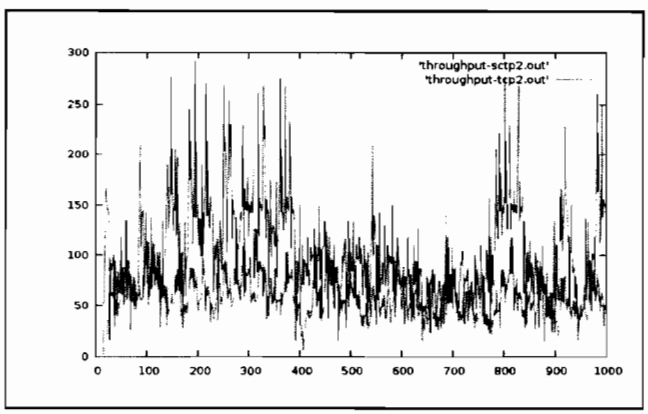

Figure 4.39: Comparison of Nodes 3 Throughput of SCTP and TCP Streams

Figure 4.40 depicts SCTP traffic throughput between nodes 4 and 14, which was less than 1 second compared with more than 2 seconds of TCP traffic.

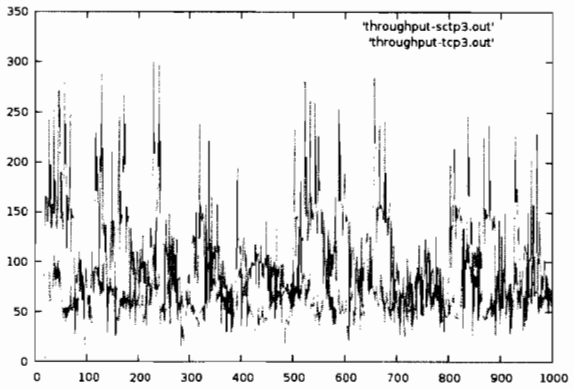

Figure 4.40 Comparison of Nodes 4 Throughput of SCTP and TCP Streams

Figure 4.41 represents the throughput of the SCTP traffic between nodes 5 and 15, and it was in the region of 1 second less than that of TCP, which was around 2 seconds.

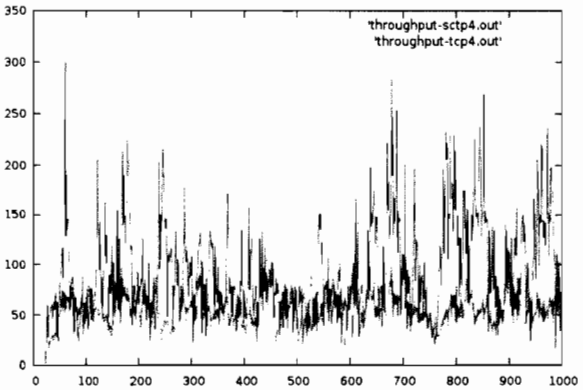

Figure 4.41 Comparison of Node 5 Throughput of SCTP and TCP Streams

Figure 4.42 explains average throughput of SCTP traffic between nodes 6 and 16, which was in the range between 0.3 and 1 second, which was less than the range between 0.5 and 2 seconds for TCP.

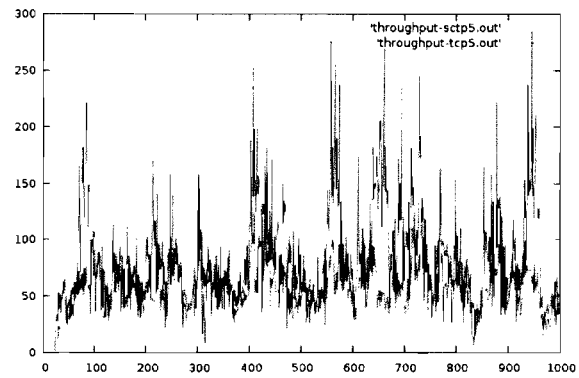

Figure 4.42 Comparison of Node 6 Throughput of SCTP and TCP Streams

Figure 4.43 shows the average throughput by the SCTP and TCP streams in a combined figure. On average, the TCP stream had three times more throughput than the SCTP stream under similar conditions. The TCP throughput is smooth, however, the SCTP throughput was affected by the network environment, where it had a variation in throughput between increasing and decreasing.

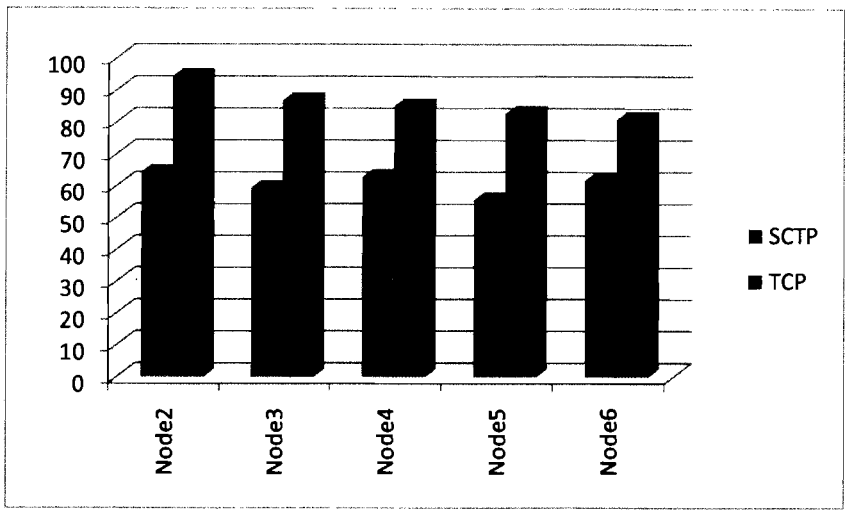

Figure 4.43 Comparison of Pair-wise Throughput of SCTP and TCP Streams

4.5. ITU Recommandation

Table 4.7: ITU Performance Metrics Recommandations (A. Alhuzali, 2010)

Performance metric	TCP Result	SCTP Result	ITU Recommendation
Delay	1269.5616 ms	1133.91 ms	≤ 100 ms
Jitter	13.59906 ms	33.3159 ms	≤ 50 ms

4.6 Summary

This chapter presented the analysis of the results obtained from the experiments carried out in this study. The analysis was based on the performance metrics for delay, delay jitter and throughput. The results of the SCTP and TCP data analysis have been compared with each other in order to determine the relative performance of SCTP and TCP in the

presence of other data traffic in the network. From the analysis carried out, it could be seen that TCP has better performance than SCTP under similar conditions.

