2017

Network Performance Analysis Using Cisco VIRL

Charitra Maharjan
Louisiana State University and Agricultural and Mechanical College

NETWORK PERFORMANCE ANALYSIS USING CISCO VIRL

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirement for the degree of
Master of Science

in

The Division of Electrical and Computer Engineering

by
Charitra Maharjan
B.E., Advanced College of Engineering and Management, 2013
August 2017

ACKNOWLEDGEMENTS

Dr. Suresh Rai, Prof. of Electrical and Computer Engineering, has been the inspirational and wonderful person who introduced me to the world of Computer Networks. I sincerely thank him for the opportunity to work with him. I would also like to appreciate for his moral support and immense guidance towards my completion of master's program.

My sincere thanks go to Dr. Bijay Karki, Prof. and Chair of Computer Science, and Dr. Xuebin Liang, Prof. of Electrical and Computer Engineering, for their consent to be the committee members and for their valuable suggestions in improving this document.

Deepest gratitude to my parents Mr. Gyan Bahadur Maharjan and Mrs. Ratna Maya Maharjan, and to my host family in USA for their love, and blessing. I sincerely thank my dear brothers Vigyan Maharjan and Prabhat Maharjan for all the moral support and inspiration.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	-	Application Program Interface
AS	-	Autonomous System
BGP	-	Broader Gateway Protocol
CDP	-	Cisco Discovery Protocol
DNS	-	Domain Name Servers
GNS3	-	Graphical Network Simulator
GUI	-	Graphical User Interface
HTTP	-	Hyper Text Transfer Protocol
IP	-	Internet Protocol
LAN	-	Local Area Network
LXC	-	Linux Container
MSS	-	Maximum Segment Size
MTU	-	Maximum Transmission Unit
OS	-	Operating System
OSI	-	Open System Interconnection
OSPF	-	Open Shortest Path First
PACP	-	Packet Capture
QoS	-	Quality of Service
SCP	-	Secure Copy
SMTP	-	Simple Mail Transfer Protocol
SSH	-	Secure Shell
TCP	-	Transmission Control Protocol
UDP	-	User Datagram Protocol
VoIP	-	Voice Over IP
VIRL	-	Virtual Internet Routing Lab
WAN	-	Wide Area Network

ABSTRACT

This thesis provides a detailed analysis of the effects of TCP and UDP traffic over a LAN and WAN medium. In addition, it also analyses some real time applications like audio, video and web browsing that is affected by TCP traffic while sharing a bottleneck node and/or link resources.

As network industry is growing continuously, the network administrator should be aware of TCP and UDP traffic that is traversing through their network. The analysis and monitoring of the traffic is crucial as it directly affects the performance of the network. Finding a cause for the poor performance of the network is quite important because it gives an idea to troubleshoot and resolve the issues effectively.

In this thesis, we have created topologies using Cisco's Virtual Internet Routing Lab (VIRL) [7]. They replicate an organizational infrastructure with client-server environment in LAN and WAN. Routing protocols such as OSPF and BGP are used to mimic the real world internet. A built-in LXC-iperf tool is used to generate the TCP and UDP traffic. During the generation of the traffic, various parameters are changed or controlled to see their effect on the network performance. As a learning and informative research, this thesis considers several Quality of Service (QoS) parameters that characterize the performance of an overall network.

In particular, this thesis obtains packet loss, throughput, and jitter as QoS parameters when the resource has both TCP and UDP traffics simultaneously. We have examined, the effects on (i) TCP throughput and packet loss and (ii) UPD packet loss and jitter of (a) real time audio, (b) video, and (c) web browsing applications. These parameters examine how the traffic be manipulated to keep minimum packet loss, minimum jitter and maximum throughput. It is needless to say that all are competing with each other (TCP/UDP traffic) for sharing bottleneck resources. We have used a sample time of 15 seconds for each of our experimental results presented in this thesis.

Our analysis shows that the best performance of the real time video and audio application is obtained when we select large size packet but its size being less than MTU of the link (without reducing the data rates). Similarly, in case of web browsing, we notice that throughput increases by increasing the window size and decreasing the latency. Efficient outcomes with the traffic analysis are achieved only if the experiments are carried out with adequate amount of attention. Overall, this work has provided us a great learning opportunity in the area of network performance using Cisco's VIRL tool.

1. INTRODUCTION

An organizational network traffic, particularly the TCP and UDP, should be reviewed and analyzed to satisfy and improve the performance. It also helps to improve Quality of service (QoS) of real time applications as they share a bottleneck node and/or link. This chapter deals with the basic concept of network traffic analysis and provides a brief layout of the thesis.

1.1 TCP/IP Suite

It is a conceptual model [9]. It consists of set of protocols that are implemented on the Internet and other similar computer networks. It defines how the data should be "packetized, addressed, transmitted, routed and received" [9] during the communication between a source and a destination. Internet Engineering Task Force (IETF [9]) maintains the technical standards of Internet protocol suite and many of its constituent protocols. It is developed prior to the Open System Interconnection (OSI) model. A typical model of the TCP/IP suite is shown in Figure 1.1. It consists of following layers.

a. Application Layer: It allows an access to the network resources and uses transport layer protocol to handle the data.
b. Transport Layer: This layer reliably delivers the message from process-to-process and also helps to recover the error. In other words, it is the backbone to dataflow between two hosts.
c. Network Layer: It helps to move packets from source to destination. The movement generally means routing of data over the network.
d. Link Layer: It is responsible for the hop-to-hop delivery of the data packet. Both NIC (Network Interface Card) and device drivers help to manage the communication.

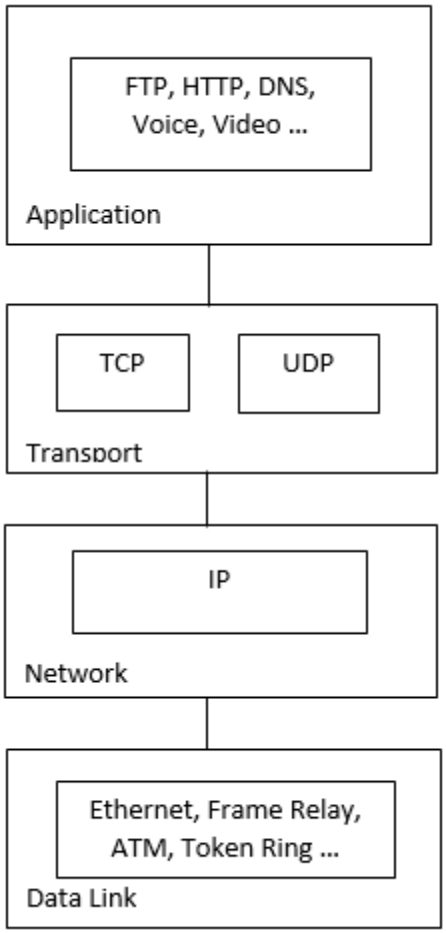


Figure 1.1: TCP/IP protocol suite [22]

1.1.1 Transmission Control Protocol (TCP)

It is a connection oriented protocol [28]. A connection is not broken until an application program has not completed messages exchange at each end. To avoid resource hogging, TCP carries the larger data as smaller packets and maintains the integrity of data at the destination node. Virtual ports are used not only to create virtual end-to-end connection but also to reuse the physical connection between two computers. Data field of Internet Protocol (IP) encapsulates TCP, which, in turn, encapsulates the higher level protocol such as Simple Mail Transfer Protocol (SMTP –

Email), Domain Name Server (DNS), Hyper Text Transfer Protocol (HTTP -web), and other protocols. A TCP header format is shown in Figure 1.2.

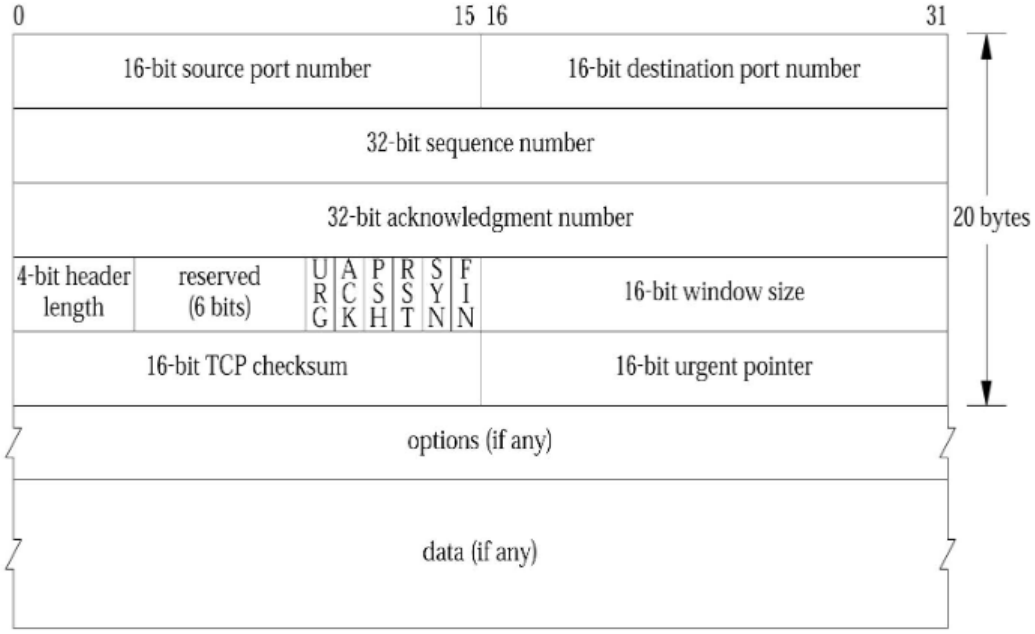


Figure 1.2: TCP header format [22]

1.1.2 User Datagram Protocol (UDP)

The UDP does not provide a reliable service. It also does not guarantee the delivery and protection of packets. It does not establish an end-to-end connection between the communicating end systems. Applications that run under the UDP send datagram at link rate of the interface. Thus, they need to be designed effectively in the sense that they should not contribute to the congestion by pumping data greater than the capacity of bottleneck node and/or link.

In addition, the UDP does not provide any security over point-to-point and end-to-end communication. The application and lower layers are responsible for the it. In a typical case, it is

achieved by using an additional protocol mechanism in order to protect the communication against message forgery, tampering or eavesdropping [23]. The UDP header format is shown in Figure 1.3.

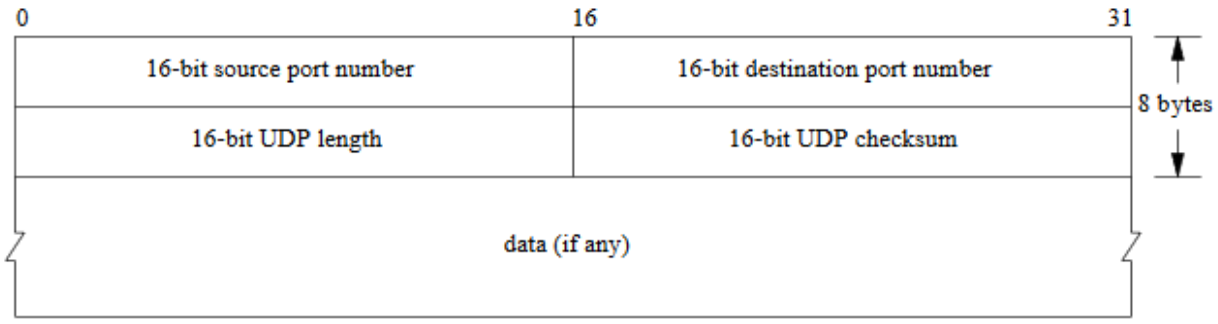


Figure 1.3: UDP header format [23]

1.2 Motivation and Thesis Goal

Many real time applications such as audio, video, web browsing etc. are available for practical and experimental uses over the Internet. It is increasing the network activity both in terms of the TCP/UDP traffic and the number of real time applications. Unfortunately, the mechanism that guarantees the Quality of Service (QoS) for a real time application has not yet been developed yet [1].

Today, all applications that we use are carried either in the form of TCP traffic or UDP traffic. A study has shown that more than 80 percent of the Internet's bandwidth is consumed by the TCP based applications such as FTP, HTTP etc. [1]. Although TCP and UDP based applications are increasing rapidly, there is no such mechanism which can guarantee the QoS of applications. Therefore, an application need to tolerate some degradation of QoS in terms of jitter, throughput and packet-loss for the data that are transmitted over LAN or WAN network [1].

UDP is a simpler transport layer protocol in comparison with TCP as it does not require a prior connection set up, retransmission, and flow control mechanism. Thus, it can be tooled to suite the retransmission and flow control schemes of an application. It also performs multicast communication, which is essential for teleconferencing [1].

When the bandwidth of an Internet is shared by the UDP and TCP protocols, the presence of one traffic affects the performance of the other. It is even more severe, when the connections share a bottleneck link/node. Not only the UDP packet-loss and jitter but also the TCP throughput and packet-loss are greatly affected by the presence of other TCP traffic and its flow control mechanism. Note, under TCP flow control, packets are retransmitted whenever packets are lost due to the congestion of network that further affect the QoS parameters.

The productivity of a company declines if the network is down even for a small period of time. Further, the essential services from public service division will be compromised. In order to avoid security breaches within the network, a network administrator needs to monitor the performance and traffic movement frequently throughout the network.

This thesis is inspired by such network traffic issues and is a step in the direction of understanding the network traffic. Essentially, a constant striving also becomes a key motivation factor in order to maintain the smooth operation within a network.

The main goal of this thesis is to investigate possible ways in order to improve the quality of service of the TCP and UDP traffic even in the presence of other heavy TCP traffic. In particular, this thesis studies the effect of the presence of TCP traffic on the jitter and packet-loss in UDP. In addition, it provides some idea on how does the TCP traffic affect the throughput and packet-loss ratio of any other TCP traffic while all are sharing a bottleneck network resource. Finally, this

thesis focuses on the study of some real time TCP and UDP traffic in terms of degradation in the quality of services like throughput, packet-loss, and jitter. In order to achieve our goal, we have used the Cisco's Virtual Internet Routing Lab (VIRL) platform. This provides virtual network devices (routers, switches), operating system, traffic generators and other relevant tools like Iperf to help experiment of our ideas. In addition, Wireshark [24] is used to capture packets which is analyzed to determine the packet loss ratio and verify the throughput of the network.

1.3 Layout of the Thesis

The layout of the thesis is as follows: Chapter 2 reviews the concept of quality of service (QoS) in detail. Chapter 3 deals with the Cisco's latest network simulation platform VIRL. Chapter 4 discusses some related work on the implementation topic. Chapter 5 starts with building a topology and provides various results that are obtained during the research. Chapter 6 presents a discussion about the importance of our work and narrates challenges that we faced during the experimentation. It also concludes the work with a discussion on the possible enhancements in future.

Appendices are included to supplement the thesis work. For example, Appendix A provides the description of tools that we have used in this thesis. It also consists of Iperf use and its short guide. A sample code that helps capture live packets is given too. Appendix B consists of various topologies that we have considered in our thesis work. Finally, Appendix C provides some additional graphs in support of our results.

2. QUALITY OF SERVICE AND PERFORMANCE TOOL

In the client-server system an application running on a client exchanges information with that on a server. The data submitted by the application is send to the operating system in order to be carried out across the network traffic. For the network to handle such traffic without compromising the service needs of certain application is called Quality of Service (QoS) parameters [16]. The QoS must satisfy the customer network administrator and network applications. In many cases, network applications attempt to occupy resources form the network while the network administrator limits the resources used by the application. This chapter describes QoS parameters that affect the network. Towards the end, benefits and objective of this study are also explained.

2.1 QoS Parameters

The traffic coming out of applications is different. It needs to be handled differently in the network. Note that applications generate the traffic at different rates. The first requirement needs the network to carry out the traffic at the same rate at which it has been generated. Further, different applications are either more or less permissive to the traffic delays in the network if it happens within a limit. It implies certain applications can cope the loss to some degree while other applications cannot [15].

2.1.1 Bandwidth

The link bandwidth or bit rate is the ability of a network to carry “volume of data over a unit time” [4]. Alternatively, it is the rate at which network carries the traffic of an application. It also measures the throughput of the network.

2.1.2 Latency

The latency refers to the time which is needed to deliver a packet from a source device to the destination device [3]. It is also the delay that measures the time it takes to travel through the network including all the intermediate nodes between a source and a destination.

2.1.3 Jitter

For a stream of packet, jitter is defined as the "mean deviation of the difference in the packet spacing at the receiver" compared to that at the sender. For any pair of consecutive packets [17], the jitter is defined as:

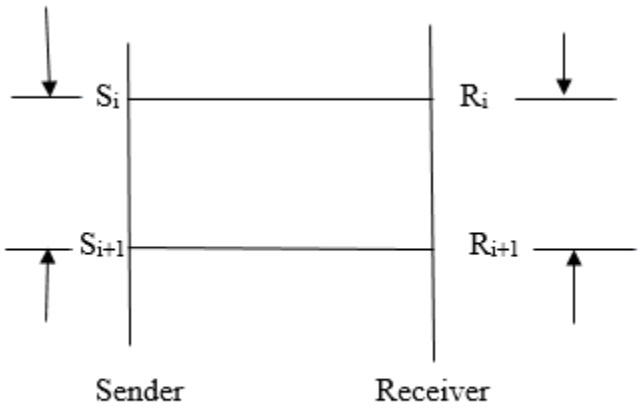


Figure 2.1: Jitter explained

$$J_i = | (R_{i+1} - R_i) - (S_{i+1} - S_i) |$$

$$= | (R_{i+1} - S_{i+1}) - (R_i - S_i) |$$

Where S_i is time at which a sender sends the packet and R_i is time at which the receiver receives it. Generally, it represents or measures the variation in latency of packets. Sometimes, it is also known as packet delay variation [4].

2.1.4 Packet Loss

The packet loss is the percentage of packets that arrive with errors or fail to reach the destination. The congestion of network at intermediate nodes or links that do not have sufficient bandwidth at different points in the network may cause it.

2.2 Resources Limitations

A host uses variety of network devices like routers, switches, hubs, and network adaptor. All are connected by the network. Each network device contains an interface which is interconnected to other device via fibers and/or cables. A combination of software and hardware helps to forward the traffic from one interface to another.

The QoS problem arises due to the limited resource availability. The bandwidth mismatch when the faster network traffic joins the slower network traffic can also create the problem. Further, the traffic passing through multiple switches and routers within the network incurs the processing delay [4]. The packets arriving on one port are buffered in the memory and are queued at specific outbound port of the device after analyzing the headers. When the receiver received the packets at much faster rate than they can be processed and sent, an overflow occurs in the queue and packets are, then, dropped. This situation is called congestion [4]. This section details the effect of resource limitation in the network.

2.2.1 Effects on Congestion

Two things contribute to the congestion within a switch or router: the "bandwidth mismatch and traffic aggregation". A bandwidth mismatch happens when the packet is "routing from a high speed network to a low speed network" [4]. For example, the packet arrives on a 5 Mbps link and it needs to be routed to a 1Mbps link. Here, the bandwidth of incoming link is five times greater than that of an outgoing link. There is insufficient capacity to hold the traffic. In this case, the router/switch will simply put the outgoing packets in a queue but packets will eventually be dropped when there is queue overflow.

The packet aggregation occurs when several incoming connections send their packets to a single outbound link. For example, five separate 1 Mbps links are carrying streams of packets and they all are sending it to a single 1 Mbps link. It causes to fill up the buffer of the router/switch very quickly, which, in turn, results in the packet loss.

Finally, an unintended traffic on the network also causes the congestion. Such traffic could come from a malfunction hardware, retransmission of packet, or traffic generation from some malicious application [4].

2.2.2 Effects on Latency

During the entire flow of a packet, the latency is accumulated. The scheduling of process causes latency at the host, which is followed by packetization of data that it generates. Packetization refers to the computation that is necessary to create the packet and move them via different layers of the network. This includes assigning buffers to sockets, generating header of UPD (or TCP). Since only one packet can send at a time, there comes the delay due to serialization.

The packets are lined up in a buffer behind the packet which is currently being in transmission, assuming multiple packets are ready to go out.

Whenever the packet hits a switch or router, it acquires some processing delay at the nodes because each packet has to move from input queue to output queue. During this period, packets are inspected in order to determine a route. Packets also encounter queuing delay at different nodes (routers or switches) where it needs to wait until outgoing port is available as it may be in use by other packets.

2.2.3 Effects on Jitter

On any single link, routers (switches or computers) "can send only one packet out at a time" [4]. Because of this, packets do not arrive at a constant rate. Further, packets have to be queued in the buffer if multiple packets need to be sent. Sometimes packets of applications suffer short delay or long delay. Short delay means the packet will be in first of queue and long delay means packets end up last in queue. If other packets result in different delay, then there is variation in the latency, which is defined as a jitter (see section 2.1.3). This process is repeated on each router and switch.

Layer 3 protocol (or IP layer) dynamically reroutes the stream of packets. It means one packet takes different path as another. Since two routers cannot give same amount of end-to-end delay, the jitter takes place. Moreover, the transport layer protocol (TCP/IP) contributes to the jitter in two ways. First, an out of turn arriving packet is a hold-back packet, which creates an end-to-end latency and increases the jitter [4]. Next, the sender retransmits a packet if the initial packet gets lost or corrupted, especially in the case of TCP (to ensure in-order and reliable packet delivery). Note, TCP uses retransmission timer to detect missing acknowledgement from the

receiver. This introduces additional delay, which is far greater than any delay and, in turn, increases the variance in the delivery time of a packet.

2.2.4 Effects on Packet Loss

The queue overflow (buffer overrun) is one of the main reasons for the packet loss. It occurs in switches and routers. New packets end up being discarded when there is no memory and the queue is full. In addition, poor transmission medium is also a factor for the packet loss. Factors like signal interference, degradation, faulty hardware cause packet loss. In case of Wireless Networks, the collision and interference are main factors for the packet loss.

2.3 QoS Aware Networks

In QoS aware networks, applications can specify the requirements for their quality like delay, bandwidth, loss, and jitter. In order to meet the demands, the network reserves enough capacity at all switching components between a source-destination pair. An application gets permission to send the data from the network only if it has demanded resources. This is called the *admission control*. Following three modes are generally used to handle it [4].

(a) No QoS

It is the default behavior of any IP router. It is also called *best-effort* network policy [4]. Here routers do their best to handle the packets. No preferential treatment is given to any packet. Moreover, the host does not fix the QoS needs of an application. Thus, desired QoS is achieved only by ensuring that the network has sufficient bandwidth and buffer resources.

(b) Soft QoS

It is also called *differentiated* QoS [4]. Here, the network does not perform *admission control*. The quality control data stream does not have to be setup explicitly. However, there exists a preferential treatment among the packets on the basis of the flow of related packets. Each packet consists of embedded QoS information and the router prioritizes one type of packet over another on the basis of rules that are configured in routers.

(c) Hard QoS

It guarantees a specified QoS requirement of any application. It requires *admission control* policy so that each component from a source to the destination will have enough resources reserved to move the traffic for each flow. In both UDP/IP and TCP/IP, a flow is a set of packets that travels from one address and port to another address and port with same protocol (eg. UDP or TCP) [4].

The problem is not having unlimited resources rather inefficiency or inability to allocate network resources appropriately to deliver a specified QoS. The network resources can be allocated using the following two approaches [4].

(i) Router-based Approach: Using this approach, each router will decide how to prioritize its traffic from a source to the destination.

(ii) Host-based Approach: Using this approach, each host will adjust its behavior appropriately like slowing down the packet transmission rate on the basis of prevailing conditions in the network.

Similarly, the data communication between source and destination is also achieved using following concepts [4].

(i) Reservation-based Approach: It enforces the *admission control* policy. Thus, each host will request particular grade of service like bandwidth, jitter, packet loss, and delay. The request of the host is, then, forwarded to all components from the source to destination. If any one of component is unable to grant specified services, the reservation is denied. In that particular case, a host will either give up or try again, or it may request network for lesser grade of service than previous.

(ii) Feedback-based Approach: Here, data is sent without any prior reservation of resources in the network. However, if network is congested, router will send feedback or the host will detect increased packet loss. In that case, it will implicitly adjust its packet transmission rate.

2.4 QoS in IP

Even though, the underlying network offers QoS control, Internet Protocol (IP) cannot take its benefit. Note that IP is initially designed to work with any packet switched network rather than with QoS control. There are four core issues, which affect the QoS in IP networks [4].

