Motilal Nehru National Institute of Technology Allahabad

Project Report on

Protocols Simulation using NS-2 Simulator

Team Members

Shanawaz Ali (2020CA086)
Samarth Pratap Singh (2020CA078)
Pradipta Kumar Sahoo (2020CA054)
Vinay Kumar (2020CA103)

Submitted To

Mr. Neeraj Tyagi
Mr. Mayukh Sarkar
Mrs. Rajitha B

Undertaking

Our project titled Protocols Simulation using NS2, submitted to Computer Science and Engineering Department, Motilal Nehru National Institute of Technology, Allahabad Uttar Pradesh for the award of Master of Computer Applications, is our original work. We have neither plagiarized nor submitted the same work for award of any other degree. In case this undertaking is found incorrect, I accept that our degree may be withdrawn.

July, 2022 Allahabad

Shanawaz Ali
Samarth Pratap Singh
Pradipta Kumar Sahoo
Vinay Kumar

Acknowledgement

The making of the project "**Protocols Simulation using NS-2 Simulator**" involves contribution of every individual member of our team, whose invisible imprint can be felt on every page of this project.

We humbly express our gratitude to **Prof. Neeraj Tyagi**, our guide for the project. We would take this opportunity to thank him for guiding us with attention and care. He took immense effort to go through every step and making necessary corrections as and when needed.

Working under his guidance has indeed been a very fruitful experience for us.

Shanawaz Ali
Samarth Pratap Singh
Pradipta Kumar Sahoo
Vinay Kumar

TABLE OF CONTENTS

Title **Page No.**

Abstract

In this project we have analysed “Protocols simulation using NS2”. Our analysis is performed with respect to parameters such as packet delivery ratios, packet loss ratios, end to end delay and throughput. Graphical representation of this analysis is performed using XGraph as a tool. Visualisation of the topology is also performed using Netnam as a tool.

NS2, is simply an event-driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviours

Our focus is to analyse and study the behaviour of various topologies in wired networks using NS2.We have created Topology using TCL script. In that we have connected nodes via duplex-link. After those protocols are applied on source nodes and destination nodes respectively. And their respective traffic generators are connected to it. Now visualize the topology using netnam. Also use trace file to generate AWK script. And, finally form a graph on it, analysing various parameters to see the changes in graph.

Introduction of NS-2

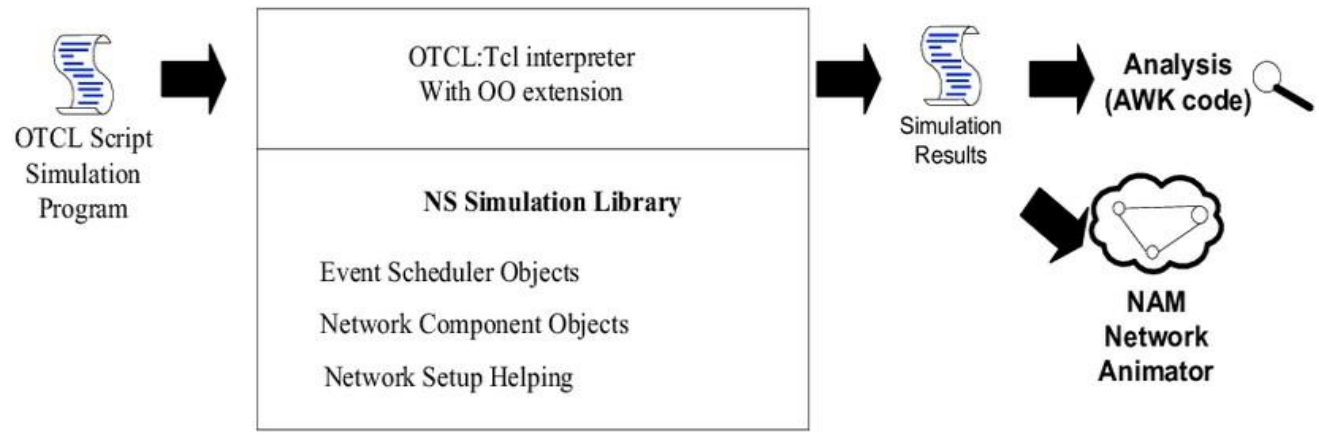
Simulation is the process of learning by doing. Whenever there is something new in the world, we try to analyze it first by examining it and in the process get to learn a lot of things. This entire course is called Simulation.

Network simulation (NS) is one of the types of simulation, which is used to simulate the networks

It provides simulation for routing and multicast protocols for both wired and wireless networks. NS is licensed for use under version 2 of the GNU (General Public License) and is popularly known as NS2. It is an object-oriented, discrete event-driven simulator written in C++ and Otcl/Tcl.

NS-2 can be used to implement network protocols such as TCP and UPD, traffic source behavior such as FTP, Telnet, Web, CBR, and VBR, router queues management mechanism such as Drop Tail, RED, and CBQ, routing algorithms, and many more. In ns2, C++ is used for detailed protocol implementation and Otcl is used for the setup. The compiled C++ objects are made available to the Otcl interpreter and in this way, the ready-made C++ objects can be controlled from the OTcl level.

Simplified User's View of NS



Installation of NS-2

Step 1: Install the basic libraries like
$] sudo apt install build-essential autoconf automake libxmu-dev

Step 2: install gcc-4.8 and g++-4.8

open the file using sudo mode
$] sudo nano /etc/apt/sources.list

Include the following line
deb http://in.archive.ubuntu.com/ubuntu bionic main universe
$] sudo apt update

$] sudo apt install gcc-4.8 g++-4.8

Step 3: Unzip the ns2 packages to home folder

$] tar zxvf ns-allinone-2.35.tar.gz

$] cd ns-allinone-2.35/ns-2.35

Modify the following make files.
~ns-2.35/Makefile.in

Change @CC@ to gcc-4.8
change @CXX@ to g++-4.8

~nam-1.15/Makefile.in
~xgraph-12.2/Makefile.in
~otcl-1.14/Makefile.in

Change in all places
@CC@ to gcc-4.8 @CPP@ or @CXX@ to g++-4.8

open the file:

~ns-2.35/linkstate/ls.h

Change at the Line no 137
void eraseAll() { erase(baseMap::begin(), baseMap::end()); } to This
void eraseAll() { this- erase(baseMap::begin(), baseMap::end()); }

All changes made

Step 4: Open a new terminal

$] cd ns-allinone-2.35/

$] ./install

Step 5 - Set the PATH Open a new Terminal,

$] gedit .bashrc

Paste the following
lines Export

PATH=$PATH:/home/yourusername/ns-allinone-2.35/bin:/home/yourusername/ns-allinone- 2.35/tcl8.5.10/unix:/home/yourusername/ns-allinone-2.35/tk8.5.10/unix

Export

LD_LIBRARY_PATH=/home/yourusername/ns-allinone-2.35/otcl- 1.14:/home/yourusername/ns-allinone-2.35/lib

Logout and Login
back OR
$] source .bashrc

Issues might occur while installing xgraph as a tool with NS-2

Solution regarding it:

Copy the PATH of xgraph and paste it in

.bashrc

$] gedit ~/.bashrc

include the following line in the

.bashrc file

alias xgraph=/home/yourusername/XGraph4.38_linux64/bin/xgraph

Goal Of This Project

- Understand how to write Tcl scripts to simulate simple network topologies and traffic patterns in NS-2.

- Analyze the trace files and understand how to evaluate the performance of networking protocols and operations using :

  XGraph tool for forming variable graphs and **Netnam** tool used for visualization .

Protocols used in the project

TCP(Transmission Control Protocol)-

TCP stands for Transmission Control Protocol a communications standard that enables application programs and computing devices to exchange messages over a network. It is designed to send packets across the internet and ensure the successful delivery of data and messages over networks.

TCP is one of the basic standards that define the rules of the internet and is included within the standards defined by the Internet Engineering Task Force (IETF). It is one of the most commonly used protocols within digital network communications and ensures end-to-end data delivery.

TCP organizes data so that it can be transmitted between a server and a client. It guarantees the integrity of the data being communicated over a network. Before it transmits data, TCP establishes a connection between a source and its destination, which it ensures remains live until communication begins. It then breaks large amounts of data into smaller packets, while ensuring data integrity is in place throughout the process.

As a result, high-level protocols that need to transmit data all use TCP Protocol. Examples include peer-to-peer sharing methods like File Transfer Protocol (FTP), Secure Shell (SSH), and Telnet. It is also used to send and receive email through Internet Message Access Protocol (IMAP), Post Office Protocol (POP), and Simple Mail Transfer Protocol (SMTP), and for web access through the Hypertext Transfer Protocol (HTTP).

UDP (User Datagram Protocol)-

User Datagram Protocol (UDP) is a communication protocol that is primarily used to establish low-latency and loss-tolerating connections between applications on the internet.