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER STUDY

5.1 Introduction

This chapter discusses the findings of the study based on the objectives and research questions as mentioned in Chapter one. The limitations and recommendations are explained in this chapter. In this chapter the findings will be discussed in detail to have a clear conclusion of the hypotheses constructed in Chapter three for the purpose of providing directions for future research.

This study focused on the SCTP protocol by measuring its performance in the best-effort network, where the network has other protocols in terms of delay, jitter and throughput. All experiments have been done by simulating two network topologies through the network simulation NS2. The first experiment was carried out by exchanging the data between the source and destination of SCTP nodes through routers. The second experiment carried out TCP exchange data with TCP in the same network as SCTP, while the network had a different protocol, UDP. The network consists of ten source nodes, ten destination nodes, along with two routers. The five source SCTP nodes would be generating FTP over SCTP traffic destined for the SCTP nodes on the other portion of the network, while the CBR over UDP generated by the rest of the nodes would be used to simulate the actual networking environment. The result indicates that the TCP has the highest average of throughput while SCTP has the highest average of delay and jitter.

The TCP is effective in terms of completion of data flow used with UDP in the same network compared with SCTP. However, some of TCP data flows were affected by UDP traffic in terms of delay and delay jitter performance metrics. Nevertheless, SCTP data flows were affected by the same protocol in terms of measuring jitter performance.

5.2 Discussion of Findings

The main purpose of this study is to measure the performance of SCTP in terms of the best-effort network like SCTP itself and other protocol. It also compares the SCTP performance with another protocol with TCP in the same network environment of TCP and other protocol. Thus, it is necessary to evaluate and compare the performance of SCTP and TCP in this network and stated as follows:

Research objective 1:

The first objective is to investigate the performance of SCTP over a Wired Network in terms of delay, jitter, and throughput in a network environment that uses STCP and UDP. Therefore, first experiment focused on the SCTP and UDP through the use of NS2 simulator, while the result from the experiments showed that the SCTP performance in terms of delay, jitter, and throughput, which also proved by the performance average of SCTP and the graphs. It can be seen that the SCTP has high delay and delay jitter in this type of network which it's competing with data flows affected by UDP in terms of delay jitter. As the result of high delay it has a low throughput.

Research objective 2:

Furthermore, the second objective is to investigate the performance of TCP over a Wired Network in terms of delay, jitter, and throughput in a network environment that uses TCP and UDP. Moreover, the experiment focused on the TCP and UDP through the use of NS2 simulator, while the result from the experiments showed that the TCP performance in terms of delay, jitter, and throughput, which proved by the performance average of TCP and the graphs. It can be seen that the TCP has low delay and delay jitter competing with data flows in this type of network. Which it's affected in some of data flows by UDP in terms of delay and delay jitter. As the result of low delay it has a high throughput.

Research objective 3:

This objective is to compare the SCTP and TCP performance results in terms of delay, jitter, and throughput. According to the experiments, the result achieved Objectives one and two by comparing SCTP and TCP performance in terms of listed metrics. Therefore, the study showed that the SCTP has higher jitter and delay than TCP, while TCP has the higher throughput when compared with SCTP. This means that the TCP performance is better than SCTP performance in the network using UDP protocol.

5.3 Limitations

The following have been identified as the limitations of this study:

1. Only three parameters have been selected for the purpose of comparison of the protocols. It is better to compare the performance of the protocols based on an extensive set of metrics.
2. The experimental setup has been rather artificial with 20 nodes with similar capacities. If the number of nodes can be made large and different types of nodes included in the experiment, the results would be more towards the real world performance.
3. Only the FTP and CBR traffic have been used in the experiment, it would be better to test the performance of the protocols based on other types of application traffic including real time and non-real time requirements.

5.4 Contributions

This study gave a deep analysis of SCTP and TCP protocols over a Wired Network with different protocols in the same network. It used UDP with the performance metrics, which were throughput, packet delay and jitter. Moreover, this study has pointed out the SCTP behavior in the wired network using SCTP and UDP protocols, and compared their behavior with TCP behavior. It has also provided important information for further studies on FTP protocols over SCTP and TCP protocols in the network that has CBR over UDP on the wired network. In conclusion, the research showed some information about NS-2 as a useful simulator for the prominent network protocols and the analysis.

5.5 Future Work

Based on the achievement of this study, the researcher suggests that future researchers who would like to work on this topic to measure the convergence time and the packet loss. It is recommended conducting research in the following fields:

1. Similar topologies with same background traffic generated in the wired network.
2. Using similar protocols with five nodes for each source and destination in the wired network.
3. Similar topologies with different packet sizes and same start time for data flows and same propagation delay in the wired network.

References

Al-Kaisan, A., Ashrafuzzaman, M., and Ahsan, S. (2007). Reducing Congestion Collapse and Promoting Fairness in the Internet by Optimizing SCTP",. *10th International Conference on Computer and Information Technology, iccit* (pp. 1-5). Bangladesh : IEEE Computer Society.

Almes, G., Kalidindi, S., and Zekauskas, M. (1999). A Round-trip Delay Metric for IPPM. *RFC2681* , 4-14.

