NUST COLLEGE OF
ELECTRICAL AND MECHANICAL ENGINEERING

PERFORMANCE COMPARISON OF DIFFERENT TCP VARIANTS IN IP AND MPLS NETWORKS

A PROJECT REPORT

MS-6 (DCE)

Submitted by

MADIHA KAZMI

(2006-NUST-MS PhD-CSE (E)-37)

MASTERS

IN

SOFTWARE ENGINEERING

YEAR

2009

PROJECT SUPERVISOR

DR.MUHAMMAD YOUNUS JAVED

COLLEGE OF
ELECTRICAL AND MECHANICAL ENGINEERING
PESHAWAR ROAD, RAWALPINDI

YEAR 2009

Performance comparison of different TCP Variants in IP and MPLS Networks

By
Madiha Kazmi
(2006-NUST-MS PhD-CSE (E)-37)

Submitted to the department of Computer engineering in partial fulfillment of the requirement for the degree of

Master of Science
In
Software Engineering

Thesis Supervisor
Brig Dr. Muhammad Younus Javed
MSC-CSE-6

College of Electrical and Mechanical Engineering
National University of Science and Technology
2009

DECLARATION

I hereby declare that no portion of the work referred to in this Project Thesis has been submitted in support of an application for another degree or qualification of this of any other university or other institute of learning. If any act of plagiarism found, I am fully responsible for every disciplinary action taken against me depending upon the seriousness of the proven offence, even the cancellation of my degree.

_____

Madiha Kazmi

ACKNOWLEDGEMENTS

First of all, I would like to thank Almighty Allah for giving me the strength and courage to complete this thesis.

Secondly, I want to pay my gratitude to my parents for their support and their ever-lasting patience in the light of my studies and thesis. I want to thank my lovely daughter Hania Kazmi for telling me how to work hard. I want to express my appreciation for my brothers for their support and their belief in me and my abilities.

Lastly my words can hardly show my gratitude to all those teachers who have been a great help to me through out this thesis.

I want to thank my supervisor, Dr. Muhammad Younus Javed for his guidance and supervision as well as his endurance and tolerance. His intellectual and intelligent suggestions have provided me with immense direction. I want to pay him additional appreciation, for his omnipresent help, ideas, patience and tolerance to even the little problems I took to him. He has been a great source of inspiration and motivation and he is the one to whom I owe the thesis as it was completely his visualization which led to development of this work.

DEDICATION

To an inspiring teacher……..

A remarkable father……

An extraordinary personality……

Dr. Muhammad Younus Javed

ABSTRACT

This project presents an analysis of the Transmission Control Protocol (TCP) and its different variants on IP and MPLS (Multi-protocol Label Switching) networks. TCP provides a trustworthy end-to-end data transfer under changeable wired networks. To overcome the problem of unreliability of IP network, TCP is used. Many service providers are now moving to MPLS over internet to transfer data, preferring it over traditional transferring strategies. Different variants of TCP show varying behavior in best effort Internet Protocol networks. This research presents an extensive investigational study of TCP variants under IP and MPLS networks by focusing Tahoe, Reno, New Reno, Sack and Vegas under File Transfer Protocol (FTP) and Constant Bit Rate (CBR) traffics. For analytical results of the proposed solution; demonstration of the IP and MPLS network is simulated over a limited number of nodes. The flow of descriptors to maintain network topology determines average delay, variance of delay, packets sent, packets received, packet dropped, throughput and congestion window size. Conclusions are drawn based on the simulation results, while comparisons between them have been elaborated.

Under different traffic flows, it has been found that all variants are unable to adapt MPLS features. However, Vegas has shown potential outcomes with almost invariable en-to-end delay after a brief time period of delay oscillation in the early phase of application. The unwavering end-to-end delay of Vegas under MPLS makes it a striking option for average to huge size real-time networks.

This research work has identified open problems related to the TCP protocol and its variants. To limit the scope of the research portion, error rate has been introduced only in single flow of FTP. This analysis has helped in investigation of congestion window patterns of the TCP variants with respect to different error rates induced over the network.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACK	Acknowledgement
BER	Bit Error Rate
BGP	Border Gateway Protocol
CBR	Constant Bit Rate
CWND	Congestion Window
FEC	Forwarding Equivalence Class
FTP	File Transfer Protocol
IP	Internet Protocol
LDP	Label Distribution Protocol
LER	Label Edge Router
LSP	Label Switched Path
LSR	Label Switching Routers
NS-2	Network Simulator-2
MPLS	Multi-protocol Label Switching
OSPF	Open Shortest Path First
QoS	Quality of Service
RSVP	Resource Reservation Protocol
RTO	Retransmission Time Out
TCP	Transmission Control Protocol
TTL	Time-To-Live

Chapter 01
Introduction

1.1 Transmission Control Protocol (TCP)

A TCP protocol is a transport layer protocol that provides a reliable and in-order deliverance of data between two nodes. TCP is also a defensive protocol i.e. highly responsive to network congestion. A set of mechanisms is put into place to detect occurrence of congestion and alleviate its affects, effectively preventing communication breakdown.

To ensure reliability of communication, TCP uses an acknowledgement packet (ACK) as a response to a successfully delivered packet ensuring accurate delivery of packets. ACKs are cumulative; ideally, consecutive ACKs will differ by the size of a packet payload. In case of a lost packet, the next packet received will return ACK of the packet received before the loss, helping the sender to recognize two identical ACKs. These are called duplicate ACKs and are considered as a signal of a packet loss.

Today's Internet traffic is carried to a large extent using TCP, and as a result there has been a major amount of research toward modeling and comprehending the impact of this protocol on file transmission times and network utilization. TCP has been and will continue to be a growing protocol, and as such, one important task of these models is to make easy comparisons between the different variants of TCP. As TCP has gone through incremental modifications, the protocol itself has grown increasingly complex, which makes analytical modeling quite difficult.

Consequently, a lot of the evaluation of TCP variations has been done using event driven simulators, such as ns-2, a network simulator developed at Lawrence Berkeley National Laboratory [1]. More recently, research has moved toward analytical modeling of the performance of TCP. This movement has been triggered by the limitations inbuilt in event driven simulations, which can be quite time consuming for fast networks and force researchers to use a packet level

granularity even when investigating large, complex networks. Also, when designed carefully, analytical models help researchers understand the effectiveness of new mechanisms being added to the protocol. As a result, many papers have been written introducing new, analytical models of performance of TCP. However, most of the analytical models focus on TCP-Reno, the most widely deployed variant of TCP, and there has been little research on analytical models of TCP-Vegas, a more recently proposed variant.

Analytical models of Vegas have been difficult to develop because of the dependence of Vegas on the delay experienced by packets in the network. Specifically, Vegas uses observed delay to detect an incipient stage of congestion and tries to adjust the sending rate before packets are lost. Thus, unlike Reno, Vegas attempts to determine the correct sending rate without relying on packet losses. Prior studies on measurement and simulation of performance of Vegas suggest that in many situations it is able to provide users higher throughput and lower loss rates than Reno. Hence, it is an important task to model performance of TCP-Vegas in order to understand how this protocol performs in a network shared with other variants of TCP and how TCP-Vegas should be incrementally deployed.

1.1.1 TCP Variants

Some of TCP Variants are discussed below.

1.8.1.1 TCP Tahoe

The TCP Tahoe has method to pay compensation for the efficiency plunge due to the congestion related Packet loss. TCP Tahoe uses three mechanisms to organize the flow and deal with congestion: slow start (SS), congestion avoidance (CA), and fast retransmit, the operation of these mechanisms depends on two variables that the protocol maintains: cwnd 3 (congestion window size) and ssthresh (threshold value of slow start). After the connection is initiated, the cwnd is set to 1 and the ssthresh to 64 (for an assumption that packet size is 1KB). The protocol enters

the SS phase in which it increases cwnd by 1 for every packet i.e. ACK received successfully. The consequence of the SS is that CWND grows exponentially i.e. is getting double per round-trip time (RTT). When cwnd surpasses ssthresh, the CA phase begins in which cwnd increases additively by 1/cwnd for each successfully ACKed packet. When the congestion event occurs, one of the following options is applied. If the loss is signaled by multiple duplicate ACKs, cwnd is halved and ssthresh is set to the new cwnd. The underlying principle for halving cwnd is that the network capacity is somewhere in between cwnd /2 and cwnd when the congestion event was initiated. If the packet loss is signaled by a timeout, the cwnd is set to 1 and ssthresh to half of the previous cwnd. At this point, the fast retransmit phase begins in which the lost packet is retransmitted immediately. TCP Reno would initiate fast recovery in which it would inflate cwnd by 3 (to account for the 3 duplicate ACKs received, indicating that 3 packets have left the network). When the ACK for the lost packet is received the Fast Recovery mechanism would inflate cwnd to ssthresh. Since the CA phase increases the window additively [2].

1.8.1.2 TCP Reno

TCP Reno is the standard implementation of the TCP protocol, which includes the congestion control algorithm proposed in 1988 by V. Jacobson. The TCP Tahoe has four mechanisms to compensate for the efficiency drop due to the congestion related packet loss. TCP Reno uses four mechanisms to control the flow and deal with congestion: slow start (SS), congestion avoidance (CA), and fast retransmit. To calculate approximately the available bandwidth in the network TCP Reno induces packet losses. When there are no packet losses, TCP Reno continues to increase its window size by one during each round trip time. Whenever a

packet loss is experienced, it reduces its window size to one half of the present window size. This is called additive increase and multiplicative decrease. They have shown that such an algorithm leads to a fair allocation of bandwidth. TCP Reno, however, fails to achieve such fairness because TCP is not a synchronized rate based control scheme, which is necessary for the convergence. As can be conspicuously, congestion avoidance mechanism adopted by TCP Reno causes a periodic oscillation in the window size due to the constant update of the window size. This oscillation in the window size leads to an oscillation in the round trip delay of the packets. This oscillation results in larger delay and jitter. Moreover an inefficient use of the available bandwidth results in many retransmissions dropped packets. Window size for each connection is updated at a rate, which depends on the round trip delay of the connection. Hence, the connections having shorter delays can update their window sizes faster compared to connections with longer delays, and thereby unfair share of the bandwidth is acquired by them. As a result, TCP Reno exhibits an undesirable prejudice against the connections with longer delays.

The settings for the connection initiation are as follows: the cwnd is set to 1 and the ssthresh to 64. The protocol enters the SS phase in which it increases cwnd by 1 for each packet successfully ACKed affects growth of SS exponentially. When cwnd exceeds ssthresh, the congestion avoidance phase begins. As this phase starts after packet loss network becomes unstable. To make the network work appropriately a careful and calculated increase of cwnd is done, so cwnd is increased by 1/cwnd for each packet whose ACK has been received by the sender. When the congestion event occurs, one of two things can happen. Either loss is signaled by multiple duplicate ACKs, cwnd is halved and ssthresh is set to the new cwnd, or the packet loss is signaled by a timeout, the cwnd

is set to 1 and ssthresh to half of the previous cwnd. At this point, the fast retransmit phase begins in which the lost packet is retransmitted immediately. TCP Reno would initiate fast recovery in which it would inflate cwnd by 3 (to account for the 3 duplicate ACKs received, indicating that 3 packets have left the network). When the ACK for the lost packet is received the Fast Recovery mechanism would inflate cwnd to ssthresh. Since the CA phase increases the window additively and decreases it multiplicatively, its algorithm is called AIMD (additive increase multiplicative decrease) [2].

1.8.1.3 TCP New Reno

TCP Reno Fast Retransmit and Fast Recovery are fairly efficient in dealing with the congestion packet if one packet is dropped in a congestion window. If multiple packets are dropped, TCP Reno will retransmit the first packet for which it received the duplicate ACK, and then it will exit the Fast Recovery phase. TCP Reno will reenter the Fast Recovery phase when it comes to know that more packets are dropped. The constant re-entering to the Fast Recovery phase affects the efficiency of the protocol. TCP New Reno tries to fix this problem by staying in the Fast Recovery phase as long as there are outstanding lost packets. That fact is recognized by the receiving the partial ACK. Partial ACK acknowledges the first packet retransmitted in the Fast Recovery phase that has not acknowledged all the packets transmitted before the Fast Recovery phase was . This implies that the retransmitted packet was not the only packet dropped in that window. TCP New Reno stays in the Fast Recovery as long as partial Asks are received [2].

1.8.1.4 TCP Sack

The major drawback of TCP Tahoe, Reno, and New Reno is that they are all dependant on the cumulative packet acknowledgements. This restrains the protocol from recognizing multiple lost packets in one RTT. If the sender wants to recover more aggressively, it might opt to retransmit the lost packet and enough of the successive packets to keep the pipe full. Such action might cause the transmission of packets that are already successfully received. TCP SACK (selective acknowledgements) allows for finer-grain information about the packets lost. SACK information carried to the sender in the TCP option field of the ACK header, contains the exact Information about the packets received. SACK option maintains the information about the block of packets received, i.e. the sequence numbers of the first and the last packet received within that block. If more than one packet has been dropped in one congestion window, SACK option will carry the information about several blocks differing by the sequence number of the dropped packets. This information allows TCP SACK to retransmit only the lost packets, thus providing a more aggressive loss recovery. The goal of the congestion control mechanisms is the prevention of the congestion collapse; the communication breakdown due to the uncontrolled congestion at the routers. With these mechanisms in place, the window size is reduced on the first sign of congestion and gradually increased with time to return the network communication to its optimum rate. This paradigm gives the congested network necessary time to recover [2].

1.8.1.5 TCP Vegas

TCP VEGAS introduces several new mechanisms to TCP including a proactive congestion avoidance technique, which does

not break the AIMD paradigm of the TCP protocol. Instead of increasing the sending rate until a packet loss occurs, TCP Vegas tries to prevent such losses by decreasing the sending rate when it senses incipient congestion even if there is no indication of packet loss. There is new retransmission mechanism in TCP VEGAS. TCP VEGAS feature two time out values. One of them is a normal coarse - grained RTO value similar to the one in TCP Reno and other is a fine –grained RTO value based on a more accurate RTT estimation. Whenever a duplicate ACK is received TCP VEGAS checks whether the difference between the current time and the timestamp recorded for the relevant segment is greater than the fine-grained RTO or not. In former case, that segment is retransmitted immediately without waiting for further duplicate ACKs.

The proactive congestion control behavior of Vegas is based on RTT measurements. Once per RTT, Vegas compares the current measured throughput with the expected throughput. The expected throughput is computed as

Expected throughput = widow size/base RTT

Where base RTT is the smallest observed RTT measurement for the connection and window size is the number of bytes currently in flight. The actual throughput is computed as

Actual throughput = rttLen/rtt

Where rtt is the average RTT of the segment acknowledged during the last RTT, whilst rttLen is the number of bytes transmitted during the last RTT. The difference (diff) between the two measurements is calculated in base RTT segments as follows:

$$diff = (windowsize / baseRTT + rttLen / rtt) * baseRTT$$

If the difference is under a certain threshold α then the congestion window is increased by a full segment size. If the difference is above a threshold β then this is taken as sign of incipient congestion and the congestion window decrease by a full segment size. Other wise, the congestion window remains unchanged. The decision process used to adjust the sending rate per RTT is summarized below:

$$cwnd = \begin{cases} cwnd + 1 \cdots if diff \lhd \alpha \\ cwnd \cdots\cdots if \alpha \le diff \le \beta \\ cwnd - 1 \ldots if diff \rhd \beta \end{cases}$$

Finally, the slow start mechanism of Vegas uses a variation of the congestion avoidance mechanism to decide when to switch to the congestion avoidance phase. Vegas monitors the expected and actual rate per RTT and increases the congestion window to make the comparisons valid. As soon as a queue build up is detected (diff >1), Vegas moves on to the congestion avoidance phase [3].