UDP speeds up transmissions by enabling the transfer of data before an agreement is provided by the receiving party. As a result, UDP is beneficial in time-sensitive communications, including voice over IP (VoIP), domain name system (DNS) lookup, and video or audio playback.

UDP is an alternative to Transmission Control Protocol (TCP). Both UDP and TCP run on top of IP and are sometimes referred to as UDP/IP or TCP/IP. However, there are important differences between the two. For example, UDP enables process-to-process communication, while TCP supports host-to-host communication.

TCP sends individual packets and is considered a reliable transport medium. On the other hand, UDP sends messages, called *datagrams*, and is considered a best-effort mode of communications. This means UDP doesn't provide any guarantees that the data will be delivered or offer special features to retransmit lost or corrupted messages.

UDP provides two services not provided by the IP layer. It provides port numbers to help distinguish different user requests. It also provides an optional checksum capability to verify that the data arrived intact.

Protocols used for creating traffic in a network: -

FTP (File Transfer Protocol)-

FTP means "File Transfer Protocol" and refers to a group of rules that govern how computers transfer files from one system to another over the internet. Businesses use FTP to send files between computers, while websites use FTP for the uploading and downloading of files from their website's servers.

FTP works by opening two connections that link the computers trying to communicate with each other. One connection is designated for the commands and replies that get sent between the two clients, and the other channel handles the transfer of data. During an FTP transmission, there are four commands used by the computers, servers, or proxy servers that are communicating. These are “send,” “get,” “change directory,” and “transfer.”

While transferring files, FTP uses three different modes: block, stream, and compressed. The stream mode enables FTP to manage information in a string of data without any boundaries between them. The block mode separates the data into blocks, and in the compress mode, FTP uses an algorithm called the Lempel-Ziv to compress the data.

CBR(Constant Bit Rate)-

Constant bit rate [CBR] Ns2 is used along with TCP and UDP to design the traffic source behavior of packets. Traffic modeling is one of the major parameter in Ns2, which uses CBR along with transport protocols. Let's know the configuration and software requirement of Ns2, which must support the proper working of CBR in Ns2.

CBR provides low latency traffic with predictable delivery characteristics for telephony and also native voice applications.

It also offer support for timing sensitive traffic.

It utilizes the full capacity of channel also to provide high quality service.

First Tcl script

First of all, you need to create a simulator object. This is done with the command

```
set ns [new Simulator]
```

Now we open a file for writing that is going to be used for the nam trace data.

```
set nf [open out.nam w]
$ns namtrace-all $nf
```

The first line opens the file 'out.nam' for writing and gives it the file handle 'nf'. In the second line we tell the simulator object that we created above to write all simulation data that is going to be relevant for nam into this file.

The next step is to add a 'finish' procedure that closes the trace file and starts nam.

```
proc finish {} {
    global ns nf
    $ns flush-
    traceclose $nf
    exec nam out.nam
    &exit 0
}
```

The next line tells the simulator object to execute the 'finish' procedure after 5.0 seconds of simulation time.

```
$ns at 5.0 "finish"
```

You probably understand what this line does just by looking at it. ns provides you with a very simple way to schedule events with the 'at' command.

The last line finally starts the simulation.

```
$ns run
```

You can actually save the file now and try to run it with 'ns example1.tcl'. You are going to get an error message like 'nam: empty trace file out.nam' though, because until now we haven't defined any objects (nodes, links, etc.) or events

You will have to use the code from this section as starting point in the other sections.

In this section we are going to define a very simple topology with two nodes that are connected by a link.

The following two lines define the two nodes. (Note: You have to insert the code in this section **before** the line '$ns run', or even better, before the line '$ns at 5.0 "finish"').

```
set n0 [$ns node]
set n1 [$ns node]
```

A new node object is created with the command '$ns node'. The above code creates two nodes and assigns them to the handles 'n0' and 'n1'.

The next line connects the two nodes.

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

This line tells the simulator object to connect the nodes n0 and n1 with a duplex link with the bandwidth 1Megabit, a delay of 10ms and a DropTail queue.

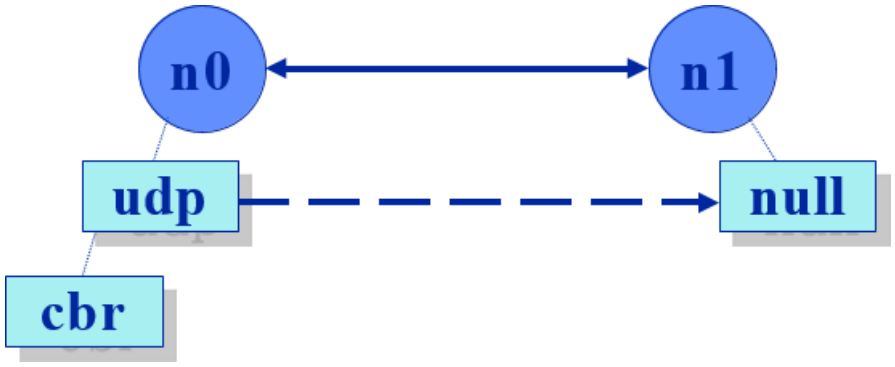
Now you can save your file and start the script with 'ns example1.tcl'. nam will be started automatically and you should see an output that resembles the picture below.

Sending data

Of course, this example isn't very satisfying yet, since you can only look at the topology, but nothing actually happens, so the next step is to send some data from node n0 to node n1. In ns, data is always being sent from one 'agent' to another. So the next step is to create an agent object that sends data from node n0, and another agent object that receives the data on node n1.

```
#Create a UDP agent and attach it to node
n0set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to
udp0set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```



These lines create a UDP agent and attach it to the node n0, then attach a CBR traffic generatot to the UDP agent. CBR stands for 'constant bit rate'. Line 7 and 8 should be self-explaining. The packetSize is being set to 500 bytes and a packet will be sent every 0.005 seconds (i.e. 200 packets per second). The next lines create a Null agent which acts as traffic sink and attach it to node n1.

```
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
```

Now the two agents have to be connected with each other.

```
$ns connect $udp0 $null0
```

And now we have to tell the CBR agent when to send data and when to stop sending. Note: It's probably best to put the following lines just before the line '$ns at 5.0 "finish"'.

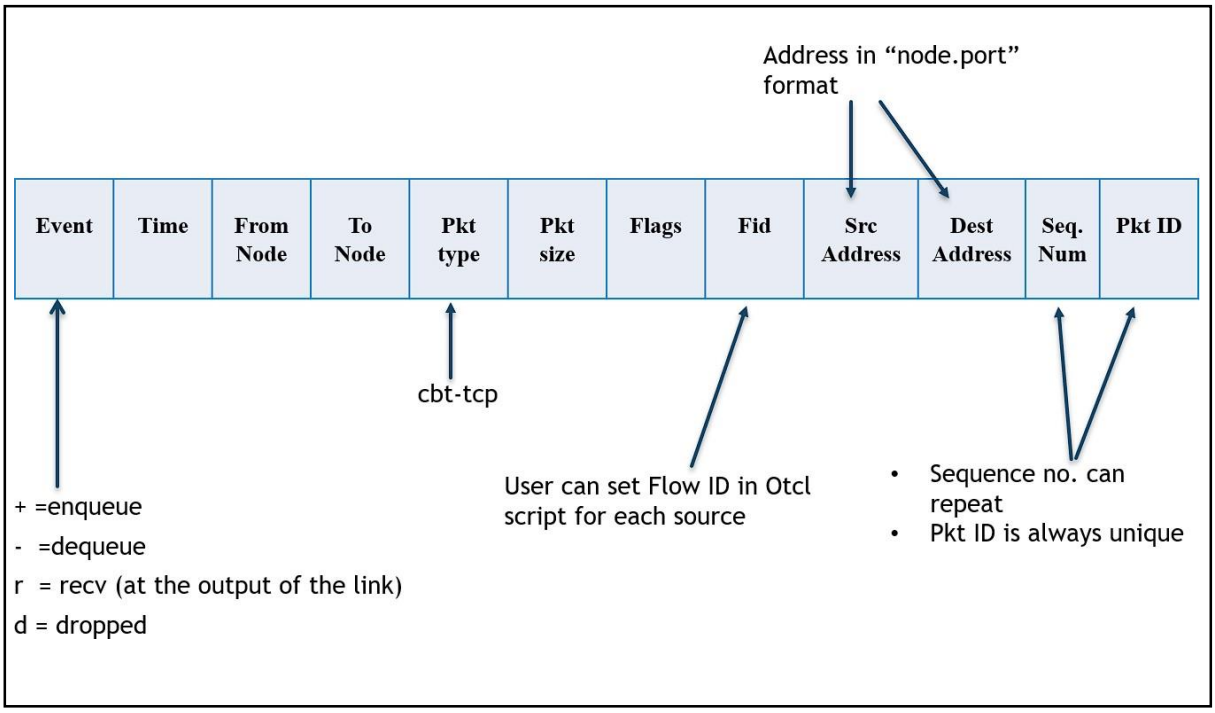
```
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

This code should be self-explaining again.