Alnuem, M., Mellor, J., and Fretwell, R. (2009). New algorithm to control TCP behavior over lossy links. *International Conference on Advanced Computer Control, DOI 10.1109/ICACC* (pp. 1-5). UK: IEEE Computer Society.

Amer, P., & Stewart, R. (2005). *Why is SCTP needed given TCP and UDP are widely available?* Virginia,USA: http://www.isoc.org/briefings/017/,Copyright C Internet Society.

Boussen, S., Tabbane, N., and Tabbane, S. (2009). Performance analysis of SCTP protocol in WiFi network. *Fourth International Conference on Computer Sciences and Convergence Information Technology* (pp. 178 – 182). Tunisia : IEEE.

Bradner, S., & McQuaid, J. (1999). Benchmarking Methodology for Network Interconnect Devices. *RFC 2544* , 4.

Byun, H., and Lim, J. (2005). Explicit window adaptation algorithm over TCP wireless networks . *IEEE Communications Conference* (pp. 1-6). IEEE Computer Society.

Chaeng, R., Lai, C., Huang, Y., and Chou, I. (2009). Multi-Stream Bandwidth Estimation for SCTP in High-Speed Networks . *Communications and Networking Fourth International Conference* (pp. 1-4). China: IEEE Computer Society.

Cheng, R., Deng, D., Chao, H., and Chen. (2010). An Adaptive Bandwidth Estimation Mechanism for SCTP over Wireless Networks. (pp. 1-6). Taiwan : IEEE Computer Society.

Cheng, R., Deng, D., Chao, H., and Chen, W. (2009). Performance analysis of SCTP protocol in WiFi network. *Fourth International Conference on Computer Sciences and Convergence Information Technology* (pp. 1-6). Taiwan : IEEE Computer Society.

Corsepius, R., Norum, E., Johns, C., Straumann, T., and Sherrill, J. (2010). RTEMS Operating System. *http://www.rtems.com* .

Dahlander, L., and Magnusson M. G. (2005). Relationships between open source software companies and communities. *Research Policy* , 481-493.

de Souza, E., & Agarwal, D. (2003). A HighSpeed TCP Study: Characteristics and Deployment Issues. *Lawrence Berkeley National Lab* (pp. 1-12). California,USA: Berkeley, CA.

Del Rey, M. (1981). Transmission Control Protocol-Darpa Internet Program-Protocol Specification. *RFC 793* , 9-13.

Demichelis, C., & Chimento, P. (2002). IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). *RFC 3393* , 6-13.

Deru, M., & Torcellini, P. (2005). *Performance Metrics Research-Project – Final Report.* Colorado: National Renewable Energy Laboratory.

Dorion, P. (2006). What is the difference between RPO and RTO (from a backup perspective)? *http://searchstorage.techtarget.com/generic/0,295582,sid5_gci1212112,00.html* .

Emma, D., Loreto, S., Pescap´e, A., and Ventre, G. (2006). Measuring SCTP Throughput and Jitter over Heterogeneous Networks. *20th International Conference on Advanced Information Networking and Applications,AINA* (pp. 1-5). IEEE Computer Society.

Fallon, S., Jacob, P., Qiao, Y., Murphy, L., Fallon, E., and Hanley, A. (2008). SCTP Switchover Performance Issues in WLAN Environments",. *IEEE CCNC proceeding* (pp. 1-5). Dublin, Ireland : IEEE Computer Society.

Floyd, S., Handley, M., Padhye, J., and Widmer, J. (September 2008). TCP Friendly Rate Control (TFRC): Protocol Specification. *RFC 5348* (pp. 7-10). London: University College London.

Fu, S., & Atiquzzaman, M. (2003). Improving End-to-End Throughput of Mobile IP using SCTP. (pp. 1-6). USA: IEEE Computer Society.

Funasaka, J., Neyagawa, T., and Ishida, K. (2010). Effect of SCTP Multistream Function Applied to Parallel Downloading. *IEEE Second International Conference on Communication Software and Networks(iccsn)* (pp. 1-5). Hiroshima: IEEE Computer Society.

Han, Y., Hwang, I., Kim, C., and Park, H. (2010). A New Attainable TCP Throughput Measurement Tool for Long Distance High Speed Networks. *IEEE Communications Letters, 10.1109/LCOMM* (pp. 1-5). IEEE Computer Society.

Hassan, M., Fahmy, S., Wu, J., Aziz, A.,. (2004). ", Chapter 4 page: – by Pearson Education, Inc, Pearson Prentice Hall, Upper Saddle River,, NJ07458. In M. a. Hassan, *High Performance TCP/IP Networking* (pp. 76-79). USA: Alan R. Apt.

Honda, M., Sakakibara, H., Nishida, Y., and Tokuda, H. (2007). SmSCTP: A Fast Transport Layer Handover Method Using Single Wireless Interface . (pp. 1-6). Japan: IEEE Computer Society.

Hunt, C. (1997). TCP/IP protocol architecure. In *TCP/IP network administration* (pp. 9-23). Sebastopol CA,USA: O'Reilly Media.

Hurtig, P., & Brunstrom, A. (2008). Improved Loss Detection for Signaling Traffic in SCTP. *IEEE Communications Society subject matter experts for publication in the ICC* (pp. 1-6). IEEE Computer Society.

Islam, N., and Kara, A. (2006). Throughput Analysis of SCTP over a Multi-homed Association",. *Computer and Information Technology,CIT '06. The Sixth IEEE International Conference* (pp. 1-6). Fukushima-ken,Japan: IEEE Computer Society.

Joe, I., YAN, S. (2009). SCTP Throughput Improvement with Best Load Sharing based on Multihoming. *Fifth International Joint Conference on INC, IMS and IDC* (pp. 1-5). Seoul, Korea: IEEE Computer Society.