1.2 Multiprotocol Label Switching (MPLS)

The unstable growth of the Internet and the introduction of complicated services require an epoch-making change. MPLS was proposed as an alternative. MPLS is an Internet Engineering Task Force (IETF) specified protocol. MPLS provides various many services through networks i.e., routing, forwarding, switching of traffic flows and effective /efficient designation. One of the most important function of MLPS is that it identifies methods to supervise the traffic flows between different hardware, machines and even different applications. MPLS is not reliant on Layer-2 and Layer-3

protocols. It also provides a way to map IP addresses to simple, fixed-length labels used by different packet- forwarding and packet-switching technologies.

MPLS is a switching technology that uses labels. In a MPLS network, arriving packets are allocated a label by a Label Edge Router (LER) in accordance to their Forwarding Equivalence Class (FEC). Packets are forwarded all along a Label Switch Path (LSP) where each Label Switch Router (LSR) puts together forwarding decisions based exclusively on the contents of the label, getting rid of the need to look for its IP address. At each hop, the LSR take off the existing label and assigns a new label for the next hop. Next hop also decides how to forward the packet by reading just the label on the packet. These established paths LSPs can guarantee a certain level of performance, to route around network congestion, or to create IP tunnels for network-based virtual private networks. The features of MPLS, make it achievable to deliver a variety of new services that can not be supported in the conventional Internet.

MPLS uses a variety of protocols for label distribution, such as Label Distribution Protocol (LDP) or piggybacked on routing protocols like Border Gateway Protocol (BGP). It also provide interface to existing routing protocol i.e. Open Shortest Path First (OSPF) and Resource Reservation Protocol (RSVP).

1.3 IP

In the beginning the intention behind introducing Internet Protocol (IP) was to transport of data via e-mail, File Transfer Protocol (FTP) and telnet sessions. Soon IP became the most demanding and leading communication protocol because of its adaptability. In this protocol routing decision are taken with in the network on the basis of IP address of sender and destination. Some of the important features of IP are de-multiplexing of protocol, reassembling of packets and fragmentation of addresses.

1.4 Tools Used

1.4.1 Network Simulator

Tools used for performance analysis is ns-2 [4].It is the most efficient way to compare the studied TCP variants. Ns-2 is a UNIX based discrete network simulator. It provides support and enhancements for routing protocols, TCP schemes and also provides support for different emerging networking technologies. It started in 1989 as an independent project (REAL network simulator), but since then has been evolved into a much larger project supported by DARPA and NSF with development contributions by Xerox PARC, University of California - Berkley, University of South California, Carnegie Mellon University, and Sun Microsystems. Today, this simulator is the standard testing tool for any development in the networking area since it allows for creation of a wide variety of different scenarios that are otherwise hard to implement practically. Thus ns-2 provides ideal environment for research studies. Ns2 is written in C++/Tcl language which supports major necessities required by researchers. For making a new protocol in ns-2, the code has to be written in C++ .Default settings for the protocol to be analyzed, should be written in ns-default.tcl. Before recompiling ns2 it is necessary to add the link of the newly added protocol at Makefile.ini. After recompiling ns-2 the newly added protocol will be ready to use.

TCP variants used here are implemented by their authors. Their source code is used in the simulations for running the scripts after adding in ns-2 by above mentioned procedure.

1.8.2 Gnuplot

Gnuplot is a graphical tool used to make graphs out of data file. Gnuplot is available for both the Linux and windows platform. Gnuplot offers analysis of data by making pie charts, bar graphs and line graphs. This tool makes it easy to analyze data. This tool has been used to generate

many graphs in this thesis, for example Graphs for throughput, delay and performance comparisons.

1.8.3 Xgraph

It is a UNIX based applications. XGraph is a utility present in ns2 for analyzing the data present in the trace file. It helps to analyze behavior of congestion window, and Acknowledgments received by reading the given trace file.

1.9 Problem Statement

- To analyze network performance by recording the traffic transmitted over the network.
- To carefully monitor traffic received from outside the Multi-Protocol Label Switching (MPLS) or sent outside MPLS, as well as transmitted within a local MPLS network. This way behaviors of different variants of Transmission Control Protocol (TCP) present in the network can be discovered, or the weaknesses of the variant can also be found in different traffics scenarios.
- To compare performance of different IP variants via network monitoring and validation.
- Detail literature study for “state of the art” technology in TCP/IP and MPLS.
- Performance comparison in delay and throughput by increasing number of flows.
- Performance comparison between the different protocols used in IP and MPLS through congestion window, delay and throughput.
- To suggest better TCP variant for IP and MPLS networks.

1.6 Description of Project

The objective of this thesis is to develop an application which monitors the network traffic in order to determine the characteristics of TCP variants in the

network. The entire traffic flowing through the network is monitored and recorded regardless of the source and destination addresses. The information is stored in trace file that can be read using any editor. The information is stored in a manner so as to provide statistics regarding network usage at any particular interval and by any particular node in the network. This information can thus be used to compare and contrast the performance of TCP variants on IP and MPLS networks.

To develop a simulation for network analysis which:

Captures all the traffic on the network and will make a thorough and detailed analysis of the packets of each host on the network. i.e., Delay or time taken by a packet to reach destination.

Also maintain statistics of the traffic.

- Number of packet read
- Number of packet sent
- Number of packet received
- average delay of all packets of network
- variance of delay
- Will perform network monitoring in single as well as multiple flow networks.

1.7 Scope of the Project

This project is about developing a simulation for providing vital statistics about network usage by any particular variant of TCP and as well as comparison of a particular variant with others on the same network topology. This information is helpful to determine the cause of saturation in the network due to the presence of a malfunctioning node or packet loss. This simulation and research have broad usage as computer networks are becoming more and more complex everyday. There is an increasing need of analysis, diagnosis and testing of functionality of TCP variants on different networks especially on combination of networks.

Particularly, all of the prior work on TCP-Vegas is based on a single sender of bulk transfer and most of the earlier work on Reno, New Reno, Tahoe, Vegas and Reno is limited to the examination of the throughput of a single sender

as a function of the loss rate in the network. But the present research extends this to multiple flows on single node and multiple flows on multiple nodes.

1.8 Organization of study

This thesis is organized in six chapters and two appendices. Present chapter is a general introduction on the subjects undertaken in the thesis and gives a brief overview of the work. The other chapters are organized as follows:

- ***Chapter 2:*** The literature review is presented. The work done previously on TCP variant, IP networks and MPLS networks has been described.
- ***Chapter 3:*** The system design is explained. The system is designed using the Network Simulator -2 (ns-2). It explains the various steps taken to design the system with the help of screen shots and flow chart.
- ***Chapter 4:*** The performance of the system is evaluated by carrying out various tests on individual modules and results are discussed in detail regarding overall performance of developed system.
- ***Chapter 5:*** Thesis conclusion and future work is included in this chapter.

Chapter 02

IP and MPLS Networks

2.1 Introduction

Service providers are increasingly using Multi-protocol Label Switching (MPLS)/Internet Protocol (IP) networks for delivering a variety of services. While MPLS uses Label Distribution Protocol (LDP), it utilizes network resources best with its well-organized, little overhead label and traffic engineering practices [5]. Real time data transfer networks require quality of service (QoS) which can be provided by Transmission Control Protocol (TCP) along with MPLS. These real time networks when use IP based networks as transmission require heavy overhead. Tahoe, Reno, Vegas, New Vegas and Sack etc. are few variants of TCP. These variants give different results in varying network scenarios; every version of TCP has its own drawbacks and benefits depending upon two factors. Firstly, the network conditions provided to them and secondly, the algorithm they are working on. Basically these algorithms are implemented and designed to give optimal performance under certain circumstances.

On Internet greater part of traffic is run by TCP. Van Jacobson in [6] introduced an algorithm that is responsible for reliable delivery of all data and fair sharing of network resources. TCP is transport layer protocol that offers a trustworthy, connection-oriented, byte-stream service. It also controls flow; as it stops sender from sending unmanageable data, which can otherwise overflow receiver's buffer. Using congestion control mechanism TCP stops sender to insert a large amount of data into network.

2.2 Background

In the field of performance evaluation of TCP over IP and MPLS network, very few useful results from performance evaluations are available. It is also very hard to compare the results of different performance evaluations as they depend heavily on the used protocols and parameter settings. One of the examples of interesting TCP performance evaluation for IP and MPLS network is the network simulations carried out at the MAJU, Pakistan [7]. The research focuses on performance evaluation of certain variants of TCP protocol over IP and MPLS

network. The investigations mainly evaluated the performance of TCP in respect to different modes of IP and MPLS network and CBR traffic with different number of flows. This research basically analyzed three variants Reno, Tahoe and Vegas under changing traffic set-ups. It points out Vegas to be an out performer, which shows a jitter free constant end-to-end delay.

In a diploma research carried out at University of Namur, numerous TCP variants were examined using network simulator NS-2, simulating different traffic scenarios with two types of traffic i.e. FTP and Telnet [8]. It was concluded that type of traffic also had a significant impact on overall performance of TCP in Universal Mobile Telecommunications System (UMTS) network.

Zhong Ren in [9] presented a scheme to integrate mobile IP and MPLS networks. Techniques for controlling and signaling this integration are argued in detail, it also points out some problems of scalability of Mobile IP. MPLS is used to switch packets instead of IP tunneling. MPLS forwarding is much faster than IP forwarding as packet processing overhead is reduced for the reason that MPLS header is smaller than IP header. A similar sort of study is performed by M. Asante in [10] by analyzing the mobile IP and MPLS union architecture. The paper highlights the benefit of this union.

There has been significant interest of researcher community in performance analysis of TCP variants. Most of the researches have been focuing on TCP congestion management in IP based networks.

Adam Wierman in [11] gave a framework for analyzing TCP variations i.e. Vegas, Sack and Reno. He noted that the customized slow start mechanism commenced in Vegas hardly aided packet loss reduction however it resulted in wasting more time in the slow start phase.

Mazleena Salleh and Ahmad Zaki Abu Bakar [12] compared TCP Tahoe, NewReno, Vegas, and Sack over self-similar traffic using NS2 for simulation. They found that NewReno did better than other TCP variants with respect to efficiency and throughput.

The two well-known variants of TCP: Reno [13] which is widely used on internet and Vegas [14] which measures performance improvement of 37 to 71

percent better throughput over the Reno. TCP Congestion Control mechanism recognizes and illustrates four congestion control algorithms to be precise: slow start, congestion avoidance, fast retransmit, and fast recovery [15]. Jeonghoon Mo, Richard J. La, Venkat Anantharam, and Jean Walrand's [16] results emphasize on former discussed research results. They demonstrated via simulation that Vegas utilized network resources more competently and fairly as compared to Reno.

Go Hasegawa and co-author [17] when compared performance of Reno and Vegas sharing bottleneck link on internet found out Reno to be a better performer. Similar results were concluded by Cheng P. Fu and Soung C. Liew in [18] where they compared performance of Reno and Vegas on asymmetric networks having bottleneck.

Thomas Bonald in [19] compared Reno and Vegas keeping RTT measurement as testing template. They focused on long-term performance criterion i.e. average throughput and average buffer taken up. Their core conclusion was that TCP Vegas was better than Reno, as it shared on hand bandwidth fairly between users of different propagation delays.

Yi-Cheng Chan in [20] has reported a few problematic sides of TCP Vegas which reduced its success in congestion avoidance. These characteristics are fairness, re-routing, constant congestion and network irregularity. Authors have also presented a router based congestion avoidance mechanism to trim down impact of Vegas problems.

Ben Soh and Joel Sing have judged New Vegas a new variation of TCP Vegas, to overcome Vegas short comings. Vegas experiences decreased throughput in networks that have large bandwidth-delay product. Writers in [21] have confirmed New Vegas to have superior throughput because of its improved expansion administration of congestion window. Problems of fairness and multi-variant session of New Vegas are yet to be answered.

2.3 Contribution

This examination is about developing a simulation for providing vital statistics about network usage by any particular variant of the TCP and as well as comparison of a particular variant with others on the same network topology. This information is helpful to determine causes of saturation in the network due to presence of a malfunctioning node or a packet lost. This simulation and research have broad usage as computer networks are becoming more and more complex everyday. There is an increasing need of analysis, diagnosis and testing of functionality TCP variants on different networks especially on combination of networks i.e. IP and MPLS networks integration.

Particularly, all of the prior work on TCP-Vegas is based on a single sender of bulk transfer and most of the earlier work on New Reno, Sack, Tahoe, Vegas and Reno is limited to examination of throughput of a single sender as a function of loss rate in the network. But the current research extended this to multiple flows on single node and multiple flows on multiple nodes.

This study considers a simulated analysis of many flows interacting through a variant of TCP in IP and MPLS network. However, in contrast to the former papers, our focus is not on modeling TCP, but on studying it in a common setting, and on the effects that variations of parameters have on the Average delay, congestion window and throughput produced by these variants.

Interestingly, in addition to the last few papers mentioned above in connection with TCP, results show lack of information. This suggests an underlying common structure not yet fully understood. There has already been some interesting work done in the field of evaluating performance of TCP over IP and MPLS network. So evidently there is a query why has this research been done on performance study of TCP over IP and MPLS network. The main difference between the present work and the related work is that it not only focuses on work done on Reno Tahoe and Vegas but also on Sack and New Reno and on two types of traffic i.e. CBR and FTP and error rate is also introduced. The idea of studying TCP in terms of categories is very interesting as the internal mechanism of thes

selected TCP variants makes them behave differently from each other in same network conditions.

Chapter 03

Design and Implementation of IP and MPLS Based Networks

3 Experimental Setup

3.1 Network Topology

General architecture of MPLS and IP network integration is shown in the Figure.3.1. This architecture consists of total number of 13 nodes which are divided into two domains (MPLS and IP).

MPLS domain consists on 7 nodes which are labeled as LSR1 to LSR7 and IP domain consist on 6 nodes named as node0 to node5. The IP domain consists of a sender and a receiver network each having three nodes. Links between nodes of MPLS and IP network are 1 Mb and 5ms node processing delay. All MPLS nodes are Label Distribution Protocol (LDP) enabled. Which provides mechanism for label distribution among MPLS enabled nodes.

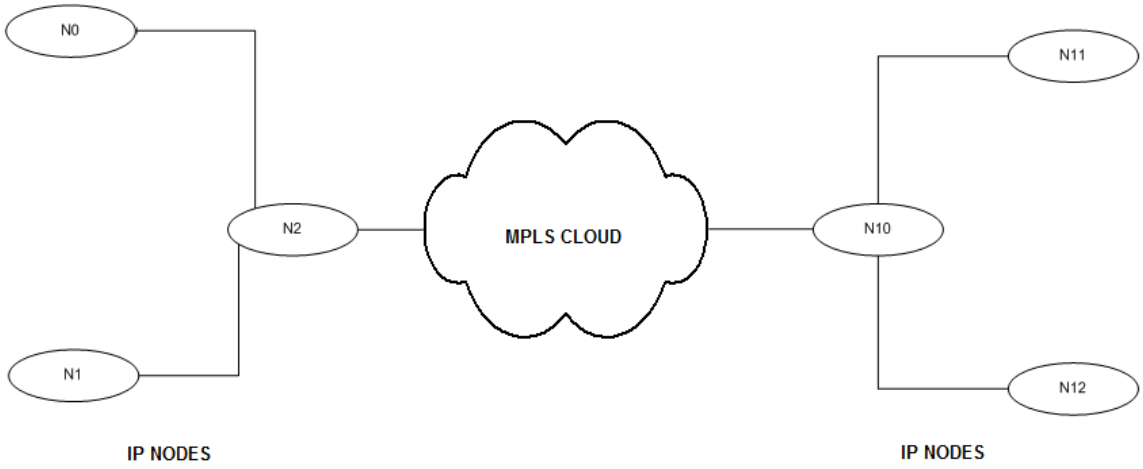

Figure 3.1: General Network topology

The traffic is Constant Bit Rate (CBR) and File Transfer Protocol (FTP) Traffic having packet size of 1500 kilo bytes with variable time interval. The simulation runs for 100 seconds. Traffic source and destination are IP based networks. Some common networks parameters are shown in Table 3.1.