Now you can save the file and start the simulation again. When you click on the 'play' button in the nam window, you will see that after 0.5 simulation seconds, node 0 starts sending data packets to node 1. You might want to slow nam down then with the 'Step' slider.

Trace File

The output file has the following format:



Most important informations in the trace record:

- Each line of the trace file corresponds to one event of packet activity
- The **first character** of one trace record indicates the action:
  - +: a packet arrives at a queue (it may or may not be dropped !)
  - -: a packet leaves a queue
  - **r**: a packet is received into a queue (buffered)
  - **d**: a packet is dropped from a queue
- The **last 4 fields** contains:
  - **source id**: id of sender (format is: **x.y = node x and transport agent y**)
  - **receiver id**: id of receiver (format is: **x.y = node x and transport agent y**)
  - **sequence number**: useful to determine if packet was new or a retransmission
  - **packet id**: is always increasing - usefule to determine the number of packets lossed.
- The **packet size** contains the number of bytes in the packet

Data Types of values:

Event	Abbreviation	Type	Value
Normal Event	r: Receive	%g %d %d %s %d %s %d %d.%d %d.%d %d %d	
	d: Drop	double	Time
	e: Error	int	(Link-layer) Source Node
	+: Enqueue	int	(Link-layer) Destination Node
	-: Dequeue	string	Packet Name
		int	Packet Size
		string	Flags
		int	Flow ID
		int	(Network-layer) Source Address
		int	Source Port
		int	(Network-layer) Destination Address
		int	Destination Port
		int	Sequence Number
		int	Unique Packet ID

Sample Output:

+ 1.84375 0 2 cbr 210 --------0 0.0 3.1 225 610
- 1.84375 0 2 cbr 210-------- 0 0.0 3.1 225 610
r 1.84471 2 1 cbr 210 ------- 1 3.0 1.0 195 600
r 1.84566 2 0 ack 40-------- 2 3.2 0.1 82 602
+ 1.84566 0 2 tcp 1000-------- 2 0.1 3.2 102 611
- 1.84566 0 2 tcp 1000 ------- 2 0.1 3.2 102 611
r 1.84609 0 2 cbr 210 ------- 0 0.0 3.1 225 610

AWK Scripts for NS2 to process data from Trace Files

AWK Scripts are very good in processing the data from the log (trace files) which we get from NS2. If you want to process the trace file manually, here is the detail Here is a sample of trace file from NS2 (However ns2 supports a new type of trace file also), but this post will make you understand the old trace format only.

AWK Script

AWK is a high level programming language which is used to process text files,named after its three orginal author's name:

A : Alfred **A**ho

W : Peter **W**einberger

K : Brian **K**ernighan

AWK Scripts are very good in processing the data from the log (trace files) which we get from NS2. If you want to process the trace file manually.

AWK PROGRAM STRUCTURE

Awk program structure contains mainly three parts;

1. Begin

2. Content

3. End

BEGIN {<initialization>}

<pattern1> {<actionSet1>}

<pattern2> {<actionSet2>}

. . .

END {<final finalActionSet>}

BEGIN : Begin deals with what to be executed prior to text file processing,normally which is used to initialize variable values or constants.

CONTENT : Script which process the text file. In this part, AWK moves lines by lines (i.e., records by records) and executes the <actionSet> if the current line match with the pattern. The actions repeats until AWK reaches the end of the file.

END : This part explains what to be executed after the text file processing ie. what to print on the terminal or to show output in terminal.

EXECUTION

Awk has two types of execution;

1) Plain Execution.

2) Match and Execute.

Plain Execution : Simply AWK statements.

Match and execute : The second type of execution is "Match and Execute", when executes plain execution statements only if the current line (of the input text file) matches with a predefined pattern. The pattern can be either : 1. Logical Expression 2. Regular Expression.

Factors on which we will going to analyze thevarious cases of topologies

Packet Delivery Ratio-

Packet delivery ratio is a very important factor to measure the performance of routing protocol in any network. The performance of the protocol depends on various parameters chosen for the simulation. The major parameters are packet size, no of nodes, transmission range and the structure of the network. The packet delivery ratio can be obtained from the total number of data packets arrived at destinations divided by the total data packets sent from sources. In other words Packet delivery ratio is the ratio of number of packets received at the destination to the number of packets sent from the source. The performance is better when packet delivery ratio is high.

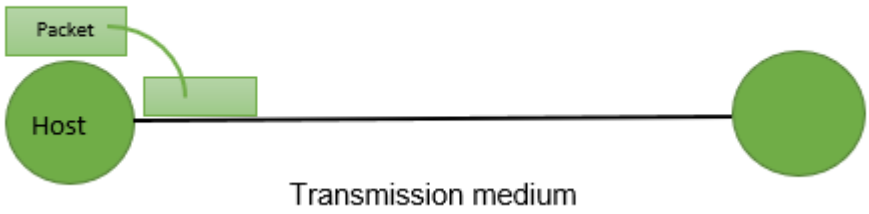
$$\text{Packet Delivery Ratio} = \frac{\Sigma(\text{Total packets received by all destination node})}{\Sigma(\text{Total packets send by all source node})} \text{----------(i)}$$

End-to-End Delay-

Average End-to-end delay is the time taken by a packet to route through the network from a source to its destination. The average end-to-end delay can be obtained computing the mean of end-to-end delay of all successfully delivered messages. Therefore, end–to-end delay partially depends on the packet delivery ratio. As the distance between source and destination increases, the probability of packet drop increases. The average end-to-end delay includes all possible delays in the network i.e. buffering route discovery latency, retransmission delays at the MAC, and propagation and transmission delay.

We have the following types of delays in computer networks:

1. Transmission Delay:
The time taken to transmit a packet from the host to the transmission medium is called Transmission delay.



For example, if bandwidth is 1 bps (every second 1 bit can be transmitted onto the transmission medium) and data size is 20 bits then what is the transmission delay? If in one second, 1 bit can be transmitted. To transmit 20 bits, 20 seconds would be required.

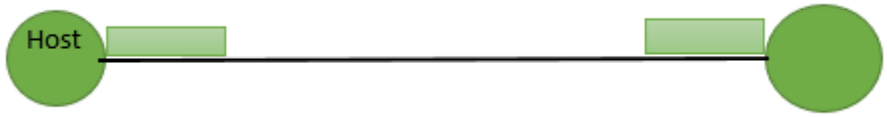
Let B bps is the bandwidth and L bit is the size of the data then transmission delay is,

$$Tt = L/B$$

This delay depends upon the following factors:

- If there are multiple active sessions, the delay will become significant.
- Increasing bandwidth decreases transmission delay.
- MAC protocol largely influences the delay if the link is shared among multiple devices.
- Sending and receiving a packet involves a context switch in the operating system, which takes a finite time.

2. Propagation delay:
After the packet is transmitted to the transmission medium, it has to go through the medium to reach the destination. Hence the time taken by the last bit of the packet to reach the destination is called propagation delay.



Factors affecting propagation delay:

1. **Distance –** It takes more time to reach the destination if the distance of the medium is longer.
2. **Velocity –** If the velocity(speed) of the signal is higher, the packet will be received faster.

$$Tp = Distance / Velocity$$

3. Queueing delay:
Let the packet is received by the destination, the packet will not be processed by the destination immediately. It has to wait in a queue in something called a buffer. So the amount of time it waits in queue before being processed is called queueing delay.

In general, we can't calculate queueing delay because we don't have any formula for that.

This delay depends upon the following factors:

- If the size of the queue is large, the queuing delay will be huge. If the queue is empty there will be less or no delay.
- If more packets are arriving in a short or no time interval, queuing delay will be large.
- The less the number of servers/links, the greater is the queuing delay.

4. Processing delay:
Now the packet will be taken for the processing which is called processing delay.

Time is taken to process the data packet by the processor that is the time required by intermediate routers to decide where to forward the packet, update TTL, perform header checksum calculations.

It also doesn't have any formula since it depends upon the speed of the processor and the speed of the processor varies from computer to computer.

Note: Both queueing delay and processing delay doesn't have any formula because they depend on the speed of the processor
This delay depends upon the following factors:

- It depends on the speed of the processor.