Kaytan, M. (2010). *TCP Versus UDP Performance In Term Of Bandwidth Usage* . Sintok-Malaysia: Universiti Utara Malaysia.

Kuhl, M., Kistner, J., Costantini, K., Sudit, M. (2007). Cyber attack modeling and simulation for network security analysis. *39th conference on Winter simulation by ACM* (pp. 1-9). IEEE Computer Society.

Leu, F., & Ko, Z. (2008). A Novel Network Mobility Scheme Using SIP and SCTP for Multimedia Applications . *International Conference on Multimedia and Ubiquitous Engineering* (pp. 564-569). Taiwan : IEEE.

Ma, L., Yu, F., and Leung, V ung. (2005). Modeling SCTP Throughput in Integrated WLAN/Cellular Networks. *IEEE International Communications Conference* (pp. 3445 - 3449). Vancouver,Canada: IEEE Computer Society.

Mcclellan, S. (2003). Active path selection for SCTP. *Patent application publication* , 1-4.

Mohammed, M. (2010). *A Comparison of Performance between TFRC and UDP over a Mobile IP Network.* Sintok, Malaysia: Universiti Utara Malaysia.

Muller, N. (2009). *SCTP Administrator's Guide.* Boston,U.S: Hewlett-Packard Development Company, L.P.

Nagamalai, D., Lee, J. (2004). Performance of SCTP over high speed Wide area network. *Conference on Cybernetics and Intelligent Systems* (pp. 1-6). Singapore: IEEE Computer Society.

R, A. A. (2010). *Performance analysis of TFRC and UDP over Mobile-IP network with competing flows.* Sintok, Kedah Dar-ul-Aman: University Utara Malaysia.

Rahim, S., & Faisal-Hasan, S. (2009). Performance Evaluation of Fast TCP and TCP Westwood+ forMultimedia Streaming in Wireless Environment. *12 th International Conference on Computer and Information Technology (ICCIT 2009)* (pp. 1-6). Dhaka, Bangladesh: IEEE Computer Society.

Rahman, S., Atiquzzaman, M., Ivancic, W., Eddy, W., and Stewart, D. (2008). Implementation of SCTP in an open source REAL-TIME operating system. (pp. 1-7). IEEE Computer Society.

Rane, J., Kumbhar, N., and Sovani, K. (2002). *Stream Control Transmission Protocol (SCTP) on FreeBSD.* India-Pune : Pune Institute of Computer Technology - Affiliated to University of Pune.

Scharf, M., & Kiesel, S. (2006). Head-of-line Blocking in TCP and SCTP: Analysis and Measurements. *IEEE Communications Society* (pp. 1-5). Germany: IEEE GLOBECOM.

Stewart, R., T¨uxen, M., and Lei, P. (2008). SCTP: What is it, and how to use it? *IEEE conference* (pp. 1-10). IEEE Computer Society.

Takemoto, Y., Funasaka, J., Teshima, S., Ohta, T., and Kakuda, Y. (2008). SCTP Performance Improvement for Reliable End-to-end Communication in Ad Hoc Networks. *IEEE conference in Future Generation Communication and Networking, (FGCN)* (pp. 1-6). Hiroshima, Japan: IEEE Computer Society.

ThinkQuest, O. (1999). TCP Features. *http://library.thinkquest.org/28289/tcpfeatures.html#* .

Valera, A., Seah, W., and Rao, S. (2003). Cooperative Packet Caching and Shortest Multipath Routing in Mobile Ad hoc Networks. (pp. 1-10). Singapore: IEEE Computer Society.

Wang, L., Kawanishi, K., and Onozato, Y. (2009). Achieving Robust Fairness of SCTP Extension for MPEG-4 Streaming. (pp. 1-5). Japan: IEEE Computer Society.

Wolaver, D. H. (1991). *Phase - locked loop circuit design.* Prentice-Hall.

Xylomenos, G., Polyzos, G., Mahonen , P., Saaranen, M. (2001). TCP performance issues over wireless links. *IEEE Communications Magazine Conference* (pp. 1-7). IEEE Computer Society.

Yang, C., Chang, W., and Huang, I. (2007). CS-SCTP: A Collaborative Approach for Secure SCTP over Wireless Networks. (pp. 1-4). Taiwan: IEEE Computer Society.

Zhao, B., Vishwanath, A., and Sivaraman, V. (2007). Performance of high-speed TCP applications in networks with very small buffers . *First International Symposium on Advanced Networks and Telecommunication Systems* (pp. 1-2). Australia: IEEE Computer Society.