(i) Bandwidth mismatch and aggregation: The packet congestion will occur whenever there is a traffic flow from higher bandwidth link to lower bandwidth link or if there is a traffic aggregation from multiple links.

(ii) Inefficient packet transmission: In order to send 1 byte of packet (single character), 58 additional bytes need to be transmitted in the form of overheads of different layers (20 bytes of TCP header, 20 bytes of IP header, 18 bytes of MAC header) [4]. Even though it is not QoS issue, it affects the (a) bandwidth of network, (b) serialization of packet at host, (c) scheduling and (d) packet queuing on routers. Thus, bigger packets offer efficiency in comparison with smaller ones while transmitting.

(iii) Unreliable delivery: IP is intrinsically an unreliable datagram delivery system. It cannot guarantee packet will reach its specific destination. The TCP was created to give software based reliability to the packet [4]. If the sender does not receive acknowledgment from the receiver within a specified time, the packet is considered to be lost. Such packets are then retransmitted. Therefore, the TCP gives reliability, however, it increases jitter.

(iv) Unpredictable packet delivery: Internet Protocol has no control over bandwidth, jitter, and delay. In addition, when packets are sent to their destination, it may take different routes, which results in change of service levels.

2.5 Iperf

The performance measurement of the network in terms of bandwidth and speed is norm in both non-productive and productive environments. In order to deploy the network dependent application servers, a detail analysis report of bandwidth and speed is very important. Sometimes, throughput of the network should be double checked while troubleshooting. For this, we need a very reliable network performance tool and one such tool is “Iperf” [10].

Iperf is an open source tool which is used to test the performance of the network [10]. The test result provided by iperf is more reliable compared to that from other online tools. It is even more reliable when we measure the performance between client-server located at different places. A sample of client-server model is shown in Figure 2.2.

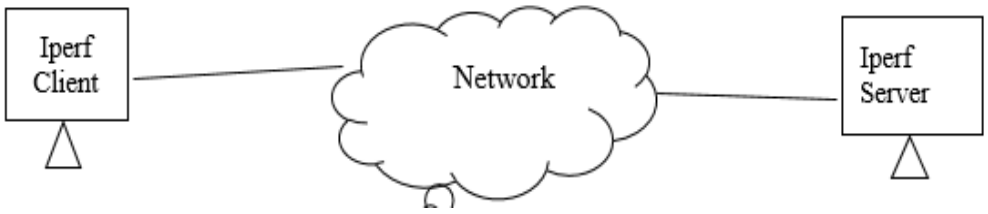


Figure 2.2: A simple client-server model

Basic Features of Iperf [10]: Iperf, generally, runs on two different computers where one behave as *client* and the other behave as *server*. It is a "command line program" [21] and it takes multiple options, which makes it suitable for different purposes. Some basic features of Iperf are as follows:

- It measures packet loss, delay, jitter etc.
- It measures the bandwidth of the network.
- It reports on MTU (Maximum Transmission Unit) and MSS (Maximum Segment Size).
- It supports for TCP window size
- It supports multithreading, which is useful for multiple simultaneous connections.
- It can generate particular UDP bandwidth streams.

2.6 Objective and Benefits

In broader perspective, the network traffic analysis helps to find (i) how different parameters of QoS affect the overall performance of any network and (ii) how to handle those parameters to have enough quality of service for different applications. One of the main objective behind a successful traffic analysis includes identification of behavior of TCP and UDP traffic in

LAN and WAN. Another objective is to find ways to enhance the performance of real time traffic while they are sharing the bottleneck node. Besides these objectives, benefits of the analysis include planning for extra links so that congestion in links and nodes can be avoided. Further, a detailed traffic report helps in shaping future network for better performance in many bottleneck situations.

The network traffic analysis on a whole is a creative task upholding the QoS of a network. In this thesis, we have used a network simulator (VIRL) to test TCP and UDP traffic including many real time applications and then have observed their impact on the network. With supportive environment and quality objectives, the network traffic analysis will surely be the highest level of network QoS assessment.

3. VIRTUAL INTERNET ROUTING LAB (VIRL)

The Virtual Internet Routing Lab [7] is a powerful network simulation platform of Cisco, which helps to develop the high fidelity models of planned or real network. It contains virtualized version of Cisco network operating system and helps to integrate with real external or physical networks, network servers, and other related elements. The VIRL runs virtual machines of Cisco's physical switches and routers. In addition, it has a powerful Graphical User Interface (GUI) to control the simulation and to design desired network.

3.1 VM-Maestro

VM-Maestro [7] is the client side application, which helps to build the topologies, manage simulations, and generate configuration and visualization that is executed on VIRL host or virtual machine. It consists of topology editor plane to draw topologies using tools and objects found in the palette pane. It also has properties pane to manipulate the various option associated with objects in topologies or the topology itself. Finally, its project pane creates, manages, and deletes the projects and topologies [refer to Figure 3.1].

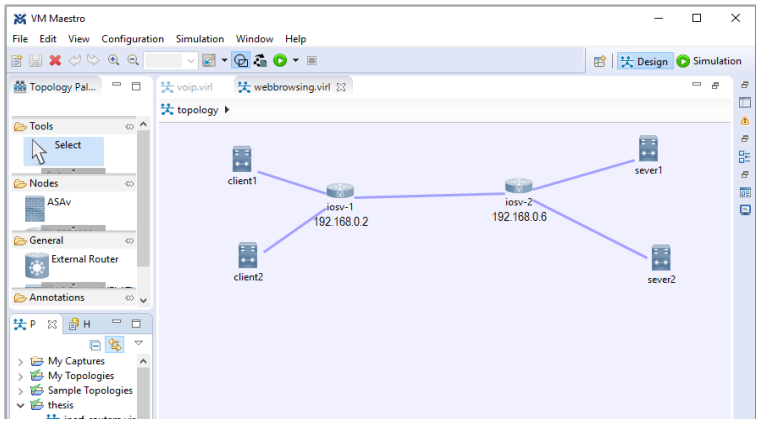


Figure 3.1: VM-Maestro [7]

VM-Maestro consists of two perspectives or modes that define the layout of various panes. They are [7]:

a. Design Perspective: It is an organization of the pane, which is optimized to design topologies using palette and properties pane. It is a default perspective.

b. Simulation Perspective: It is also an organization of the pane, which is optimized to run simulation including space for router console.

3.2 Management Access

VM-Maestro helps to connect console ports of VIRL nodes that are helpful to configure, test, and troubleshoot a network. However, in order to use management platform (i) to configure, (ii) to send troubleshoot information to log servers, (iii) to download and apply updates, and (iv) to use controllers or network application with network Application Program Interface (API), we need Internet Protocol (IP) connectivity to management interface of nodes [7]. VIRL provides three mechanisms for such connectivity and they are (1) private simulation network, (2) private project network, and (3) shared flat network.

3.2.1 Private Simulation Network

By default, it creates a single 10.255.0.0/16 subnet for each simulation. The Linux Container (LXC) has connectivity to only those nodes, which are running within that single simulation. Figure 3.2 shows the IP connectivity in a private simulation network.

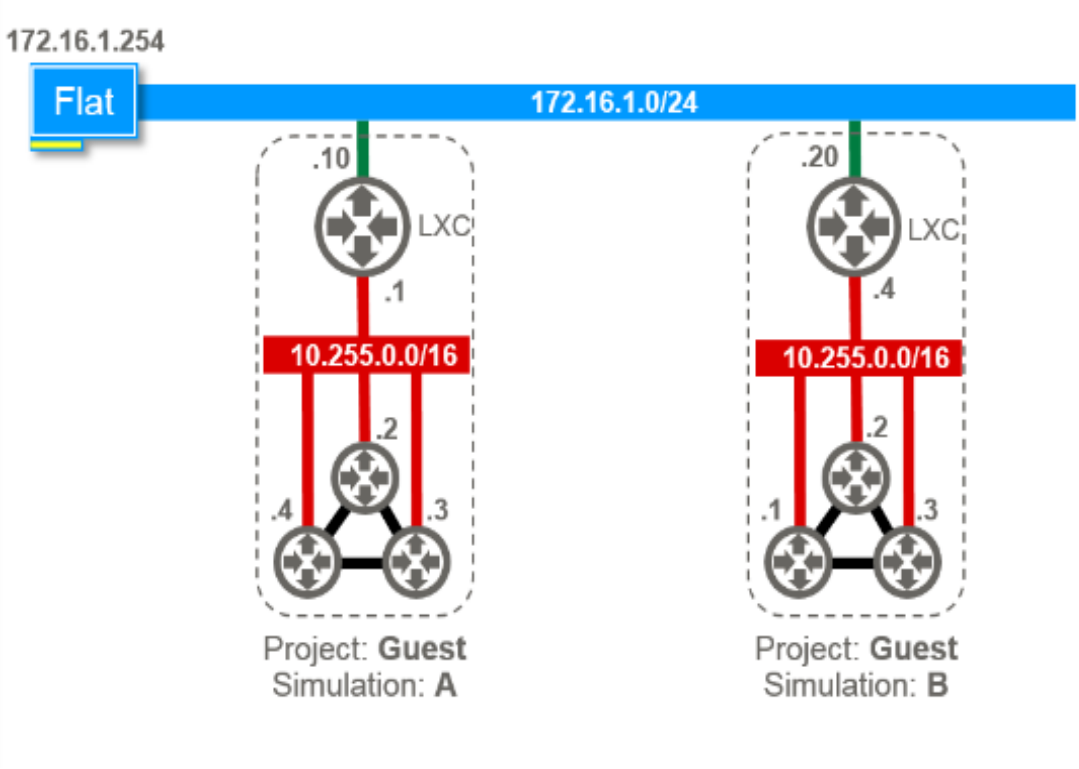


Figure 3.2: Private simulation network [7]

Simulation A and Simulation B under the project "Guest" are two separate private simulation networks. The dotted lines depict the scope of nodes that is visible to LXC. In each private simulation network, one interface of LXC is connected to Flat network of 172.16.1.0/24 and another interface is connected to the management interface (10.255.0.0/16) of nodes. The LXC of simulation A cannot see LXC of simulation B. Thus, LXC of simulation A cannot access nodes which are running in simulation B even though they are running under same project "Guest".

3.2.2 Private Project Network

Similar to the private simulation, a single 10.255.0.0/16 subnet is created for each project. Regardless of which user owns simulation, the LXC has connectivity to all nodes which are running within that project. Figure 3.3 illustrate the concept of private project networking.

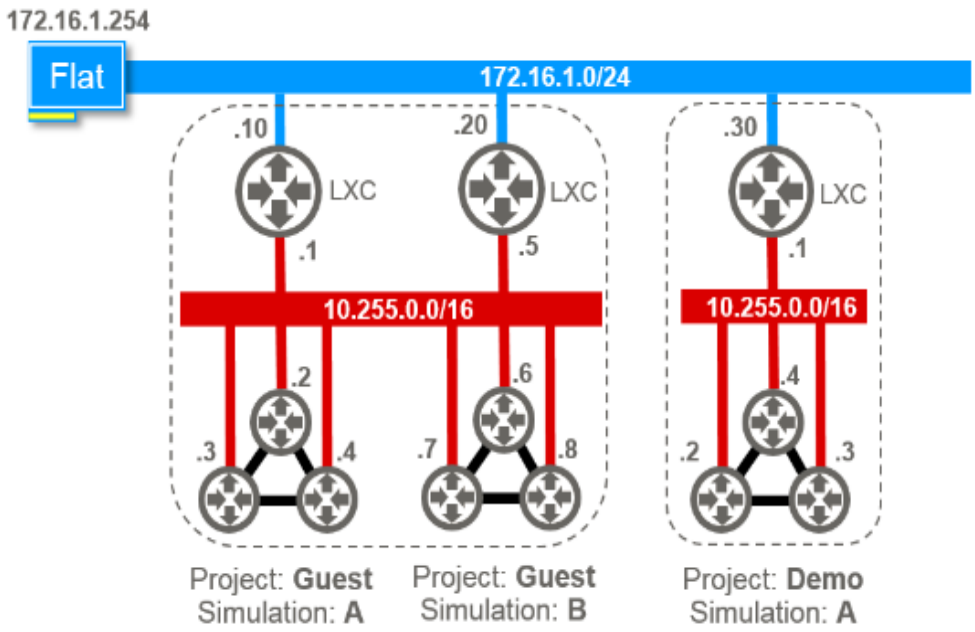


Figure 3.3: Private project network [7]

Simulation A and Simulation B for the project "Guest" are under the same private project network, whereas the Simulation B under the project "Demo" is another private project network. The dotted lines detail the scope of nodes that are visible to LXC. similar to the private simulation network, the one interface of LXC is connected to Flat network 172.16.1.0/24 and another interface is connected to management interface (10.255.0.0/16) of nodes. Here, the LXC of the project "Guest" cannot see and, thus, cannot use the nodes of the project "Demo" or the nodes that are part of private simulation network.

3.2.3 Shared Flat Network

It is separate from the above two private networking in following ways:

a. The management interfaces nodes are directly placed on the flat network 172.16.1.0/24 by default.

b. No LXC is used or needed to access management interfaces because a subnet 10.255.0.0/16 is not created.

c. Regardless of the project or user, nodes have visibility to all other nodes in the simulation and also to all LXCs associated with other simulations.

d. On flat network, nodes have direct connectivity to all devices.

Figure 3.4 shows a sample of shared flat networking.

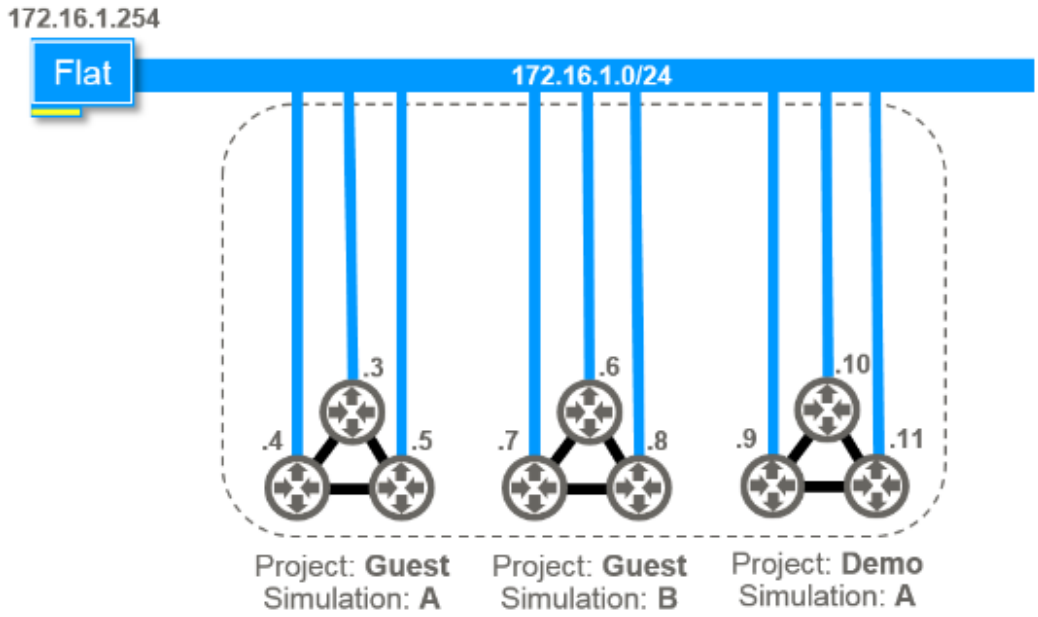


Figure 3.4: Shared flat network [7]

All the nodes of different simulations (A and B) under different projects (Guest and Demo) are directly connected to the Flat network 172.16.1.0/24. The dotted line depicts the visibility of each node. Thus, each node in simulation A of project "Guest" is visible to every other node in other simulations (B and A) of projects "Guest" and "Demo", respectively. So, each node is accessible to every other node regardless of any simulations and projects.

3.3 Conclusion

This brief discussion of Virtual Internet Routing Lab (VIRL) provides its importance. One of the reason for choosing VIRL is availability of its automatic configuration of nodes, which is done by using the feature called “AutoNetKit”. In addition to that, VIRL comes with a complete set of legal and licensed Cisco images with new OS release provided in regular basis [7]. Furthermore, it is a powerful and portable tool in comparison with other simulation platforms available in the market (e.g., Graphical Network Simulation [30], Cisco Packet Tracer [31] etc.). In addition, it does not require bulkier equipment and hours of cable wiring. By linking to additional physical devices, the lab can easily be extended to study medium to large size networks.

4. IMPLEMENTATION

With an aim to study the real LAN and WAN network performance, we have used Cisco's new simulation platform called VIRL. As described in Chapter 3, the VIRL comes up with virtual images of different networking components like routers, switches, Iperf IoS, and others. It is very easy to build topology and generate the configuration file automatically in VIRL using a feature called "AutoNetKit". This Chapter answers: (i) how to create a topology, (ii) how to connect VIRL with an external terminal, (iii) how to capture the traffic offline and online using Wireshark [24], (iv) how to tune TCP and UDP for a desired performance, and (v) what is the importance of this study. In short, we discuss nuts and bolts of our work.

4.1 Creating a Topology

We create a simple topology in VIRL using VM-Maestro with the following steps:

Step 1. Select desired nodes 

from the palette pane and drop them down onto topology pane as shown in Figure 4.1.

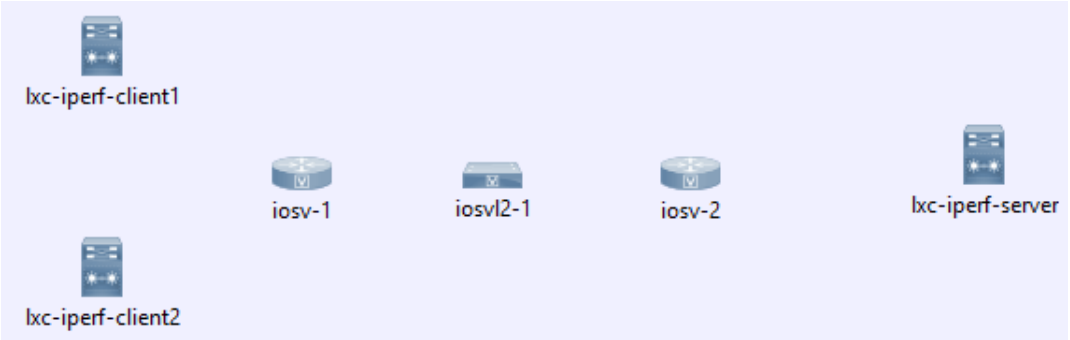


Figure 4.1: Typical nodes in a VIRL [7].

Step 2. Connect nodes using a tool  from the palette pane and change names of the component as desired (Figure 4.2).

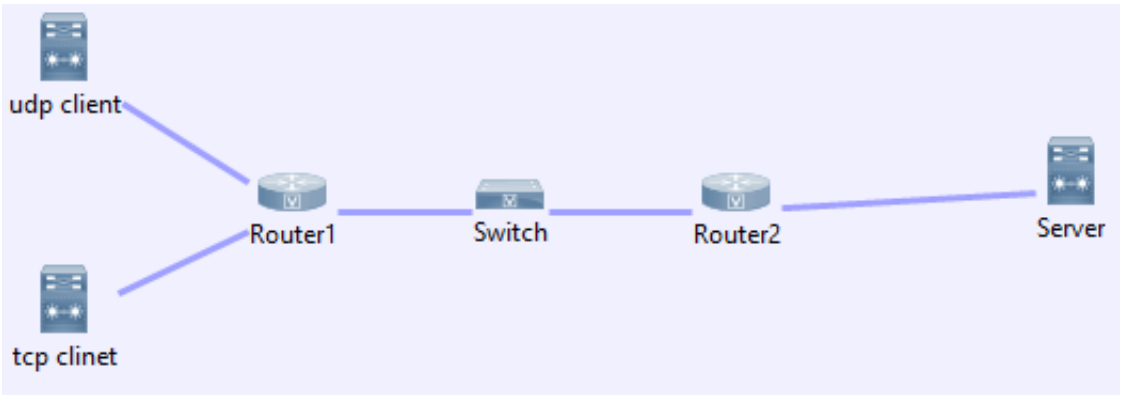
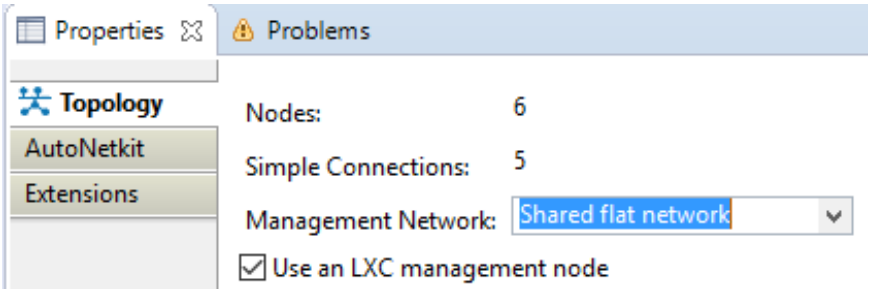
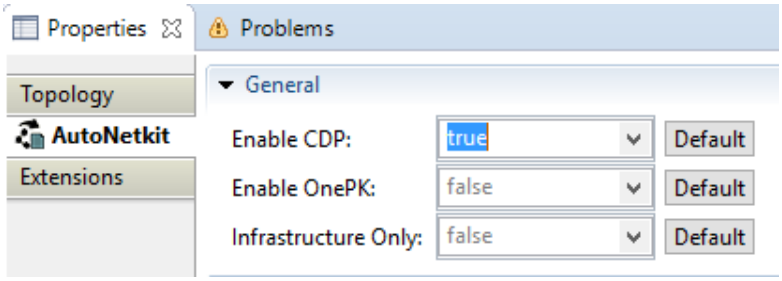


Figure 4.2: A simple topology.

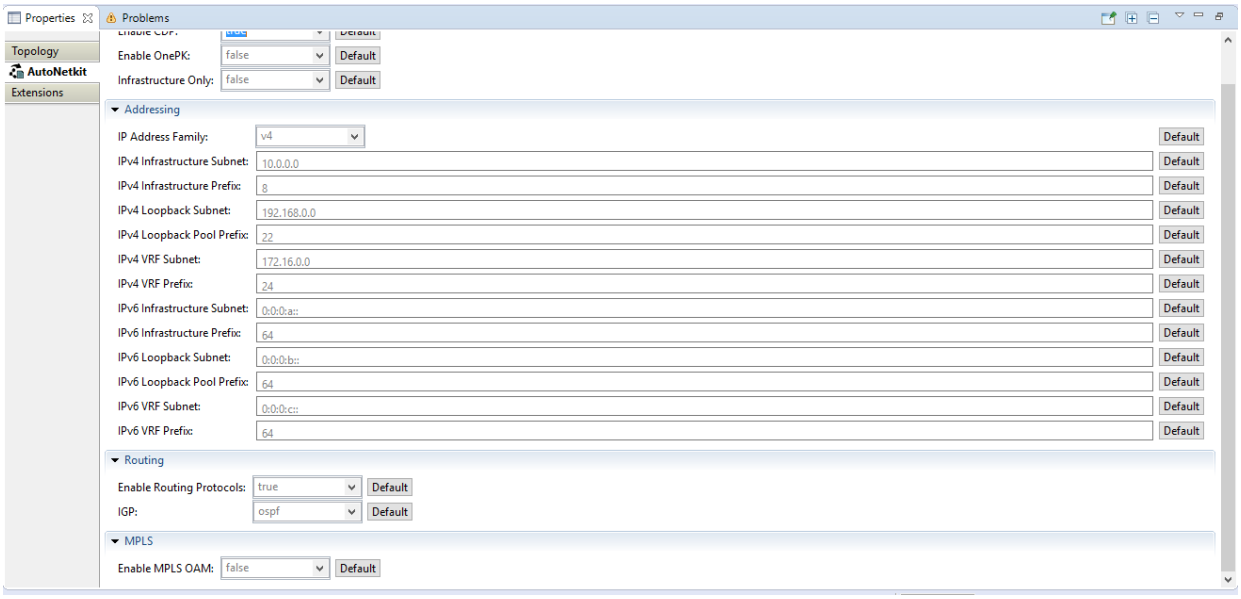
Step 3. Go to properties pane and choose ‘shared flat network’ as given below.



Step 4. From the AutoNetKit, enable Cisco Discovery Protocol (CDP), a link layer protocol developed by Cisco (see below).