Attributes	Values
Bandwidths	1MB
Link Processing Delay	5ms
Number of hops	7
Number IP nodes	6
Packet size	1500
Number of MPLS nodes	7

Table 3.1: Network parameter

The different TCP variants in MPLS/IP network are analyzed using different scenarios. The numbers of flows of the link were varied in order to check the effect of different flows on the throughputs and delay. These traffics are run in both situations i.e. CBR and FTP.

- **Single Traffic**
  - **Single Traffic CBR**
  - **Single Traffic FTP**

- **Multiple Traffic**
  - **Two flows Traffic**
    - **Two flows Traffic CBR**
    - **Two flows Traffic FTP**
  - **Four flows Traffic**
    - **Four flows Traffic CBR**
    - **Four flows Traffic FTP**
  - **Eight flows Traffic**
    - **Eight flows Traffic CBR**

- **Eight flows Traffic FTP**

3.2 Tool Command Language (TCL) Script

3.2.1 Simulation initialization

The first step of any tcl script file is to set a new network simulator, this preamble initializes the simulation.

3.2.2 Predefine tracing

Second step is predefined tracing. Files are opened to write trace-data for NAM and Xgraph or any other graph plotting tool like Gnuplot used by this research .Two files are opened and naming “mpls.nam” which records all information of nam and “mpls.tr” which traces all events in a simulation.

There are many ways to collect output or trace data while simulation is running. Generally, trace data is prompt directly during the simulation is run, or stored in a file to be post-processed and analyzed. There are two main types of monitoring facility supported by the network simulator. The first, called traces, records each individual packet as it is received, sent, or dropped at a link or queue. The other type of monitoring is called *monitors*, which counts various interesting quantities such as packet and byte arrivals, departures, etc. Monitors examine counts linked to all packets.

3.2.3 Trace File

There are 14 trace entries .A screen shot of trace file is shown in Figure 3.2.Column-wise details are as under:

3.2.3.1 First column:

- “+” : enque operations
- “-” : deque operations
- “r” : receive events
- “d” : drop event

3.2.3.2 Second column

The simulated time (in seconds) at which each event occurred is listed in the second column.

3.2.3.3 Third and fourth columns

The next two fields indicate between which two nodes tracing is happening.

3.2.3.4 Fifth column

The next field is a descriptive name for the type of packet seen for example tcp, telnet, cbr, ack, LDP etc.

3.2.3.5 Sixth column

The next field is the packet's size, as encoded in its IP header.

3.2.3.6 Seventh-ninth column

The next field contains the flags. The flags are defined in the flags[] which is an array in source file (trace.cc).

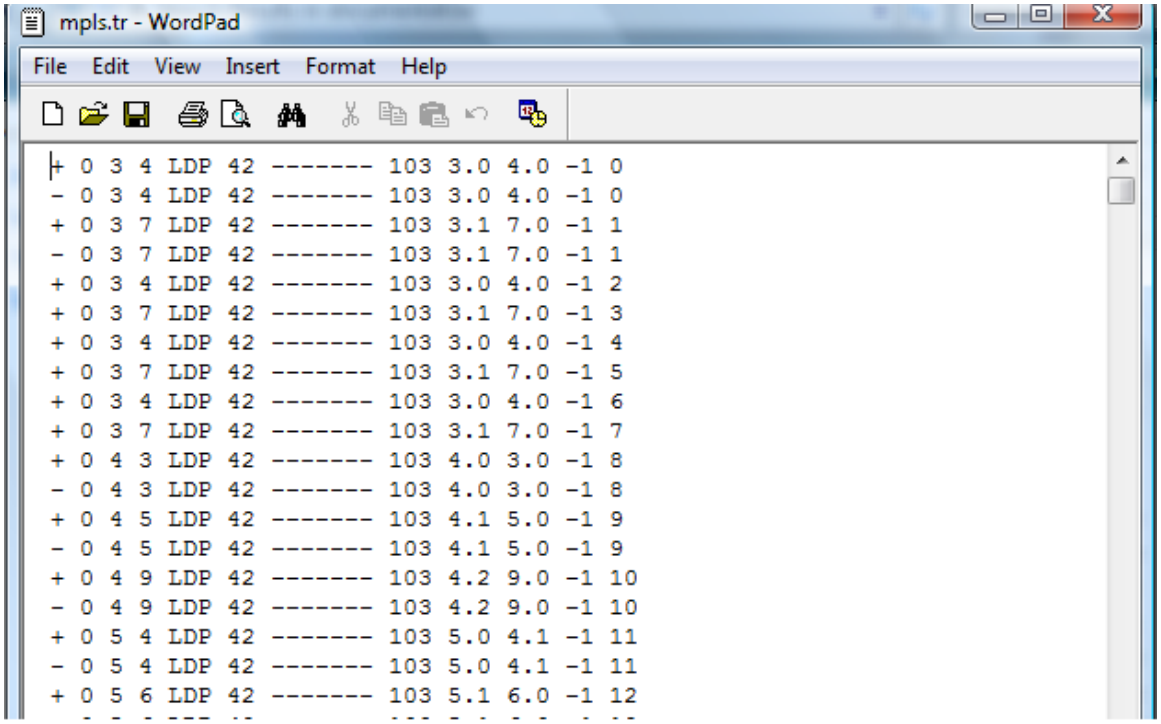

Figure 3.2: Screen shot of a trace file showing different columns

Following are few of the flags are used:

- "E" for Congestion Experienced (CE).
- "N" for ECN-Capable-Transport (ECT) in IP header.
- "C" for ECN-Echo.

- "A" for Congestion Window Reduced (CWR) indication in the TCP header.
- "P" is for priority.
- "F" for TCP Fast Start.

3.2.3.7 Tenth column

The next field gives the IP "flow identifier" field as defined for IP version 6.1.

3.2.3.8 Eleventh and twelfth column

The subsequent two fields indicate the packet's source and destination node addresses, respectively.

3.2.3.9 Thirteenth column

The following field indicates the sequence number.

3.2.3.10 Last column

The last field is a unique packet identifier. Every packet that is generated during simulation is allocated a fresh unique identifier.

3.2.4 Finish procedure defined

In next portion of script, Finish procedure is written which closes the trace file and opens Xgraph and NAM but it is called in the end.

3.2.5 Awk Script

To extract the required information from the traces generated by NS-2 some scripting language is required. There are different scripting language such as perl, grep and awk.

Awk script is used to calculate the required data from the traces generated by the Network Simulator. The awk scripting used in code is discussed below.

3.2.5.1 Awk script for creation of send-receive information trace file

This script takes trace file (**mpls.tr**) as input and saves all the entries in the output file (**sendrecieve.tr)** having value of columns 1 equal to "+" i.e., en-queue operation was performed, also the value of column 3 is 0 and value column 4 is 2 i.e., tracing is between node 0 and node 2; value of column 5 is tcp and value of column 6 is 1500 i.e. the packet is of type tcp having size equal to 1500 bytes. Moreover entries in the output file

having value of columns 1 equal to “r” i.e., read operation is performed; also value of column 3 is 10 and value column 4 is 11 this shows that tracing is between node 10 and node 11, value of column 5 is tcp and value of column 6 is 1500 i.e. the packet is of type tcp having a size of 1500 bytes.

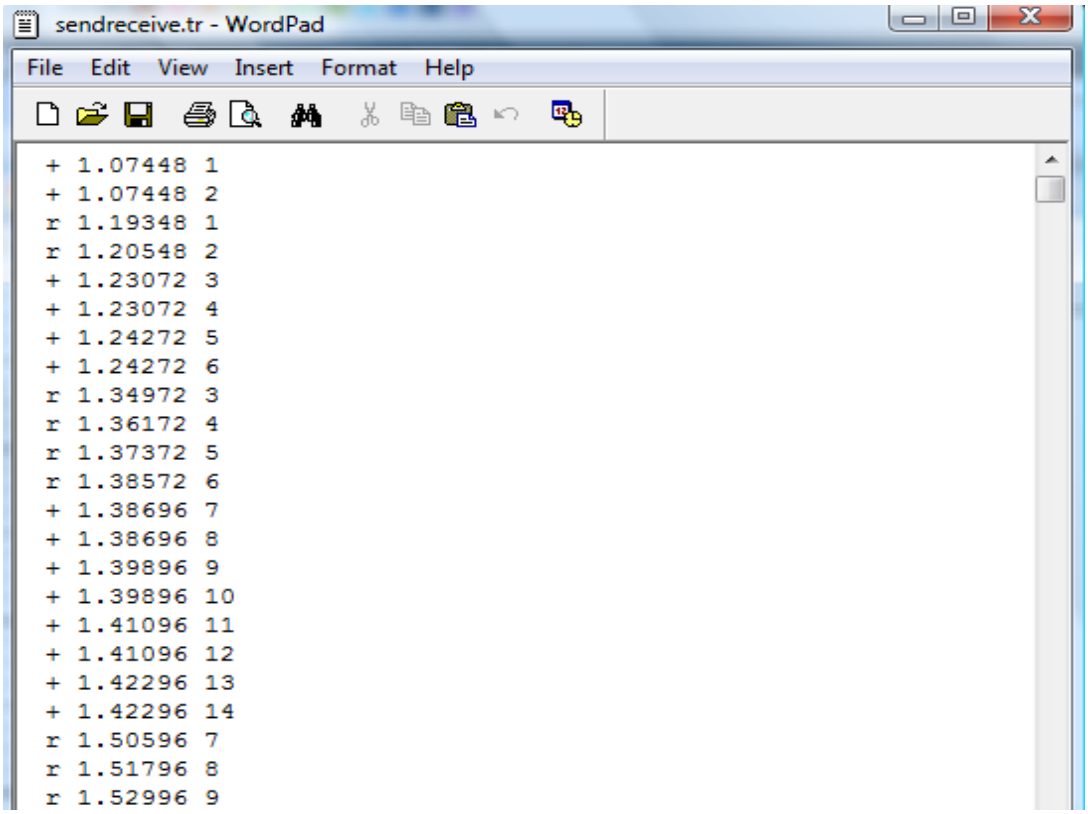

Figure 3.3: Screen shot of a trace file showing different columns

The resultant output file (**sendrecieve.tr)** shown in Figure 3.3 contains column 1, 2 and 11 which is used to extract more information. Where value of column1 shows operation performed on packets, value of column 2 is the time on which event occured during simulation and 11th column is the packet’s source.

3.2.5.2 Awk Script For Creation Of Congestion Window Information Trace File

This script takes trace file (congwnd.tr) as input and gives newcongwnd.tr as output; which saves column 1 and 7 of input tracefile. This output file is used to draw graphs for analysis of congestion window behavior of MPLS-TCP variants with different error rates in single flow FTP traffic.

3.2.6 Script For Execution Of delay.cpp

This script runs delay.cpp's executable, takes senrecieve.tr as input and gives two files as output. This script is repeated with increasing number of flow i.e. for single flow it is written once only, twice for two flows, four times for four flows and eight times for eight flows. Each time it is called it takes an input file and generates two output files.

3.2.7 Node Creation

Following IP nodes are created: node0, node1, node2, node3, node4 and node5. The given MPLS nodes are also formed: LSR1, LSR2, LSR3, LSR4, LSR5, LSR6 and LSR7.

3.2.8 Setting Link Parameters

- The link is set as duplex : This creates a bi-directional link between node1 and node2. This parameter basically creates a duplex-link from two simplex links, one from node1 to node2 and the other from node2 to node1.
- Bandwidth is set to 1 Mb : it is Link bandwidth in bits per second.
- Delay is set to 5ms : it is Link propagation delay in seconds.
- Queue management DropTail : The type of queue management used in the link. Default value is DropTail.

3.2.9 Configure the Label Distribution Protocol (LDP) on all MPLS Node

Install/configure LDP agents on all MPLS nodes, and set path restoration function that reroutes traffic around a link failure in a LSP to a substitute LSP. There are 2 options

- "new": If a path doesn't exist, create a different and fresh path .
- "drop": New alernative path is not created but loop length is set to address all MPLS nodes.

3.2.10 Set LDP-Message Color

- ldp-request-color as blue
- ldp-mapping-color as red
- ldp-withdraw-color as magenta
- ldp-release-color as orange
- ldp-notification-color as yellow

Define trigger strategy, Label Distribution Control Mode and Label Allocation and Distribution Scheme. When the following line is skipped over, data-driven option is set for trigger strategy. This strategy can also be applied on every LSR.

3.2.11 Scheduling of events

It specifies the type of scheduler to be used for simulation. Different types of scheduler available are List, Calendar, Heap and RealTime. This simulation uses 'List' scheduling of events.

3.2.12 TCP Traffic Source

Next step is creation of a TCP agent and connecting it to an application like FTP or Telnet, which will generate data. At various layers agents are endpoints where network-layer packets are created and used in the implementation of protocols.

3.2.12.1 Protocol Agents

There are number of agents supported in the simulator. Few of which used by this simulation are:

The one-way TCP sending agents currently supported are:

- Agent/TCP : a “tahoe” TCP sender
- Agent/TCP/Reno : a “Reno” TCP sender
- Agent/TCP/Newreno : Reno with a modification
- Agent/TCP/Sack1 : TCP with selective repeat
- Agent/TCP/Vegas : TCP Vegas

The one-way TCP receiving agents currently supported are:

- Agent/TCPSink : TCP sink with one ACK per packet
- Agent/TCPSink/DelAck : TCP sink with configurable delay per ACK
- Agent/TCPSink/Sack1 : selective ACK sink
- Agent/TCPSink/Sack1/DelAck : Sack1 with DelAck

The two-way experimental sender currently supports only a Reno form of TCP:

- Agent/TCP/FullTcp

3.2.12.1.1 TCP Tahoe

The "Tahoe" TCP agent Agent/TCP performs congestion control and round-trip-time estimation in a way similar to the version of TCP released with the 4.3BSD "Tahoe" UNIX system release from UC Berkeley. The congestion window size is enlarged by one packet per new ACK received during slow-start and is increased by 1 cwnd for each new ACK received during congestion avoidance.

3.2.12.1.1.1 Responses to Congestion

Tahoe assumes loss of a packet (due to congestion) when whenever retransmission timer expires or it observes NUMDUPACKS duplicate ACKs. In both scenarios, Tahoe sets threshold to half of the current window size or 2, whichever is bigger. It then initializes congestion window's size back to the value of windowInit_. This is the point in simulation when TCP enters slow-start.

3.2.12.1.2 TCP Reno

The Reno is TCP agent works similar to the Tahoe TCP agent, except it also includes fast recovery, where the current congestion window is inflated by the number of duplicate ACKs the TCP sender has received before receiving a new ACK. A new ACK refers to any ACK with a value higher than the highest observed until now. During a fast retransmit the TCP Reno agent does not return to slow-start. It also sets the congestion

window to half the current window size and resets ssthresh_ to match this value.

3.2.12.1.3 TCP NewReno

This agent is basically an extension of the TCP Reno agent, but it performs different action when new ACKS are received.

The sender must receive an ACK for the highest sequence number sent to exit fast recovery. Thus, new partial ACKs (those which represent new ACKs but do not represent an ACK for all outstanding data) do not reduce the window (and possibly lead to a stall, characteristic of Reno).

3.2.12.1.4 Vegas TCP

This agent implements TCP Vegas. It was given by Ted Kuo.

3.2.12.1.5 Sack TCP

This agent implements selective repeat, based on selective ACKs provided by the receiver.

3.2.12.2 TCP Receivers (sinks)

These TCP senders represent one-way data senders. They must take in with a “TCP sink” object.