For study purpose we ignore processing and queuing delays and focus on propagation and transmission delays

Packet Loss

Packet Loss is the ratio of the number of packets that never reached the destination to the number of packets originated by the source. Mathematically it can be shown as equation (iii).
PL= (nSentPackets- nReceivedPackets)/ nSentPackets (iii)
Where nReceivedPackets = Number of received packets
nSentPackets = Number of sent packets

Packet Loss Ratio

Packet Loss Ratio is the ratio of the number of packets that never reached the destination to the number of packets originated by the source. Mathematically it can be shown as

PLR = (nSentPackets- nReceivedPackets)/ nSentPackets * 100
Where nReceivedPackets = Number of received packets
nSentPackets = Number of sent packets

Average Throughput

It is the average of the total throughput. It is also measured in packets per unit TIL. TIL is Time Interval Length. Mathematically it can be shown
Average Throughput = (recvdSize/(stopTimestartTime))*(8/1000)

Where recvdSize = Store received packet's size
stopTime = Simulation stop time
startTime = Simulation start time

XGraph

The xgraph program draws a graph on an X display given data read from either data files or from standard input if no files are specified. It can display up to 64 independent data sets using different colors and/or line styles for each set. It annotates the graph with a title, axis labels, grid lines or tick marks, grid labels, and a legend. There are options to control the appearance of most components of the graph.

A data set consists of an ordered list of points of the form “directive X Y”. For directive “draw”, a line will be drawn between the previous point and the current point. Specifying a “move” directive tells xgraph not to draw a line between the points. “draw” is the default directive. The name of a data set can be specified by enclosing the name in double quotes.

The interface used to specify the size and location of this window depends on the window manager currently in use. Once the window has been opened, all of the data sets will be displayed graphically with a legend in the upper right corner of the screen.

Xgraph also presents three control buttons in the upper left corner of each window: Hardcopy, Close and about xgraph accepts a large number of options most of which can be specified either on the command line, in the user’s .Xdefaults or .Xresources file, or in the data files themselves.

Multiple Sources with Multiple destinations wired networks

The very basic steps involved to run and analyze a program

Step 1: go to the folder where the .tcl file placed

Step 2: open terminal in that folder

Step 3: run ns <filename>.tcl (this wll run the tcl script)
after running this command our trace and nam file would be created with the name as provide in the script

Step 4: run nam <filename>.nam (this will show the graphical repesention)

Step 5(Running awk scripts): awk -f AwkFile.awk TaceFile.tr > points
the output of the awk script would be stored in points file (here)

Step 6: xgraph points (this will show us the respective graph)

Case – I

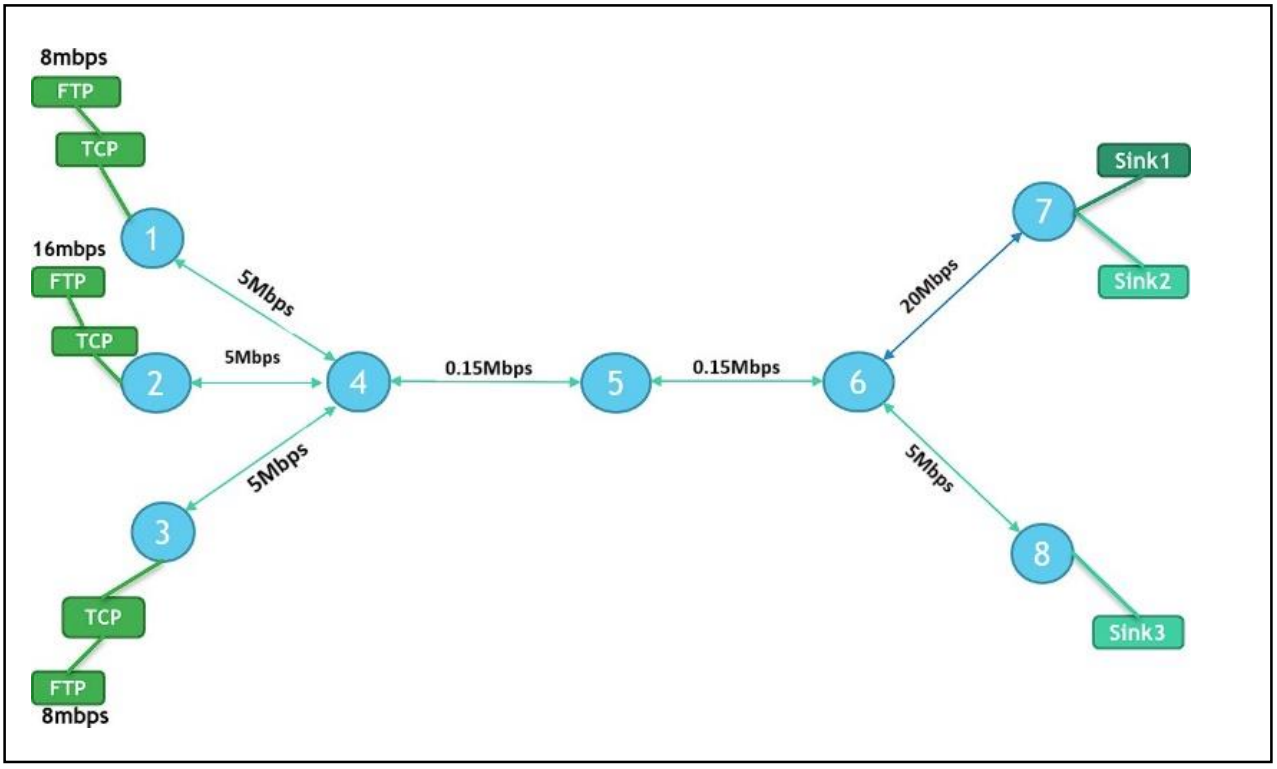
All nodes 1,2,3 are sending packets using TCP protocols with the help of FTP as traffic source generator

The bandwidth of each link is mentioned just above the link in each node-to-node connection

All the sources with their destinations

1. Node 1 is the source and node 7 is its destination with transmission speed of 8mbps
2. Node 2 is the source and node 7 is its destination with transmission speed of 16mbps
3. Node 3 is the source and node 8 is its destination with transmission speed of 8mbps

Node 1 is running throughout the simulation
Node 2 and Node are running from 5th second till 10th seconds only



TCL script

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

# Define different colors
# for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
$ns color 3 Yellow

# Open the NAM trace file
set nf [open p1_disp.nam w]
$ns namtrace-all $nf

set tracefile [open p1_log.tr w]
$ns trace-all $tracefile

puts "---------------------------------------------------------------------------------- "
puts " The NAM and Log Trace files have been created. Do run the following
commands in order:- "

puts ""
puts " For viewing the simulation : nam p1_disp.nam"
puts ""

puts "----------------------------------------------------------------------------------"
puts " 1.For plotting Packet Delivery Ratio :-"
puts "----------------------------------------------------------------------------------"
puts " A) Run : awk -f pdr.awk -v src=<src> -v dest=<dest>
p1_log.tr>p1_pdr"
puts "  B) Run : xgraph p1_pdr"

puts ""
puts "---------------------------------------------------------------------------------- "
puts " 2.For plotting Packet Loss Ratio :-"
puts "----------------------------------------------------------------------------------"
puts " A) Run : awk -f plr.awk -v src=<src> -v dest=<dest>
p1_log.tr>p1_plr"
puts "  B) Run : xgraph p1_plr"
puts ""
puts "----------------------------------------------------------------------------------"
puts " 3.For plotting End to End Delay :-"
puts "----------------------------------------------------------------------------------"
puts "  A) Run : awk -f e2e_delay.awk -v src=<src> -v dest=<dest>
p1_log.tr>p1_e2e_delay"
puts " B) Run : xgraph p1_e2e_delay"
puts ""
puts "----------------------------------------------------------------------------------"
puts " "
```

```
# Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace

        # Close the NAM trace file
        close $nf

        exec awk -f throughput.awk -v dest=6 p1_log.tr &
        after 300
      puts "\n"
        exit 0
}
# Create four nodes

set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]

# Create links between the nodes
$ns duplex-link $n1 $n4 5Mb 10ms DropTail
$ns duplex-link $n2 $n4 5Mb 10ms DropTail
$ns duplex-link $n3 $n4 5Mb 10ms DropTail
$ns duplex-link $n4 $n5 0.15Mb 30ms DropTail
$ns duplex-link $n5 $n6 0.15Mb 20ms DropTail
$ns duplex-link $n6 $n7 20Mb 10ms DropTail
$ns duplex-link $n6 $n8 5Mb 10ms DropTail

# Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n4 $n5 10
$ns queue-limit $n5 $n6 10

# Give node position (for NAM)
$ns duplex-link-op $n1 $n4 orient right-down
$ns duplex-link-op $n2 $n4 orient right
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n5 $n6 orient right
$ns duplex-link-op $n6 $n7 orient right-up
$ns duplex-link-op $n6 $n8 orient right-down
```