APPENDIX A: NS-2 Code

1.0. SCTP Topology

```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows
$ns color 1 Blue
$ns color 2 Red
$ns color 3 Yellow
$ns color 4 Green
$ns color 5 Orange
$ns color 6 Purple
$ns color 7 maroon
$ns color 8 navy
$ns color 9 black
$ns color 10 gray

#open trace file
set tf [open sctpudp.tr w]
$ns trace-all $tf

#Open the nam trace file
set nf [open sctpudp.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
        global ns tf nf
        $ns flush-trace
      #Close the trace file
        close $tf
        close $nf
      #Execute nam on the trace file
        exec nam sctpudp.nam &
        exit 0
}

#Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]
set n10 [$ns node]
```

```
set n11 [$ns node]
set n12 [$ns node]
set n13 [$ns node]
set n14 [$ns node]
set n15 [$ns node]
set n16 [$ns node]
set n17 [$ns node]
set n18 [$ns node]
set n19 [$ns node]
set n20 [$ns node]
set n21 [$ns node]

#Create links between the nodes
$ns duplex-link $n2 $n0 1Mb 10ms DropTail
$ns duplex-link $n3 $n0 1Mb 10ms DropTail
$ns duplex-link $n4 $n0 1Mb 10ms DropTail
$ns duplex-link $n5 $n0 1Mb 10ms DropTail
$ns duplex-link $n6 $n0 1Mb 10ms DropTail
$ns duplex-link $n7 $n0 1Mb 10ms DropTail
$ns duplex-link $n8 $n0 1Mb 10ms DropTail
$ns duplex-link $n9 $n0 1Mb 10ms DropTail
$ns duplex-link $n10 $n0 1Mb 10ms DropTail
$ns duplex-link $n11 $n0 1Mb 10ms DropTail
$ns duplex-link $n0 $n1 0.5Mb 100ms DropTail
$ns duplex-link $n1 $n0 0.5Mb 100ms DropTail
$ns duplex-link $n12 $n1 1Mb 30ms DropTail
$ns duplex-link $n13 $n1 1Mb 30ms DropTail
$ns duplex-link $n14 $n1 1Mb 30ms DropTail
$ns duplex-link $n15 $n1 1Mb 30ms DropTail
$ns duplex-link $n16 $n1 1Mb 30ms DropTail
$ns duplex-link $n17 $n1 1Mb 30ms DropTail
$ns duplex-link $n18 $n1 1Mb 30ms DropTail
$ns duplex-link $n19 $n1 1Mb 30ms DropTail
$ns duplex-link $n20 $n1 1Mb 30ms DropTail
$ns duplex-link $n21 $n1 1Mb 30ms DropTail


#Create a sctp agent and attach it to node n2
set sctp2 [new Agent/SCTP]
$ns attach-agent $n2 $sctp2
set sink2 [new Agent/SCTP]
$ns attach-agent $n12 $sink2
$ns connect $sctp2 $sink2
#$sctp2 set fid_ 1

# Create a FTP traffic source and attach it to sctp2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $sctp2
$ftp2 set packetSize_ 1000
$ftp2 set rate_ 0.01Mb
$ftp2 set random_ false

#Create a sctp agent and attach it to node n3
set sctp3 [new Agent/SCTP]
```

```
$ns attach-agent $n3 $sctp3
set sink3 [new Agent/SCTP]
$ns attach-agent $n13 $sink3
$ns connect $sctp3 $sink3
#$sctp3 set fid_ 2


# Create a ftp traffic source and attach it to sctp3
set ftp3 [new Application/FTP]
$ftp3 attach-agent $sctp3
$ftp3 set packetSize_ 1000
$ftp3 set rate_ 0.01Mb
$ftp3 set random_ false

#Create a sctp agent and attach it to node n4
set sctp4 [new Agent/SCTP]
$ns attach-agent $n4 $sctp4
set sink4 [new Agent/SCTP]
$ns attach-agent $n14 $sink4
$ns connect $sctp4 $sink4
#$sctp4 set fid_ 3

# Create a ftp traffic source and attach it to sctp4
set ftp4 [new Application/FTP]
$ftp4 attach-agent $sctp4
$ftp4 set packetSize_ 1000
$ftp4 set rate_ 0.01Mb
$ftp4 set random_ false

#Create a sctp agent and attach it to node n5
set sctp5 [new Agent/SCTP]
$ns attach-agent $n5 $sctp5
set sink5 [new Agent/SCTP]
$ns attach-agent $n15 $sink5
$ns connect $sctp5 $sink5
#$sctp5 set fid_ 4

# Create a ftp traffic source and attach it to sctp5
set ftp5 [new Application/FTP]
$ftp5 attach-agent $sctp5
$ftp5 set packetSize_ 1000
$ftp5 set rate_ 0.01Mb
$ftp5 set random_ false

#Create a sctp agent and attach it to node n6
set sctp6 [new Agent/SCTP]
$ns attach-agent $n6 $sctp6
set sink6 [new Agent/SCTP]
$ns attach-agent $n16 $sink6
$ns connect $sctp6 $sink6
#$sctp6 set fid_ 5

# Create a ftp traffic source and attach it to sctp6
set ftp6 [new Application/FTP]
```

```
$ftp6 attach-agent $sctp6
$ftp6 set packetSize_ 1000
$ftp6 set rate_ 0.01Mb
$ftp6 set random_ false


#Create a udp agent and attach it to node n7
set udp7 [new Agent/UDP]
$ns attach-agent $n7 $udp7
set sink7 [new Agent/Null]
$ns attach-agent $n17 $sink7
$ns connect $udp7 $sink7
#$tcp7 set fid_ 6

# Create a CBR traffic source and attach it to udp7
set cbr7 [new Application/Traffic/CBR]
$cbr7 attach-agent $udp7
$cbr7 set packetSize_ 1000
$cbr7 set rate_ 0.01Mb
$cbr7 set random_ false

#Create a udp agent and attach it to node n8
set udp8 [new Agent/UDP]
$ns attach-agent $n8 $udp8
set sink8 [new Agent/Null]
$ns attach-agent $n18 $sink8
$ns connect $udp8 $sink8
#$tcp8 set fid_ 7

# Create a CBR traffic source and attach it to udp8
set cbr8 [new Application/Traffic/CBR]
$cbr8 attach-agent $udp8
$cbr8 set packetSize_ 1000
$cbr8 set rate_ 0.01Mb
$cbr8 set random_ false


#Create a udp agent and attach it to node n9
set udp9 [new Agent/UDP]
$ns attach-agent $n9 $udp9
set sink9 [new Agent/Null]
$ns attach-agent $n19 $sink9
$ns connect $udp9 $sink9
#$tcp9 set fid_ 8

# Create a CBR traffic source and attach it to udp9
set cbr9 [new Application/Traffic/CBR]
$cbr9 attach-agent $udp9
$cbr9 set packetSize_ 1000
$cbr9 set rate_ 0.01Mb
$cbr9 set random_ false

#Create a udp agent and attach it to node n10
set udp10 [new Agent/UDP]
```

```
$ns attach-agent $n10 $udp10
set sink10 [new Agent/Null]
$ns attach-agent $n20 $sink10
$ns connect $udp10 $sink10
#$tcp10 set fid_ 9

# Create a CBR traffic source and attach it to udp10
set cbr10 [new Application/Traffic/CBR]
$cbr10 attach-agent $udp10
$cbr10 set packetSize_ 1000
$cbr10 set rate_ 0.01Mb
$cbr10 set random_ false

#Create a udp agent and attach it to node n11
set udp11 [new Agent/UDP]
$ns attach-agent $n11 $udp11
set sink11 [new Agent/Null]
$ns attach-agent $n21 $sink11
$ns connect $udp11 $sink11
#$tcp11 set fid_ 10

# Create a CBR traffic source and attach it to udp11
set cbr11 [new Application/Traffic/CBR]
$cbr11 attach-agent $udp11
$cbr11 set packetSize_ 1000
$cbr11 set rate_ 0.01Mb
$cbr11 set random_ false


#Schedule events for the CBR agents
$ns at 10 "$ftp2 start"
$ns at 15 "$ftp3 start"
$ns at 15 "$ftp4 start"
$ns at 20 "$ftp5 start"
$ns at 20 "$ftp6 start"
$ns at 30 "$cbr7 start"
$ns at 35 "$cbr8 start"
$ns at 30 "$cbr9 start"
$ns at 40 "$cbr10 start"
$ns at 50 "$cbr11 start"

#Call the finish procedure after 5 seconds of simulation time
$ns at 1000 "finish"

#Run the simulation
$ns run
```