3.2.12.3 Two-Way TCP Agents (FullTcp)

The Agent/TCP/FullTcp object is a new addition to the suite of TCP agents supported in the simulator. It is under development. This agent is different from and is not compatible with the other agents, but does utilize somewhat the same architecture. It differs from these agents in the following ways:

- connections are established (SYN/FIN packets are exchanged)
- bidirectional data transfer is supported
- sequence numbers are in bytes rather than packets

3.2.13 Creating Sender agent and attaching to a node

Script creates new sender agent and attaching it to sender node.

- **Maxcwnd** is the upper bound on the congestion window for the TCP connection. If it is set to zero it will be ignored and this is also its default value.

- **windowInit** is the initial size of the congestion window on slow-start.
- **packetSizet** is the size in bytes to use for all packets from this source.
- **tcpTick** is the TCP clock granularity for measuring roundtrip times. It is set by default 100ms which is not a standard value.

Simulation sets the tcp agent's maximum window to 40, set packetSize to 1460 bytes, set tcpTick to 0.001, set windowInit to 2. All tcp variants have been set on same initial values. It is also defined here that the agent will be FTP/CBR and associates FTP/CBR with the TCP sender.

3.2.14 Creating Receiver agent and attaching to node

Receiver agent is created and attached to a reciever node and establish TCP connection.

3.2.15 Trace MPLS/LDP packets

Trace results of MPLS/LDP packets at a given LSR are dumped at the prompt.

3.2.16 Simulation time

The simulation runs for 100s. It arranges FTP/CBR session to range between 1.0 seconds to 98.0 seconds. The packet sending is stopped 0.2 seconds before the simulation ends to avoid the packet loss making it sure that application received the last sent packet.

The simulation comes to an end when the scheduler invokes the finish procedure. This procedure closes all trace files, and invokes nam visualization on one of the trace files.

3.2.17 Start the simulation

Finally, start the simulation using run command.

Figure 3.4 shows the basic flow of simulation.

3.3 Error rate induction

Error model is one which introduces packet losses into a simulation. Error model simulates link-level errors or losses by two ways: either marking the

packet's error flag or dumping the packet to a drop target. During simulations, errors can be induced from a simple model using the packet error rate; the other way is from more complex statistical and empirical models. To support a numbers of models, the unit of error can be specified in terms of packet, bits, or time-based.

If error unit is not specified, it will be in packets by default, and the random variable will be uniformly distributed from 0 to 1.

Current simulation is creating error models with the packet error rate of 1 percent (0.01) , 2 percent (0.02), 3 percent (0.03), 4 percent (0.04) and 5 percent (0.05) for all the tcp variants Reno, New Reno, Sack, Tahoe and Vegas under the scenario of FTP traffic.

3.3.1 Steps in creation of loss module

Following steps are taken in creation of a loss module

- Set its packet error rate to value(1-5) percent
- Set the unit and random variable(this step is optional)
- Set the error unit: packets (the default is packet)
- Set target for dropped packets
- Attach random variable to loss module

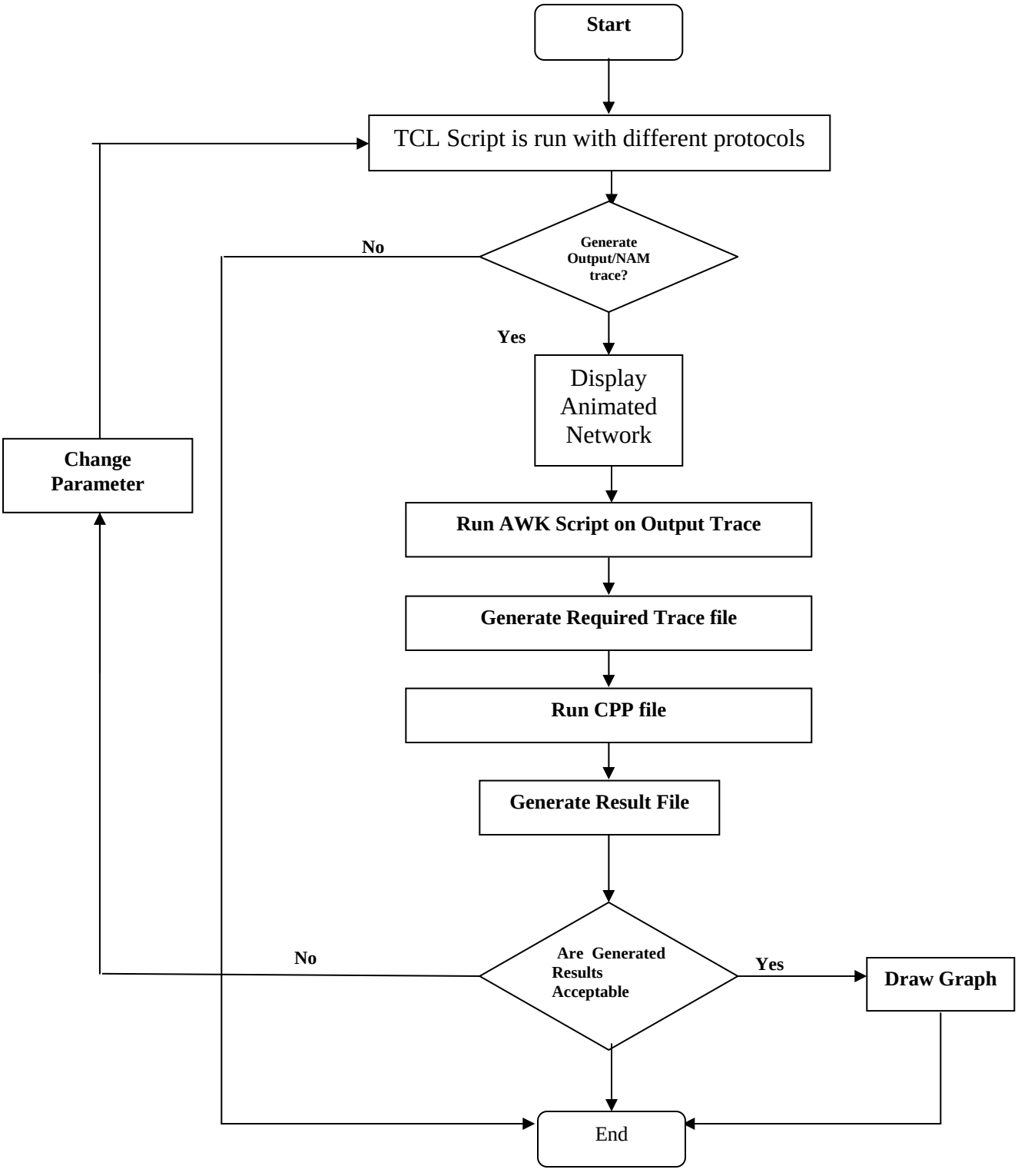

Figure 3.4: Flow Chart

3.4 delay.cpp

This file takes trace file (sendrecieve.tr) as input and gives two files as output. The first output contains packet sequence number and delay for that packet. The second output carries following information:

- Number of entries read
- Number of entries sent
- Number of entries received
- Average Delay of entries
- Variance of delay

3.5 NAM Visualization of Topology

Figure 3.5 shows the nam visualization of network topology. The circles with numbers show the nodes whereas lines connecting them show links between nodes.

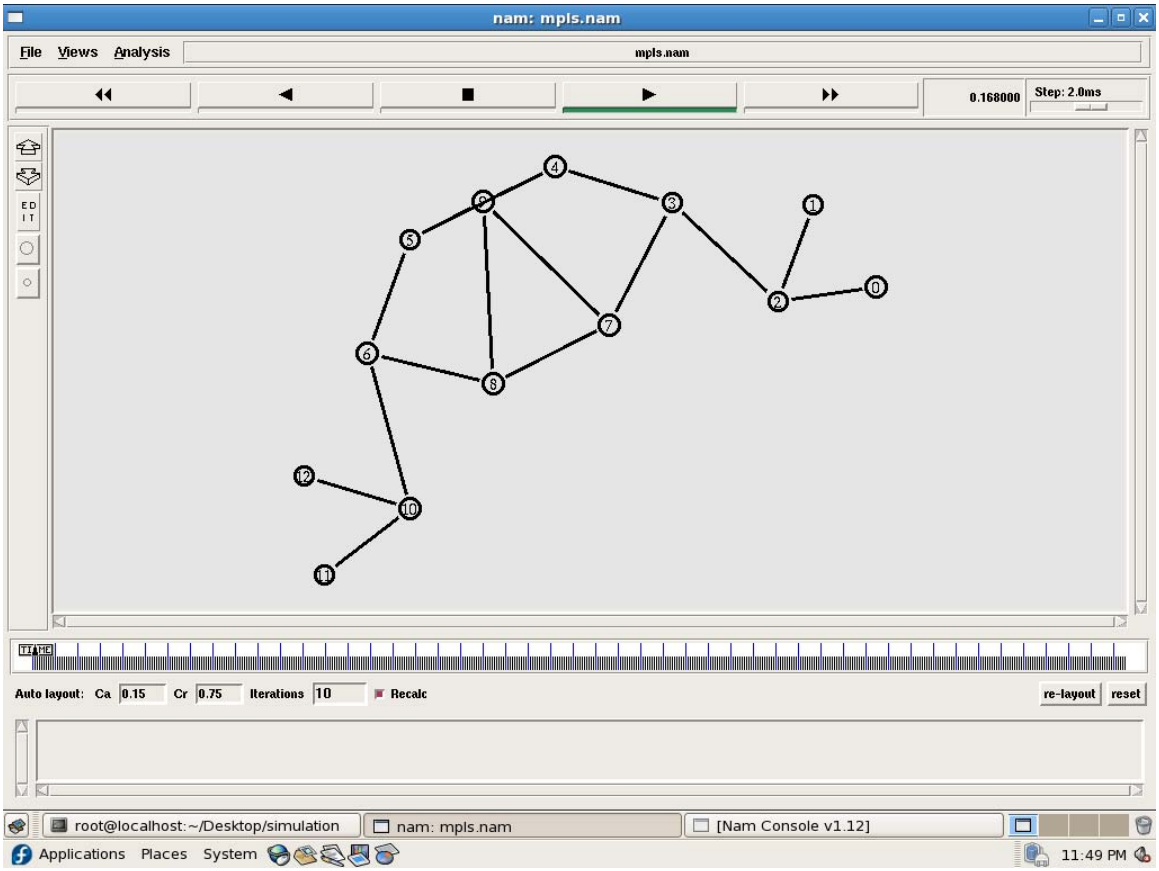

Figure 3.5: Screen shot of nam visualization

Node 0, 1, 2, 10, 11 and 12 are IP nodes in which first three are sender nodes and last three are receiver nodes. The intermediate ones from 4 to 9 are MPLS nodes and are LDP enabled.

3.6 Running TCL Script

Figure 3.6 shows the running of tcl script. In this case name of the script is "mpls.tcl" and command used to run the script is ns <name of script>.

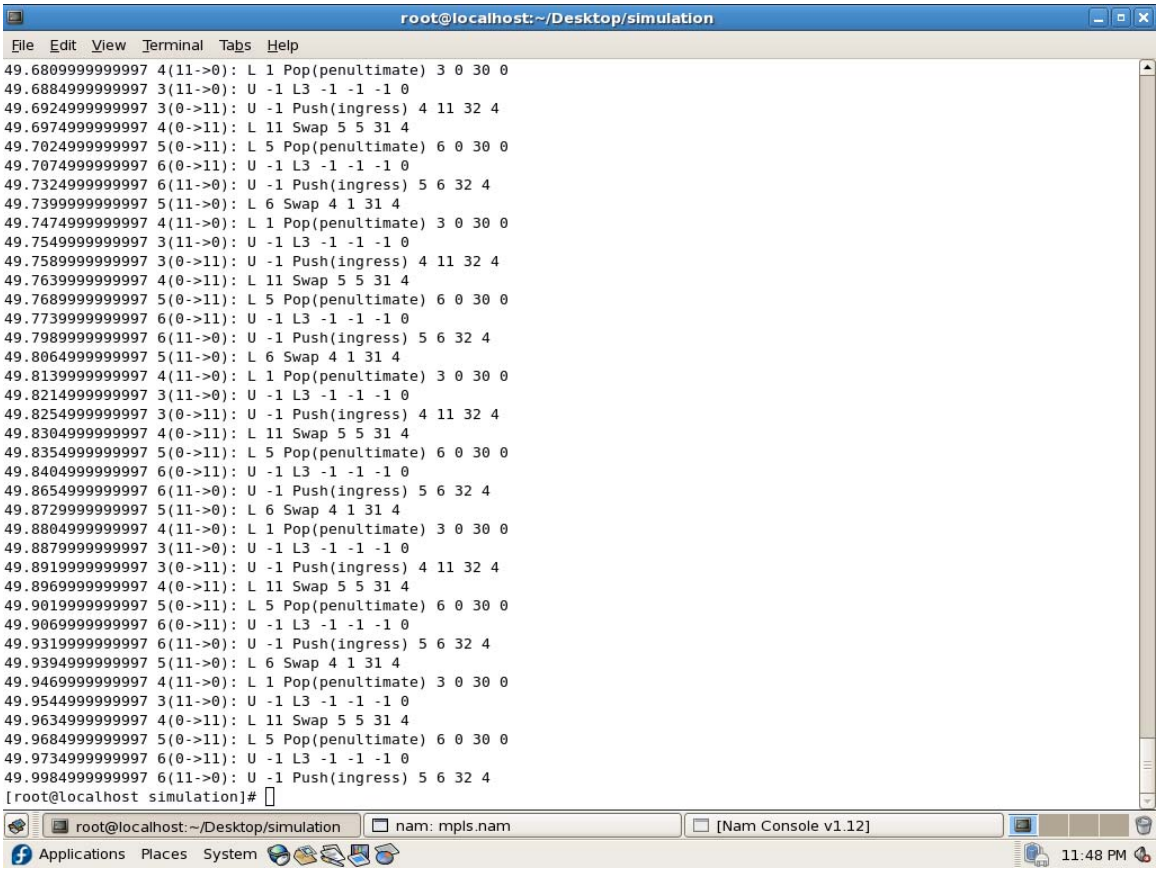

Figure 3.6: Example of MPLS Packet Trace

When the 'trace-mpls' API is used, an example of trace result might appear as Figure 3.6 shows. First field indicates the simulated time (in seconds) at which each event occurs. The next field indicates the address of the node that processes the packet. The next two fields indicate the packet's source and destination node addresses. The subsequent field indicates whether the received packet is unlabeled (U) or labeled (L). The next field is value of an incoming-label. Moreover a label operation is represented such as Push, Pop, and Swap. The subsequent two fields indicate the packet's out-going interface and out-

going label. The last two fields indicate the shim header's TTL and size. A shim header is a special header placed between layers two and three of the OSI model. The shim header carries the label which is used to forward MPLS packets.

3.7 Quality of service parameter

3.7.1 Delay

Delay is the time calculated by finding difference between the time sender application takes in passing the packet to transport protocol and reciever application in receiving of it. Delay is of great importance to some applications. One way Delay in MPLS and IP network is calculated using the different TCP variants by the following formula:

$$Delay = Tr - Ts$$

Where Ts is the time stamp of packet sending and Tr is time stamp of packet receiving respectively. Mean delay is recorded for results and calculated using this formula.

$$Mean\ \ Delay\ = \frac{Total\ \ Delay}{N}$$

Where n is the total number of packets sent in a simulation time.