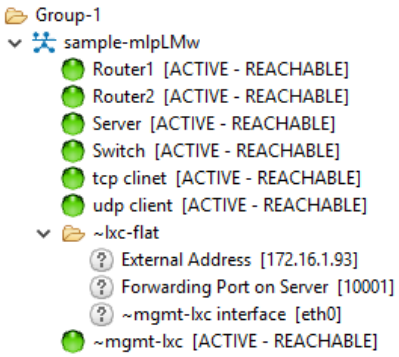
Step 5. Review various IP address properties and choose default values as below. We are using "OSPF" routing scheme.



Step 6. Select 'Build initial Configuration' tool  from the toolbar. After this, the configuration of routers and visualization of nodes can be viewed on clicking 'Yes' on the prompted nodes.

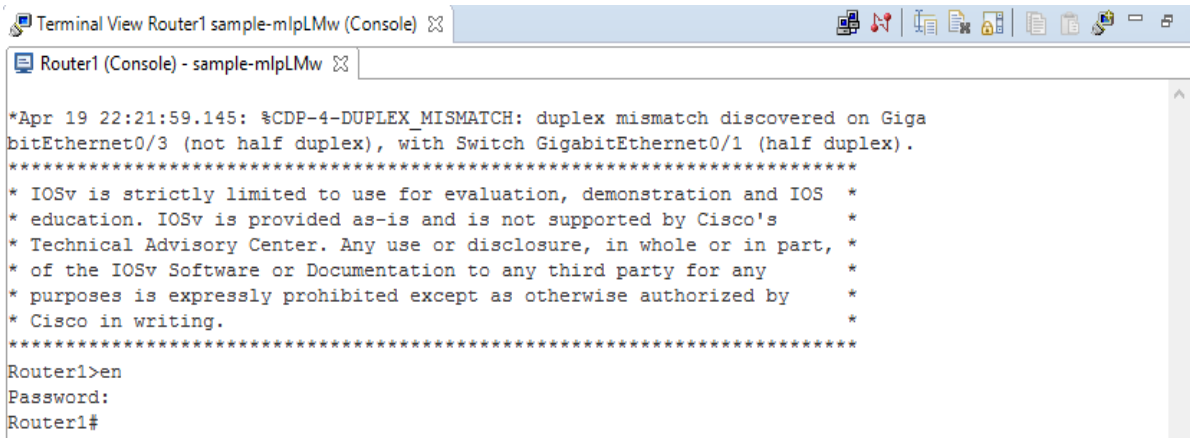
Step 7. Select 'Launch Simulation' tool  from the toolbar.

Step 8. VM Maestro will switch to a simulation perspective. Wait and watch until all nodes become active as illustrated below.



Step 9. Click on any node and select “telnet” and then from sub-menu select ‘to its console port’.

Step10. In console pane, press ‘Enter’ in order to get console prompt and enter the command ‘en’ and password ‘cisco’ to open a node in enable mode as shown below.



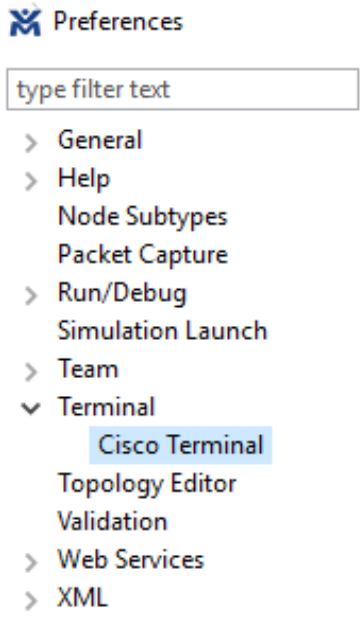
Using this approach, we are able to create a topology (Figure 4.2) that is analogues to real world network. All the screenshot of topologies that are necessary to conduct this research are displayed in Appendix B.

4.1.1 Connecting VIRL to external terminal ‘Putty’

‘Putty’ is a network file transfer application, which supports several network protocols including Secure Shell (SSH), Telnet, Secure Copy (SCP) etc. [29]. It is very inconvenient to work with inbuilt console pane of VIRL when we are running configuration of multiple nodes. In addition, switching form one node to another and observing the topology at the same time makes the work much more difficult. Thus, it will be fruitful to use an external terminal ‘Putty’ to connect and work with nodes that is running VIRL. In order to do that the following steps are required.

Step 1. Go to ‘file’ menu in toolbar and select ‘Preferences’

Step 2. From the preferences, select ‘Cisco Terminal’ under section Terminal (see below).



Step 3. Use external terminal option and fill the telnet command, telnet argument, SSH command and SSH argument as below (Figure 4.3).

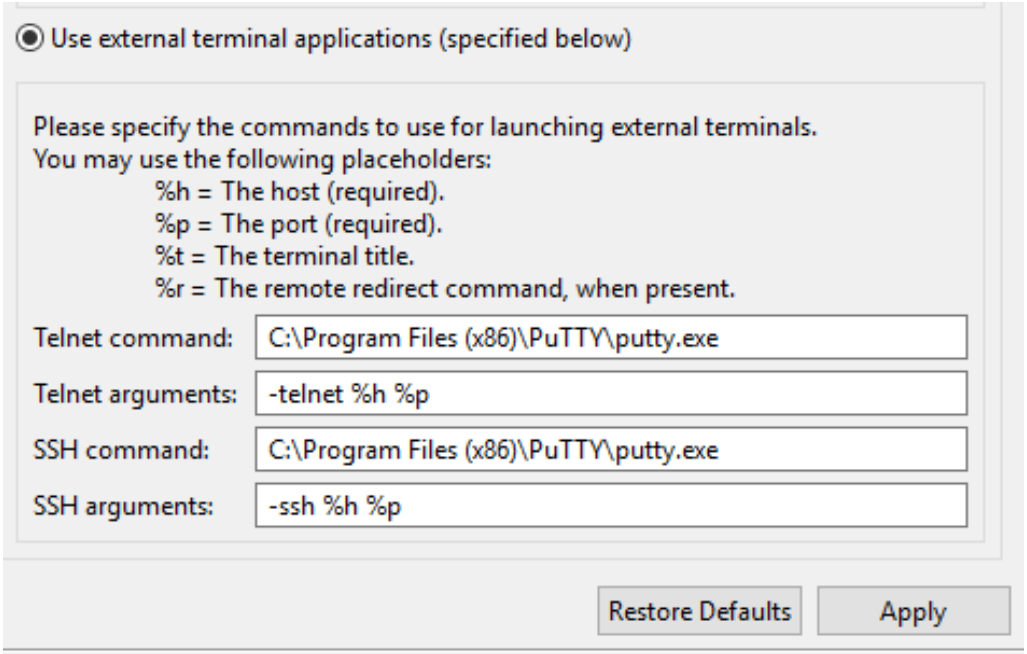


Figure 4.3: External terminal application

Step 4. Now select any active node, go to its ‘console port’. It will switch to external terminal ‘Putty’.

Step 5. Press ‘enter’ to get console prompt and enter ‘en’ and password ‘cisco’ as before to go to the enable mode of nodes.

Once the ‘Putty’ is connected to VIRL, each node’s terminal window can be opened separately in a larger size compared to the inbuilt window. The topology can also be viewed in parallel that helps to see how traffic is moving from a source to a destination. This makes easier to generate the traffic from a client to and see its effect on a server simultaneously.

4.2 Packet Capture with Wireshark

In order to do the traffic analysis and analyze the performance of the network, it is essential to capture the packet that is traversing from a source to a destination. We have used Wireshark [24] for this job. It offers two modes of packet capturing.

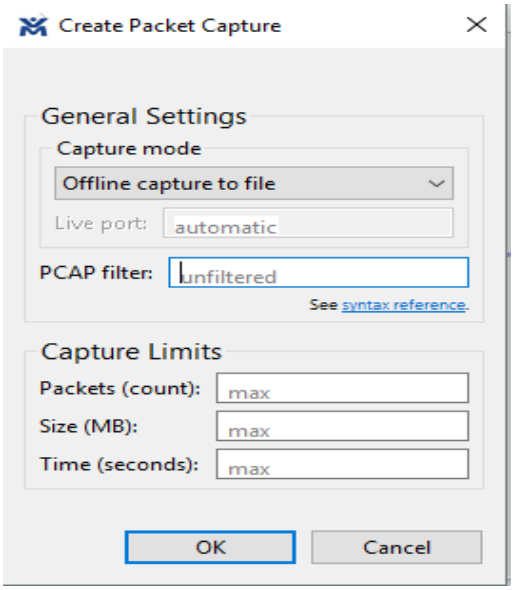
(a) Offline Packet Capture

This option allows user to download the capture file and save it locally for the analysis. Here, VIRL server collects the data and stores in a file. Files can be saved by using PCAP filter and can even be left blank, which can be opened later directly by using external packet sniffer tool ‘Wireshark’ [24].

Following steps help capture the packet offline.

Step a. Right click on destination Ethernet port from link, select ‘packet capture’, and click ‘Create New.’ It will open ‘Create Packet Capture’ prompt.

Step b. Select ‘Offline capture to file’ from capture mode and leave rest or default (see below).



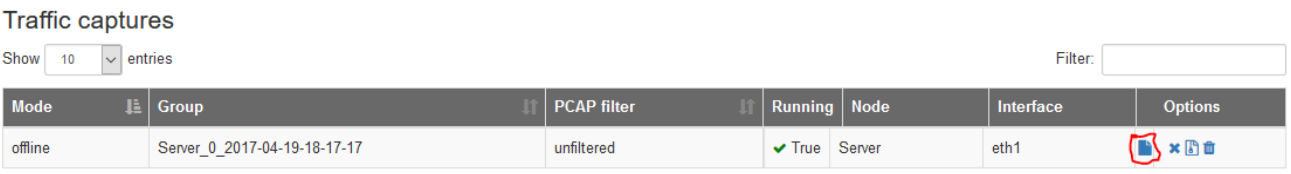
Now, when traffic is sent form a source to a destination, Wireshark will capture and store it in a file. The stored file can be download. Save it locally from VIRL server. The steps involved in this process are as follows:

Step 1. Enter the IP address of VIRL server.

Step 2. Go to ‘User Workspace Management’, and enter “username” and “password”.

Step 3. Click name of topology that is running in VIRL server.

Step 4. Go to the section ‘Traffic Capture’ and click on download icon circled by red color as indicated below.

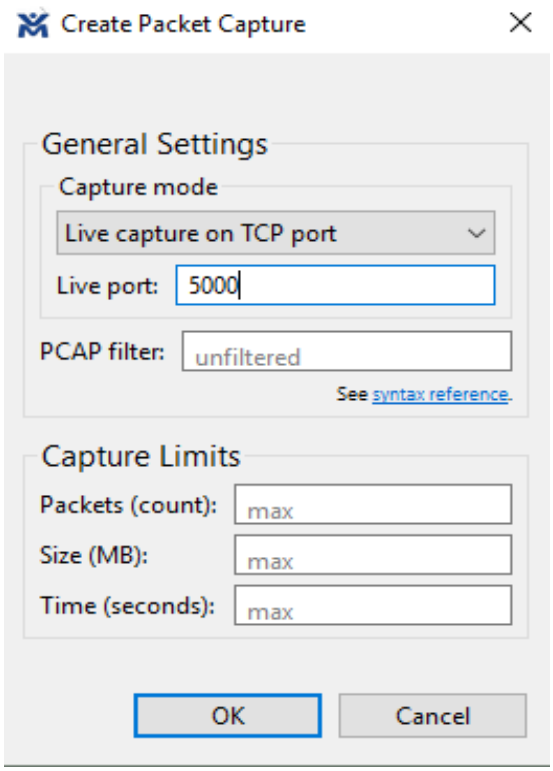


(b) Live Packet Capture

This option allows the user to connect Wireshark directly to the listening port of VIRL server. This displays flow of packets as it happens. The necessary script file for the 'live packet capture' is in Appendix A. Steps involved in this case is quite similar to that of offline packet capture. They are:

Step 1. Right click on the destination Ethernet port from link, select 'packet capture' and click 'Create New.' It will open 'Create Packet Capture' prompt.

Step 2. Select 'Live capture on TCP port' from capture mode and enter live port in between 1025 to 65535. Leave rest as default (see below).



Step 3. Run the executable script file ‘live_pcap_gui’. It will prompt a window in which we enter the IP address of VIRL server and live port (eg. 5000). This will directly connect Wireshark to VIRL server listening port (eg. 5000). Figure 4.4 illustrates the concept.

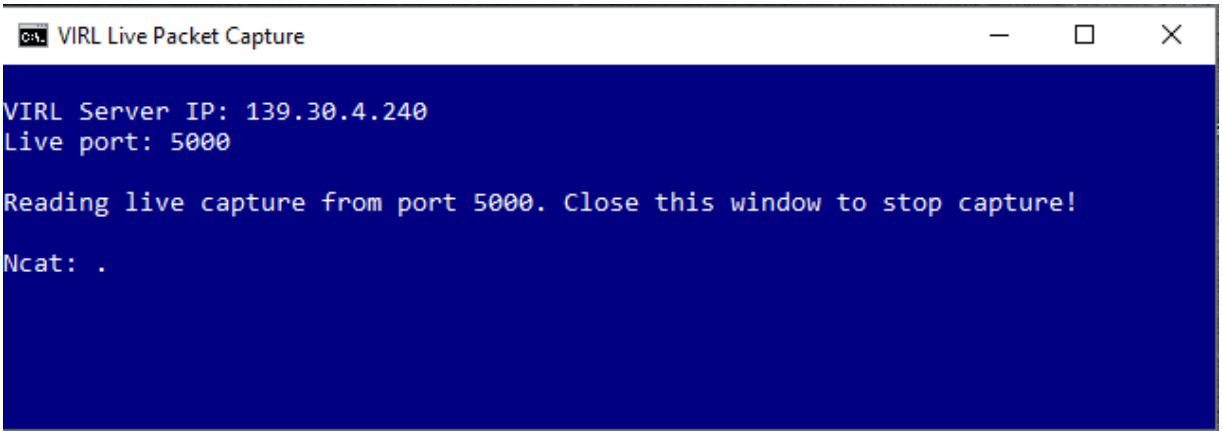


Figure 4.4: Live packet capture

Live packet capture is very useful because we can see the live flow of packets from a source to a destination. Further, we can use pass filters to capture specific packets (TCP, UDP) according to our needs. The captured packet is saved and analyzed for various purposes.

4.3 TCP Tuning

Iperf [14] helps to tune TCP connections over a particular path. The window size is the most fundamental tuning element for TCP. It is the maximum amount of data a sender can send to the other end without an acknowledgement [10]. If there is no packet loss, the size of window can limit throughput as $Throughput \leq Rwin \,/\, RTT$ [21]. Here $Rwin$ is TCP window size and RTT denotes the round trip time for the path. Many hosts and Operating Systems (OSes) have their upper limit on the size of TCP window. However, if there is a packet loss, it will further impose limit on the throughput because when there is loss, TCP rate is limited by the congestion avoidance algorithm. In this case the throughput becomes [2].

$$Throughput \leq MSS * C / RTT \sqrt{P} \text{ ……..…..…..(1)}$$

Here MSS is the maximum segment size, C is constant, and P (Pi) is the packet-loss ratio, which is the ratio of number of retransmitted packets to the total transmitted packets.

Another tuning issue for the TCP is with parallel TCP streams. For 'n' TCP connections and for fixed C, the equation (2) becomes,

$$BWagg \leq C \left[\frac{MSS1}{RTT1\sqrt{P1}} + \frac{MSS2}{RTT2\sqrt{P2}} + \cdots + \frac{MSSn}{RTTn\sqrt{Pn}}\right] \text{ ……..(2)}$$

Here $BWagg$ is the aggregate throughput. Note, RTT will be same across all the TCP connections and MSS remains identical and constant across all the TCP connections between hosts. After simplifying the equation (2), we get:

$$BWagg \leq \left[C * \frac{MSS}{RTT}\right] \left[\frac{1}{\sqrt{P1}} + \frac{1}{\sqrt{P2}} + \cdots + \frac{1}{\sqrt{Pn}}\right] \text{ …….....(3)}$$

It is, thus, clear that MSS and RTT are relatively static compared to the dynamic nature of packet-loss ratio (P). When there is more parallel TCP connection, P becomes dominant factor, which will directly affect the throughput. Basic command for tuning the TCP client and server side is given using "Iperf" as follows (see Figures 4.5 and 4.6).

$*iperf –c server_ip -p server_port*

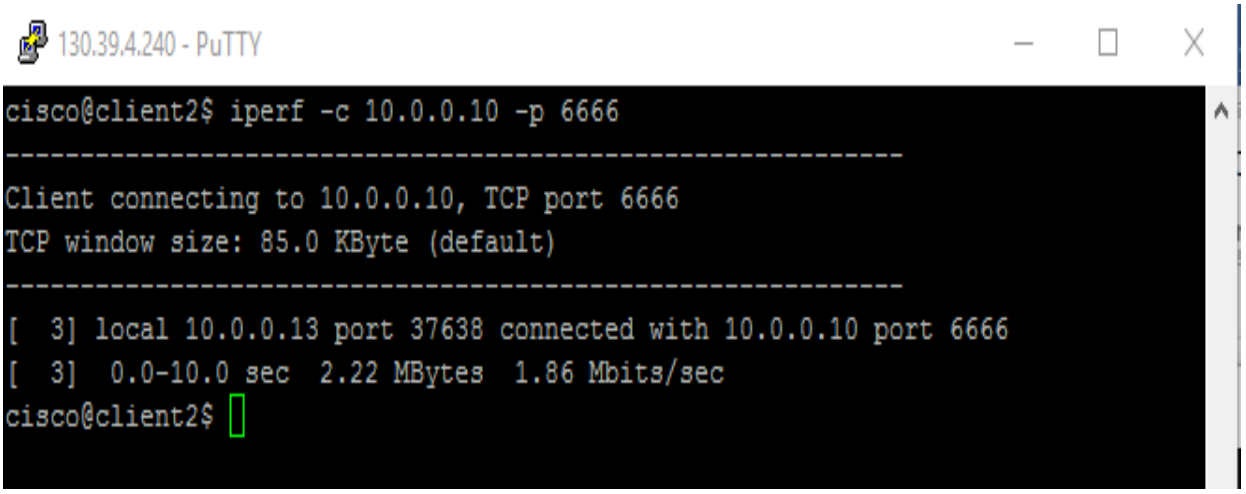


Figure 4.5: TCP client

$*iperf –s -p server_port*

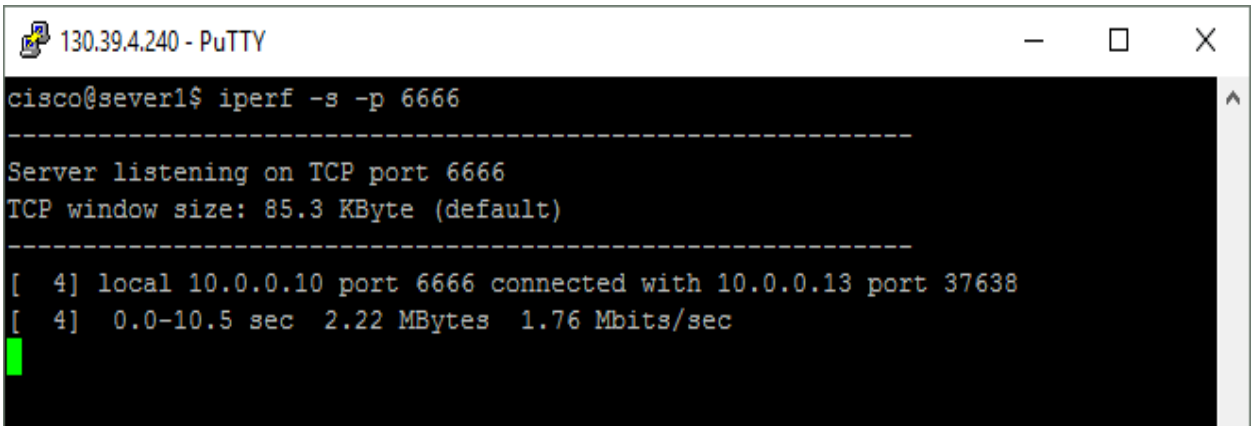


Figure 4.6: TCP server

The TCP server is run first to make it ready to accept the traffic from the TCP client. By default, the client sends traffic for 10 seconds with a window size of 85.3KB. The various parameters are tuned at the client to generate the different traffic data. After sending the traffic for

10 seconds, the client and server both showed the throughput in Mbits/sec, but we used the value obtained in the client side only.

4.4 UDP Tuning

When we use Iperf to conduct the UDP test, it provides some more information about the network which is very useful to find the bottleneck in the network. Not only the TCP window size but also network jitter and packet-loss affect the quality of service of any network. The UDP test is generally done by passing the –u argument in the command line that gives valuable information about the jitter and packet loss [14]. By default, Iperf uses TCP if we do not specify –u argument.

Moreover, various argument can be passed along with –u argument like –l and –b, which specify the size of datagram that can be sent over the network and speed at which datagram is sent, respectively. It is assumed that large packets are fruitful to send compared to the small ones because even if we need to send 1 byte of packet via UDP 46 additional bytes of different headers need to be sent (8 bytes of UDP header, 20 bytes of IP header, and 18 bytes of Ethernet MAC header).

Note, a smaller packet has more overhead in comparison with the larger ones. Iperf, by default, uses 1470 byte of data when we use –l argument and 1 Mbps data rate when we use –b option, respectively. During the test, UDP server continuously calculates jitter. In particular, it computes the relative transit time between the client “send” time and receiver “receive” time. A basic command for the UDP test along with its snapshot in VIRL is given below (refer to Figures 4.7 and 4.8).

$*iperf –c server_ip -u –p server_port*

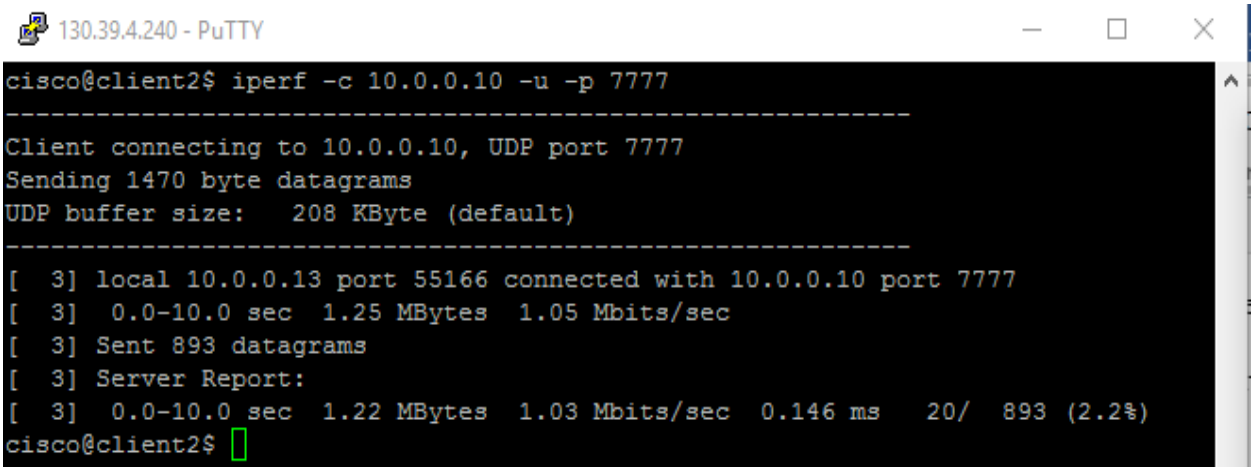


Figure 4.7: UDP client

$*iperf –s –u –p server_port*

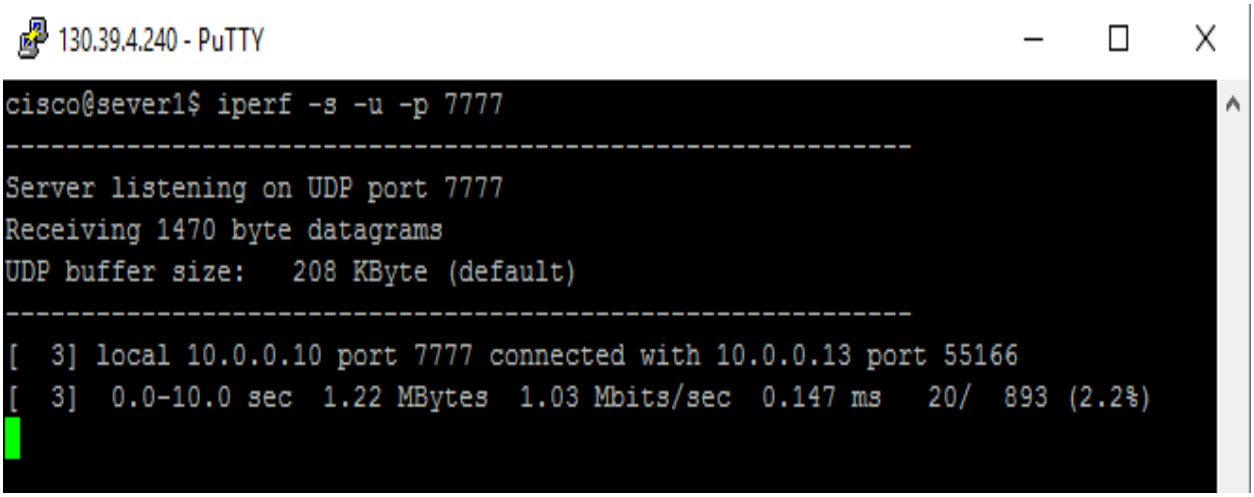


Figure 4.8: UDP server

Like in the TCP, the UDP server is run first. By default, the UDP client sends traffic for 10 seconds. After this time interval, the client gives jitter in milliseconds (ms) and packet loss in percentage as shown in Figures 4.7 and 4.8, which is used for analysis later. The various parameters are changed with different values to generate different traffic data.