```
# Monitor the queue for link (n4-n5). (for NAM)
$ns duplex-link-op $n4 $n5 queuePos 0.5


# Setup first TCP connection
set tcp1 [new Agent/TCP]
$tcp1 set class_ 2
$ns attach-agent $n1 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $n7 $sink1
$ns connect $tcp1 $sink1
$tcp1 set fid_ 1

# Setup first FTP over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP
$ftp1 set packet_size_ 5000
$ftp1 set rate_ 8mb
$ftp1 set random_ false
# Setup second TCP connection
set tcp2 [new Agent/TCP]
$tcp2 set class_ 2
$ns attach-agent $n2 $tcp2


set sink2 [new Agent/TCPSink]
$ns attach-agent $n7 $sink2
$ns connect $tcp2 $sink2
$tcp2 set fid_ 2

# Setup second FTP over TCP connection
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set type_ FTP
$ftp2 set packet_size_ 10000
$ftp2 set rate_ 16mb
$ftp2 set random_ false

# Setup third TCP connection
set tcp3 [new Agent/TCP]
$tcp3 set class_ 2
$ns attach-agent $n3 $tcp3
```

```
set sink3 [new Agent/TCPSink]
$ns attach-agent $n7 $sink3
$ns connect $tcp3 $sink3
$tcp3 set fid_ 3

# Setup third FTP over TCP connection
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
$ftp3 set type_ FTP
$ftp3 set packet_size_ 50000
$ftp3 set rate_ 8mb
$ftp3 set random_ false

# Schedule events for the FTP agents
$ns at 0.1 "$ftp1 start"
$ns at 24 "$ftp1 stop"
$ns at 5 "$ftp2 start"
$ns at 10.0 "$ftp2 stop"
$ns at 5.0 "$ftp3 start"
$ns at 10.0 "$ftp3 stop"