2.0. TCP Topology

```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows
$ns color 1 Blue
$ns color 2 Red
$ns color 3 Yellow
$ns color 4 Green
$ns color 5 Orange
$ns color 6 Purple
$ns color 7 maroon
$ns color 8 navy
$ns color 9 black
$ns color 10 gray

#open trace file
set tf [open newtcpudp.tr w]
$ns trace-all $tf

#Open the nam trace file
set nf [open newtcpudp.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
        global ns tf nf
        $ns flush-trace
      #Close the trace file
        close $tf
        close $nf
      #Execute nam on the trace file
        exec nam newtcpudp.nam &
        exit 0
}

#create nodes
#Router Nodes
set r0 [$ns node]
set r1 [$ns node]

#Source Nodes SCTP
set s_tcp1 [$ns node]
set s_tcp2 [$ns node]
set s_tcp3 [$ns node]
set s_tcp4 [$ns node]
set s_tcp5 [$ns node]


#Source Nodes UDP
set s_udp1 [$ns node]
```

```
set s_udp2 [$ns node]
set s_udp3 [$ns node]
set s_udp4 [$ns node]
set s_udp5 [$ns node]

#Destination Nodes SCTP
set d_tcp1 [$ns node]
set d_tcp2 [$ns node]
set d_tcp3 [$ns node]
set d_tcp4 [$ns node]
set d_tcp5 [$ns node]

#Destination Nodes UDP
set d_udp1 [$ns node]
set d_udp2 [$ns node]
set d_udp3 [$ns node]
set d_udp4 [$ns node]
set d_udp5 [$ns node]

#Create links between the nodes
$ns duplex-link $s_tcp1 $r0 1Mb 10ms DropTail
$ns duplex-link $s_tcp2 $r0 1Mb 10ms DropTail
$ns duplex-link $s_tcp3 $r0 1Mb 10ms DropTail
$ns duplex-link $s_tcp4 $r0 1Mb 10ms DropTail
$ns duplex-link $s_tcp5 $r0 1Mb 10ms DropTail
$ns duplex-link $s_udp1 $r0 1Mb 10ms DropTail
$ns duplex-link $s_udp2 $r0 1Mb 10ms DropTail
$ns duplex-link $s_udp3 $r0 1Mb 10ms DropTail
$ns duplex-link $s_udp4 $r0 1Mb 10ms DropTail
$ns duplex-link $s_udp5 $r0 1Mb 10ms DropTail
$ns duplex-link $r0 $r1 0.5Mb 100ms DropTail
$ns duplex-link $r1 $r0 0.5Mb 100ms DropTail
$ns duplex-link $d_tcp1 $r1 1Mb 30ms DropTail
$ns duplex-link $d_tcp2 $r1 1Mb 30ms DropTail
$ns duplex-link $d_tcp3 $r1 1Mb 30ms DropTail
$ns duplex-link $d_tcp4 $r1 1Mb 30ms DropTail
$ns duplex-link $d_tcp5 $r1 1Mb 30ms DropTail
$ns duplex-link $d_udp1 $r1 1Mb 30ms DropTail
$ns duplex-link $d_udp2 $r1 1Mb 30ms DropTail
$ns duplex-link $d_udp3 $r1 1Mb 30ms DropTail
$ns duplex-link $d_udp4 $r1 1Mb 30ms DropTail
$ns duplex-link $d_udp5 $r1 1Mb 30ms DropTail

#Create a tcp agent and attach it to node s_tcp1
set tcp1 [new Agent/TCP]
$ns attach-agent $s_tcp1 $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $d_tcp1 $sink1
$ns connect $tcp1 $sink1
#$tcp2 set fid_ 6

# Create a ftp traffic source and attach it to tcp1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
```

```
#$ftp2 set type_ FTP
#$ftp2 set packetSize_ 1000
#$ftp2 set rate_ 0.01Mb
#$ftp2 set random_ false

#Create a tcp agent and attach it to node s_tcp2
set tcp2 [new Agent/TCP]
$ns attach-agent $s_tcp2 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $d_tcp2 $sink2
$ns connect $tcp2 $sink2
#$tcp8 set fid_ 7

# Create a ftp traffic source and attach it to tcp2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
#$ftp2 set type_ FTP
#$ftp3 set packetSize_ 1000
#$ftp3 set rate_ 0.01Mb
#$ftp3 set random_ false


#Create a tcp agent and attach it to node s_tcp3
set tcp3 [new Agent/TCP]
$ns attach-agent $s_tcp3 $tcp3
set sink3 [new Agent/TCPSink]
$ns attach-agent $d_tcp3 $sink3
$ns connect $tcp3 $sink3
#$tcp9 set fid_ 8

# Create a ftp traffic source and attach it to tcp3
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
#$ftp4 set packetSize_ 1000
#$ftp4 set rate_ 0.01Mb
#$ftp4 set random_ false

#Create a tcp agent and attach it to node s_tcp4
set tcp4 [new Agent/TCP]
$ns attach-agent $s_tcp4 $tcp4
set sink4 [new Agent/TCPSink]
$ns attach-agent $d_tcp4 $sink4
$ns connect $tcp4 $sink4
#$tcp10 set fid_ 9

# Create a ftp traffic source and attach it to tcp4
set ftp4 [new Application/FTP]
$ftp4 attach-agent $tcp4
#$ftp2 set type_ FTP
#$ftp5 set packetSize_ 1000
#$ftp5 set rate_ 0.01Mb
#$ftp5 set random_ false

#Create a sctp agent and attach it to node s_tcp5
```

```
set tcp5 [new Agent/TCP]
$ns attach-agent $s_tcp5 $tcp5
set sink5 [new Agent/TCPSink]
$ns attach-agent $d_tcp5 $sink5
$ns connect $tcp5 $sink5
#$tcp11 set fid_ 10

# Create a ftp traffic source and attach it to tcp5
set ftp5 [new Application/FTP]
$ftp5 attach-agent $tcp5
#$ftp2 set type_ FTP
#$ftp6 set packetSize_ 1000
#$ftp6 set rate_ 0.01Mb
#$ftp6 set random_ false


#Create a udp agent and attach it to node s_udp1
set udp1 [new Agent/UDP]
$ns attach-agent $s_udp1 $udp1
set sink7 [new Agent/Null]
$ns attach-agent $d_udp1 $sink7
$ns connect $udp1 $sink7
#$tcp7 set fid_ 6

# Create a CBR traffic source and attach it to udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 0.01Mb
$cbr1 set random_ false

#Create a udp agent and attach it to node s_udp2
set udp2 [new Agent/UDP]
$ns attach-agent $s_udp2 $udp2
set sink8 [new Agent/Null]
$ns attach-agent $d_udp2 $sink8
$ns connect $udp2 $sink8
#$tcp8 set fid_ 7

# Create a CBR traffic source and attach it to udp2
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 0.01Mb
$cbr2 set random_ false

#Create a udp agent and attach it to node s_udp3
set udp3 [new Agent/UDP]
$ns attach-agent $s_udp3 $udp3
set sink9 [new Agent/Null]
$ns attach-agent $d_udp3 $sink9
$ns connect $udp3 $sink9
#$tcp9 set fid_ 8
```

```
# Create a CBR traffic source and attach it to udp3
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp3
$cbr3 set packetSize_ 1000
$cbr3 set rate_ 0.01Mb
$cbr3 set random_ false

#Create a udp agent and attach it to node s_udp4
set udp4 [new Agent/UDP]
$ns attach-agent $s_udp4 $udp4
set sink10 [new Agent/Null]
$ns attach-agent $d_udp4 $sink10
$ns connect $udp4 $sink10
#$tcp10 set fid_ 9

# Create a CBR traffic source and attach it to udp4
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 0.01Mb
$cbr4 set random_ false

#Create a udp agent and attach it to node s_udp5
set udp5 [new Agent/UDP]
$ns attach-agent $s_udp5 $udp5
set sink11 [new Agent/Null]
$ns attach-agent $d_udp5 $sink11
$ns connect $udp5 $sink11
#$tcp11 set fid_ 10

# Create a CBR traffic source and attach it to udp5
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp5
$cbr5 set packetSize_ 1000
$cbr5 set rate_ 0.01Mb
$cbr5 set random_ false

#Schedule events for the CBR agents
$ns at 10 "$ftp1 start"
$ns at 15 "$ftp2 start"
$ns at 15 "$ftp3 start"
$ns at 20 "$ftp4 start"
$ns at 20 "$ftp5 start"
$ns at 30 "$cbr1 start"
$ns at 35 "$cbr2 start"
$ns at 30 "$cbr3 start"
$ns at 40 "$cbr4 start"
$ns at 50 "$cbr5 start"

#Call the finish procedure after 5 seconds of simulation time
$ns at 1000 "finish"

#Run the simulation
$ns run
```