3.7.2 Throughput

$$PercentageThroughput = \frac{Numberof\ \mathrm{Re}cieved\ Packets}{Numberof\ \ Sent\ Packets} x100$$

3.7.3 Packet Loss

$$PacketLoss = NumberofSentPackets - Numberof\mathrm{Re}cievedPackets$$

Chapter 04

Results and Discussions

The performance of network links can be described by two main parameters: the bandwidth and the delay. The more bandwidth the higher would be end-to-end throughput and better quality-of-service for the applications. Delay is important in order to guarantee prompt response from the network. And in recent times, with advanced high speed networks, extra latency can even reduce consumption of a TCP stream.

With disperse nature of the internet, many users share network resources; resultantly internet is unpredictable. As a direct result of this, an end user can never obtain expected performance. In the recent times it has become very important to observe the state of the network in order to estimate requirements for network planning, extra provisioning and bandwidth monitoring. Users want to verify whether they get the expected throughput and desired provisioning of network clouds.

This simulation consists of total number of 13 nodes; MPLS domain consists on 7 nodes and IP domain consist on 6 nodes. The IP domain consists of two networks each having three nodes. Links between nodes of MPLS and IP network are 1 Mb and 5ms processing delay. The total simulation time is 100 seconds. Both traffic source and destination are IP based.

The different TCP variants in MPLS/IP network are analyzed using different cases. The numbers of flows of the link were varied in order to check the effect of different flows on the throughput and delay. These traffics are run in both scenarios CBR and FTP. A detail analysis is presented and discussed.

4.1 Constant Bit Rate(CBR)

CBR traffic was simulated on single and multiple flows and results were traced. These results are then presented in form of tables and figures for TCP NewReno, TCP Reno, TCP Sack, TCP Tahoe and TCP Vegas.

4.1.1 Single Traffic

A CBR connection is established between node0 and node11 and this simulation is executed for New Reno, Reno, Sack, Tahoe and Vegas. Table 4.1(a) demonstrates average delays in milliseconds, while Table 4.1(b) depicts the percentage throughput of different variants on single flow.

Protocol	**Delay(millisec)**
TCP New Reno	202.555
TCP Reno	202.555
TCP Sack	202.555
TCP Tahoe	202.555
TCP Vegas	149.277

Table 4.1 (a): Delay of Single Flow CBR

Protocol	**Percentage Throughput**
TCP New Reno	99.7
TCP Reno	99.7
TCP Sack	99.7
TCP Tahoe	99.7
TCP Vegas	99.8

Table 4.1 (b): Percentage Throughput of Single Flow CBR

There is small number of packet loss. Average delays and throughputs of all the variants are approximately same and TCP Vegas shows 25% lesser average delay than other variants.

The behavior of TCP variants can also be observed by recording the average delay of packets on basis of their sequence numbers .i.e. average delay up to packet number 50,100,150 and so on for every variant is recorded, as shown in

Table 4.2.Figure 4.1 is depicting same information as in Table 4.2.As it is evident from the Figure that performance of Vegas is superior to other variants.

Number of Packets	New Reno	Reno	Sack	Tahoe	Vegas
50	0.169064	0.169064	0.169064	0.169064	0.130308
100	0.185912	0.185912	0.185912	0.185912	0.12552
150	0.191528	0.191528	0.191528	0.191528	0.130333
200	0.194336	0.194336	0.194336	0.194336	0.135159
250	0.196021	0.196021	0.196021	0.196021	0.138056
300	0.197144	0.197144	0.197144	0.197144	0.139986
350	0.197946	0.197946	0.197946	0.197946	0.141365
400	0.198548	0.198548	0.198548	0.198548	0.1424
450	0.199016	0.199016	0.199016	0.199016	0.143204
500	0.19939	0.19939	0.19939	0.19939	0.143848
550	0.199697	0.199697	0.199697	0.199697	0.144374
600	0.199952	0.199952	0.199952	0.199952	0.144813
650	0.200168	0.200168	0.200168	0.200168	0.145184
700	0.200353	0.200353	0.200353	0.200353	0.145503
750	0.200514	0.200514	0.200514	0.200514	0.145779
800	0.200654	0.200654	0.200654	0.200654	0.14602
850	0.200778	0.200778	0.200778	0.200778	0.146233
900	0.200888	0.200888	0.200888	0.200888	0.146422
950	0.200987	0.200987	0.200987	0.200987	0.146591
1000	0.201075	0.201075	0.201075	0.201075	0.146744
1050	0.201155	0.201155	0.201155	0.201155	0.146882
1100	0.201228	0.201228	0.201228	0.201228	0.147007
1150	0.201295	0.201295	0.201295	0.201295	0.147122
1200	0.201356	0.201356	0.201356	0.201356	0.147227

Table 4.2: End-to-end delay for TCP Variants New Reno, Reno, Sack, Tahoe and Vegas under CBR Single Flow

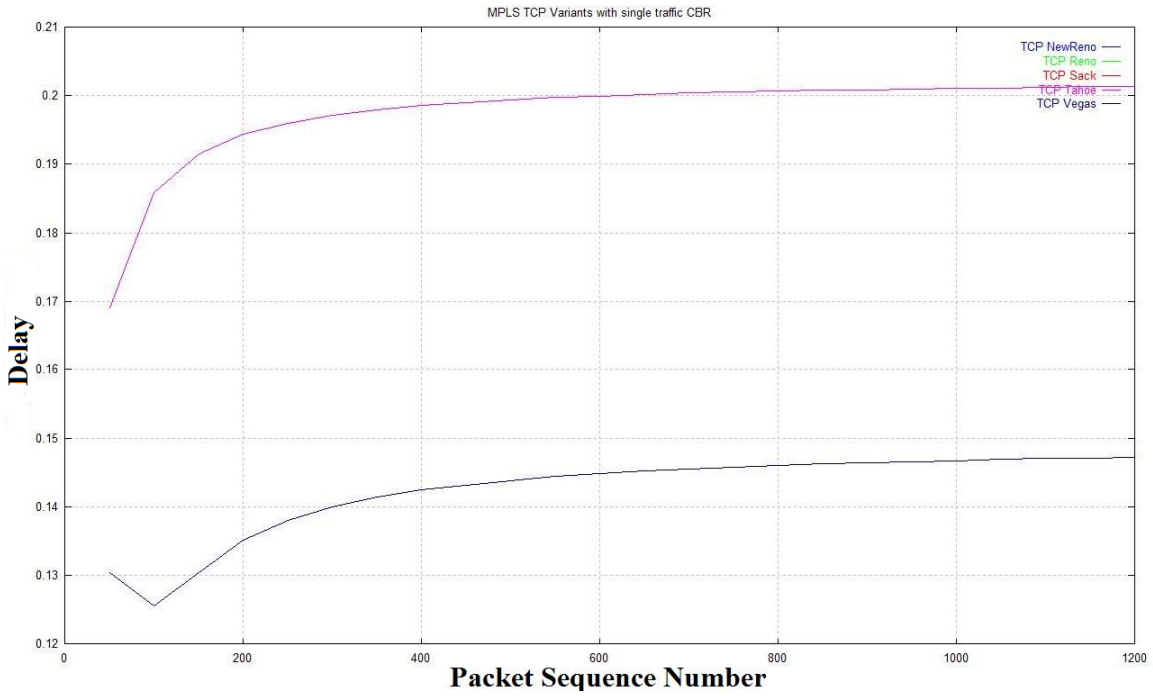

Figure 4.1: Comparison of TCP Variants with Single Traffics under CBR

4.1.2 Two Flows

This part of simulation creates two flows; both flows are of CBR type and start at same time. Node0 is sending data to Node 11 and Node1 is sending data to Node12.The intermediate cloud is MPLS and both senders and receivers are IP based. Tables 4.3(a) and 4.3(b) demonstrate delay and percentage throughput of different variations of TCP.

Average delay of all the variants is approximately same and TCP Vegas shows 40% lesser average delay than other variants. TCP Vegas does well again on two flows than other variants while analyzing average delay. Figure 4.2 gives an idea about average delays of all variants under observation at an interval after 50, 100, 150, 200 and so on until 1200 packets respectively.

Protocol	FlowID	Delay(millisec)
TCP New Reno	Flow1	440.808
	Flow2	441.058
TCP Reno	Flow1	440.808
	Flow2	441.058
TCP Sack	Flow1	440.808
	Flow2	441.058
TCP Tahoe	Flow1	440.808
	Flow2	441.058
TCP Vegas	Flow1	173.061
	Flow2	173.150

Table 4.3 (a): Delay of Two Flows CBR

Protocol	Percentage Throughput
TCP New Reno	99.5
TCP Reno	99.5
TCP Sack	99.5
TCP Tahoe	99.5
TCP Vegas	99.8

Table 4.3 (b): Percentage Throughput of Two Flows CBR

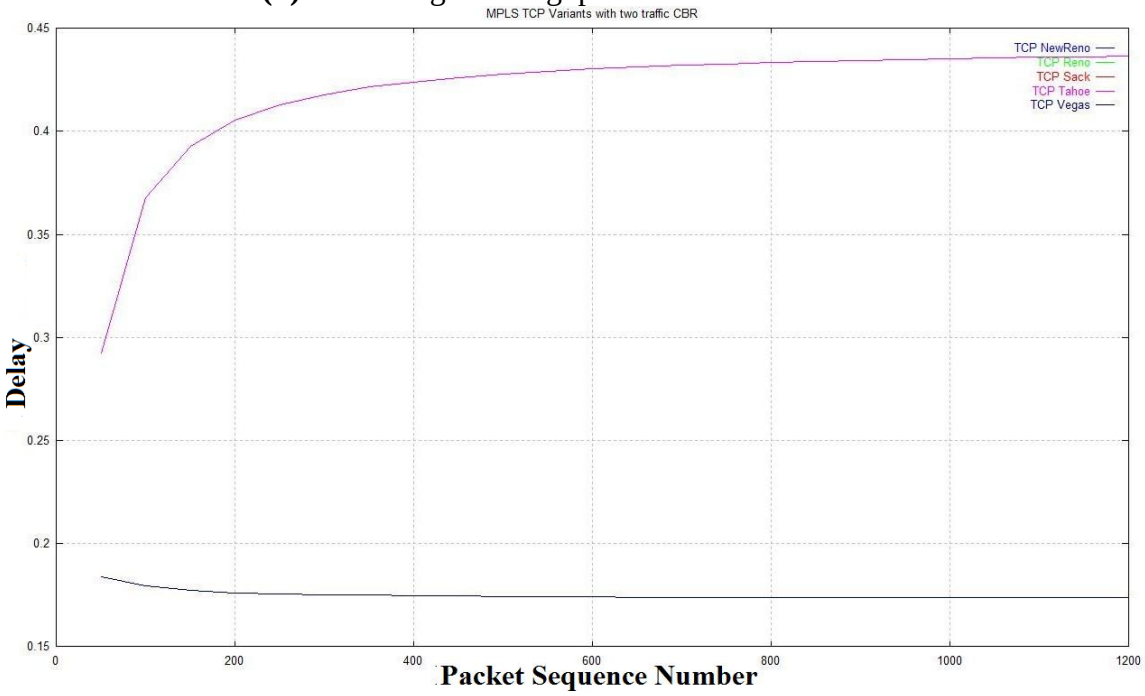

Figure 4.2: Comparison of TCP Variants with Two Traffics under CBR

4.1.3 Four Flows

Simulation produces four flows; all CBR flows were started simultaneously. There are two sessions on Node0 that are sending data to Node 11 and two sessions on Node1 which are sending data to Node12. The intermediate cloud is MPLS and both senders and receivers are IP networks. Tables 4.4(a) and 4.4(b) illustrate average delay of each flow and percentage throughput of variants of TCP under observation.

Protocol	**Flow ID**	**Delay(millisec)**
TCP New Reno	Flow1	585.640
	Flow2	596.261
	Flow3	603.354
	Flow4	597.696
TCP Reno	Flow1	570.435
	Flow2	610.045
	Flow3	587.092
	Flow4	593.988
TCP Sack	Flow1	601.222
	Flow2	611.663
	Flow3	625.661
	Flow4	604.966
TCP Tahoe	Flow1	573.041
	Flow2	576.960
	Flow3	578.665
	Flow4	578.468
TCP Vegas	Flow1	243.911
	Flow2	243.576
	Flow3	243.959
	Flow4	243.959

Table 4.4 (a): Delay of Four Flows CBR

Protocol	Percentage Throughput
TCP New Reno	98.5
TCP Reno	98.5
TCP Sack	98.3
TCP Tahoe	98.5
TCP Vegas	99.7

Table 4.4 (b): Percentage Throughput of Four Flows CBR

Vegas dropped very few packets and had highest throughput. On the whole average delay is amplified. Average delay of all the variants is approximately similar and only TCP Vegas demonstrated 40% smaller average delay. This proves better performance of TCP Vegas on four flows than other variants.

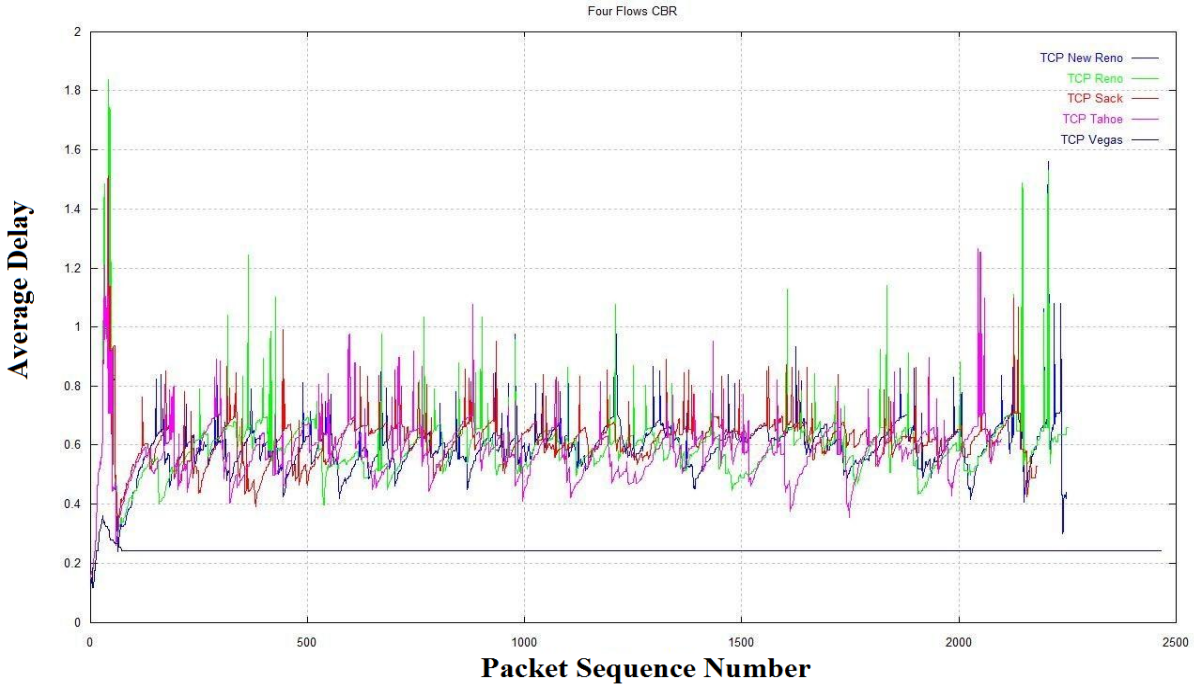

Figure 4.3: Four Flows CBR

At the start of simulation as exposed in Figure 4.3, fluctuation is seen because of network instability in Vegas but rest of variants show this instability till the end of

the simulation. The smoothness of line of Vegas behavior proves that after sending few packets the delay remains the same for rest of traffic, this proves that the delays of the packets remain same throughout simulation time.