# Call the finish procedure after
# 5 seconds of simulation time
$ns at 25.0 "finish"

# Run the simulation
$ns run
```

Terminal Output

This is the output of our tcl script as shown in the terminal below contains

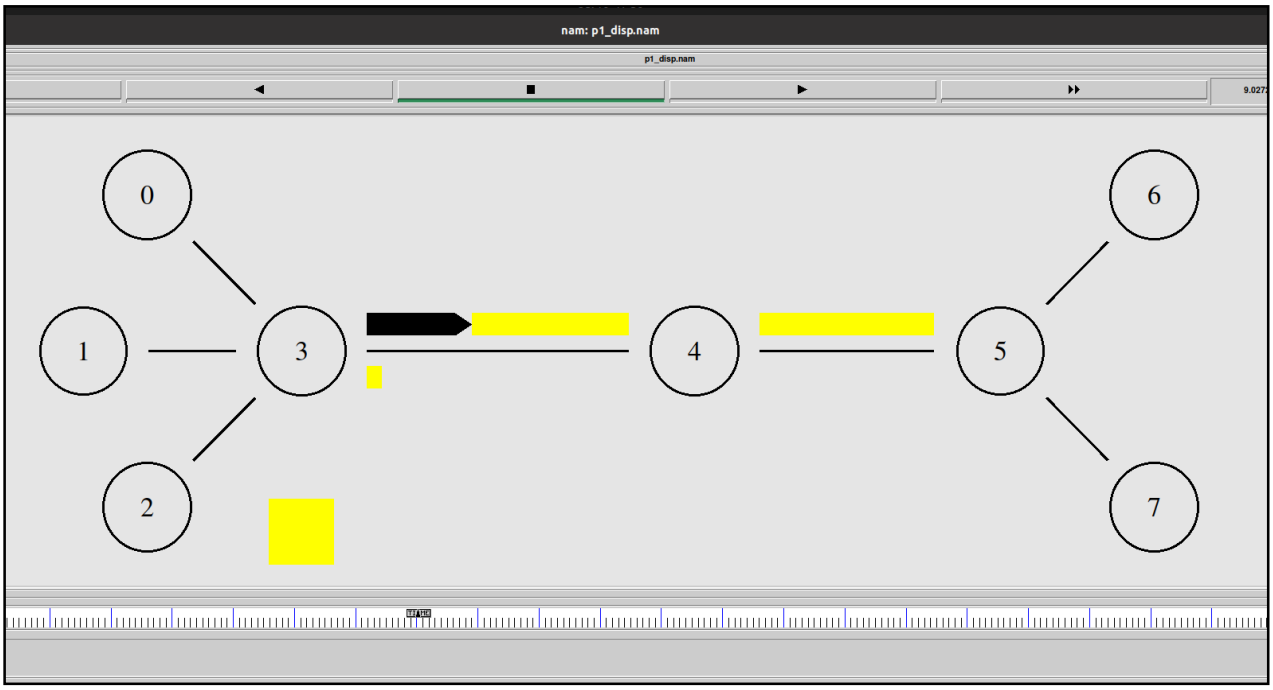
- The name of the trace file generated which can be used for further analyses (p1_log.tr).
- The name of the NAM file which can we used to see the visual representation of our topology (p1_disp.nam)
  To see nam output just type **nam p1_disp.nam** in the terminal.
- Three factors which can be analyzed using the trace file with the help of awk script and xgraph are
  - Packer delivery ratio
  - Packet Loss Ratio
  - End to End Delay
- Each awk command has a src (source node) and a dest (destination node) variable which is used to analyze factor just between those two nodes.
  The output of the awk script is stored in a file which is then further used to plot the graphs

  For example, if we want to calculate packet delivery ration from node 2 to node 7 then run **awk -f pdr.awk -v src=2 -v dest=7 p1_log.tr>p1_pdr**

  In the terminal after this p2_pdr file would be generated which is used to plot the graph by running **xgraph p1_pdr** in the terminal
- Throughput from the source to the destination

NAM Output

Here back and yellow color block represent flow of the packets.
The yellow-colored squared block is a packet which is being dropped from the node 3.



Packet Delivery Ratio

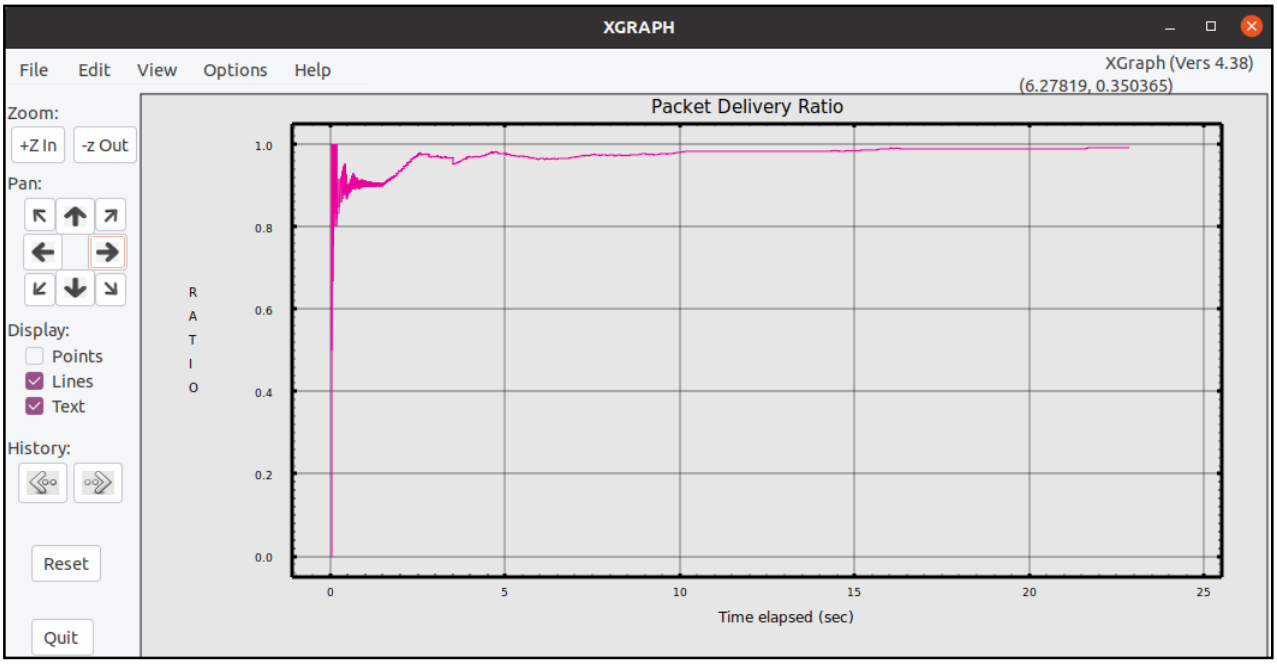
Here,

x-axis represents time elapsed in simulation
y-axis represents the ratio

Since we know packet delivery is the ratio of number of packets sent from the source to the numbers of the packet received to the destination so its value would be always between 0-1.

Here, Node 1 is sending packets for all the time but as soon as Node 2 and Node 3 start sending packets there is some decrease in ratio after 5th second which is then restored as those node's (2 and 3) traffic was closed after 10th second.

The PDR (Packet Delivery Ratio) is almost equal to 1 here because we have used TCP protocol for all the nodes and which indeed is a reliable protocol and hence there is very less packet loss.



Packet Lost Ratio

Here,

x-axis represents time elapsed in simulation
y-axis represents the ratio

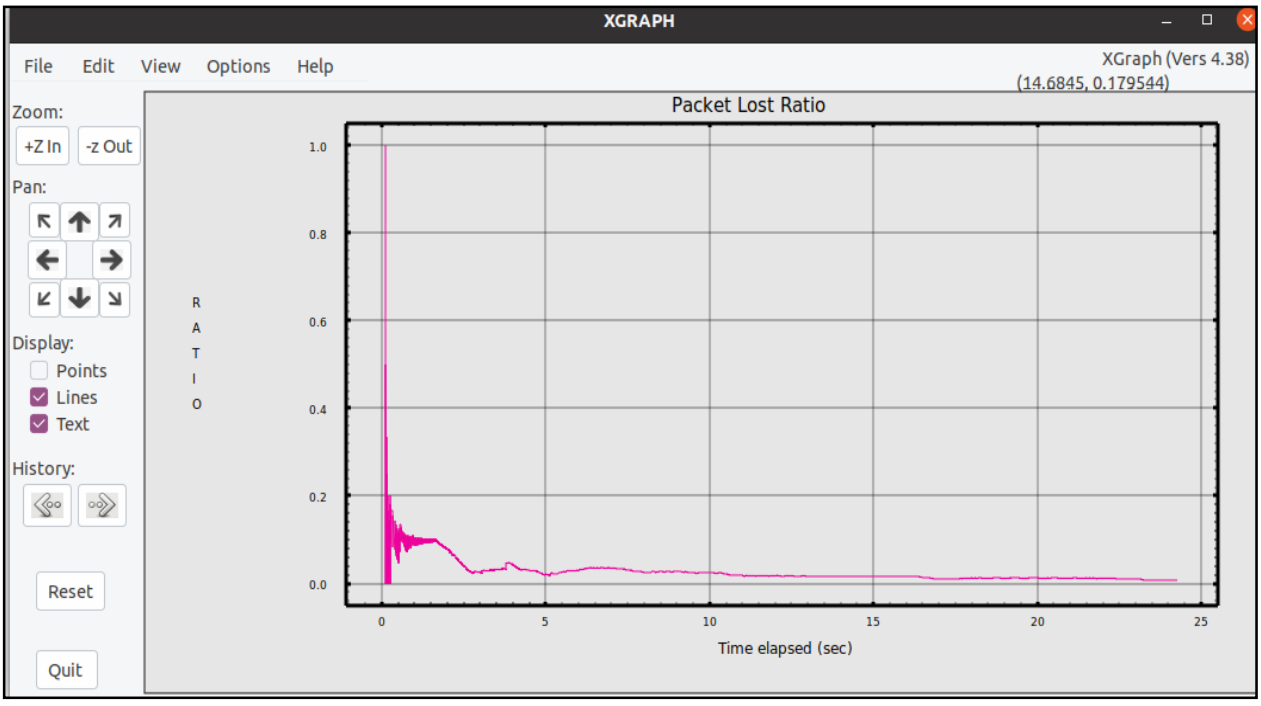
Since we know packet loss is the ratio of number of packets lost from the source to the numbers of the packet received to the destination so its value would be always between 0-1.

Here Node 1 is sending packets for all the time but as soon as Node 2 and Node 3 start sending packets there is some increase in ratio after 5th second which is then restored as those node's (2 and 3) traffic was closed after 10th second

.

The PLR (Packet Loss Ratio) is almost equal to 0 here because we have used TCP protocol for all the nodes and which indeed is a reliable protocol and hence there is very less packet loss.

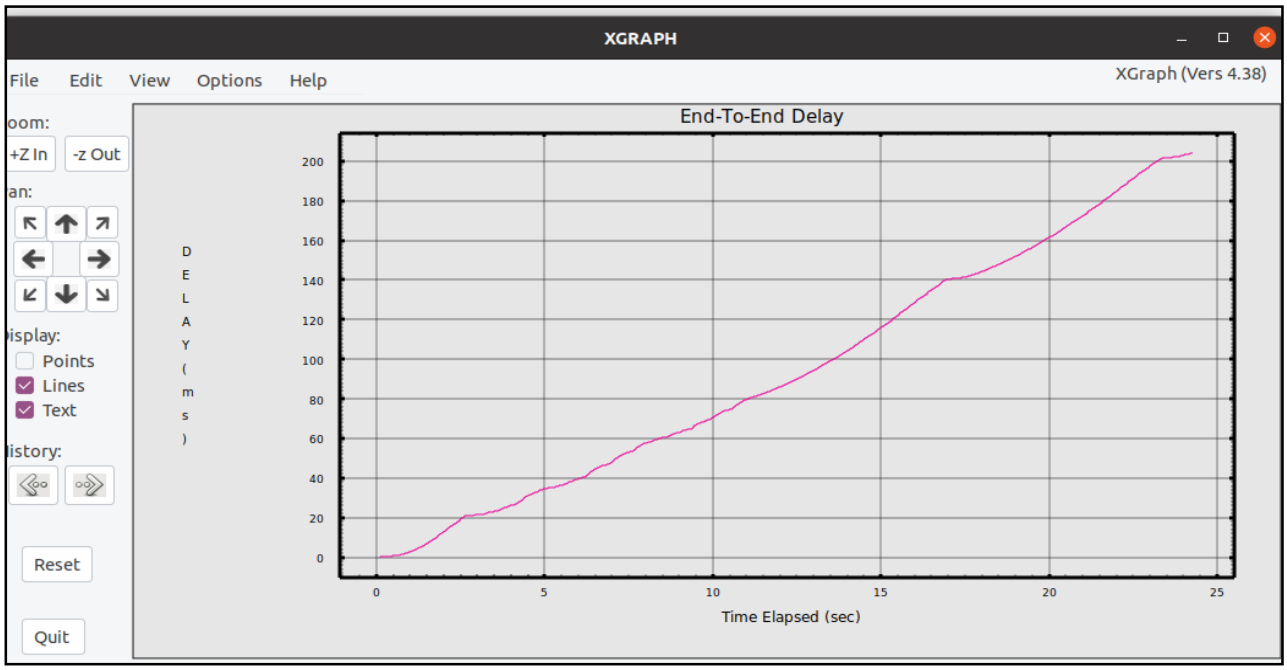
The graph of PLR is exactly opposite of PDR.



End-To-End Delay

Here
x-axis represents time elapsed in simulation
y-axis represents the delay with time elapsed (in milliseconds)

the delay of the packets are been keep adding and the overall delay is shown in the graph.



Case-II

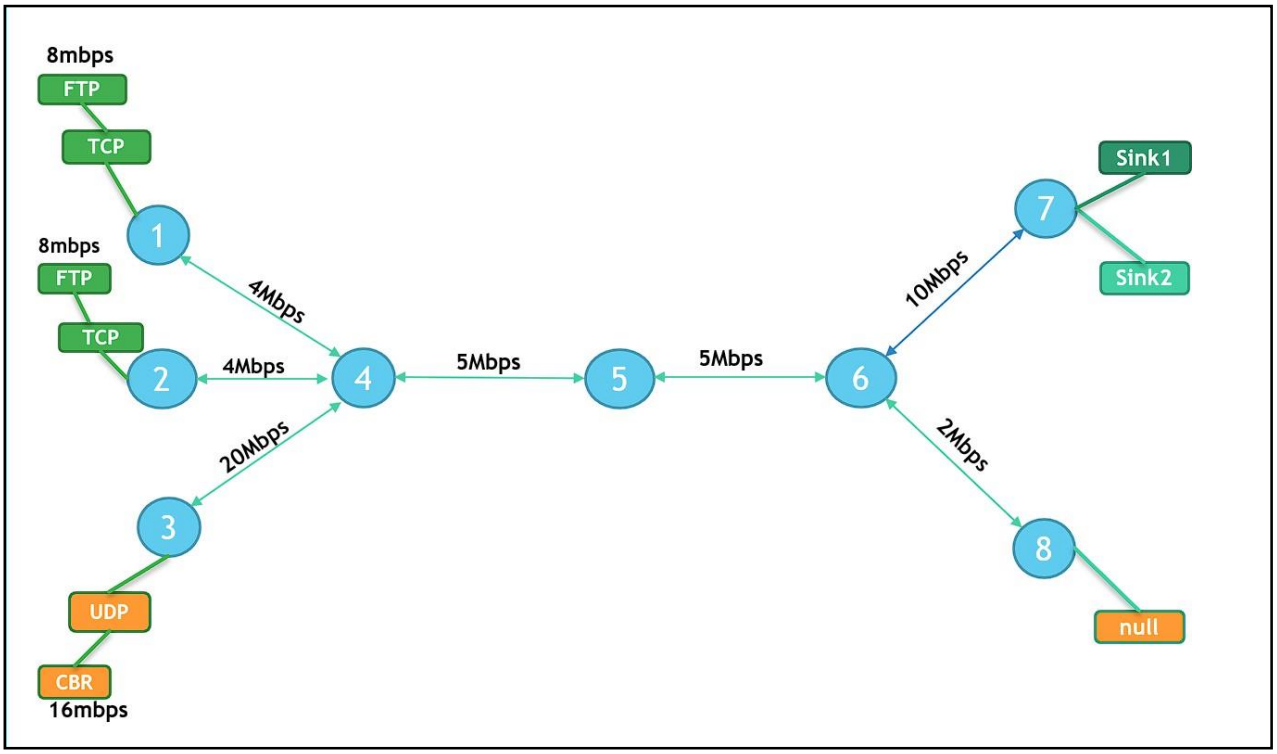
All nodes 1 and are sending packets using TCP protocols with the help of FTP as traffic source generator where as node 3 is sending packets using UDP protocol with the help if CBR as a traffic source generator.

The bandwidth of each link is mentioned just above the link in each node-to-node connection.

All the sources with their destinations

1. Node 1 is the source and node 7 is its destination with transmission speed of 8mbps
2. Node 2 is the source and node 7 is its destination with transmission speed of 8mbps
3. Node 3 is the source and node 8 is its destination with transmission speed of 16mbps

Here we have increased the data rate of the UDP source and decreased queuing capacity of the common links.



TCL Script

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

# Define different colors
# for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
$ns color 3 Yellow

# Open the NAM trace file
set nf [open p2_disp.nam w]
$ns namtrace-all $nf

set tracefile [open p2_log.tr w]
$ns trace-all $tracefile

puts " "
puts    "-----------------------------------------------------------------------------------------"
puts " The NAM and Log Trace files have been created. Do run the following
commands in order:- "
puts ""
puts " For viewing the simulation : nam p2_disp.nam"
puts ""
puts    "-----------------------------------------------------------------------------------------"
puts " 1.For plotting Packet Delivery Ratio :-"
puts    "-----------------------------------------------------------------------------------------"
puts " A) Run : awk -f pdr.awk -v src=<src> -v dest=<dest>
p2_log.tr>p2_pdr"
puts "  B) Run : xgraph p2_pdr"
puts ""
puts    "-----------------------------------------------------------------------------------------"
puts " 2.For plotting Packet Loss Ratio :-"
puts    "-----------------------------------------------------------------------------------------"
puts " A) Run : awk -f plr.awk -v src=<src> -v dest=<dest>
p2_log.tr>p2_plr"
puts "  B) Run : xgraph p2_plr"
puts ""
puts    "-----------------------------------------------------------------------------------------"
puts " 3.For plotting End to End Delay :-"
puts    "-----------------------------------------------------------------------------------------"
puts "  A) Run : awk -f e2e_delay.awk -v src=<src> -v dest=<dest>
p2_log.tr>p2_e2e_delay"
puts " B) Run : xgraph p2_e2e_delay"
puts ""
puts    "-----------------------------------------------------------------------------------------"
puts " "
```

```
# Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace

# Close the NAM trace file
close $nf

exec awk -f throughput.awk -v dest=6 p2_log.tr &
after 300
puts "\n"
exit 0
}

# Create four nodes

set n1 [$ns node]

set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]

# Create links between the nodes
$ns duplex-link $n1 $n4 4Mb 10ms DropTail
$ns duplex-link $n2 $n4 4Mb 10ms DropTail
$ns duplex-link $n3 $n4 20Mb 10ms DropTail
$ns duplex-link $n4 $n5 5Mb 30ms FQ
$ns duplex-link $n5 $n6 5Mb 20ms FQ
$ns duplex-link $n6 $n7 10Mb 10ms DropTail
$ns duplex-link $n6 $n8 2Mb 10ms DropTail

# Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n4 $n5 20
$ns queue-limit $n5 $n6 20

# Give node position (for NAM)
$ns duplex-link-op $n1 $n4 orient right-down
$ns duplex-link-op $n2 $n4 orient right
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n5 $n6 orient right
$ns duplex-link-op $n6 $n7 orient right-up
$ns duplex-link-op $n6 $n8 orient right-down
```

```
# Monitor the queue for link (n4-n5). (for NAM)
$ns duplex-link-op $n4 $n5 queuePos 0.5


# Setup first TCP connection
set tcp1 [new Agent/TCP]
$tcp1 set class_ 2
$ns attach-agent $n1 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $n7 $sink1
$ns connect $tcp1 $sink1
$tcp1 set fid_ 1

# Setup first FTP over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP
$ftp1 set packet_size_ 500
$ftp1 set rate_ 8mb
$ftp1 set random_ false

# Setup second TCP connection
set tcp2 [new Agent/TCP]
$tcp2 set class_ 2
$ns attach-agent $n2 $tcp2

set sink2 [new Agent/TCPSink]
$ns attach-agent $n7 $sink2
$ns connect $tcp2 $sink2
$tcp2 set fid_ 2

# Setup second FTP over TCP connection
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set type_ FTP
$ftp2 set packet_size_ 500
$ftp2 set rate_ 8mb
$ftp2 set random_ false


# Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n3 $udp
set null [new Agent/Null]
```

```
$ns attach-agent $n7 $null
$ns connect $udp $null
$udp set fid_ 3

# Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 500
$cbr set rate_ 16mb
$cbr set random_ false


# Schedule events for the FTP agents
$ns at 0.1 "$ftp1 start"
$ns at 24 "$ftp1 stop"
$ns at 5 "$ftp2 start"
$ns at 10.0 "$ftp2 stop"
$ns at 5.0 "$cbr start"
$ns at 10.0 "$cbr stop"

# Call the finish procedure after
# 5 seconds of simulation time
$ns at 24.0 "finish"

# Run the simulation
$ns run
```