APPENDIX B: Performance Metrics Code

1.0. Throughput

```
BEGIN {
      recv = 0
      currTime = prevTime = 0

}

{
      # Trace line format: normal
      if ($2 != "-t") {
            event = $1
            time = $2
            if (event == "+" || event == "-") node_id = $3
            if (event == "r" || event == "d") node_id = $4
            flow_id = $8
            pkt_id = $12
            pkt_size = $6
            flow_t = $5
            level = "AGT"
      }
      # Trace line format: new
      if ($2 == "-t") {
            event = $1
            time = $3
            node_id = $5
            flow_id = $39
            pkt_id = $41
            pkt_size = $37
            flow_t = $45
            level = $19
      }

      # Init prevTime to the first packet recv time
      if(prevTime == 0)
            prevTime = time

      # Calculate total received packets' size
      if (level == "AGT" && flow_id == flow && node_id == dst &&
          event == "r" && pkt_size >= pkt) {
            # Rip off the header
            hdr_size = pkt_size % pkt
            pkt_size -= hdr_size
            # Store received packet's size
            recv += pkt_size
            # This 'if' is introduce to obtain clearer
            # plots from the output of this script
            if((time - prevTime) >= tic*10) {
                  printf("  %10g %10s %5d %5d %15g %18g\n", \
                        flow,flow_t,src,dst,(prevTime+1.0),0)
```

```
                printf("  %10g %10s %5d %5d %15g %18g\n", \
                      flow,flow_t,src,dst,(time-1.0),0)
            }
            currTime += (time - prevTime)
            if (currTime >= tic) {
                printf(" %15g %18g\n", \
                       time,(recv/currTime)*(8/1000))
                recv = 0
                currTime = 0
            }
            prevTime = time
      }

}

END {
      printf("\n\n")
}
```