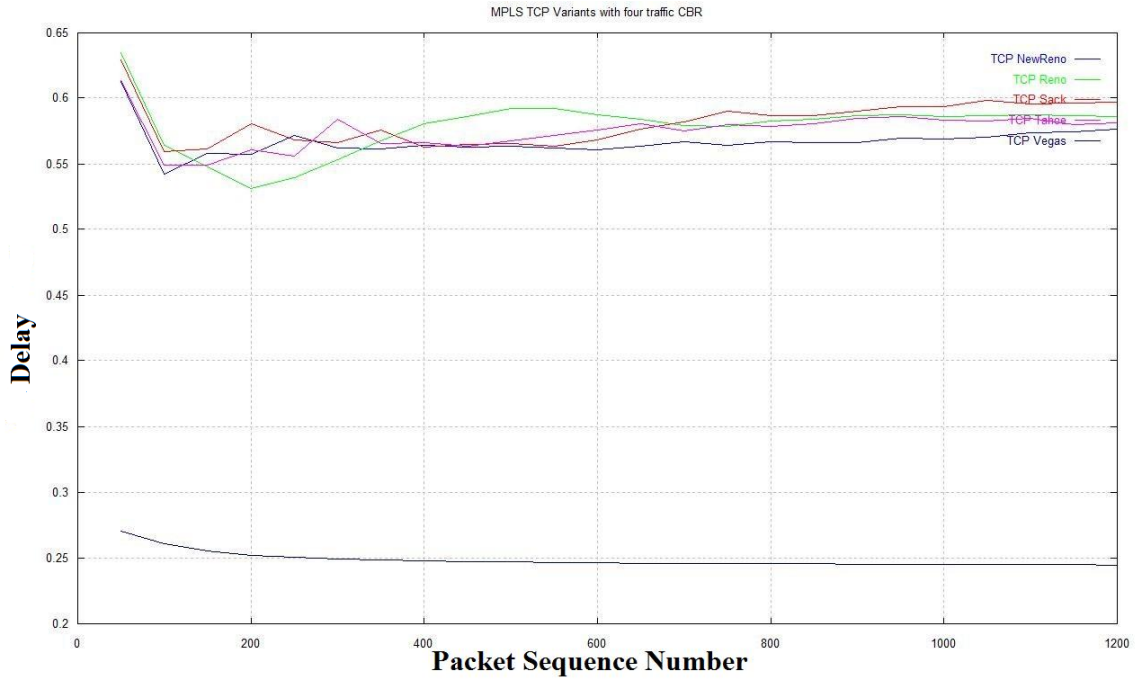

Figure 4.4: Comparison of TCP Variants with Four Traffics under CBR

Figure 4.4 is another outlook of packet's average delay plotted against their sequence number when the simulation sends some numbers of packets i.e. 50,100,150,200 and so on. TCP Reno has highest delay peak reaching 0.65 seconds until the end of simulation. TCP Reno is the TCP variant that induces packet losses as a sign of congestion.

4.1.4 Eight Flows

Simulation generates eight flows; all flows are of CBR type and start concurrently. There are four sessions on Node0 and four sessions on Node1 that are sending data to Node 11 and Node12 respectively. The intermediate cloud is

MPLS and both senders and receivers are IP based. Tables 4.5(a) and 4.5(b) give details average delay of all eight flows and percentage throughput of different variants of TCP.

Protocol	**Flow ID**	**Delay(millisec)**
TCP New Reno	Flow1	700.279869
	Flow2	735.202271
	Flow3	674.296937
	Flow4	720.512184
	Flow5	735.973601
	Flow6	749.842139
	Flow7	721.417000
	Flow8	691.349018
TCP Reno	Flow1	658.346388
	Flow2	660.996949
	Flow3	636.468069
	Flow4	639.180377
	Flow5	707.880977
	Flow6	699.855385
	Flow7	623.428567
	Flow8	673.569345
TCP Sack	Flow1	682.262498
	Flow2	687.364703
	Flow3	664.239830
	Flow4	681.605197
	Flow5	735.834975
	Flow6	692.867648
	Flow7	691.358682
	Flow8	763.340669
TCP Tahoe	Flow1	628.499090
	Flow2	630.102096
	Flow3	646.907666

	Flow4	633.740541
	Flow5	657.795324
	Flow6	688.076014
	Flow7	620.713057
	Flow8	646.306347
TCP Vegas	Flow1	313.645456
	Flow2	313.955819
	Flow3	314.289876
	Flow4	314.623111
	Flow5	313.748911
	Flow6	314.079965
	Flow7	314.415467
	Flow8	313.531206

Table 4.5 (a): Delay of Eight Flows CBR

Protocol	**Percentage Throughput**
TCP New Reno	95.7
TCP Reno	96.5
TCP Sack	96.2
TCP Tahoe	96.1
TCP Vegas	99.6

Table 4.5 (b): Percentage Throughput of Eight Flows CBR

On eight flows New Reno shows even worst performance. TCP Sack delay reaches highest peaks of 0.5 seconds. The variation of delay for individual packet shows instability of New Reno in increasing flows environment.

The traffic sent rate is increasing by number of flows; the delay also increases because more packets are induced by the traffic generator that further increases queuing delay, as total end-to-end delay is composed

of transmission delay, propagation delay, and queuing delay. More packets sent mean more queuing delay. Only few packets were lost in Vegas simulation whereas more packets were dropped in other variants of TCP. Collectively average delay increased with increase in flows but still Average delay of all the variants is roughly identical and TCP Vegas shows 50% lesser average delay than other variants. TCP Vegas gives highest throughput and lowest average delay.

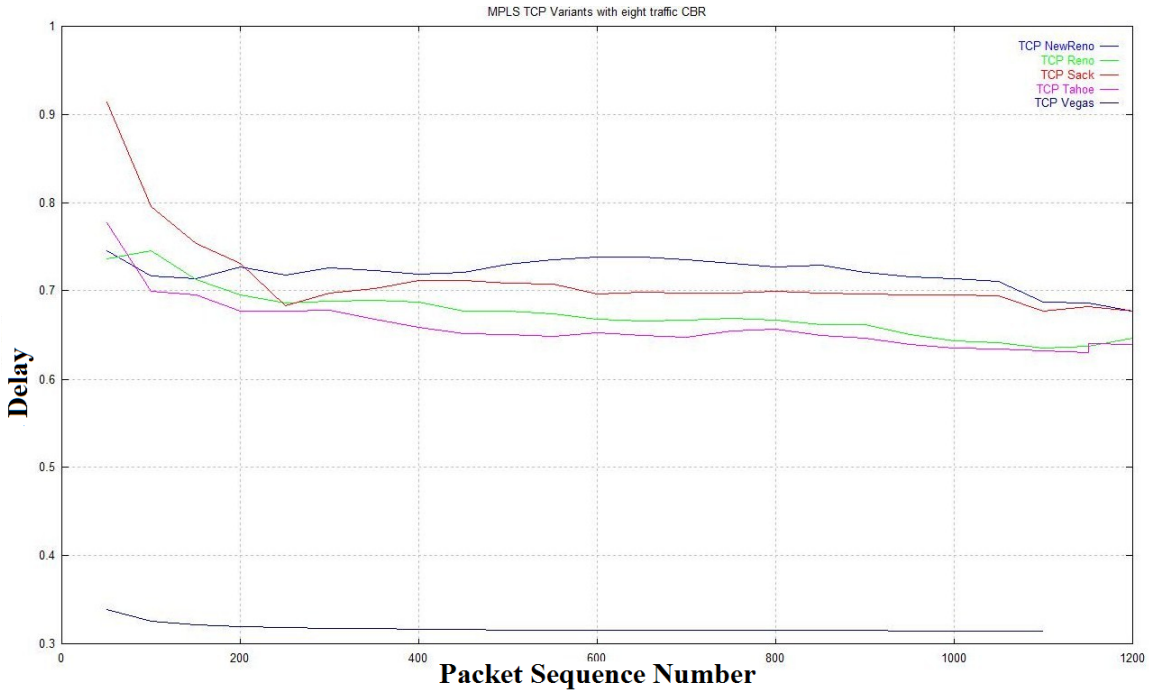

Figure 4.5: Comparison of TCP Variants with Eight Traffics under CBR

Figure 4.5 is portraying packets' average delay plotted against their sequence number for different intervals i.e. 50,100,150,200 and so on packets were sent before that observation. TCP sack has highest delay reaching 0.9 second, but from, 200th packet till the end of simulation New Reno is giving highest peaks of average delay.

4.2File Transfer Protocol (FTP)

FTP traffic has been run on single and multiple flows and results are recorded thereby. These results are then presented in structure of tables and figures for TCP NewReno, TCP Reno, TCP Sack, TCP Tahoe and TCP Vegas.

4.2.1 Single Traffic

An FTP connection is set up between node0 and node1 and this simulation is executed for New Reno, Reno, Sack, Tahoe and Vegas. Table 4.6(a) shows the values of these simulation average delays in milliseconds, while Table 4.6(b) depicts the percentage throughput of different variants on single flow and it can be examined that all variants are giving 100% throughput.

Protocol	**Delay(millisec)**
TCP New Reno	202.551
TCP Reno	202.551
TCP Sack	202.551
TCP Tahoe	202.551
TCP Vegas	149.270

Table 4.6 (a): Delay of TCP Variants on Single Flow FTP

Protocol	**Percentage Throughput**
TCP New Reno	100
TCP Reno	100
TCP Sack	100
TCP Tahoe	100
TCP Vegas	100

Table 4.6 (b): Percentage Throughput of TCP Variants on Single Flow FTP

On FTP flow there is no packet loss. Average delay of all the variants is more or less alike and TCP Vegas shows 25% lesser average delay than other variants. The accomplishments of TCP variants can also be observed by recording

the delay of packets .i.e. average delay of simulation traced until 50, 100, 150 and so on. Figure 4.6 is graphical representation of information represented in Table 4.7.

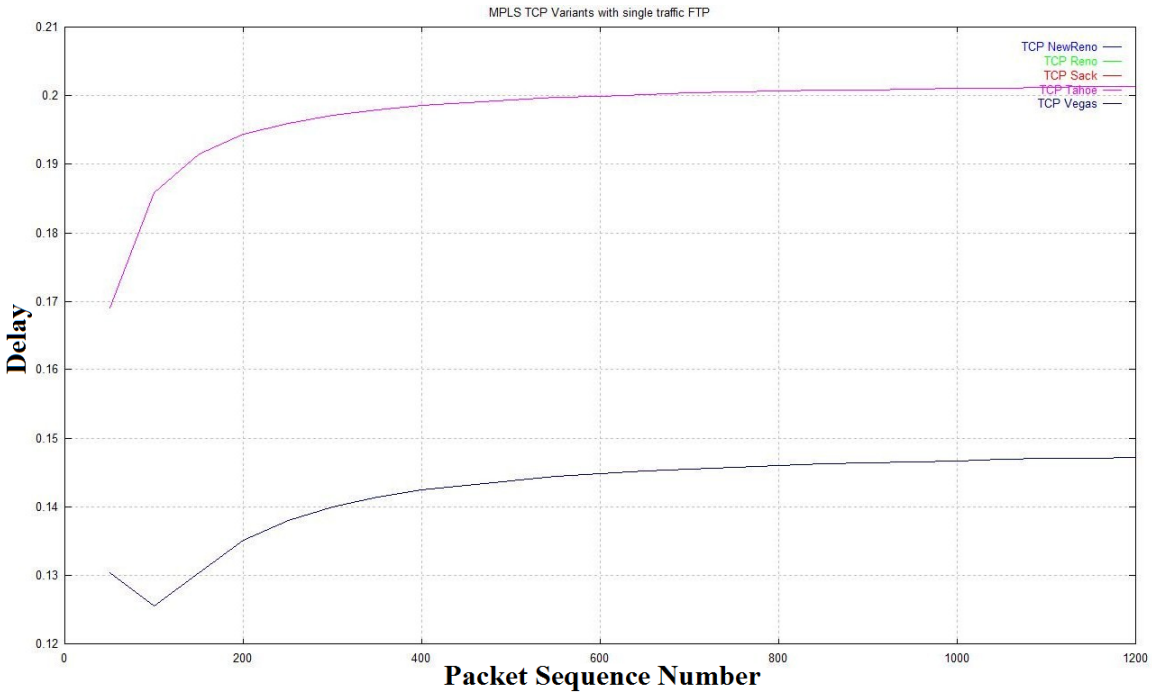

Figure 4.6: Comparison of TCP Variants with Single Traffics under FTP

Number of Packet	New Reno	Reno	Sack	Tahoe	Vegas
50	0.169064	0.169064	0.169064	0.169064	0.130308
100	0.185912	0.185912	0.185912	0.185912	0.12552
150	0.191528	0.191528	0.191528	0.191528	0.130333
200	0.194336	0.194336	0.194336	0.194336	0.135159
250	0.196021	0.196021	0.196021	0.196021	0.138056
300	0.197144	0.197144	0.197144	0.197144	0.139986
350	0.197946	0.197946	0.197946	0.197946	0.141365
400	0.198548	0.198548	0.198548	0.198548	0.1424
450	0.199016	0.199016	0.199016	0.199016	0.143204
500	0.19939	0.19939	0.19939	0.19939	0.143848
550	0.199697	0.199697	0.199697	0.199697	0.144374
600	0.199952	0.199952	0.199952	0.199952	0.144813
650	0.200168	0.200168	0.200168	0.200168	0.145184
700	0.200353	0.200353	0.200353	0.200353	0.145503
750	0.200514	0.200514	0.200514	0.200514	0.145779
800	0.200654	0.200654	0.200654	0.200654	0.14602
850	0.200778	0.200778	0.200778	0.200778	0.146233
900	0.200888	0.200888	0.200888	0.200888	0.146422
950	0.200987	0.200987	0.200987	0.200987	0.146591
1000	0.201075	0.201075	0.201075	0.201075	0.146744
1050	0.201155	0.201155	0.201155	0.201155	0.146882
1100	0.201228	0.201228	0.201228	0.201228	0.147007
1150	0.201295	0.201295	0.201295	0.201295	0.147122
1200	0.201356	0.201356	0.201356	0.201356	0.147227

Table 4.7: End-to-end delay for TCP Variants New Reno, Reno, Sack, Tahoe and Vegas under FTP Single Flow

4.2.2 Two Flows

This module of simulation produces two flows, both flows are of FTP type and all flows start at the same time. Node0 is sending data to Node 11 and Node1 is sending data to Node12. The intermediate cloud is MPLS and both senders and receivers are IP based.

The Table 4.8(a) is about average delay in milliseconds whereas Table 4.8(b) depicts percentage throughputs. In this model of FTP flow there is no packet loss. Average delay of all the variants is roughly similar but TCP Vegas shows 40% lesser average delay than other variants. In other words, TCP Vegas performs superior on two flows than other variants keeping average delay under discussion.

Protocol	**FlowID**	**Delay(millisec)**
TCP New Reno	Flow1	440.777
	Flow2	441.031
TCP Reno	Flow1	440.777
	Flow2	441.031
TCP Sack	Flow1	440.777
	Flow2	441.031
TCP Tahoe	Flow1	440.777
	Flow2	441.031
TCP Vegas	Flow1	173.062
	Flow2	173.152

Table 4.8 (a): Delay of TCP Variants on Two Flows FTP

Protocol	Percentage Throughput
TCP New Reno	100
TCP Reno	100
TCP Sack	100
TCP Tahoe	100
TCP Vegas	100

Table 4.8 (b): Percentage Throughput of TCP Variants on Two Flows FTP

Figure 4.7 shows the behavior of TCP variants observed by recording the delay of the packets plotted with their sequence numbers .i.e. average delay after packet number 50, 100, 150 and so on, sent by every variant, is traced.