4.5 Conclusion

The discussion on the implementation shows how we created various topologies for our study. The external connectivity using the “Putty” gave us more visibility of the nodes and topology. Further, the offline and/or online packet capture using the “Wireshark” helped us to analyze the traffic in the network. The TCP and UDP tuning gave us ideas on the generation of traffic and its analysis. Moreover, this Chapter provides following importance of traffic analysis:

(i) When application consumes more bandwidth during working hours, we can decide whether the particular application should be allowed or not depending on the situation.

(ii) The traffic reports also give us vital information that helps to find out any anomalies in the network. This not only saves the time but also the cost that will involve in securing the networks, when there is a security breach.

5. RESULTS AND ANALYSIS

A detailed description of results obtained from testing the different network built in virtual laboratory for various scenarios is discussed in this chapter. Various network topologies are created by working as one of the user of VIRL server [Refer to Appendix B]. In each topology, the configuration file is generated automatically using "AutoNetKit". Broadly speaking, we have considered three main scenarios i.e., LAN, WAN, and real time traffic.

5.1 Local Area Network (LAN)

The LAN topologies are created with all default configuration and settings for both TCP and UDP. The following things need to be done before entering into traffic generator, Iperf, inside VIRL topology from the external terminal putty. Refer to Figure 5.1.

Login as: Group-1 # username given by the VIRL admin

Password: u3EVGA # password to enter

Management login: ssh cisco@mgmt_int_ip # login into management interface using its ip

Password: cisco # default password

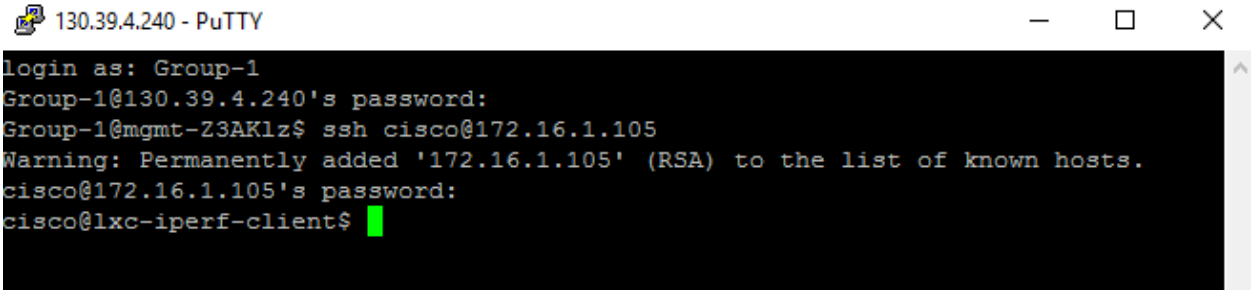


Figure 5.1: Login Client Using Putty

Before generating traffic from a client to the server (source to destination), the server needs to start first before the client. It is ready to receive the traffic from the client with the following command.

$iperf –s –p 6000 # running iperf as server and listening traffic on port 6000

Similarly, a command that helps to run the client for generating traffic is given below.

$iperf –c server_ip –p 6000 # send the traffic to server at port 6000

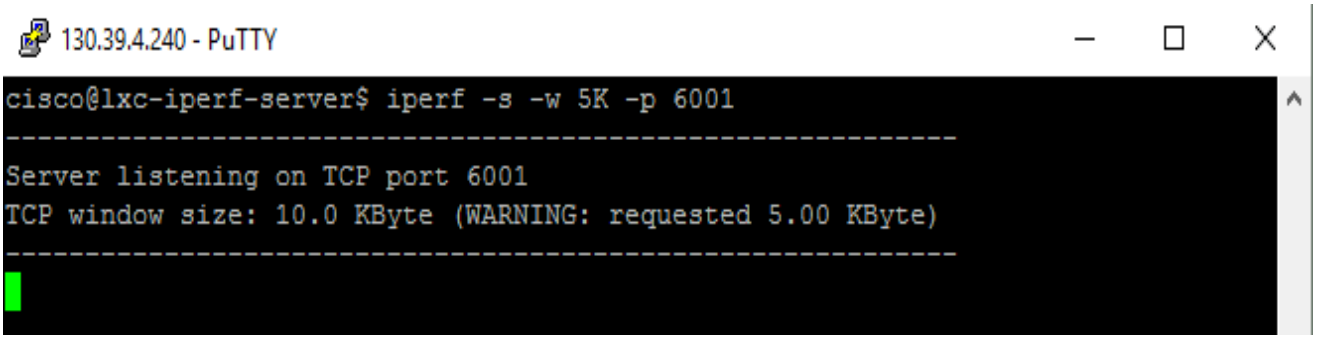
By default, it sends TCP traffic for 10 seconds with window size 85.3 KB. Further, the server receives the traffic on its port 5000.

A. TCP Traffic

Appendix B shows the topology of five nodes (2 iperf nodes, 2 routers, and 1 switch) that we have created to study TCP. We change two variables, namely, the window size (w) and parallel TCP connection (P), and generate results for different combination of variables. Moreover, one Iperf node is tuned to the server and another to the client.

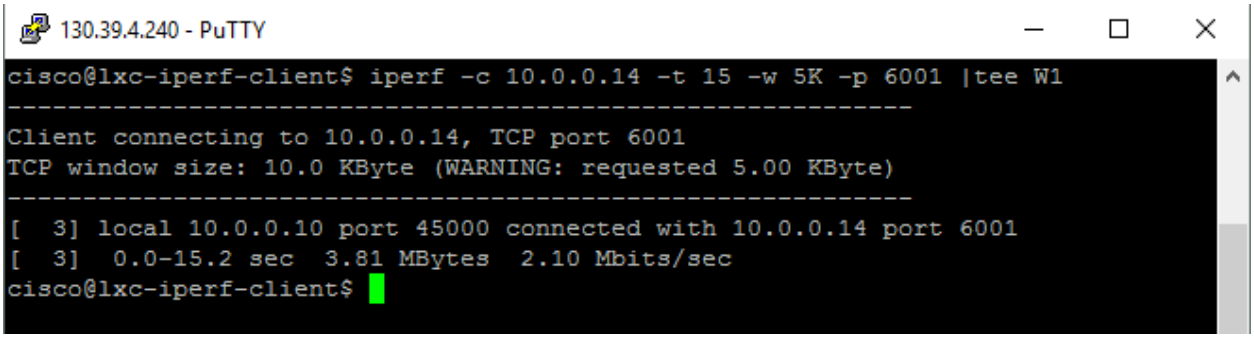
At Server Side

$iperf –s -w 5K –p 6001 # here even window size 5K is given as input, it will, by default assign double of it i.e 10K



At Client Side

$iperf –c 10.0.0.14 –t 15 –w 5K –p 6001 | tee W1 # running as a client and send traffic to server using server ip 10.0.0.14 for 15 seconds having window size w 10K.



The throughput is measured directly from the Iperf client after time-out of the traffic. But to measure the packet-loss ratio, we need to capture the traffic data in a file. The traffic reflects client-server data for specific interval of time. The packet-loss ratio is obtained by dividing the total TCP retransmitted packets by the total transmitted packets. Figure 5.2 illustrates a typical captured data file.

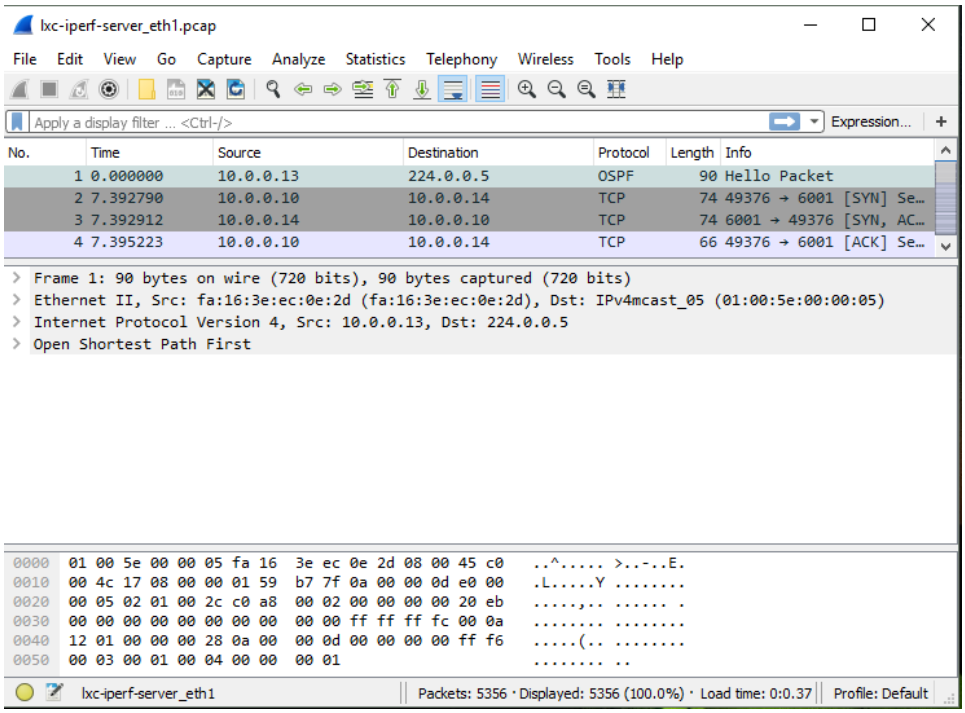


Figure 5.2: Capture File

The display filter command that is necessary to enter in the Wireshark [24] while it is opening the captured file to get total TCP transmitted packet is:

ip.src==10.0.0.10 && tcp.srcport==49376 && ip.dst==10.0.0.14 && tcp.dstport==6001

and similarly for total TCP retransmitted packet is:

ip.src==10.0.0.10 && tcp.srcport==49376 && ip.dst==10.0.0.14 && tcp.dstport==6001 && tcp.analysis.retransmission

Here, TCP source port is generated automatically and can be seen in the captured file.

Results are obtained from the client side by varying the window size (w) from 10KB to 200KB for different parallel TCP connections (from 1 to 10). For parallel TCP connection, variable *P* along with its value needs to be entered in the command from the client side as below.

$iperf –c 10.0.0.14 –t 15 –w 10K –P 2 –p 6002 / tee w2

The above command helps to connect the server with two parallel TCP connections each with the window size of 10K. The main important thing that needs to be noted during experiment is window size, which should be matched in both client and server. The Figure 5.3 and 5.4 illustrates the plot between packet loss ratio verses parallel TCP connection, and throughput verses parallel TCP connections for different window sizes, respectively.

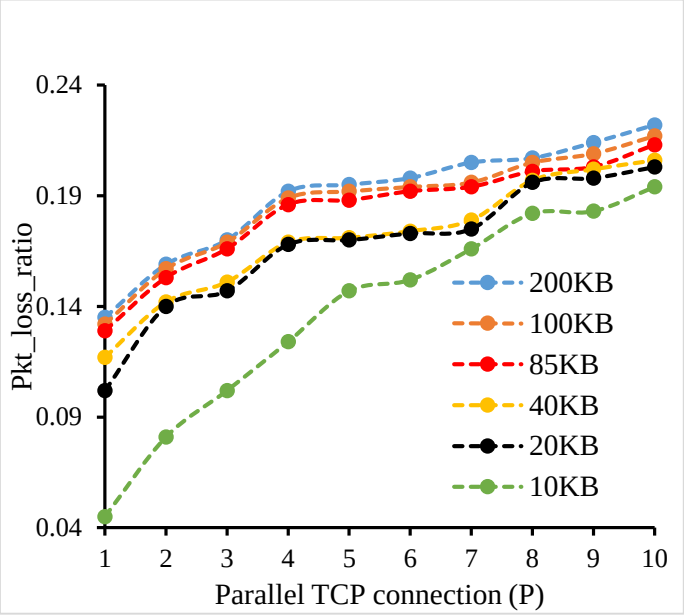


Figure 5.3: Packet loss ratio verses parallel TCP connections

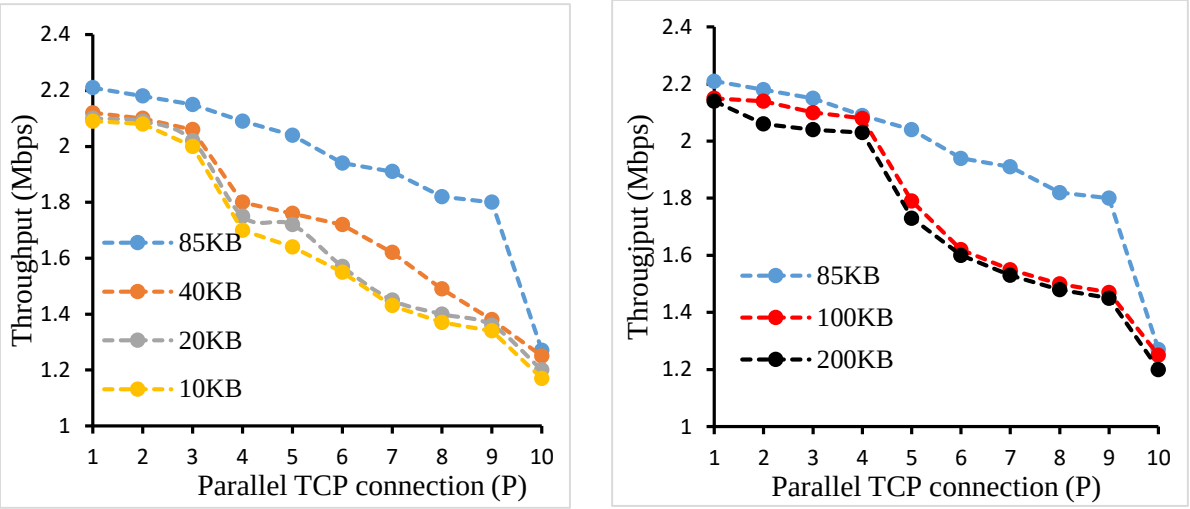


Figure 5.4: Throughput vs parallel connections from (a) 10KB to 85KB and (b) 85KB to 200KB

From Figure 5.3, it is clear that increasing the parallel TCP connection increases the probability of packet-loss ratio. This happens due to the overflow of buffer of first router. Actually, when we are increasing the number of parallel TCP connections, it is sending more traffic to the server. As a result, the packets are lined up in a buffer queue, and when packets coming to the router exceeds its capacity, they are simply dropped.

In addition, increasing the size of window also increases packet loss ratio (see Figure 5.3). Window size means the amount of data that the sender can sent at a time. By sending more traffic that also overflow the buffer of the router and the probability of packet loss increases. Hence, both parallel TCP connection and window size have inverse relation with the packet-loss ratio.

Figure 5.4 illustrates that by increasing parallel connections, we decrease throughput in the network. We observed that increasing parallel TCP connections cause an increase in the packet-loss ratio, and from equation (3) of Chapter 4, the throughput is inversely proportional to the packet-loss ratio. So, higher the packet-loss ratio lower will be the throughput of the network.

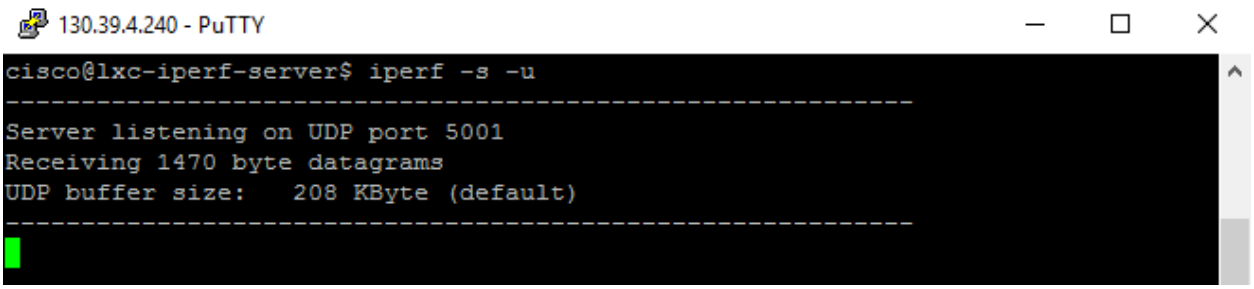
Generally, the window size and throughput has direct correlation. That means higher the window size higher will be the throughput of the network. But, in practice, the nodes between source to destination may not be able to handle large amount of data. Thus, there is always a critical value of the window size at which we get maximum throughput. In our case, when increase the window size from 10KB to 85KB, throughput increases accordingly (see Figure 5.4). However, when the window size is increased further (from 85KB to 200KB), the throughput starts to decrease. A critical window size happens at 85KB. It is advantageous to know the critical value of the window size for a network to achieve the maximum throughput.

B. UDP Traffic

Similar to TCP analysis, we created a topology with five nodes for our experiment with all default setting (see Appendix B1.2). There is one difference, the congestion control algorithm is not included in UDP. We have used this for real-time applications like voice, video, etc. Two variables, namely, packet transmission rate (b) and datagram size (l), are varied. Results, packet loss and jitter, are obtained for different combinations of these variables. The following commands show UDP traffic generation at the server and the client sides.

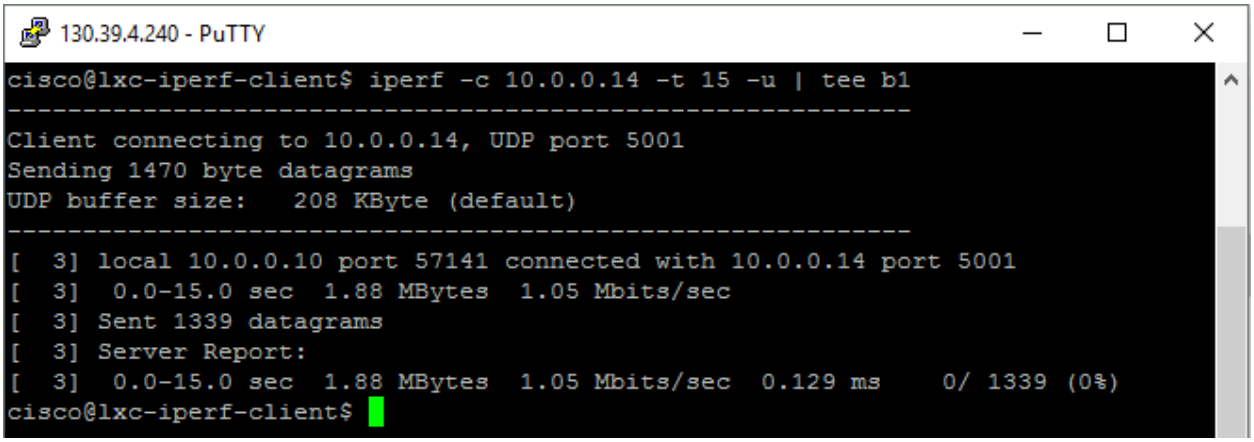
At Server Side

$iperf –s –u # running the iperf node as UDP server. By default, it will listen on the port 5001



At Client Side

$iperf –c 10.0.0.14 –t 15 –u | tee b1 # running the iperf node as UDP client, send traffic for 15 seconds, and save it in a file b1. By default, it will send datagram of 1470B at transmission rate of 1 Mbps.



Results are obtained by varying data transmission rate from 64 Kbps to 2 Mbps for three different datagram sizes (360B, 735B, and 1470B) as most of the datagram for the real time traffic falls in the range 360B to 1470B. The datagram size is fixed at both client and server. The sample commands for this experiment is given below.

At Sever Side

$iperf –s –u –l 360B

At Client Side

$iperf –c 10.0.0.14 -t 15 –u –l 360B –b 64K | tee b2

We have not observed any loss up to 2 Mbps for the above three datagram sizes. However, we noticed a significant jitter (see Figure 5.5).

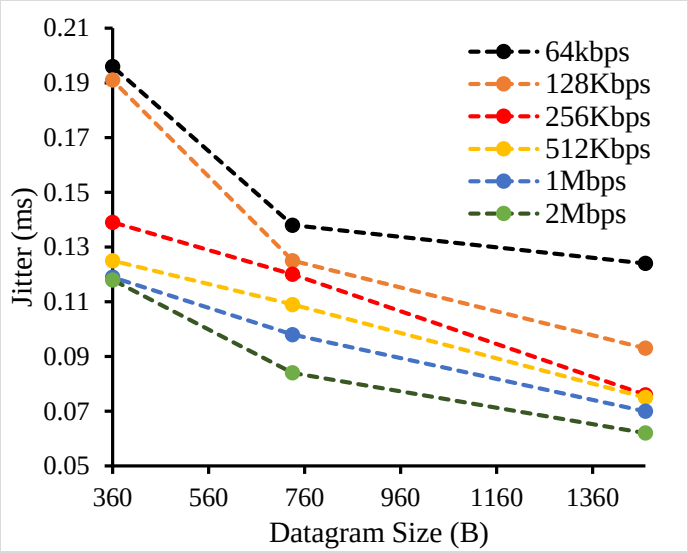


Figure 5.5: Jitter verses datagram size

From the jitter verses datagram plot, it is clear that whenever the size of datagram increases from 360B to 1470B, the corresponding value of the jitter decreases for any particular transmission rate from 64 Kbps to 2 Mbps. It is because the larger packets reduce the overhead that needs to be transmitted and that, in turn, reduces the head of line blocking in routers. As a result, it decreases the overall congestion and jitter.

In addition, the transmission rate has inverse relation with the jitter up to 2 Mbps. It means higher the rate, lower is the jitter. It happens because when the transmission rate increases, the rate at which the receiver receive the packets increases causing the decrease in time interval between the successive received packets. also increase. As a result, there is less jitter.

In our case, when the transmission rate increases greater than 2 Mbps, there is not only a significant increase in the jitter but there is also significant a loss. It is because, when the rate exceeds the capacity of a router to handle the packets, there is an excess congestion. So, the packets

are dropped, which increases time interval between successive packets at the receiver resulting into a higher jitter.

It becomes very important to know up to what speed the packets need to be sent in any network for a better performance. Note, the size of datagram should not be greater than the MTU of the link. Otherwise, the packets will be fragmented into smaller packets causing a negative impact on the jitter.