2.0. Jitter

```
BEGIN {
      num_recv = 0
            }

{
      # Trace line format: normal
      if ($2 != "-t") {
            event = $1
            time = $2
            if (event == "+" || event == "-") node_id = $3
            if (event == "r" || event == "d") node_id = $4
            flow_id = $8
            pkt_id = $12
            pkt_size = $6
            flow_t = $5
            level = "AGT"
      }
      # Trace line format: new
      if ($2 == "-t") {
            event = $1
            time = $3
            node_id = $5
            flow_id = $39
            pkt_id = $41
            pkt_size = $37
            flow_t = $45
            level = $19
      }
```

```
      # Store packets send time
      if (level == "AGT" && flow_id == flow && node_id == src &&
          sendTime[pkt_id] == 0 && (event == "+" || event == "s") &&
pkt_size >= pkt) {
            sendTime[pkt_id] = time
      }

      # Store packets arrival time
      if (level == "AGT" && flow_id == flow && node_id == dst &&
          event == "r" && pkt_size >= pkt) {
            recvTime[pkt_id] = time
            num_recv ++
      }
}

END {
      # Compute average jitter
      jitter1 = tmp_recv = 0
      prev_time = delay = prev_delay = processed = currTime = 0
      prev_delay = -1
      for (i=0; processed<num_recv; i++) {
            if(recvTime[i] != 0) {
                  tmp_recv++
                if(prev_time != 0) {
                      delay = recvTime[i] - prev_time
                      e2eDelay = recvTime[i] - sendTime[i]
                      if(delay < 0) delay = 0
                      if(prev_delay != -1) {
                            jitter1 += abs(e2eDelay -
prev_e2eDelay)

                      }
                      # This 'if' is introduce to obtain clearer
                      # plots from the output of this script
                      if(delay >= tic*10) {
                            printf("  %10g %10s %5d %5d %15g
%18g\n", \

      flow,flow_t,src,dst,(prev_time+1.0),0)
                            printf("  %10g %10s %5d %5d %15g
%18g\n", \
                                  flow,flow_t,src,dst,(recvTime[i]-
1.0),0)
                      }
                      currTime += delay
                      if (currTime >= tic) {
                            printf("%15g %16g\n", \
                                  recvTime[i],jitter1*1000/tmp_recv)
                                        jitter1=0
                            currTime = 0
                            tmp_recv = 0
                      }
                      prev_delay = delay
                      prev_e2eDelay = e2eDelay
```

```
                    }
                    prev_time = recvTime[i]
                    processed++
                }
        }
}

END {
        printf("\n\n")
}

function abs(value) {
        if (value < 0) value = 0-value
        return value
}
```

3.0 Delay

```
BEGIN {
      num_recv = 0

}

{
      # Trace line format: normal
      if ($2 != "-t") {
            event = $1
            time = $2
            if (event == "+" || event == "-") node_id = $3
            if (event == "r" || event == "d") node_id = $4
            flow_id = $8
            pkt_id = $12
            pkt_size = $6
            flow_t = $5
            level = "AGT"
      }
      # Trace line format: new
      if ($2 == "-t") {
            event = $1
            time = $3
            node_id = $5
            flow_id = $39
            pkt_id = $41
            pkt_size = $37
            flow_t = $45
            level = $19
      }

      # Store packets send time
      if (level == "AGT" && flow_id == flow && node_id == src &&
          sendTime[pkt_id] == 0 && (event == "+" || event == "s") &&
pkt_size >= pkt) {
            sendTime[pkt_id] = time
      }

      # Store packets arrival time
      if (level == "AGT" && flow_id == flow && node_id == dst &&
          event == "r" && pkt_size >= pkt) {
            recvTime[pkt_id] = time
            num_recv ++
      }
}

END {
      # Compute average e2eDelay
      e2eDelay = tmp_recv = 0
      prev_time = delay = prev_delay = processed = currTime = 0
      prev_delay = -1
```

```
      for (i=0; processed<num_recv; i++) {
            if(recvTime[i] != 0) {
                  tmp_recv++
                if(prev_time != 0) {
                      delay = recvTime[i] - prev_time
                      e2eDelay = recvTime[i] - sendTime[i]
                      if(delay < 0) delay = 0
                      if(prev_delay != -1) {
                            jitter1 += abs(e2eDelay -
prev_e2eDelay)

                      }
                      # This 'if' is introduce to obtain clearer
                      # plots from the output of this script
                      if(delay >= tic*10) {
                            printf("  %10g %10s %5d %5d %15g
%18g\n", \

      flow,flow_t,src,dst,(prev_time+1.0),0)
                            printf("  %10g %10s %5d %5d %15g
%18g\n", \
                                  flow,flow_t,src,dst,(recvTime[i]-
1.0),0)
                      }
                      currTime += delay
                      if (currTime >= tic) {
                            printf("%15g %16g\n", \
                                  recvTime[i],delay)
                                        currTime = 0
                            tmp_recv = 0
                      }
                      prev_delay = delay
                      prev_e2eDelay = e2eDelay
                }
                prev_time = recvTime[i]
                processed++
            }
      }
}

END {
      printf("\n\n")
}

function abs(value) {
      if (value < 0) value = 0-value
      return value
}
```