Terminal Output

This is the output of our tcl script as shown in the terminal below contains

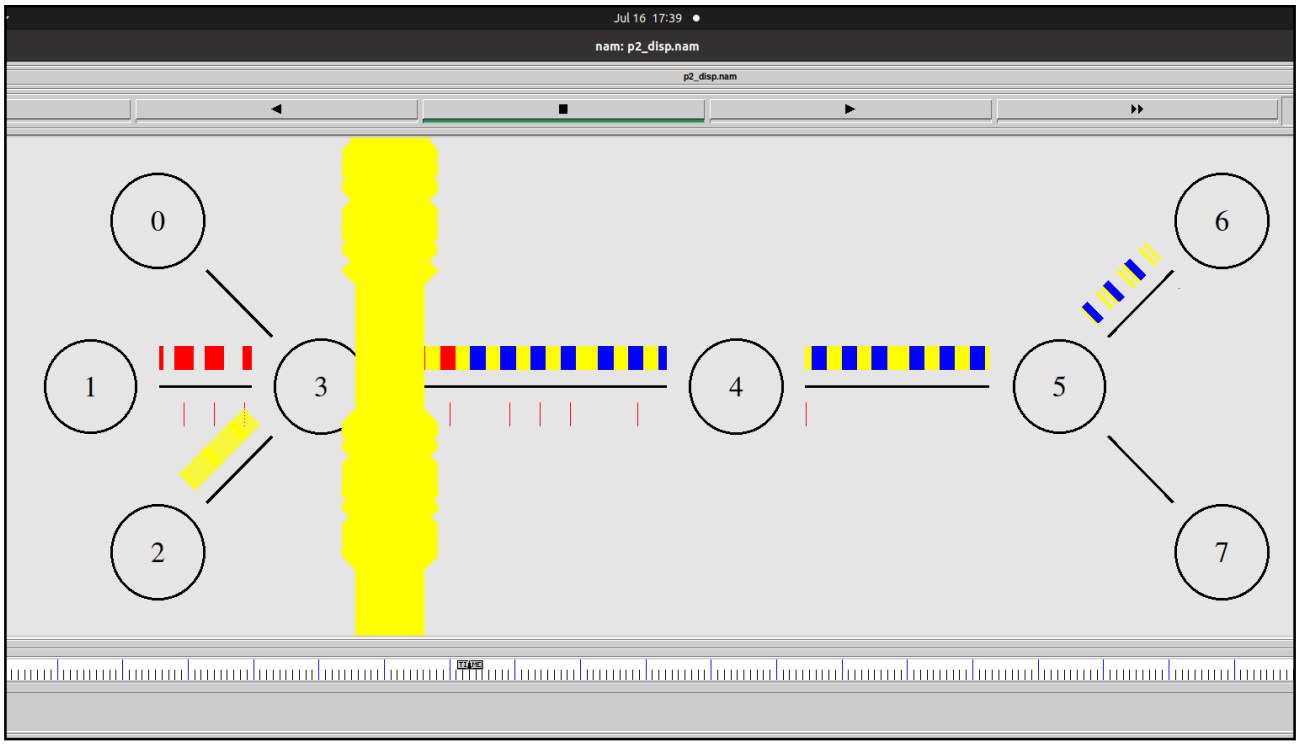
- The name of the trace file generated which can be used for further analyses (p2_log.tr)
- the name of the NAM file which can we used to see the visual representation of our topology (p2_disp.nam)
- Three factors which can be analyzed using the trace file with the help of awk script and xgraph are
    - Packer delivery ratio
    - Packet Loss Ratio
    - End to End Delay
- Each awk command has a src (source node) and a dest (destination node) variable which is used to analyze factor just between those two nodes.
  The output of the awk script is stored in a file which is then further used to plot the graphs.