C. Both TCP and UDP Traffic

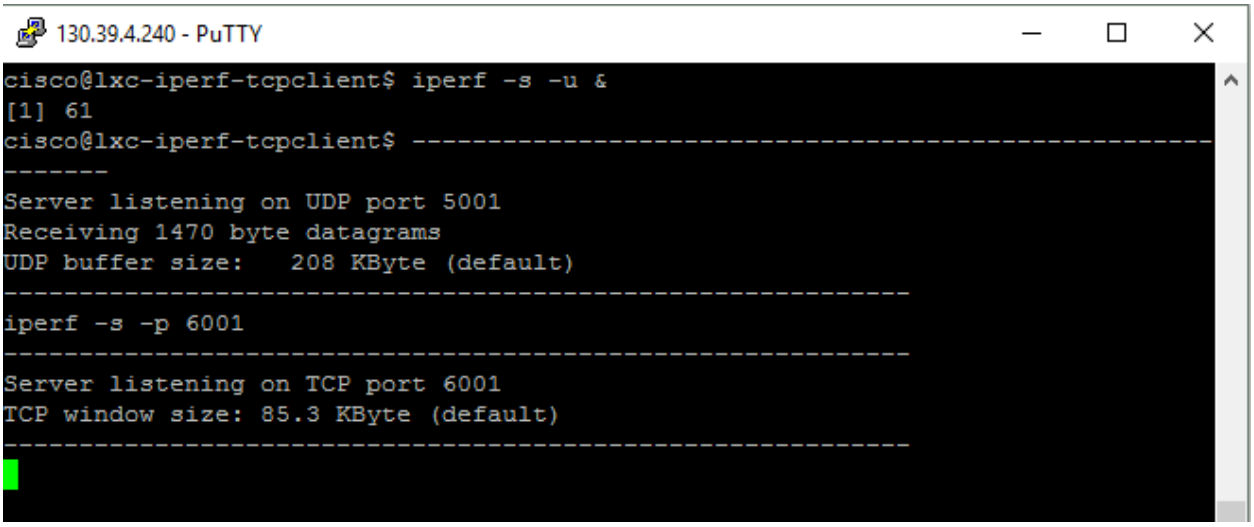
To understand the effect of TCP and UDP in a LAN environment, a topology with 6 nodes (having 3 Iperf nodes) is created with all default configuration and setting in the flat network mode (see Appendix B1.3). Out of two Iperf nodes, one is a TCP client and the other is a UDP client). Both clients are connected to the same bottleneck router. The third Iperf node is set to work as a server for both TCP and UDP.

The variable, parallel TCP connections, is changed from 3 to 12 from TCP client at the step of 3. It sends the traffic to the TCP sever for fixed a window size of 85KB. At the same time, the UDP client also transmit the traffic to UDP server by varying the datagram size and transmission rate. All the results are obtained from the UDP client to see the effect of TCP on UDP. The sample commands, used to conduct this experiment, are shown below.

At Server Side

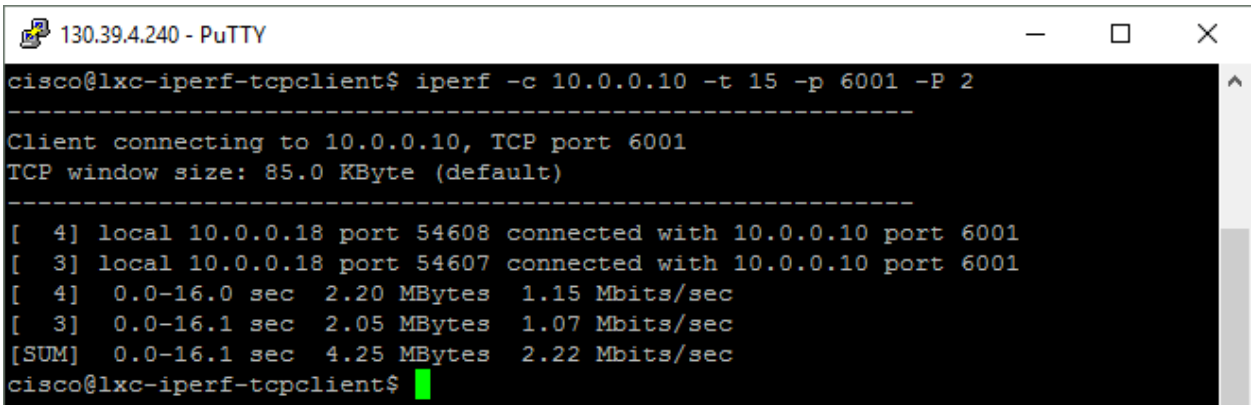
$iperf –s –u & # running as UDP server on background and listen on port 5001

$iperf –s –p 6001 # running as TCP server on front and listen on port 6001



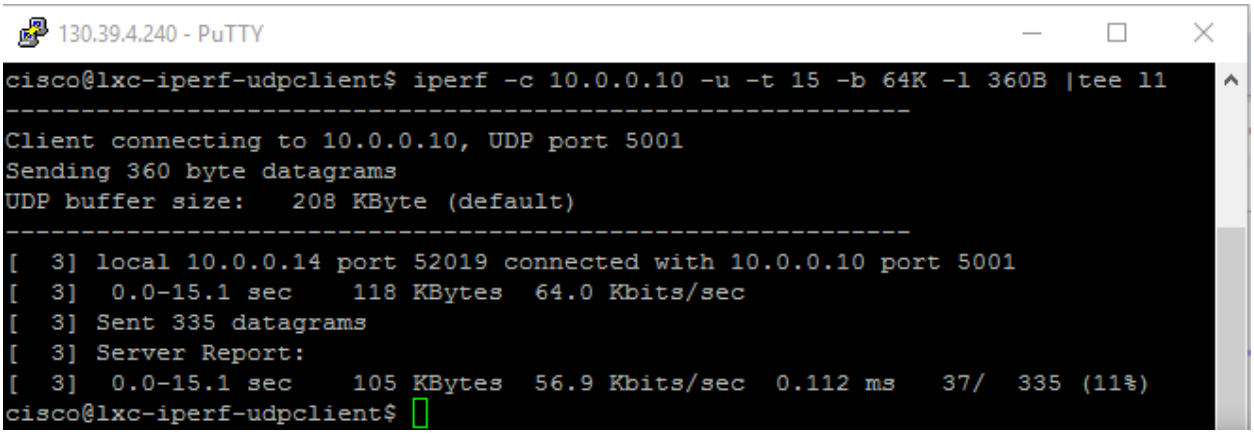
At TCP Client

$iperf –c 10.0.0.10 –t 15 –p 6001 –P 2 # send traffic to TCP server with two parallel TCP connection.



At UDP Client

$iperf –c 10.0.0.10 –t 15 –b 64K –l 360B | tee l1 # send traffic to UPD server and save it to l1



Results are obtained for three different datagram sizes (360B, 735B, and 1470B) and for three transmission rates (64 kbps, 512Kbps, and 2 Mbps) in the presence of parallel TCP connections. Figure 5.6 and 5.7, respectively, depicts the change in the (i) jitter vs datagram size and (ii) packet loss vs datagram size for different parallel TCP connections.

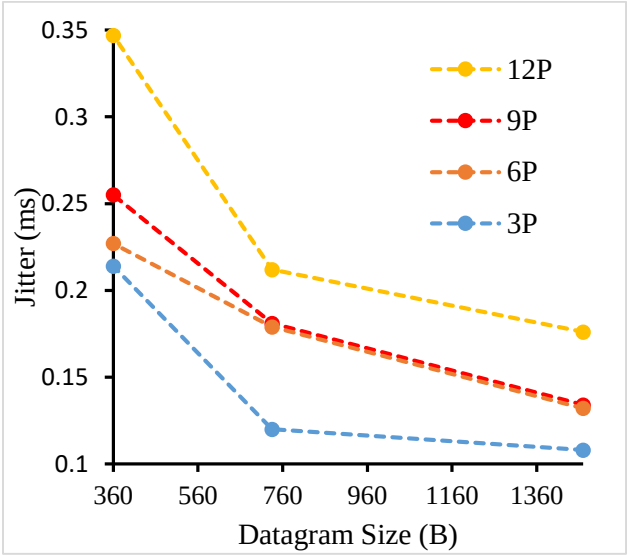


Figure 5.6: Jitter verses datagram size for different parallel connections at 64 Kbps

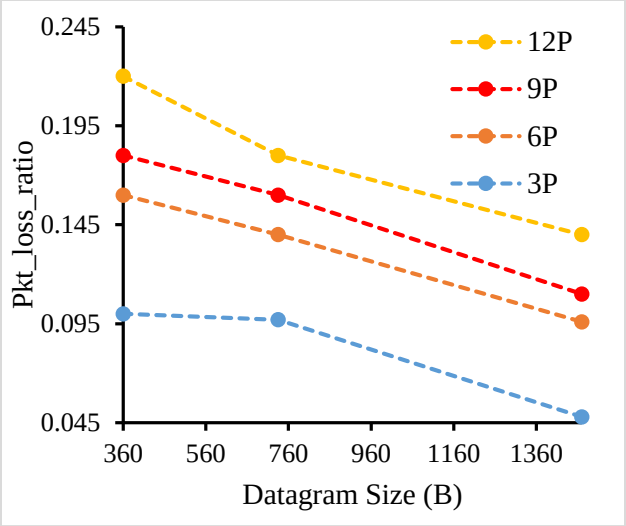


Figure 5.7: Packet loss ratio vs datagram size for different parallel connection at 64 Kbps

Similar kind of results are plotted for date rates 512 Kbps and 2 Mbps (see Figure in the Appendix C1.1 and C1.2). The plot of jitter verses datagram size, and packet loss ratio verses datagram size showed that when the datagram size increases from 360B to 1470B, it lowers both jitter and the packet loss for any particular TCP connection. It is because a larger datagram reduces the congestion, and packet being dropped on the bottleneck router than smaller packets. As a result, it will also reduce the time interval between successive packets at the receiver. Thus, the larger packet sizes not only decrease the jitter but also packet loss.

Moreover, the number of parallel TCP connections impacts on the UDP jitter and packet loss. The traffic at the bottleneck router increases when the number of parallel connection increases. It causes an increase in the probability of congestion and the packet being dropped. Hence, parallel connection has direct relation with the jitter and the packet loss.

To help to reduce the jitter, it is good to increase the speed of the UDP transmission packet. But, it does not reduce the packet loss. In fact, it increases the packet loss ratio because the increase in the transmission rate fills up the buffer at the bottleneck router at a faster rate. Hence, the

possible solution to reduce to effect of TCP on UDP is to increase the capacity of the bottleneck router.

5.2 Wide Area Network (WAN)

Our WAN topology has three Autonomous Systems (AS) connected by the Broder Gateway Protocol (BGP). The latency from the client to server is very high in comparison to LAN, and there are multiple links that may be chosen by traffic to reach the destination. Similar to LAN, our experiments are conducted for three different conditions to observe the performance in each category. They are TCP, UDP, and both TCP and UDP simultaneously. The necessary commands to run the server and to generate the traffic from the clients are similar to LAN.

A. TCP Traffic

The topology of 14 nodes is shown in Appendix B2.1. We have changed the two variables, namely, the window size (w) and parallel TCP connections (P) and have generated results from their different choices. The commands that are necessary to generate the traffic and to run the server are similar to that given for the LAN.

During each experiment, we have chosen a matching window size at the server and the client. The throughput is measured directly from the TCP client whereas the packet loss ratio is computed from the captured file by using the display filter command in Wireshark [24]. Results, obtained by varying parallel TCP connection from 1 to 10 for different window sizes, are plotted for the analysis. Figure 5.8 through 5.10 showed the plot of packet loss ratio verses parallel TCP connection and throughput verses parallel TCP connection for various window sizes, respectively.

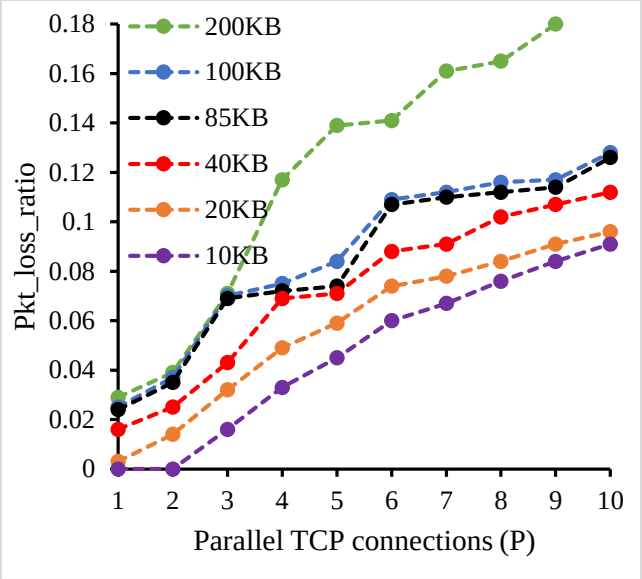


Figure 5.8: Packet loss ratio verses parallel TCP connection

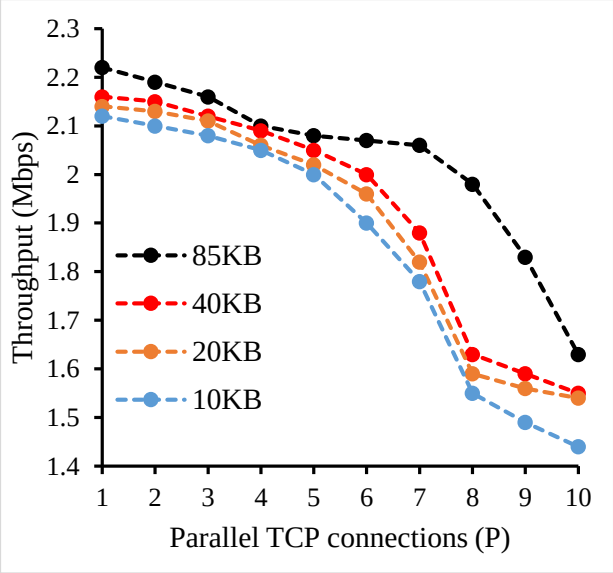


Figure 5.9: Throughput verses parallel TCP connection for window sizes from 10KB to 85KB.

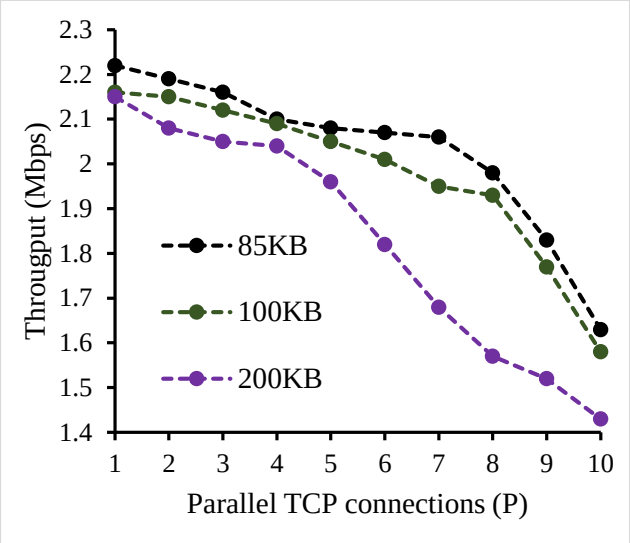


Figure 5.10: Throughput verses parallel TCP connection for window sizes from 85KB to 200KB.

As illustrated in Figure 5.8, the probability of packet-loss ratio increases with the increase in parallel TCP connections for a particular window size. It is because an increase in the parallel connections changes the amount of traffic, which, in turn, increases the congestion at the router. This causes the packet being dropped. Moreover, the plot also shows that an increased in the window size increases the packet loss ratio. Actually, when we have larger window size, it sends more traffic to the server. So, the buffer at the router fills up quickly. Consequently, the probability of packet loss ratio further increased.

Similarly, Figures 5.9 and 5.10 illustrates the throughput of the network. Note that it increases with an increase in the window size but only up to a certain extent. In our network, the router becomes unable to handle the traffic efficiently after 85KB. It is because the buffer fills up more quickly forcing the incoming packets to drop. In addition, the number of parallel connections causes an increase in the traffic, which makes the router congested causing them to drop the throughput of the network.

In comparison to LAN, the effects of multiple path latency are also observed. The latency has an inverse relation with the throughput. The effect of multipath is a dominant factor over latency. In WAN, the multiple links not only decrease the packet loss ratio but also helps to increase the throughput.

B.UDP Traffic

Similar to the TCP, a topology of 14 nodes is created and the same is shown in Appendix B2.1. We have varied two variables, namely, the datagram size and transmission rate. We have generated results from their different combinations. Commands that are necessary for each traffic generation are similar to LAN. During each traffic generation, the size of the datagram is matched at both client and server ends.

The results are obtained for three different datagrams sizes (360B, 735B, and 1470B) at various transmission rates varying from 64Kbps to 2Mbps. Unlike LAN, the packet loss is not obtained up to 2Mbps. Figure 5.11 shows a plot of jitter verses datagram at various transmission rates.

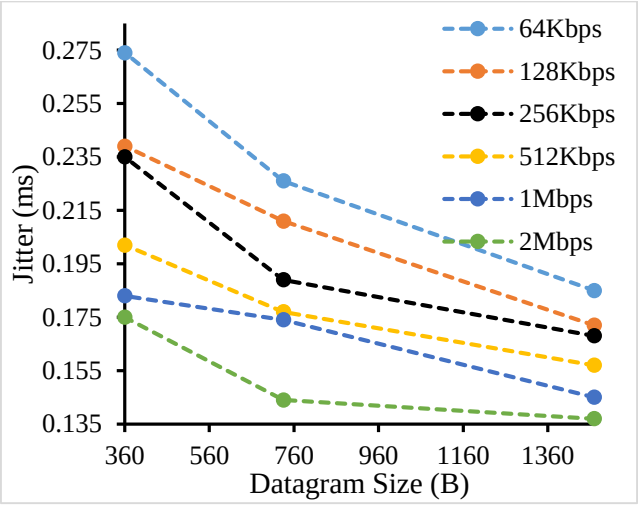


Figure 5.11: Jitter verses datagram in UDP traffic

As shown in Figure 5.11, the datagram size increases from 360B to 1470B and the transmission rate changes from 64Kbps to 2Mbps. The jitter is inversely proportional to the data rate. It is due to less congestion at the router. In other words, large data packets would reduce the head of line blocking at the router. A higher transmission rate would further decrease the jitter because the receiver would receive packets at much faster rate. However, there is a limit in the datagram size and transmission rate up to which the increase in the performance takes place.

Similar to TCP, the multipath shows some effect on the jitter (in comparison to a LAN). In WAN topology, the jitter is comparatively higher because the packet might take different routes to reach same destination. Some packet would reach destination faster and some would reach slower. So, there is no consistency in the received packet at the receiver causing the jitter to increase. Note, the latency has no effect on UDP because whatever that time may be, the client would send given number of packet per second.

C. Both TCP and UDP Traffic

In the topology of 14 nodes (Appendix B2.1), two clients (TCP and UDP) are connected to the same bottleneck node as in LAN, and a single server works as both TCP and UDP sever. The TCP and UDP traffic is simultaneously sent traffic to the server. The results are obtained for various datagram sizes (360B, 735B, and 1470B), and different transmission rates (64Kbps to 2Mbps) while TCP client would vary its parameter (parallel connections) from 3 to 12. Commands that perform this experiment are similar to LAN. The plot of results is shown in Figures 5.12 and 5.13, respectively.

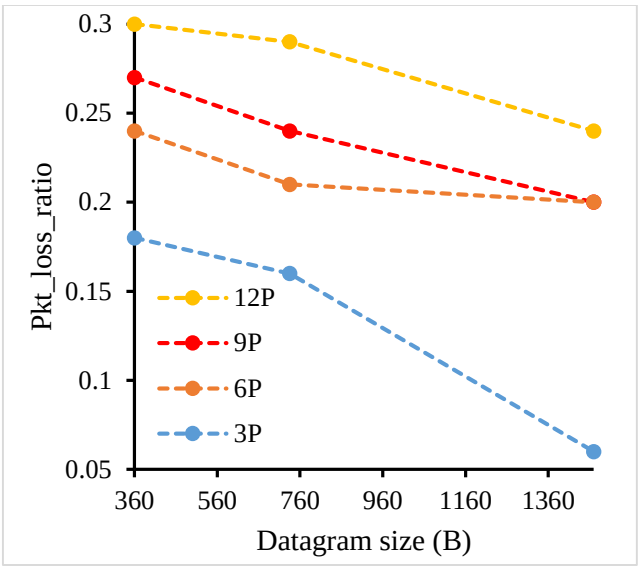


Figure 5.12: Packet loss ratio verses datagram for different parallel connections at 64 Kbps

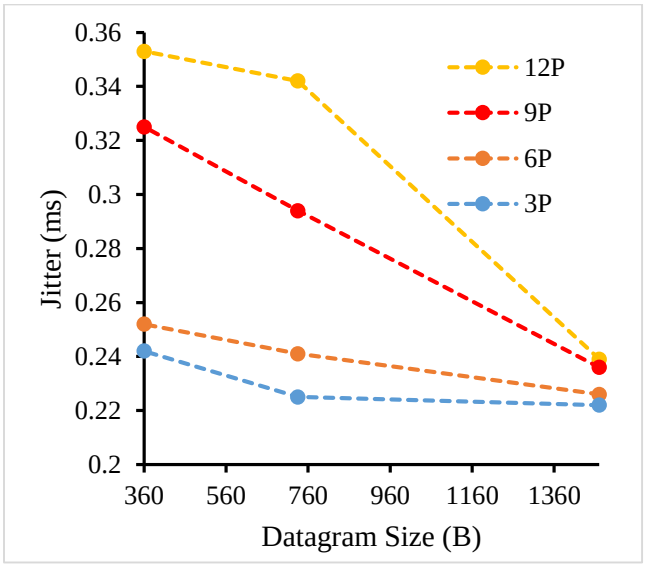


Figure 5.13: Jitter verses datagram for different parallel connections at 64 Kbps

Similar kind of results are obtained at 512Kbps and 2Mbps, which are shown in Appendix C2.1 and C2.2. The plot of jitter verses datagram and packet loss ratio verses datagram emphasize that jitter and packet loss have inverse relation with the datagram size for any particular TCP connection. The reason is when a large datagram is sent, the overhead that needs to be added

becomes less. This would reduce congestion at the router. So, the probability of packets being dropped and time interval at which it receives the successive packets would reduce. Hence, jitter and packet-loss would decrease.

The number of parallel TCP connections directly impacts on both jitter and packet-loss ratio of UDP. It is because when this number increases, the amount of traffic changes causing an increases in the congestion on bottleneck router. As a result, packets are dropped. And the packet-loss ratio is more. In UDP, dropped/lost packets are not retransmitted. So, the time interval between the successive packets at the receiver would increase resulting in a higher jitter.

5.3 Effect of TCP on Real Time Traffic

Today, most of the traffic in any network is either in TCP form or in UDP form. The analysis of QoS parameters for both TCP and UDP traffic in LAN and WAN provides us the notion of network/traffic importance. To know the effect of TCP on real time traffic, experiments are conducted for three traffic scenarios. They are video, voice, and web browsing. Among them, the first two is UDP traffic (voice, and video) where the remaining one portrays a TCP traffic (web browsing). Experiments are further extended to find ways to minimize the effect of TCP.

A. Video Traffic

The video traffic is sent in the form of UDP traffic. The transmission rates are usually from 128Kbps to 2 Mbps. They are sent in three packet sizes: 576B, 1000B, or 1470B. The experiment is conducted for a particular case of 576B datagram size

For this, the topology of 6 nodes with higher latency around 120ms (two way) is created. it is shown in Appendix B3.1. Two clients (TCP and UDP) are connected to the same bottleneck

router at one end. They send the traffic to their individual servers at the other end. Commands necessary to run servers and to generate the traffic from each individual client are given below.

$iperf –s –p 6001 # running as TCP server and listen on port 6001

$iperf –s –u –l 576B # running as UDP server, accepting 576B datagram and listen on 5001 by default

$iperf –c 10.0.0.18 –t 15 –p 6001 –P 2 # sending TCP traffic with two parallel connection

$iperf –c 10.0.0.14 –u –l 576 B –b 128K –t 15 |tee l1 # sending UDP traffic

During each traffic generation, the TCP window size at the TCP client and the server is kept at the default (85KB) value whereas UDP datagram size at its client and the server is matched. The TCP client changed its variable (parallel TCP connection) from 3 to 12 at the step of 3. The UDP client also changes its parameters (transmission rate from 128Kbps to 2Mbps) to get different results for 576B of datagram. Figure 5.14 and 5.15 illustrates a plot of jitter verses data rate and the packet loss rate verses data rate for 576B of datagram at different parallel TCP connections.