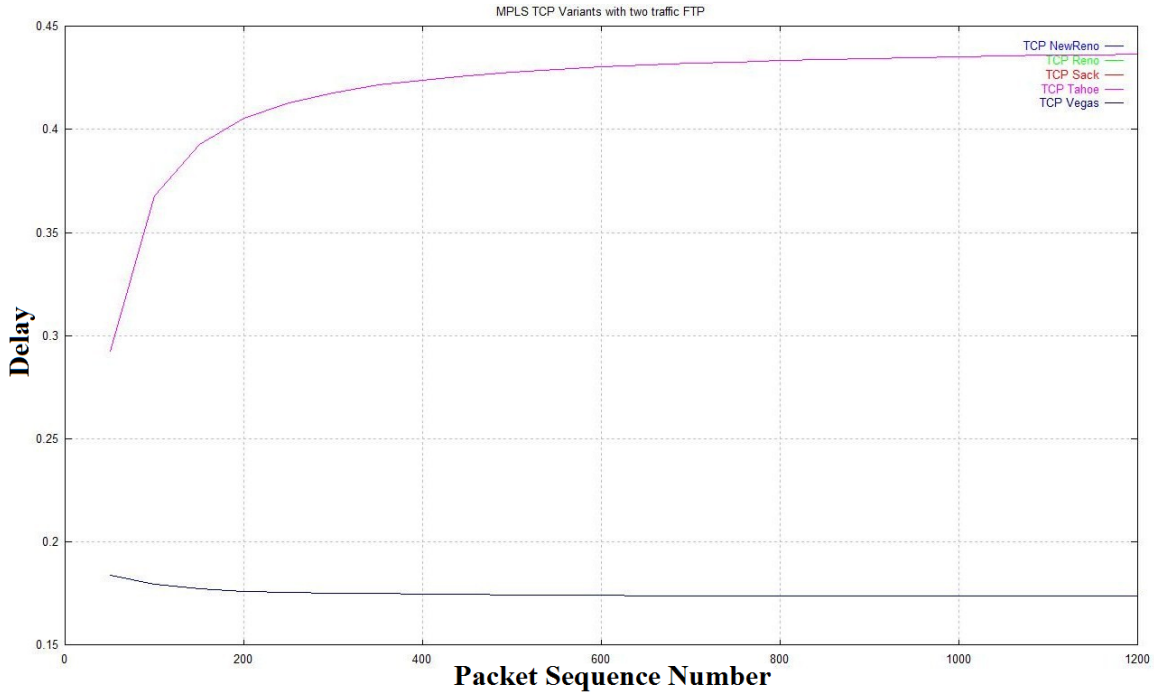

Figure 4.7: Comparison of TCP Variants with Two Traffics under FTP

4.2.3 Four Flows

Simulation creates four flows; all flows are of FTP type; they start parallel to each other. There are two sessions on Node0 that is sending data to Node 11 and two sessions on Node1 which is sending data to Node12. Together senders and receivers are IP based communicating via MPLS network. Tables 4.9(a) and 4.9(b) exhibit average delay and percentage throughput of different variants of TCP.

Protocol	**Flow ID**	**Delay(millisec)**
TCP New Reno	Flow1	605.975
	Flow2	605.082
	Flow3	615.735
	Flow4	608.651
TCP Reno	Flow1	569.332
	Flow2	609.487
	Flow3	586.963
	Flow4	592.969
TCP Sack	Flow1	605.138
	Flow2	612.934
	Flow3	627.058
	Flow4	606.446
TCP Tahoe	Flow1	572.651
	Flow2	576.190
	Flow3	578.452
	Flow4	577.673
TCP Vegas	Flow1	243.926
	Flow2	243.585
	Flow3	243.975
	Flow4	243.694

Table 4.9 (a): Delay of TCP Variants on Four Flows FTP

Protocol	Percentage Throughput
TCP New Reno	99.9
TCP Reno	99.2
TCP Sack	99.0
TCP Tahoe	96.1
TCP Vegas	100

Table 4.9 (b): Percentage Throughput of TCP Variants on Four Flows FTP

Vegas is giving 100% throughput as there is no packet lost. Generally a rise in average delay is observed but still all the variants are more or less at same level except TCP Vegas which shows 40% lesser average delay than other variants. This shows that TCP Vegas performs better on four flows than other variants in case of average delay analysis, similarly packet loss is much lesser than other variants of TCP.

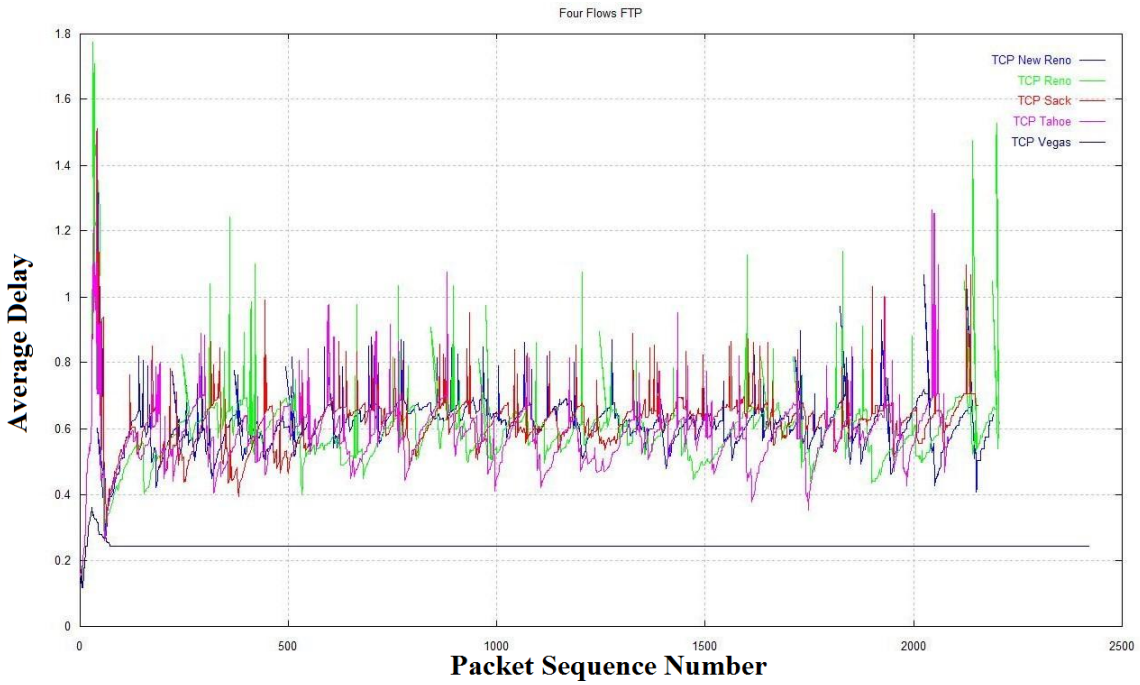

Figure 4.8: Four Flows FTP

The Figure 4.8 gives a glimpse of behaviors of TCP variants by plotting average delay of packets in seconds across y-axis. Each sharp edge shows the abrupt change in delay of that particular variant. Vegas shows a little jitter in the beginning but later on illustrates a smooth performance, this show that the delay of the packets remains the same throughout simulation time. There is no packet loss in TCP Vegas. TCP Reno has highest delay reaching 1.8 second, until the end of simulation Reno is giving highest peaks of delay.

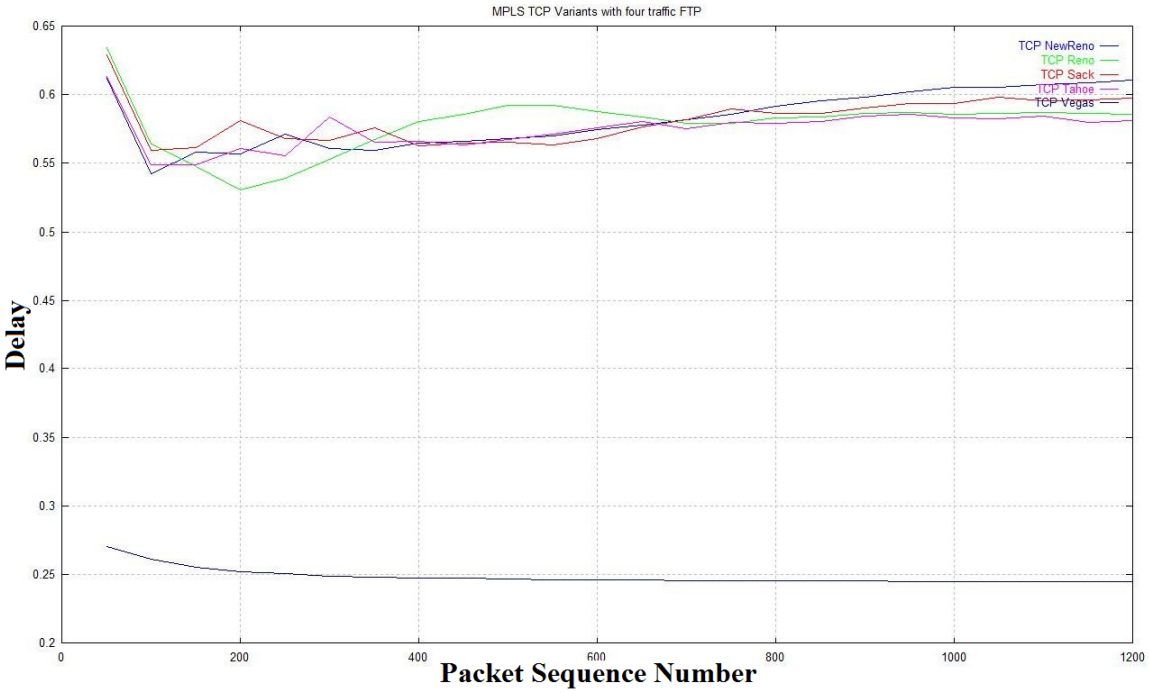

Figure 4.9: Comparison of TCP Variants with Four Traffics under FTP

Figure 4.9 illustrates the behavior of TCP variants observed by recording the delay of packets on basis of their sequence numbers .i.e. average delay of packets after transmission of 50,100,150 and so on packets for every variant.

4.2.4 Eight Flows

Eight FTP flows are generated that are started at same time. Four sessions are maintained at Node0 that is sending data to Node 11 and four sessions at

Node1 that is sending data to Node12. The intermediate cloud is MPLS and both senders and receivers are IP based. Tables 4.10(a) and 4.10(b) show average delay in milliseconds and percentage throughput of different variants of TCP.

No packet loss is monitored in Vegas, while there are evidences that packets are lost by other variants.

Protocol	**Flow ID**	**Delay(millisec)**
TCP New Reno	Flow1	691.869927
	Flow2	662.255458
	Flow3	688.122176
	Flow4	699.388716
	Flow5	687.965784
	Flow6	728.847338
	Flow7	689.433813
	Flow8	661.223415
TCP Reno	Flow1	660.768065
	Flow2	660.907158
	Flow3	636.338427
	Flow4	638.675302
	Flow5	708.683366
	Flow6	700.968139
	Flow7	623.384563
	Flow8	673.609520
TCPSack	Flow1	685.240279
	Flow2	692.608726
	Flow3	666.291201
	Flow4	683.729984
	Flow5	739.583802
	Flow6	733.796826
	Flow7	695.336639
	Flow8	767.577396

TCPTahoe	Flow1	626.950386
	Flow2	629.545637
	Flow3	645.618190
	Flow4	632.915162
	Flow5	656.544641
	Flow6	686.928511
	Flow7	618.480730
	Flow8	645.734697
TCP Vegas	Flow1	313.654211
	Flow2	313.970171
	Flow3	314.307196
	Flow4	314.649593
	Flow5	313.759531
	Flow6	314.096555
	Flow7	314.434729
	Flow8	313.537906

Table 4.10 (a): Delay of TCP Variants on Eight Flows FTP

Protocol	**Percentage Throughput**
TCP New Reno	96.5
TCP Reno	97.4
TCP Sack	96.8
TCP Tahoe	97.0
TCP Vegas	100

Table 4.10 (b): Percentage Throughput of TCP Variants on Eight Flows FTP

Cumulatively there is a rise in average delay; however TCP Vegas shows 50% lesser average delay than other variants, Average delay of all the other variants is almost similar. This shows that TCP Vegas performs better on eight flows than other variants keeping average delay under observation; packet loss of

Vegas is zero but other variants of TCP did not give 100% throughput. New Reno gave lowest throughput.

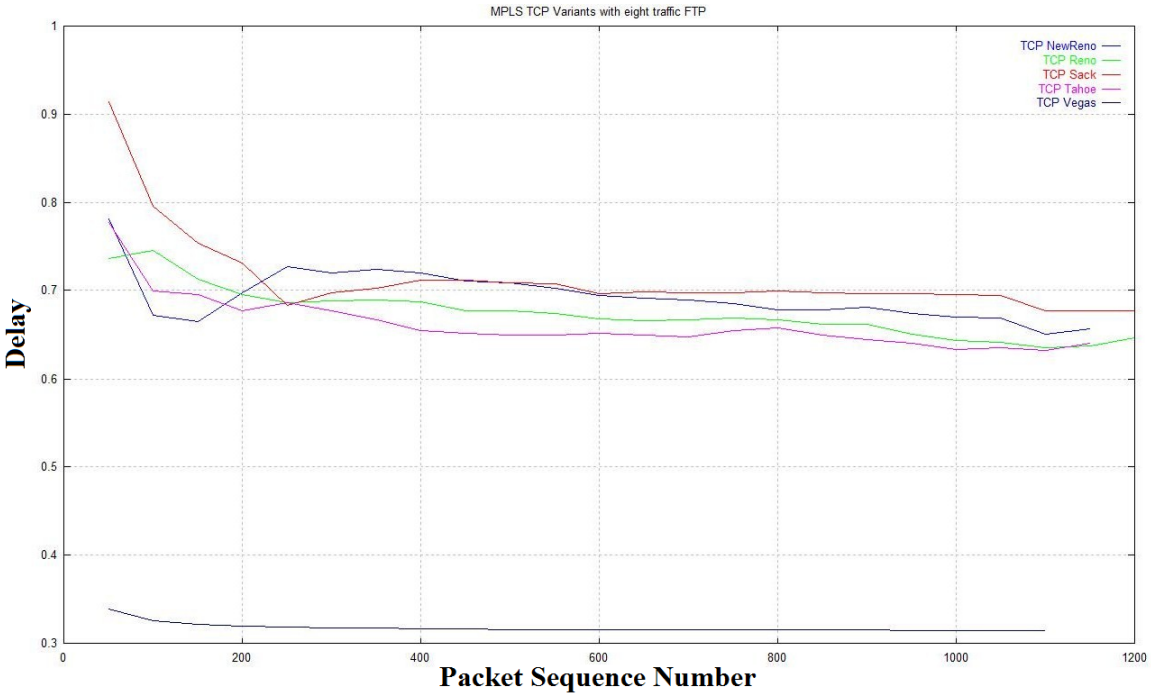

Figure 4.10: Comparison of TCP Variants with Eight Traffics under FTP

Figure 4.10 shows the behavior of TCP variants observed by recording the delay of packets on basis of their sequence numbers .i.e. delay of packet number 50,100,150 and so on for every variant is documented.

4.3 Error rate induction in FTP Single Flow

TCP provides reliable end to end transport layer protocol. Nowadays approximately 90% traffic on internet is using this protocol. Congestion avoidance in TCP allows the application to increase by one packet whenever an acknowledgement is received; allowing full utilization of available bandwidth.

Most important feature of TCP is its Congestion Control strategy. In wired network, whenever there is a packet loss, it indicates that network is congested. Inducing error rate explicitly shows the behavior of different TCP variants, as

some variants reduce congestion window unnecessarily. Performance of TCP is affected by various factors like link capacity, RTT, random losses, short flows etc.

As IP is an unreliable network, adding TCP to it is for providing reliability via sliding window scheme, ACK, sequence number and control flow to avoid overflowing of receiver buffer [22].