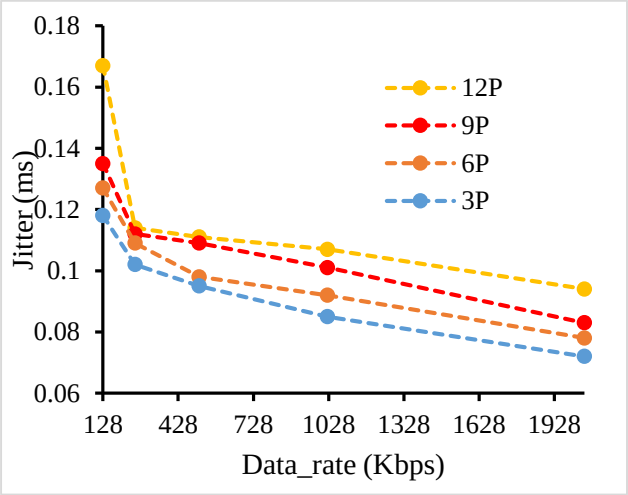


Figure 5.14: Jitter verses data rate at 576B of datagram size

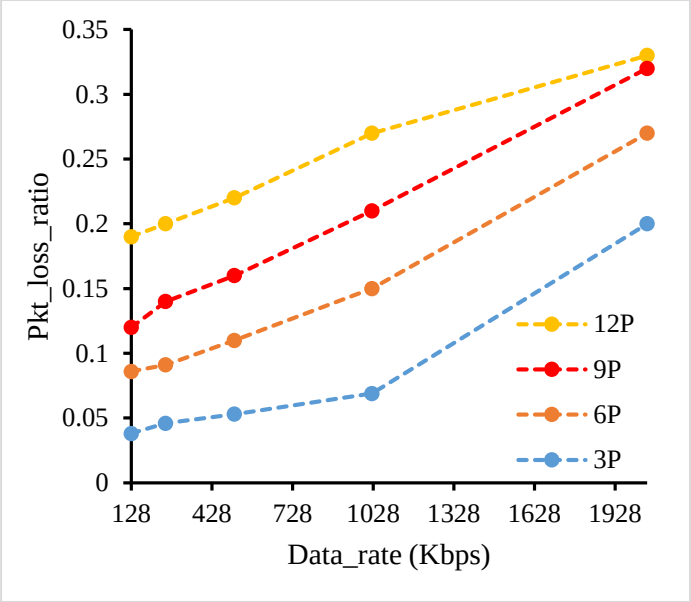


Figure 5.15: Packet loss rate verses data rate at 576B of datagram size

From Figures 5.14 and 5.15, it is clear that with the increase in parallel TCP connections, both the jitter and the packet loss rate of UDP decrease. Our main goal is to decrease the jitter and packet loss rate without decreasing the transmission rate even in the presence of TCP traffic while they are competing for the same bandwidth.

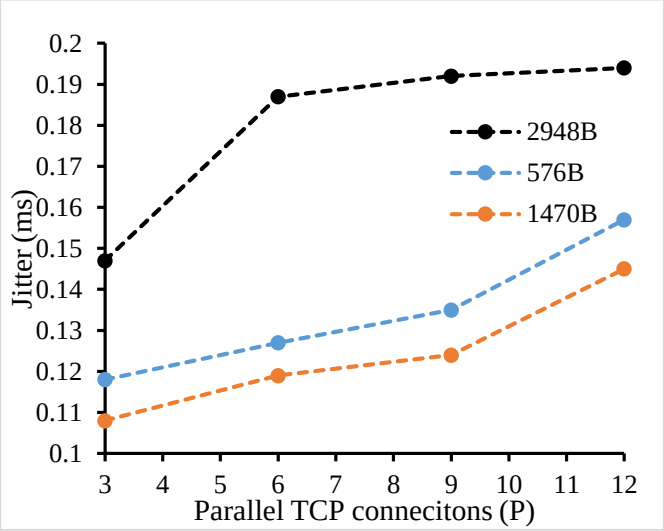


Figure 5.16: Jitter verses parallel TCP at 128Kbps

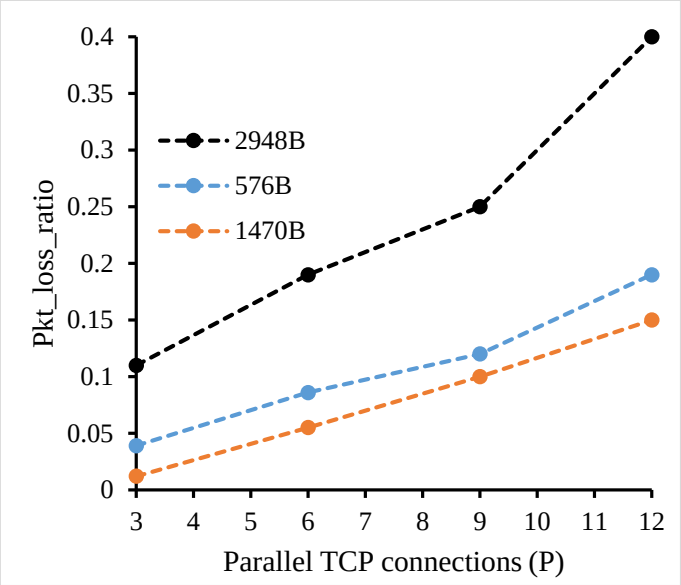


Figure 5.17: Packet-loss ratio verses parallel TCP at 128Kbps

Another experiment is conducted where at each transmission rate different results are obtained by increasing the datagram size from 570B to 1470B for different parallel TCP connections. A typical plot at 128Kbps is shown in Figure 5.16 and 5.1.7. [Rest of the plots at 512Kbps and 2Mbps is shown in Appendix C3.1 and C3.2.]

The plots show that when the packet size is increased, both the jitter and packet loss ratio decrease. However, if we try to increase the datagram size greater than 1470B, both the jitter and the packet loss ratio increase. It is because when the packet size is more than 1500B (MTU) of link, packets are fragmented into smaller chunks. The smaller packet means more overhead and more congestion. This will not only cause loss but also the packet aggregation at the receiver will increase jitter. Decreasing the transmission rate may also decrease the packet loss rate but it will increase the jitter. Hence, increasing the datagram size will be one of the solutions to reduce the impact of TCP.

B. VoIP Traffic

The voice traffic (or Voice over IP) is also sent in the form of UDP traffic. Data rates usually range from 8Kbps to 64Kbps and packet sizes from 60B to 240B. Let us consider the datagram of size 80B to conduct this experiment. The topology, configurations, and commands necessary to generate the traffic and to run servers are similar to video traffic shown in Appendix B3.1. Different results are obtained for the datagram (80B) by varying the transmission rates from 8Kbps to 64Kbps at UDP client and parallel TCP connections from 3 to 12 at the step of 3 at TCP client. Figures 5.18 and 5.19 illustrates two typical plots.

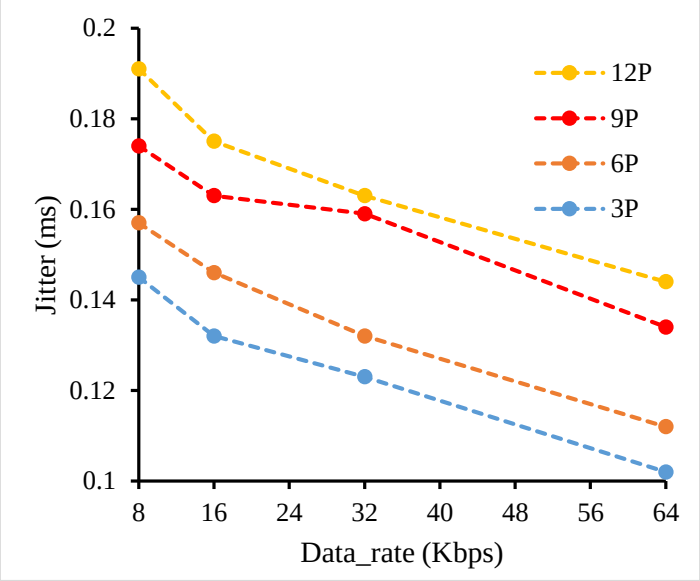


Figure 5.18: Jitter verses data rate for 80B of datagram size

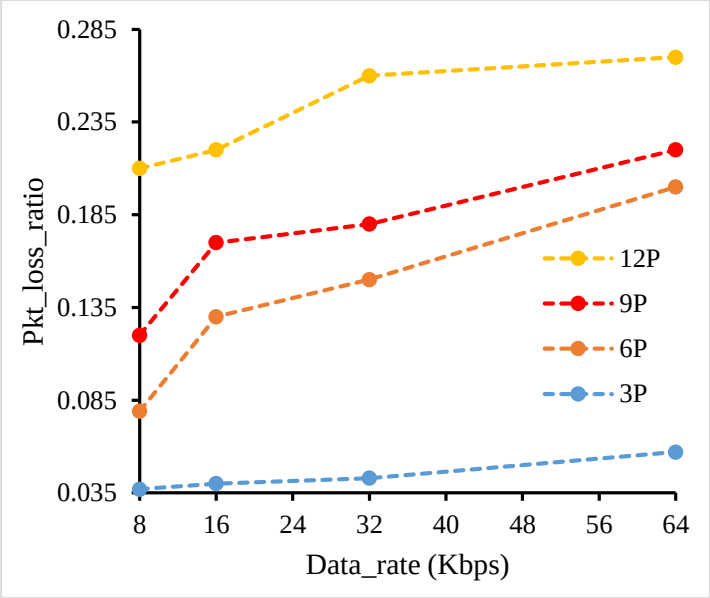


Figure 5.19: Packet loss ratio verses data rate for 80B of datagram size

These figures show that when the number of parallel TCP connections increases, both the jitter and the packet-loss ratio decrease. Our main goal is to reduce jitter and loss ratio without decreasing the data rate even in the presence of TCP traffic. So, at each transmission rate, the experiment is further conducted by varying the datagram from 80B to 320B for different parallel TCP connections. Typical plots at 8Kbps are given below in Figures 5.20 and 5.21. Other plots at 16Kbbps, 32Kbps, and 64Kbps appear in Appendix C4.1, C4.2 and C4.3.

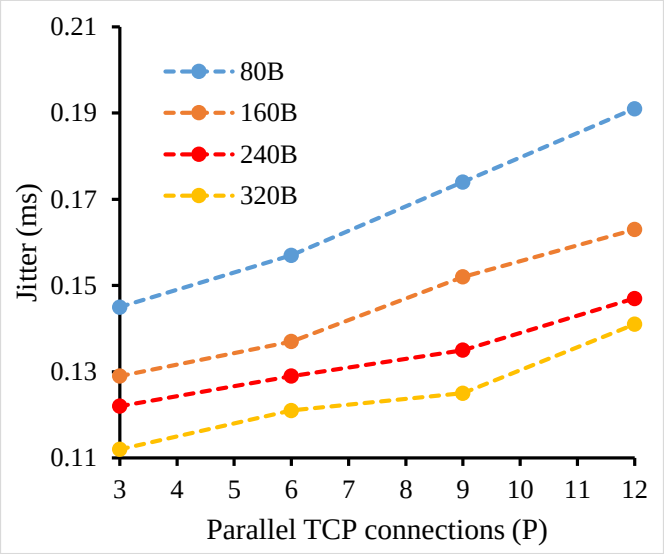


Figure 5.20: Jitter verses parallel TCP at 8Kbps

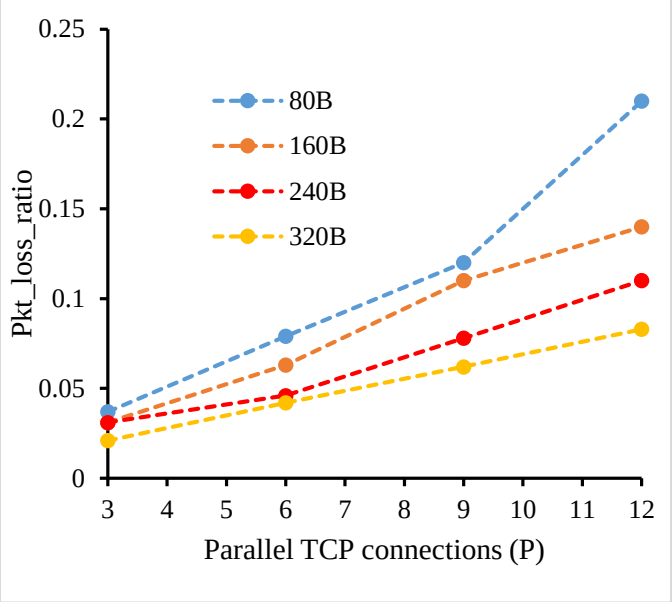


Figure 5.21: Packet loss ratio verses parallel TCP at 8Kbps

Figures 5.20 and 5.21 show that when size of datagram is increased from 80B to 320B, the corresponding value of both jitter and packet loss ratio is decreased. It is because a larger packet contributes to less overhead and less congestion at the bottleneck router. As a result, it decreases

probability of packet being dropped from that node. It also reduces the time to aggregate packets at the receiver resulting less jitter.

C. Web Traffic

The web traffic is sent using TCP. For web browsing, both the window size, and delay are less than 24KB and 400ms, respectively. For our experiment, we consider a size of 16KB and a delay of 120ms. Our topology consists of 6 nodes with two TCP clients connected to the same bottleneck node at one end and two TCP servers at the other end as shown Appendix B3.2.

One TCP client sends traffic by varying the number of parallel TCP connections at the window size of 85KB while another client sends web traffic at window size of 16 KB. Commands necessary to generate traffic and run servers are similar to those discussed in the previous cases. The main goal is to increase the throughput and decrease the packet loss ratio of web browsing traffic even in the presence of other parallel TCP traffic while they all are competing for the bandwidth. Result obtained by varying the window size of web browsing from 16KB to 128KBare plotted and shown below in Figures 5.22 and 5.23.

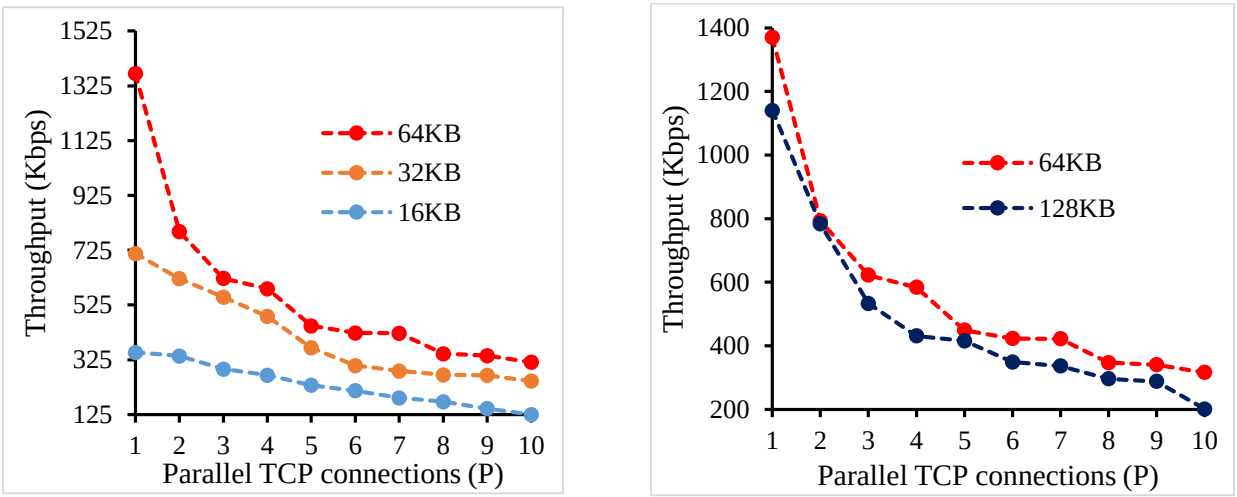


Figure 5.22 Throughput verses parallel TCP from 16KB to 128KB window size

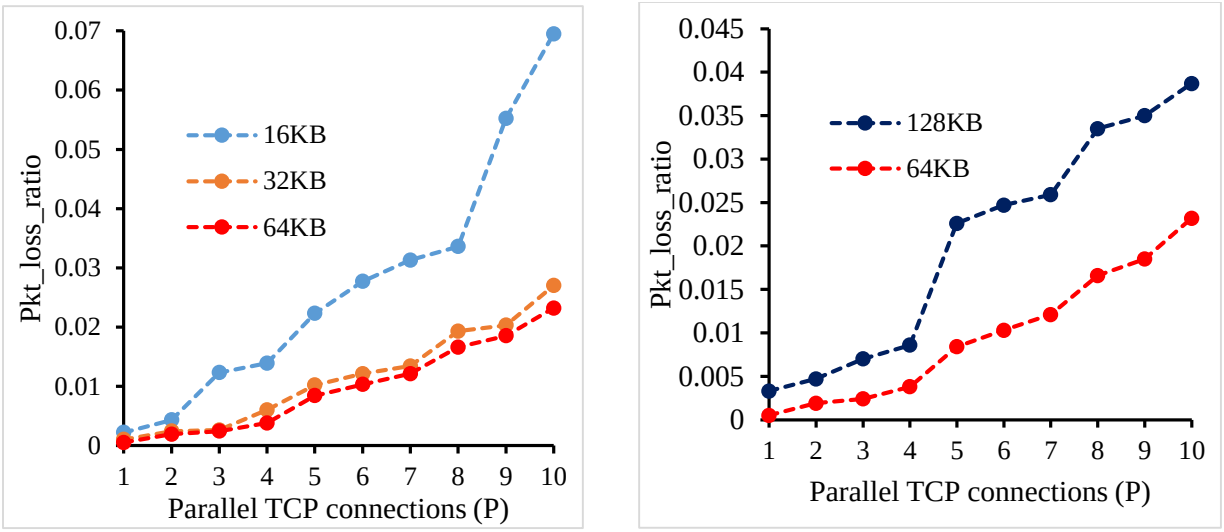
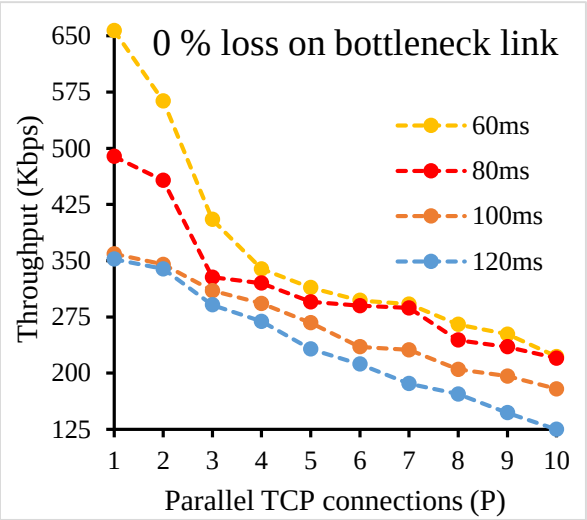


Figure 5.23: Packet loss ratio verses parallel TCP from 16KB to 128KB window size

Figures 5.22 and 5.23 illustrates that the throughput of web browsing increases and loss ratio decreases when the window size is increased up to 64 KB. Beyond this, throughput is decreased and loss ratio is increased because of the congestion at the bottleneck router is due to the heavy traffic. The incoming packets are dropped causing higher packet loss and lesser throughput.

In this particular condition, increasing the buffer size at the router will be an optimal solution. However, the other factor that may affect the performance is latency. To understand its impact, in detail, we conducted another experiment.

For this, we consider the same topology by decreasing the latency from 120ms to 60ms. One is for zero percent loss on bottleneck link and another is for two percent loss on bottleneck to incorporate the scenario for congested link.



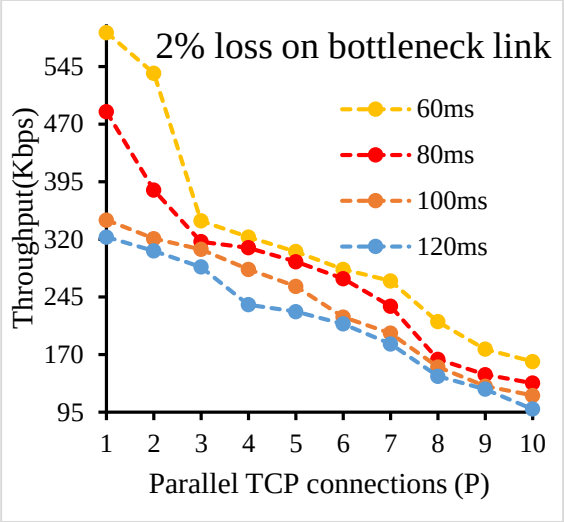


Figure 5.24: Throughput verses parallel TCP at different latency

From Figure 5.24, it is clear that the throughput of the network increases when the latency is decreased at any particular TCP connection (at any particular congested link) Note, when the latency decreases, the time to reach the destination is reduced which would increase the throughput as throughput and latency have an inverse relation. The supporting plots at each latency for the above two cases are shown in Appendix C5.1 and C5.2.

5.4 Conclusion

This chapter gave us insight view of the effect of TCP/UDP traffics on various QoS parameters in different topologies such as LAN, WAN, and some real time applications. In this chapter TCP, UDP, and both TCP and UDP traffics on LAN and WAN networks are studied in detail. We also studied effect of latency on throughput for a congested link. A summary of results obtained from our study and some future work are discussed in Chapter 6.

6. DISCUSSION AND FUTURE WORK

This chapter provides appropriate contribution of our study, and the possible future works. In short, we have studied the following:

- Effect of TCP traffic on LAN and WAN networks.
- Effect of UDP traffic on LAN and WAN networks
- Effect of TCP and UDP in LAN and WAN networks
- Effect of TCP traffic on Real Time Traffics
- Effect of Latency on Throughput.

From the analysis of TCP traffic (in LAN and WAN networks), we observed that both the parallel TCP connections and the window size have direct relation with the packet-loss ratio. However, the parallel TCP connections are inversely proportional to the throughput of the network. We also observed that the window size and the throughput are correlated only up to the certain value of window size (in our case 85KB).

Similarly, in the UDP traffic (LAN and WAN networks), we observed that both the datagram size and the data rate have inverse relation with jitter. But, it is limited by the MTU and the capacity of the link. We did not notice any loss up to the data rate 2Mbps. However, experimenting beyond 2 Mbps, we found that the size of datagram and the data rate directly affect both the jitter and the packet-loss. Knowing the critical values of data rate and TCP window size are essential for the better performance of a network.

For simultaneous TCP and UDP traffic (LAN and WAN networks), we also observed the direct effect of the datagram size on the packet-loss ratio for any number of parallel TCP

connections. It is not present in either the TCP or the UDP traffic. In addition, the number of parallel TCP connections have direct impact on both the UDP jitter and the packet-loss ratio for a fixed TCP window size. The main difference between LAN and WAN networks are in their latency and presence of multiple paths. From our study, we observed that multiple path is a dominant factor. Thus, in TCP traffic, multiple links decrease the packet-loss, which, in turn, increase the throughput of the network. Moreover, in UDP traffic, the jitter in WAN network is comparatively higher than that of the LAN networks.

In real time UDP applications (voice and video traffic), both the jitter and the packet-loss ratio are decreased without reducing their transmission rate even in the presence of TCP traffic (while they all are competing for bandwidth and/or sharing bottleneck node). This is achieved by increasing the datagram size. Similarly, in the TCP application (Web), the throughput is increased and packet-loss ratio is decreased even in the presence of other parallel TCP connections. This performance is achieved by increasing the window size near its critical value. Finally, the effect of latency on the throughput is studied for two different congested links. It shows that the latency is inversely proportional to the throughput for any congested link. It also tells us that higher the congestion on link lower is the throughput of the network for any latency and any number of TCP connections.

This thesis has emphasized on the importance of quality of service for the better performance in any network. We have presented the various parameters that can affect the quality of service of TCP and UDP traffic in different networks. Moreover, this thesis demonstrated the working of data plane traffic generator tool, Iperf, in virtualized environment to test performance in different networks. The virtual laboratory setup is analogous to the real world organizational

networks. These kind of contributions have really motivated us to utilize and enhance the methodologies to unlock the more factors that can affect network's performance.

This thesis explains the processes in detail that can be performed during the network performance analysis on the real networks. Irrespective of the network topologies, the factors that affect the QoS are analyzed by using the Iperf tool. In addition, for WAN network, a virtual prototype of an organizational network is built with three different Autonomous System (AS) using the transport layer protocol BGP. BGP is the current main internet domain protocol.

In conclusion, the network performance analysis helps the admin to mitigate the factors that are affecting the QoS of any particular traffic. Although, the network engineers are striving to mitigate the existing issues, there will be always new challenges emerging. A successful network performance analysis with a proper methodology can guarantee the QoS for many traffics in different networks.

Using this study as a base, the impact of multipath on the network's performance will be the most fruitful future work. However, it will also be wise to see the dominancy between the latency and the multipath in different scenarios as the network are becoming more and more complex. Moreover, the study of effects on the network by control plane traffics will also be good idea because of increasing market of Software Define Networking (SDN).

REFERENCES

[1] Sawashima, Hidenari, Yoshiaki Hori, and Hideki Sunahara. “Characteristics of UDP Packet Loss: Effect of TCP Traffic.” N.p., n.d. Web. 05 Feb. 2016.

[2] Hacker, Thomas J., Brian D. Athey, and Brian Noble. “The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network.” IEEE, 2002. Web. 10 Feb. 2016.

[3] “Network Latency.” N.p., n.d. Web. 13 Feb. 2016. <http://smutz.us/techtips/NetworkLatency.html>

[4] Krzyzanowski, Paul. “Quality of Service.” N.p., 28 Jan. 2013. Web. 19 Feb 2016.

[5]. Rogier, Boris. “Network Performance: Links between Latency, Throughput and Packet Loss.” Performancevision, 26 May 2016. Web. 05 Apr. 2017.

[6] Lakshman, T.V., and Upamanyu Madhow. “The Performance of TCP/IP for Networks with High Bandwidth-delay Products and Random Loss.” IEEE, June 1997. Web. 11 Mar. 2016.

[7] “VIRL Learning Lab Tutorial.” N.p., n.d. Web. 15 Mar. 2016. <http://virl-dev/innovate.cisco.com/tutrial.php>

[8] Chen, Yan, Toni Farley, and Nong Ye. “QoS Requirements of Network Applications on the Internet.” IOS Press, 2004. Web. 19 Mar. 2016.

[9] Miller, Marshal, and Chris Chase. “TCP/IP Protocol Suite.” *TCP/IP Protocol Suite.* N.p., n.d. Web. 23 Mar. 2016.

[10] Pillai, Sarath. “Iperf: How to Test Network Speed, Performance, Bandwidth.” N.p., 04 Feb. 2013. Web. 09 Apr. 2016. <http://www.slashroot.in/iperf-how-test-network-speedperformancebandwidth>

[11] Lewis, Chris, and Steve Pickavance. “Implementing Quality of Service Over Cisco MPLS VPNs.” *Introduction to QoS.* Cisco Press, 09 Feb. 2016. Web. 05 Dec. 2016.

[12] Deshpande, Sachin, and Srinivas Kandala. “Models for MPEG2 and Video Conferencing.” IEEE, Nov. 2000. Web. 04 Nov. 2016.

[13] Monfort, Jean-Yves. “Basic Requirements to Quality of Service (IP Centric).” International Telecommunication Union, 23-25 May 2003. Web. 14 Apr. 2016.

[14] “Iperf- The Easy Tutorial.” N.p., 10 Dec. 2010. Web. 21 Apr. 2016. <http://www.openmaniak.com/iper.php>

[15] “Quality of Service Technical White Paper.” Microsoft Corporation, 1999. Web. 26 Apr. 2016. <www.cs.columbia.edu/~hgs/internet.qosover.pdf>

[16] “Network Latency and Packet Loss Effect on Performance.” N.p., n.d. Web. 03 May 2016. <https://www.noction.com/blog/network_latency_packet_loss_effects>

[17] Amir, Elan, and Hari Balkrishnan. “An Evolution of the Metricom Ricochet Wireless Network.” University of California at Berkeley, 07 May 1996. Web. 09 May 2016.

[18] Dugan. Jon. “Iperf Tutorial.” Esnet, 2010. Web. 19 May 2016

[19] “What is TCP (Transmission Control Protocol).” *SearchNetworking*. N.p., n.d. Web. 25 May 2016.

[20] “The User Datagram Protocol (UDP).” N.p., n.d. Web. 08 June 2016. <http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/udp.html>

[21] Gueant Vivean. “iperf – The network bandwidth measurement tool.” N.p., n.d. Web. 04 June 2016. <https://iperf.fr>

[22] “TCP Header Format.” N.p., n.d. Web. 10 Dec. 2016. <https://fenix.tecnico.ulisboa.pt>

[23] “TCP/IP Illustrated.” N.p., n.d. Web. 10 Dec. 2016. <http://www.imengineering.com>

[24] Hoffman, Chris. “How to Use Wireshark to Capture, Filter and Inspect Packets.” 30 July 2016. Web. 27 Nov. 2016.

[25] “Zenmap – Official Cross-platform Nmap Security Scanner GUI.” N.p., n.d. Web. 27 Apr. 2016.

[26] Peplnjak, Ivan. “BGP Tutorial: The Routing Protocol That Makes the Internet Work.”. N.p., n.d. Web. 16 May. 2016.

[27] Lowe, Scott. “A Brief Introduction to Linux Containers with LXC.”. N.p., n.d. Web. 25 Sept. 2016.

[28] Partsenidis, Chris. “What Is Iperf and How Is It Used?” *SearchNetworking*. N.p, n.d. Web. 5 Oct. 2016.

[29] “PuTTy”. N.p, n.d Web. 25 Sept. 2016. <https://en.wikipedia.org/PuTTy>

[30] “Using the GNS3 Network Simulator.” N.p., n.d Web. 29 May 2017. <https://www.pluralsight.com/blog/it-ops/using-gns3-network-simulator>

[31] “Packet Tracer – Cisco.” N.p., n.d. Web. 29 May 2017. <www.cisco.com/web/learning/netcad/course_catalog/PacketTracer.html>

APPENDIX A: IPERF GUIDE AND SAMPLE CODE FOR LIVE PACKET CAPTURE

A1: Iperf Guide

Table A1.1: General Options [21]

General Options		
Command Line Option	Environment Variable Option	Description
-f, --format	$IPERF_FORMAT	A letter specifying the format to print bandwidth numbers in. Supported formats are 'b' = bits/sec 'B' = Bytes/sec 'k' = Kbits/sec 'K' = KBytes/sec 'm' = Mbits/sec 'M' = MBytes/sec 'g' = Gbits/sec 'G' = GBytes/sec
-i, --interval	$IPERF_INTERVAL	It will set the interval time in seconds between the periodic bandwidth, jitter and loss reports. By default, it is zero. That means no periodic reports are printed
-l, --len	$IPERF_LEN	It shows the length of buffer to read or write. By default, it is 8 KB for TCP and 1470 B for UDP. Also for UDP, it is size of datagram.
-p, --port	$IPERF_PORT	This is port for the sever to listen on and the client to connect to. It should be same in both server and client. By default, it is 5001
-u, --udp	$IPERF_UDP	It tells to use UDP rather than TCP
-w, --window	$TCP_WINDOW_SIZE	It will set socket buffer to specified value. For TCP, it will set TCP window size but for UDP it is just the buffer on which datagram are received
-h, --help		Print out a summary of command and quit
-v, --version		Print version information an quit

Table A1.2: Server Specific Options [21]

Server Specific Options		
Command line option	Environment variable option	Description
-s, --server	$IPERF_SERVER	It will run Iperf in server mode
-D		It will run the server as a daemon.
-R		If Iperf service is running, it will remove it
-P	$IPERF_PARALLEL	The number of connections to handle by the server before closing. Default is 0(which means to accept connections forever)

Table A1.3: Client Specific Options [21]

Client Specific Options		
-b, --bandwidth	$IPERF_BANDWIDTH	It is speed in bits/sec at which UDP sends data. This implies –u option. By default, it is 1 Mbit/sec
-c, --client	$IPERF_CLIENT	It will run the Iperf in client mode, connection to an Iperf server that is running on host.
-d, --dualtest	$IPERF_DUALTEST	It will run Iperf in dual mode. That means, it will cause the server to connect back to the client.
-t, --time	$IPERF_TIME	It is the duration of time in second to transmit the data. By default, it is 10 seconds.
-P, --parallel	$IPERF_PARALLEL	It is number of simultaneous connection that is connecting to the server.
-T,--ttl	$IPERF_TTL	It is the time-to-live for the outgoing multicast packets. In other words, it is the number of routers to go through. By default, it is 1

A2: Sample Code for the Live Packet Capture

```
TITLE VIRL Live Packet Capture
MODE con:cols=80 lines=12
COLOR 1F
set NETCAT_PATH=%PROGRAMFILES(x86)%\Nmap\ncat.exe
set WIRESHARK_PATH=%PROGRAMFILES%\Wireshark\Wireshark.exe
echo.
set /P VIRL_HOST="VIRL Server IP: "
set /P PCAP_PORT="Live port: "
echo.
echo Reading live capture from port %PCAP_PORT%. Close this window to stop capture!
echo.
"%NETCAT_PATH%" %VIRL_HOST% %PCAP_PORT% | "%WIRESHARK_PATH%" -k -i
-
```

APPENDIX B: TOPOLOGIES FOR LAN, WAN AND REAL TIME TRAFFICS

B1: LAN Topology

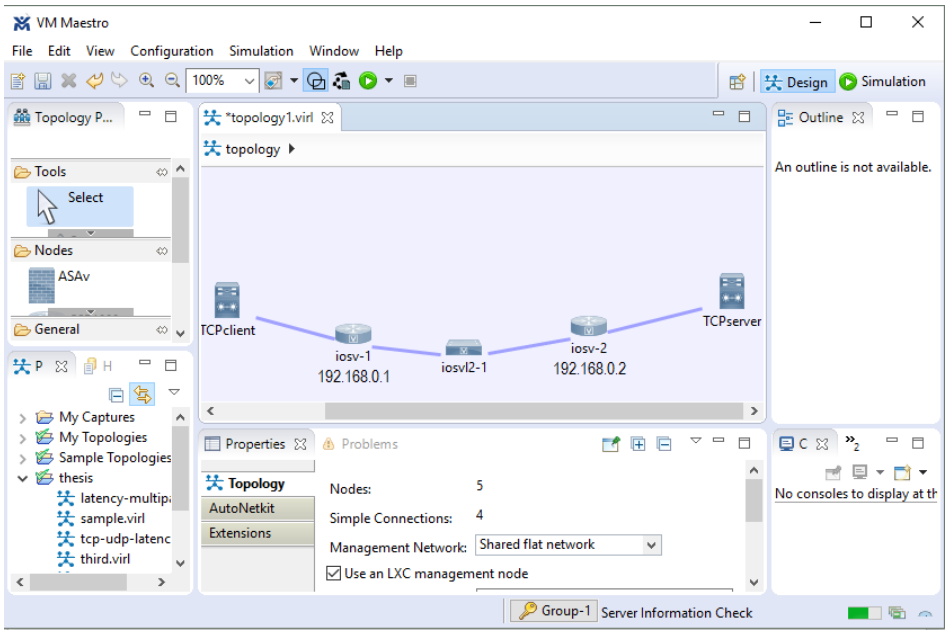


Figure B1.1: Topology used for TCP

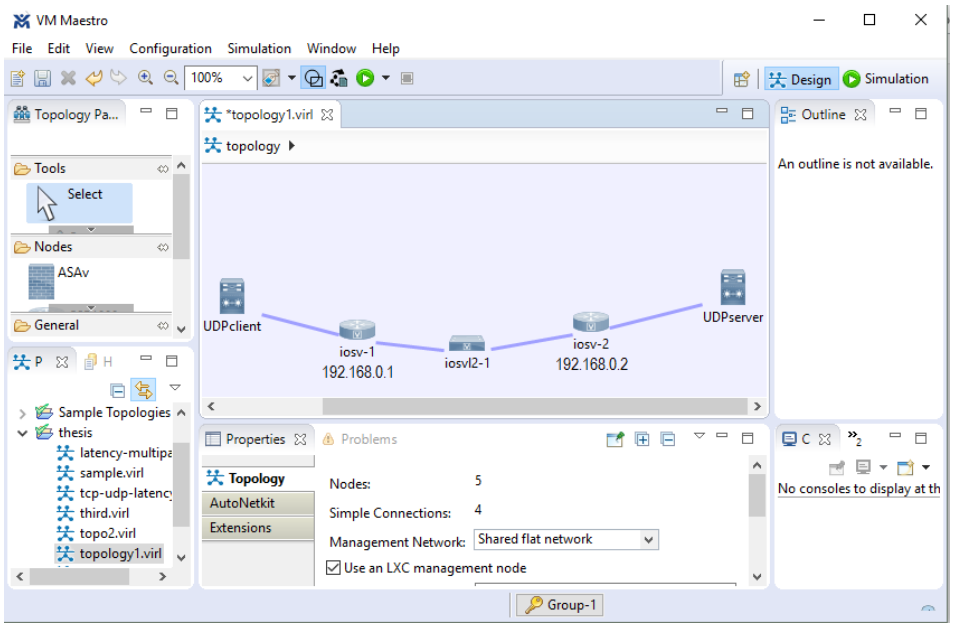


Figure B1.2: Topology used for UDP

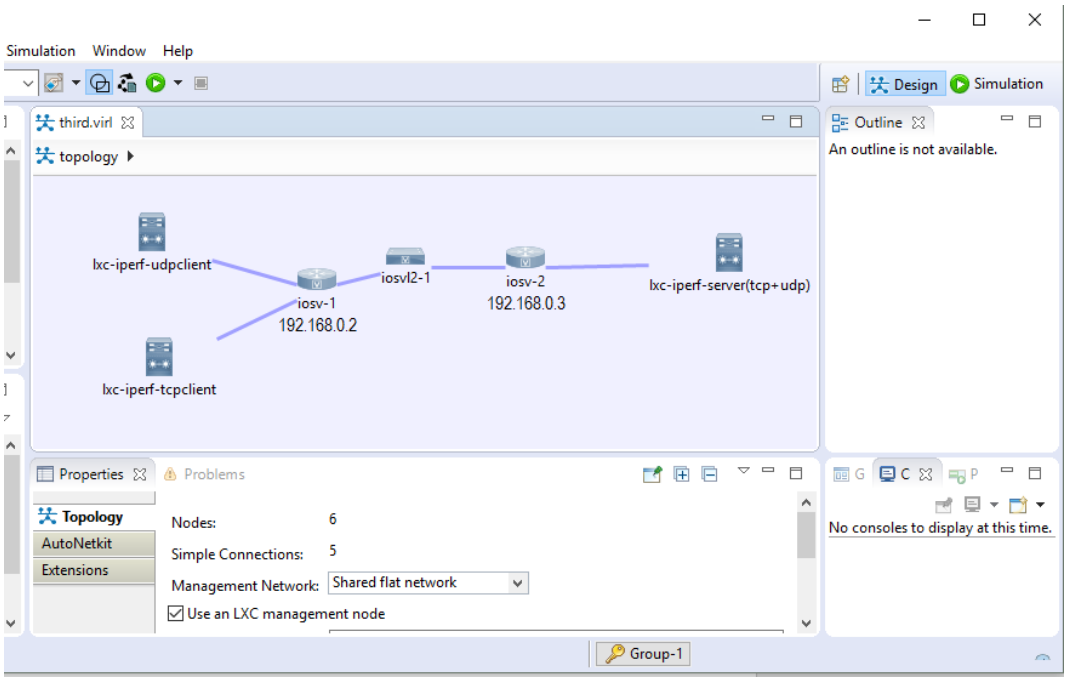


Figure B1.3: Topology used to observe the effect of TCP/UDP

B2: WAN Topology

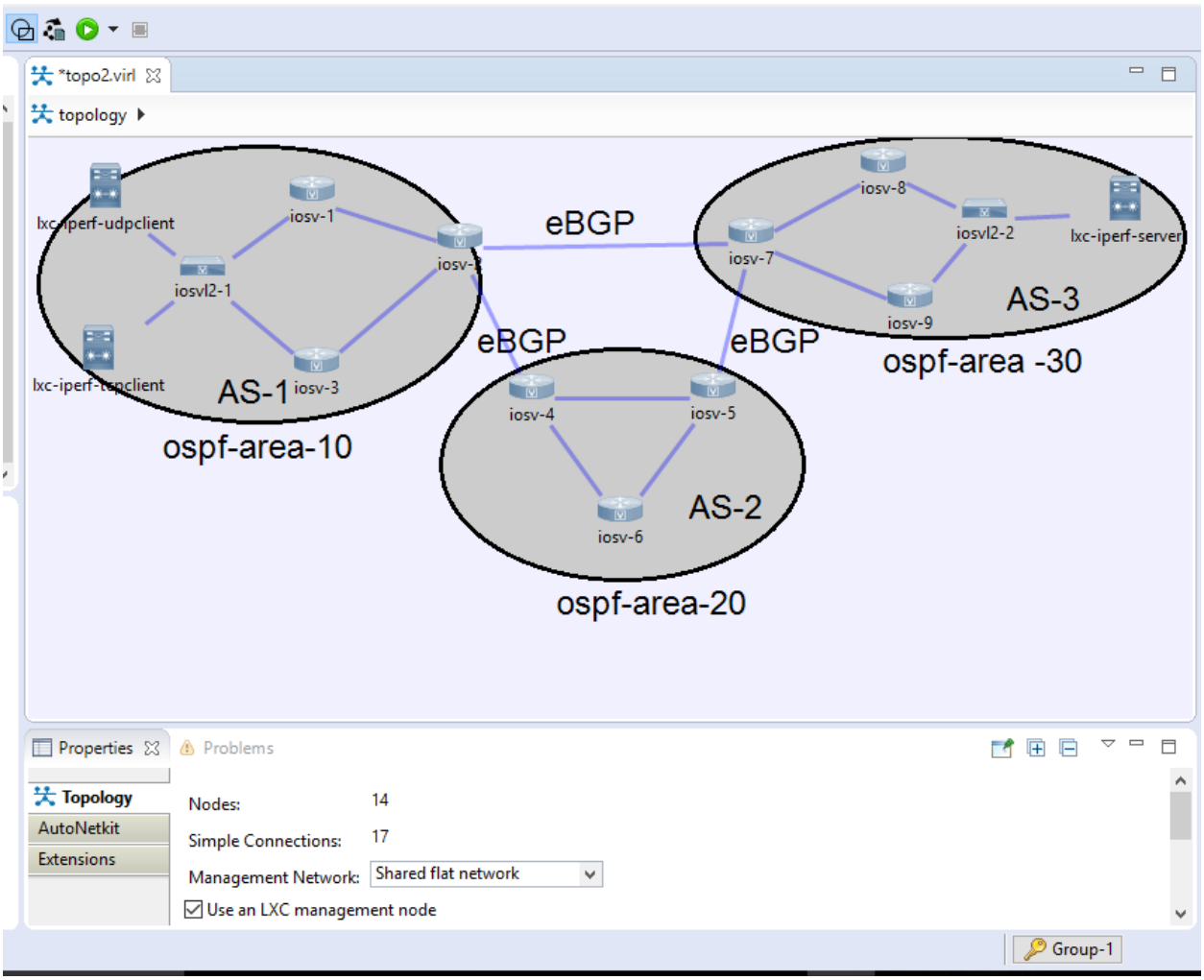


Figure B2.1: A single topology used for WAN

B3: Real Time Traffics Topology

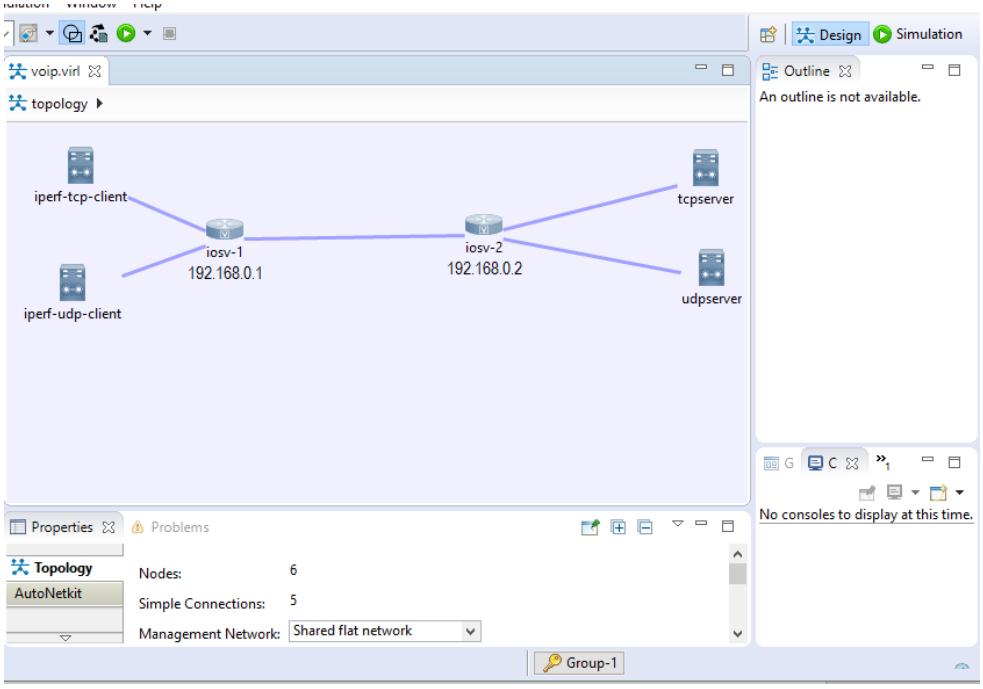


Figure B3.1: Topology used for voice and video traffic

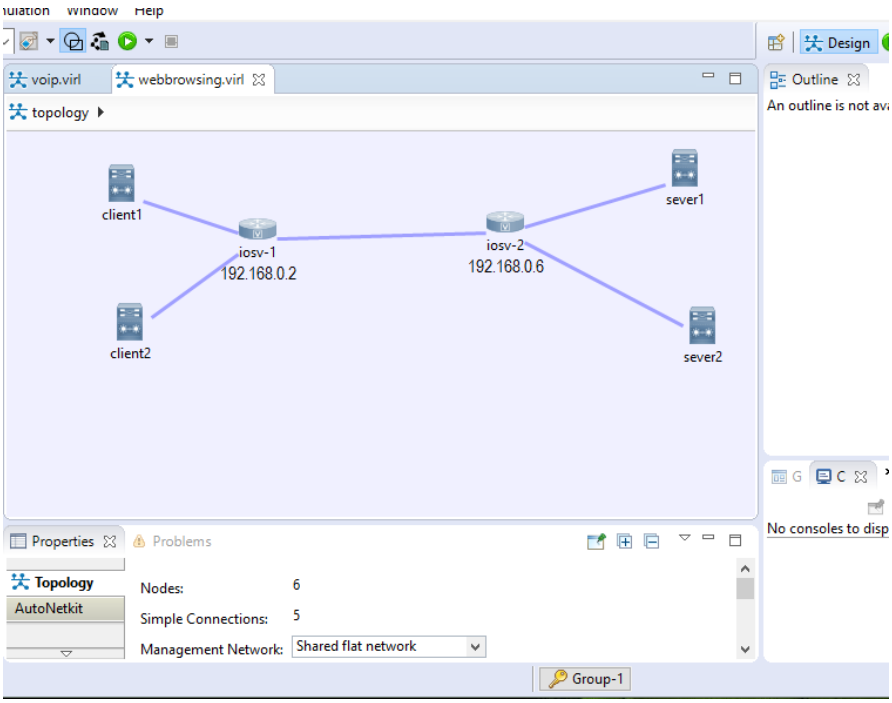


Figure B3.2: Topology used for web browsing traffic

APPENDIX C: SUPPORTING GRAPHS FOR LAN, WAN AND REAL TIME TRAFFICS

C1: Both TCP and UDP Traffic (LAN)