Protocol	**Delay (sec)**	**Variance (millisec)**
TCP New Reno	0.145702	0.001609
TCP Reno	0.145680	0.002137
TCP Sack	0.145641	0.001558
TCP Tahoe	0.143764	0.001775
TCP Vegas	0.131559	0.000884

Table 4.11: Delay and Variance of TCP Variants with 1% Error Rate

Protocol	**Delay (millisec)**	**Variance (millisec)**
TCP New Reno	0.135669	0.002390
TCP Reno	0.138227	0.003512
TCP Sack	0.135028	0.001944
TCP Tahoe	0.137339	0.002414
TCP Vegas	0.129579	0.003029

Table 4.12: Delay and Variance of TCP Variants with 2% Error Rate

Protocol	**Delay (millisec)**	**Variance (millisec)**
TCP New Reno	0.136803	0.003754
TCP Reno	0.139429	0.005499
TCP Sack	0.134300	0.002736
TCP Tahoe	0.136965	0.003030
TCP Vegas	0.131611	0.003224

Table 4.13: Delay and Variance of TCP Variants with 3% Error Rate

Protocol	Delay (millisec)	Variance (millisec)
TCP New Reno	0.142735	0.010064
TCP Reno	0.143514	0.007368
TCP Sack	0.135631	0.003895
TCP Tahoe	0.141868	0.005738
TCP Vegas	0.136873	0.008297

Table 4.14: Delay and Variance of TCP Variants with 4% Error Rate

Protocol	Delay (millisec)	Variance (millisec)
TCP New Reno	0.143130	0.006641
TCP Reno	0.149531	0.011490
TCP Sack	0.139308	0.005425
TCP Tahoe	0.143978	0.006306
TCP Vegas	0.140118	0.010976

Table 4.15: Delay and Variance of TCP Variants with 5% Error Rate

TCP Tahoe is the scheme of TCP that deploys slow start mechanism to prevent the problem of congestion. It is a reactive mechanism. New Reno is an active variant of TCP which is used for multiple packet losses. It provides the solution for oscillating congestion window to resolve problem faced by TCP Reno. For the solution to the problems of TCP's inability to tell about the multiple packet-dropping, TCP Sack was proposed. TCP Vegas is the proactive variant of TCP that anticipates the intended congestion on the basis of round trip times of the data packets. A comparison of the values of all flavors of TCP under test is shown in Table 4.11 to 4.15 on IP and MPLS network.

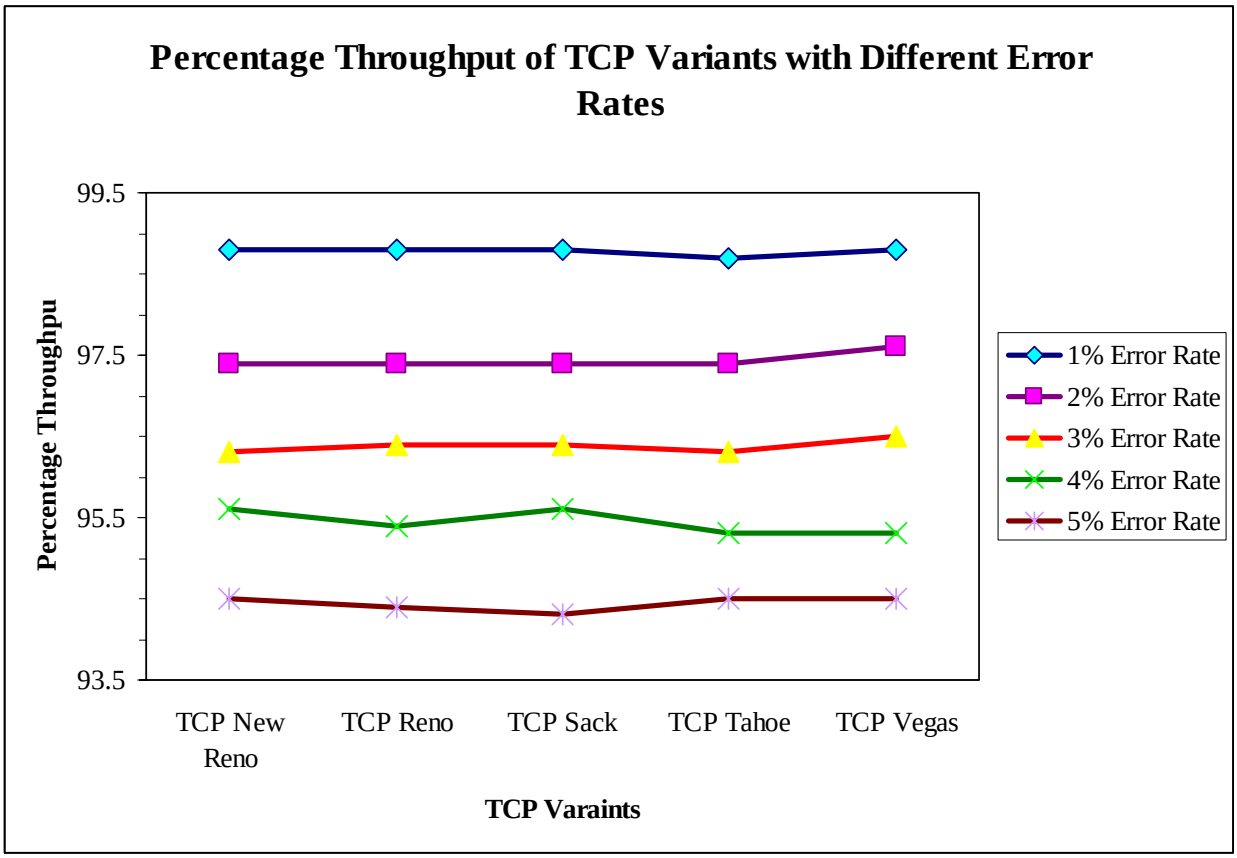

Figure 4.11: Percentage throughput of TCP variants with different Error Rates

Figure 4.11 gives an idea about percentage throughput achieved by TCP variants. As the probability of packet loss is increased 1 to 5% the throughput of variants deteriorates. This is because every time a retransmitted packet is lost, the frequency of dropped packet increases and all variants of TCP suffer from numerous drops and timeouts. When random packet loss was introduced, throughput of all variants remained the same i.e. 98, 97, 96, 95 and 94 percent for 1, 2, 3, 4 and 5 percent error rate respectively. Vegas gave lowest end-to-end delay till 3% error rate but dramatically TCP Sack gave delay even lower than Vegas for 4 and 5 % error rate.

The congestion window of all the sources frequently goes down to the smaller value. New Reno had largest value of average congestion window in 1% error rate case. But, Vegas had largest value and Tahoe had smallest value of average

congestion window throughout the experiment scenarios, this shows Tahoe's inferiority to Vegas.

It is evident that as loss rate increases, throughput decreases. So, it can be concluded that there exists an inverse relationship between loss rates and throughput. As error rate increases throughput decreases gradually. Similarly congestion window size also decreases as observed in Table 4.16.

Protocol	**Average Congestion Window 1% Error Rate**	**Average Congestion Window 2% Error Rate**	**Average Congestion Window 3% Error Rate**	**Average Congestion Window 4% Error Rate**	**Average Congestion Window 5% Error Rate**
TCP New Reno	13.6817	9.88161	8.238325	7.110332	5.899398
TCP Reno	13.32309	9.504204	8.021566	6.503233	5.660716
TCP Sack	13.66305	9.766847	8.297817	7.012776	5.865765
TCP Tahoe	12.30899	9.252068	7.836436	6.232895	5.406906
TCP Vegas	13.11434	10.87495	10.07931	8.910358	8.471908

Table 4.16: Average Congestion Window Size of TCP Variants with different Error rates

Chapter 05
Conclusions and Future Work

5.1 Conclusion

TCP is arguably the most significant protocol on the internet today. The most important feature of TCP is its complex algorithms for congestion control. TCP tries to achieve the best bandwidth rate vigorously on any network. It keeps on pushing high transfer rate but continuously reduces this transfer rate on detecting errors from time to time. Observing the behavior of TCP reveals alot about behaviors of different variants of TCP on IP and MPLS network. It produced charts that plot throughput, congestion window, average end-to-end delay, variance of delay, sent / received data, sequence number analysis and packet lost etc.

This chapter will recapitulate the key results obtained while simulating the protocols and it will also show whether it has resolved the research problem rose in the beginning of the thesis. The future work is also mentioned briefly, and will conclude the report.

5.1.1 Summary of Results

This project studies the behavior and performance of TCP when applied over wired IP and MPLS network by surveying the TCP versions based on End-to-End scheme which have been designed for best-effort traffic. A variety of results are achieved from the simulation analysis using performance parameters when TCP versions were applied over the IP and MPLS network.

A number of variants of TCP exist, that are classically well-known for their particular congestion control and packet loss recovery methods integrated into the protocol. The research sticks to the simulation results as evidence that TCP variants have minor effect on the overall results except in few cases. Simulation observation based on TCP Reno, New Reno, Sack, Tahoe and Vegas clearly describes about the performance evaluation through measuring throughput, delay and congestion window for FTP and CBR.

Thus, TCP Vegas showed superior performance and improved throughput over the other variants. As the Vegas being delay based protocol, when it observes RTT increases, it assumes that it is due to network congestion and decreases the sending rate. In particular, the throughput performance of TCP Vegas over a lossless wired network exhibits a 40-50 % improvement over Reno, New Reno, Tahoe and Sack implementations. Simulation was run for different number of flows: single to two, four and eight flows for both FTP and CBR traffics. The performance of all the variants gradually decreases with increase in the number of competitive flows.

In scenario of FTP the Vegas even did not lose a single packet till eight flows because of its better congestion avoidance strategy. And also gave highest throughput throughout the experimentation. Vegas utilized the bandwidth more efficiently than other flavors of TCP.

Also 1 to 5% error rate was induced to carefully observe the congestion window size, throughput and delay of all above mentioned variants. When random packet loss was introduced, throughput of all variants remained same i.e. 98, 97, 96, 95 and 94 percent for 1, 2, 3, 4 and 5 percent error rate respectively. Vegas gave lowest end-to-end delay till 3% error rate but dramatically TCP Sack gave delay even lower than Vegas for 4 and 5 % error rate. Vegas had largest average congestion window size and Tahoe had smallest average congestion window, this shows Tahoe's inferiority to Vegas.

Another important finding about this IP and MPLS network which have nominal IP neighboring, is that Vegas can be a very good and reliable choice to be used in real-time data transmission because of its stable jitter free end-to-end delay behavior.

5.2 Future Work

This portion now discusses several directions for future work, both in addressing the limitations of this work and in exploring new possibilities. In future we hope to increase the application scale by running the network with particular attention on real-time applications (VOIP and Streaming multimedia). This simulation did not conduct any real world testing. Although the major goal is not to have an applicability study in this work, it is crucial to check in a real word system.

In addition, it did not pay attention to the multicast, wireless, or anything like that kind of network environment. This test has focused on a fixed network environment. Once it is proved whether IP and MPLS work fine in a real world, it is encouraged to move on to one of those areas. This work has focused on MPLS-based technologies. An in-depth analysis of current and future optical network technology (GMPLS) may be the future area of work.

Finally, we assumed that the packet size is fixed. There can be another chance to look into more detail for a variable packet size. Similarly, other network parameters can be altered to examine behavior of the variants, for example number of IP and MPLS nodes, bandwidth, delay and error rate. Also, selection of other variants of TCP protocols (especially New Vegas) can be useful for the performance evaluation or other parameters of performance could be considered for simulation.

Simulation is done in NS-2, another possibility is doing the same work through another tool like OPNET.

REFERENCES

[1] S. M. S. Floyd. ns-LBL Network Simulator, 1997.

[2] Kai Xu, Ye Tian, Nirwan Ansari, K. Xu et al., “Improving TCP performance in integrated wireless communications networks”, Computer Networks 47 219-237,2004

[3] Dongkyun Kim Juan-Carlos Cano and P. Manzoni, “A comparison of the performance of TCP-Reno and TCP-Vegas over MANETs”, C-K. Toh, 2006 IEEE

[4] T.D. Dyer and R. V. Boppana. “A comparison of TCP performance over three routing protocols for mobile ad hoc networks”. In proceedings of the 2001 ACM international Symposium on Mobile ad hoc networkin & computing, pages 56-66. ACmpress, 2001.

[5] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. Standards Track, RFC 2581, Internet Engineering Task Force, 1999.

[6] V. Jacobson and M. Karels. “Congestion Avoidance and Control.” ACM SIGCOMM, 1988. pp. 273-288.

[7] M. Saeed Akbar, Syed Zubair Ahmed, M. Abdul Qadir “Quantitative Analytical Performance of TCP Variants in IP and MPLS Networks” Center for Distributed and Semantic Computing (CDSC), Mohammad Ali Jinnah University, Islamabad campus

[8] X. Dubois.” Performance of Different TCP Versions over Common/Dedicated UMTS Channels”. Master Thesis, University of Namur, 2005.

[9] Zhong Ren Chen-Khong Tham Chun-Choong Foo Chi-Chung Ko, " Integration of Mobile IP and MPLS", IEEE ICC 2001, vol. 7, pp 2123-2127, Helsinki, Finland, June 2001.

[10] M. Asante and R.S. Sherratt (UK). “Mobile IP Convergence in MPLS-based Switching”. In Proceeding of Wireless and Optical Communication MultiConference,

July,2004.

[11] Adam Wierman, Takayuki Osogami, and Jorgen Olsen. A Unified Framework for Modeling TCP-Vegas, TCP-SACK, and TCP-Reno. Proceedings of MASCOTS, 2003

[12] Mazleena Salleh and Ahmad Zaki Abu Bakar . Comparison of TCP Variants over Self-Similar Traffic, Proceedings of Computers, Communications, & Signal Processing with Special Track on Biomedical Engineering, Page(s):85 - 90, Nov. 2005

[13] H. Balakrishnan, V. N. Padmanabhan, S. Sechan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links", IEEEIACM Trans. Networking, vol. 5, no. 6, pp. 756-769, Dec. 1997.

[14] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet", IEEE J. Select. Areas Commun., vol. 13, pp. 1465-1480, Oct. 1995.

[15] J. Ols'en. Stochastic Modeling and Simulation of the TCP Protocol. PhD thesis, Department of Mathematics, Uppsala University, Sweden, Oct. 2003.

[16] J. Mo, R. J. La, V. Anantharam, and J. Walrand, “Analysis and comparison of TCP Reno and Vegas,” in Proc. INFOCOM ’99, vol. 3, New York, Mar. 1999, pp. 1556–1563.

[17] Go Hasegawa, Kenji Kurata and Masayuki Murata. Analysis and Improvement of Fairness between TCP Reno and Vegas for Deployment of TCP Vegas to the Internet. Proceedings of the 2000 International Conference on Network Protocols, Page: 177, 2000

[18] Cheng P. Fu and Soung C. Liew. A Remedy for Performance Degradation of TCP Vegas in Asymmetric Networks, Communications Letters, IEEE Volume 7, Issue 1, Page(s): 42 - 44, Jan 2003

[19] Thomas Bonald .Comparison of TCP Reno and TCP Vegas via Fluid Approximation, November 1998. Available as INRIA Research Report

[20] Yi-Cheng Chan, Chia-Tai Chan, Yaw-Chung Chen and Cheng- Yuan Ho "Performance Improvement of Congestion Avoidance Mechanism for TCP Vegas", Proceedings of the 10th International Conference on Parallel and Distributed Systems (ICPADS'04) 1521-9097/04 2004 IEEE

[21] Joel Sing and Ben Soh, "TCP New Vegas: Performance Evaluation and Validation", Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06) 0- 7695-2588-1/06 2006 IEEE

[22] S. Floyd. Issues of TCP with SACK, January 1996. Tech. Report, http://www.icir.org/floyd/sacks.html. (Cited on page 6 and 58.)