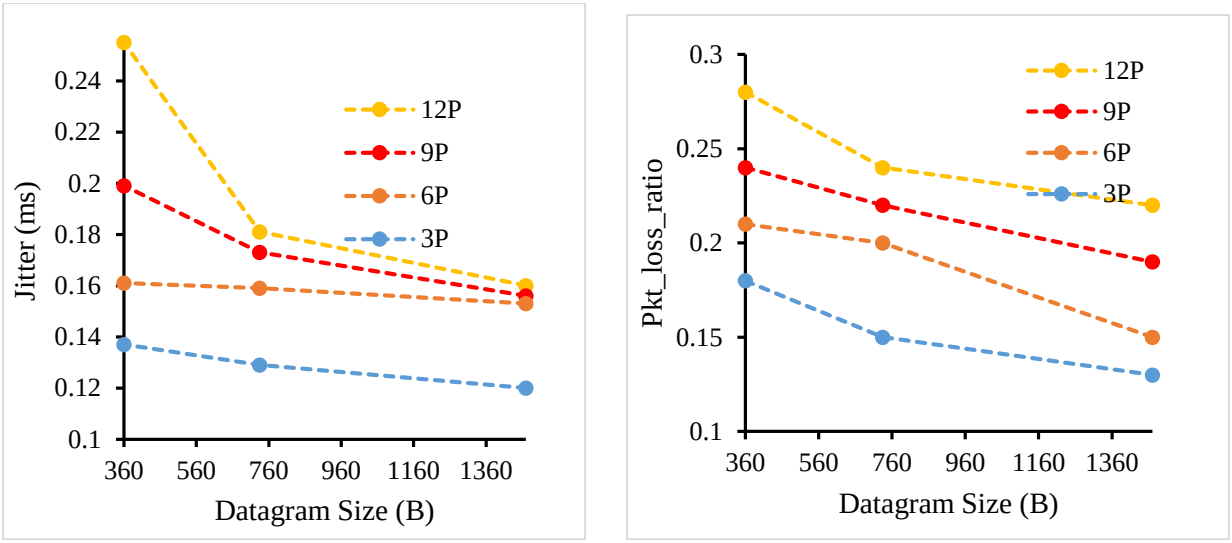


Figure C1.1 Jitter and packet loss ratio with datagram at 512Kbps (LAN)

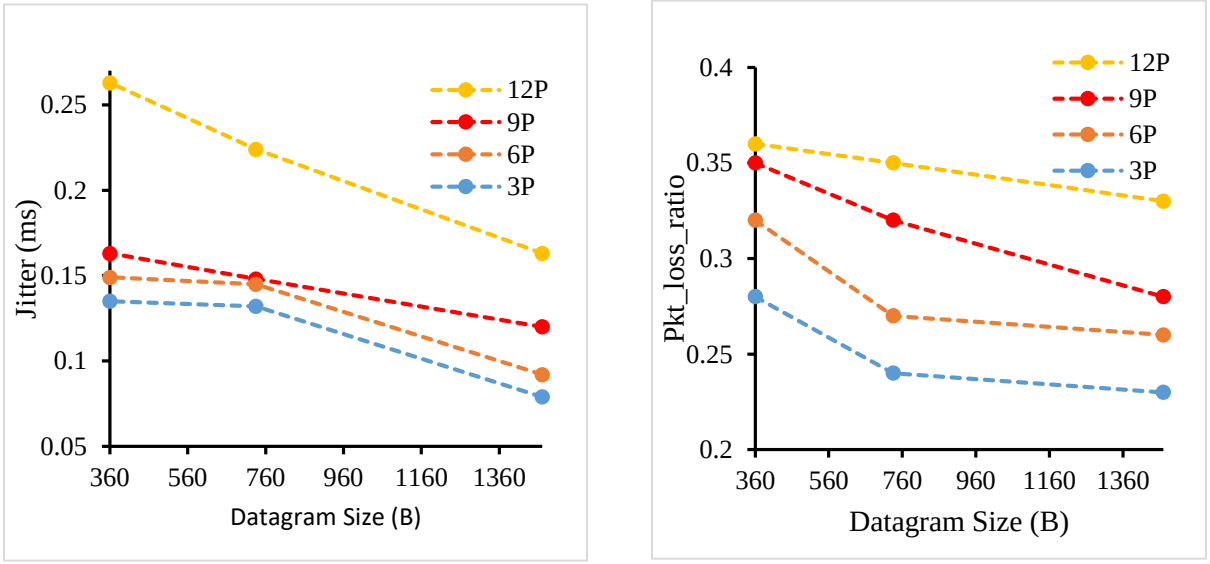


Figure C1.2 Jitter and packet loss ratio with datagram at 2Mbps (LAN)

C2: Both TCP and UDP Traffic (WAN)

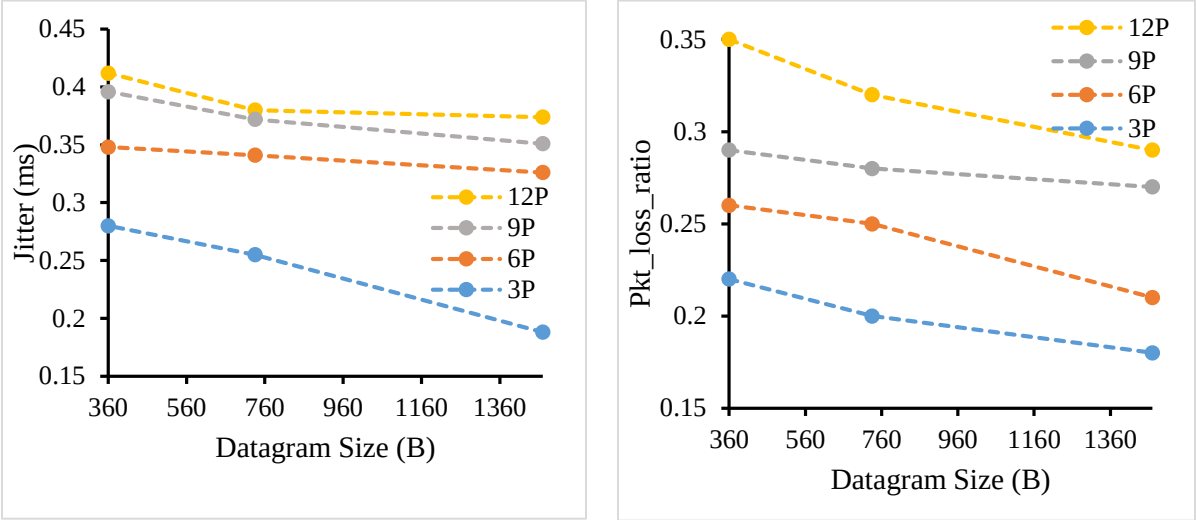


Figure C2.1 Jitter and packet loss ratio with datagram at 512Kbps (WAN)

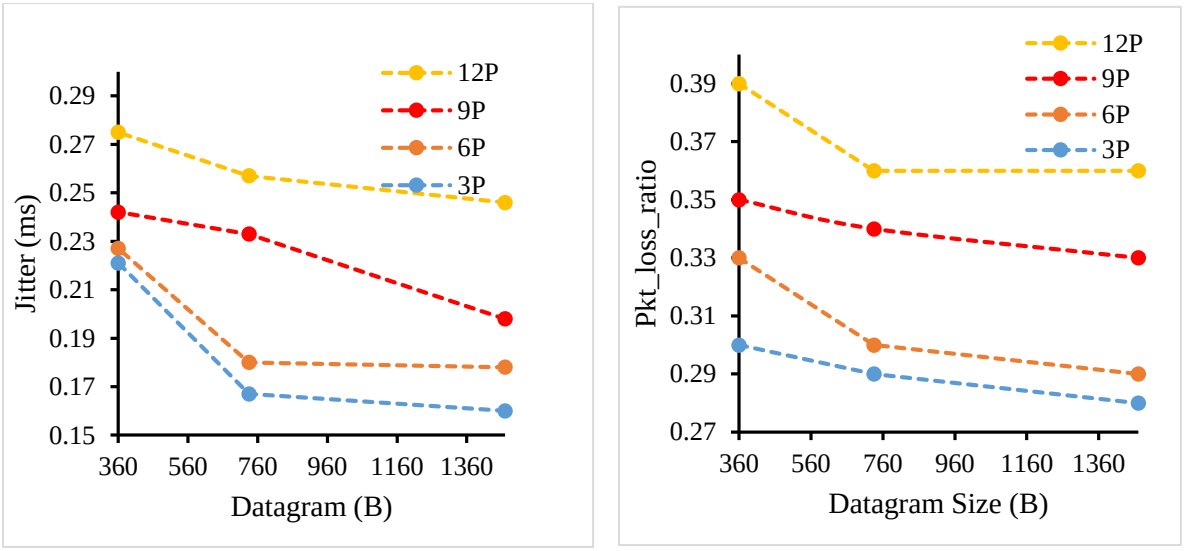


Figure C2.2 Jitter and packet loss ratio with datagram at 2Mbps (WAN)

C3: Video Traffic

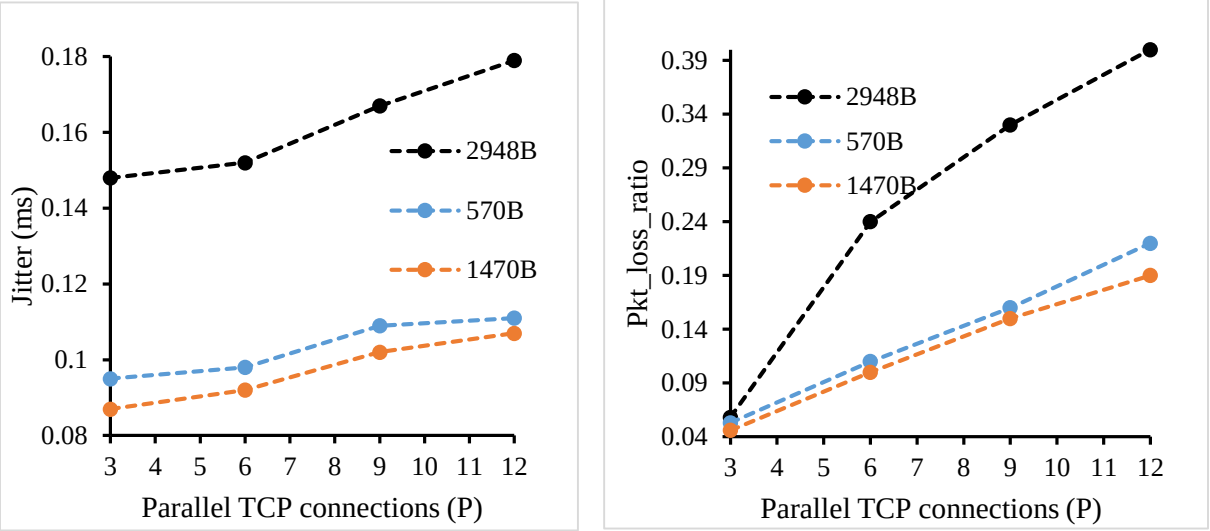


Figure C3.1 Jitter and packet loss ratio with parallel TCP at 512Kbps (Video)

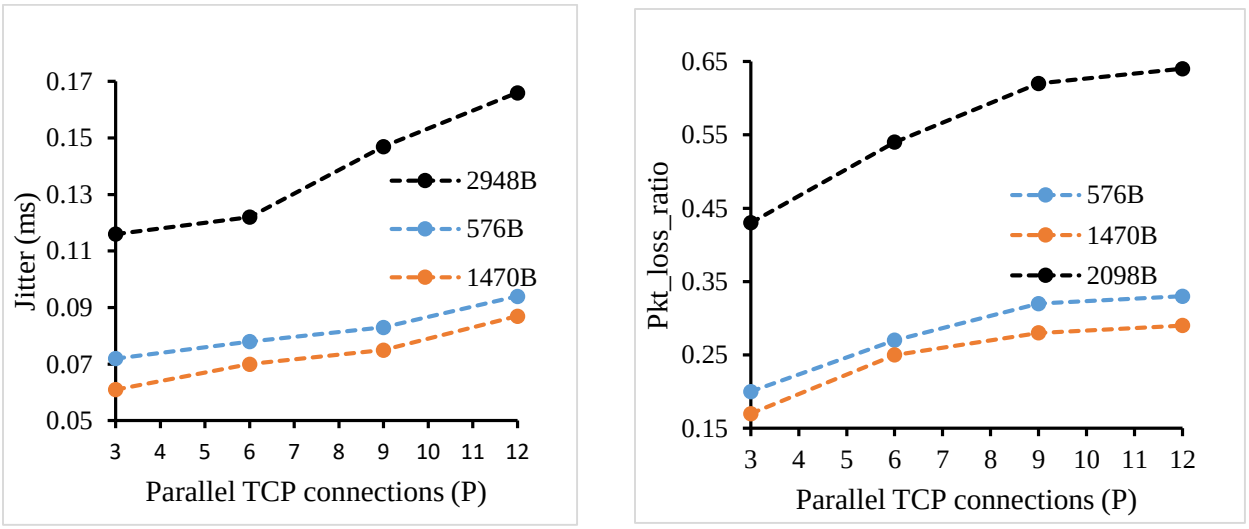
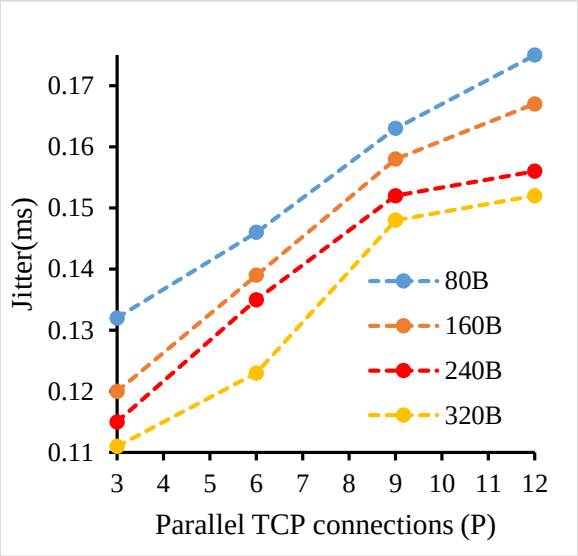


Figure C3.2: Jitter and packet loss ratio with parallel TCP at 2Mbps (Video)

C4: VoIP Traffic



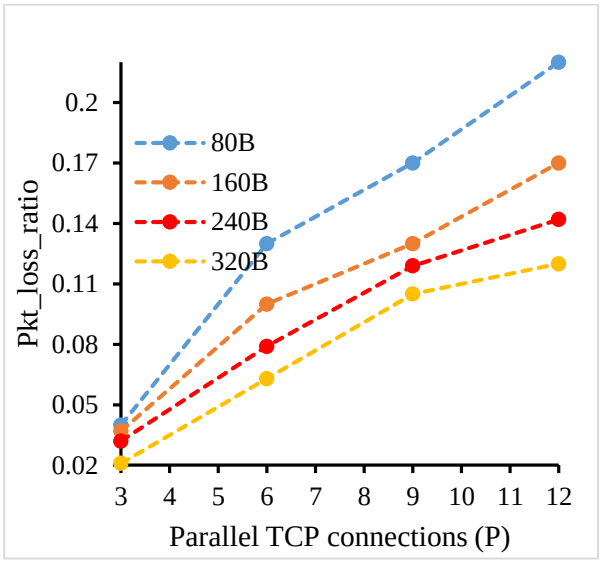
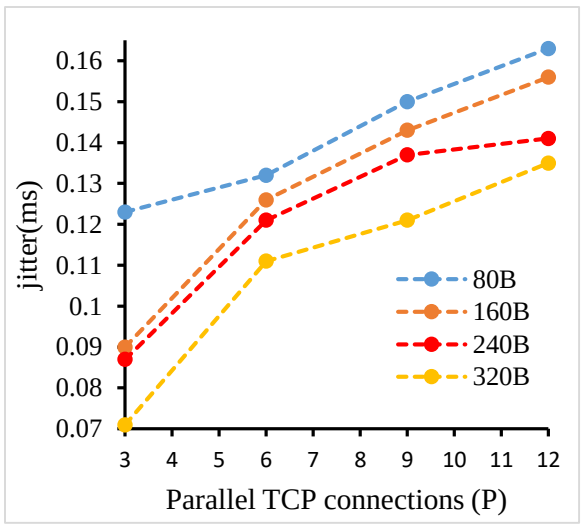


Figure C4.1: Jitter and packet loss ratio with parallel TCP at 16Kbps (Voice)



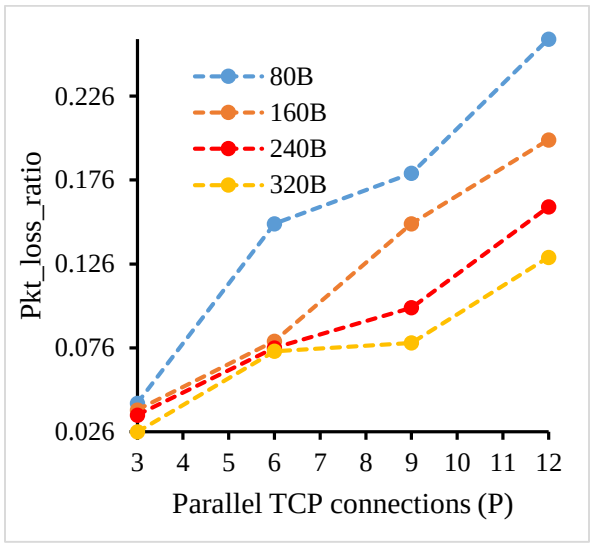


Figure C4.2: Jitter and packet loss ratio with parallel TCP at 32 Kbps (Voice)

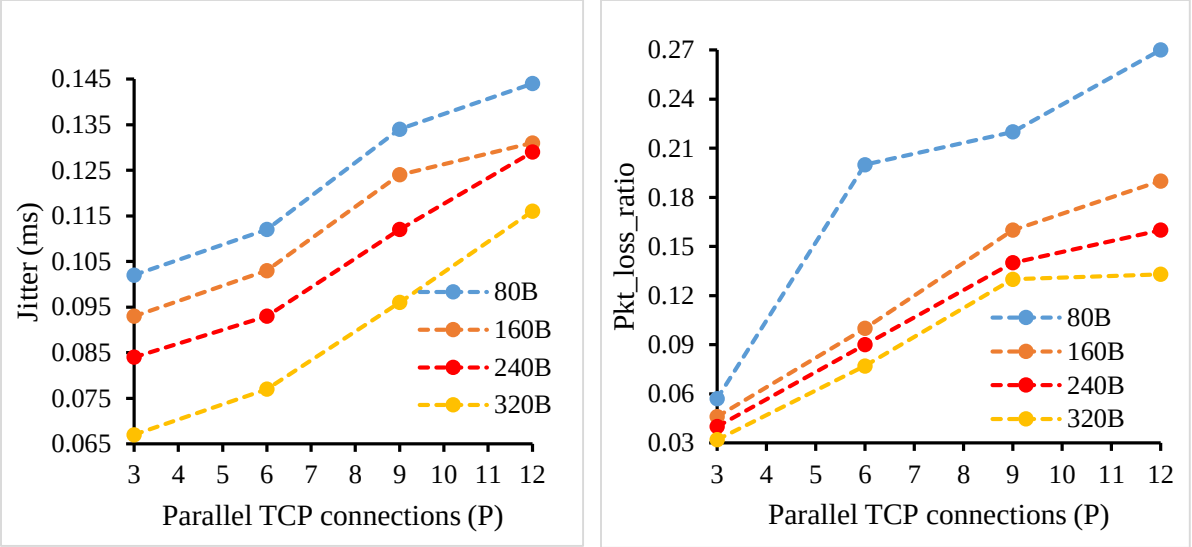


Figure C4.3: Jitter and packet loss ratio with parallel TCP at 64Kbps (Voice)

C5: Effect of Latency on Throughput (Web Browsing)

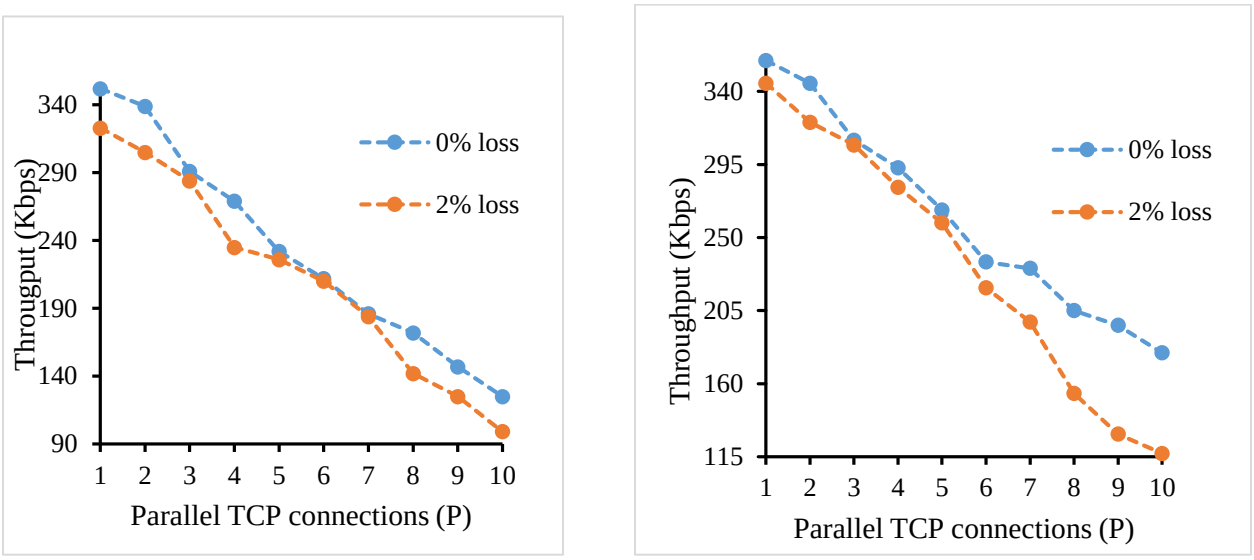


Figure C5.1: Throughput verses parallel TCP at 120ms and 100ms latency

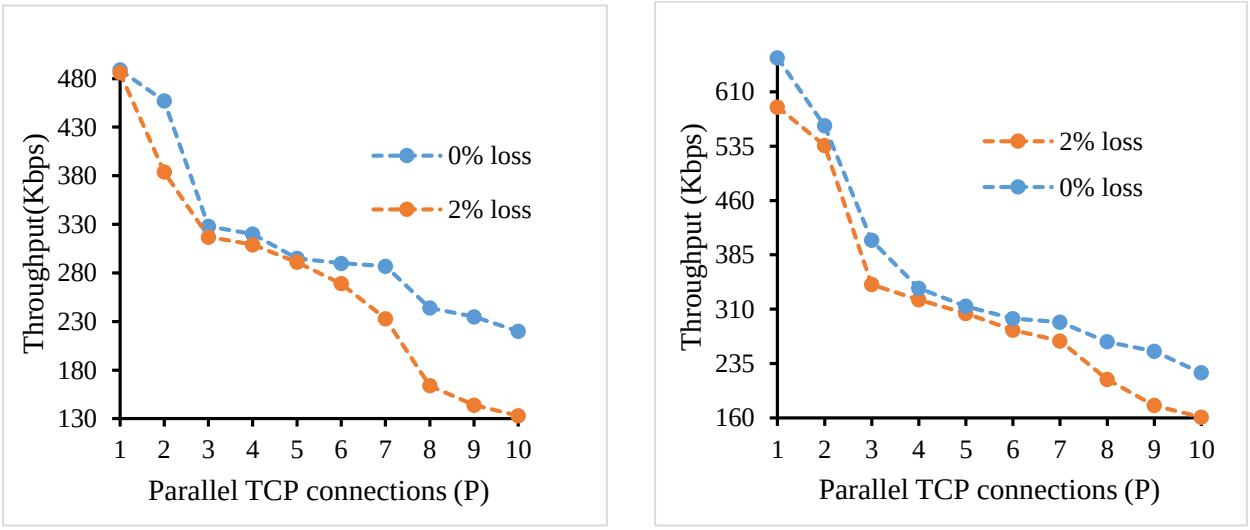


Figure C5.2: Throughput verses parallel TCP at 80ms and 60ms latency

VITA

Charitra Maharjan, a native of Lalitpur, Nepal, was born on 1 June 1989 to Mr. Gyan Bahadur Maharjan and Mrs.Ratna Maya Maharjan. After finishing his schooling from Deepmala Secondary English School in 2006, he graduated 12th from United Academy, Kumaripati Lalitpur. He studied Electronics and Communication Engineering at Advanced College of Engineering and Management in Lalitpur, Nepal, from 2009 through 2013 toward obtaining his Bachelor of Engineering. After graduating, he got approved for admission at Louisiana State University, Baton Rouge for the fall 2015. Since then, he is pursuing his Master's Program as a graduate student in Electrical Engineering Department. During his time at LSU, he has been working as a Research Assistant for Computer Science Department under the Dr. Bijaya Karki in parallel molecular dynamics simulation projects. Following receipt of his master's degree, he plans to work in the computer networking industry.