  For example, if we want to calculate packet delivery ration from node 2 to node 7 then run
  awk -f pdr.awk -v src=2 -v dest=7 p2_log.tr>p2_pdr
  in the terminal after this p2_pdr file would be generated which is used to plot the graph by running **xgraph p2_pdr** in the terminal
- Throughput from the source to the destination

NAM Output

Here red and blue color block represent flow of the packets using TCP protocols whereas yellow color blocks represent UDP protocols packets

The yellow-colored squared block is a packet which is being dropped from the node 3



Packet Delivery Ratio

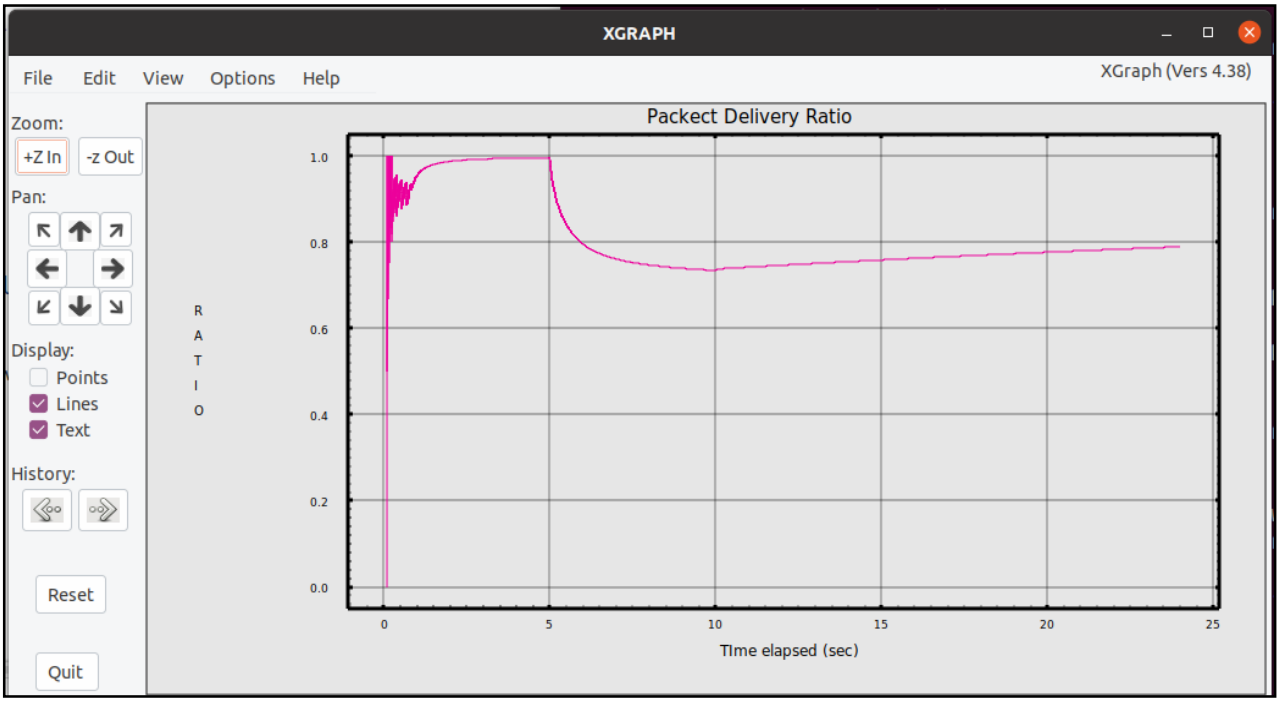
Here,

x-axis represents time elapsed in simulation
y-axis represents the ratio

Since we know packet delivery is the ratio of number of packets sent from the source to the numbers of the packet received to the destination so its value would be always between 0-1.

Here Node 1 is sending packets for all the time but as soon as Node 2 and Node 3 start sending packets there is significant decrease in ratio after 5th second which is then restored as those node's (2 and 3) traffic was closed after 10th second.

Due to one UDP source the number of dropping packets are increases as UDP is a un reliable protocol.



Packet Loss Ratio

Here,

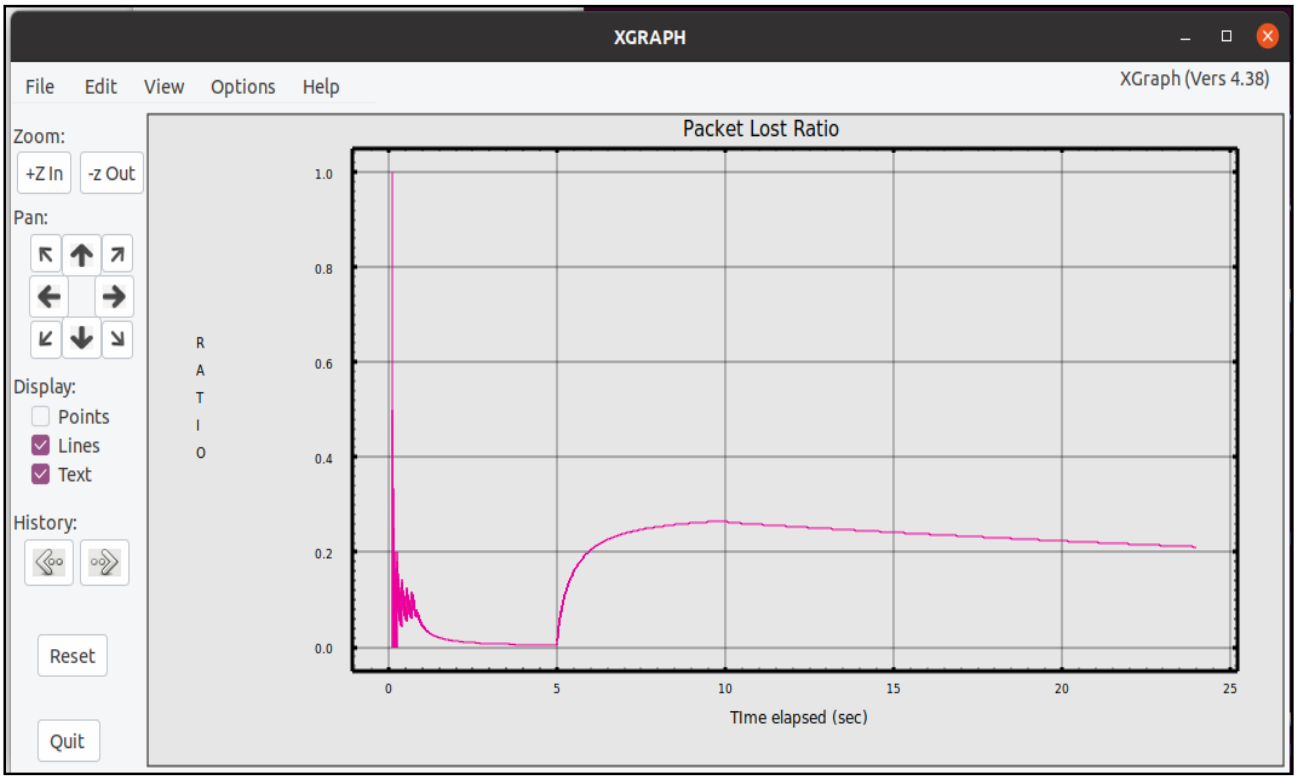
x-axis represents time elapsed in simulation
y-axis represents the ratio

Since we know packet loss is the ratio of number of packets lost from the source to the numbers of the packet received to the destination so its value would be always between 0-1

Here Node 1 is sending packets for all the time but as soon as Node 2 and Node 3 start sending packets there is some increase in ratio after 5th second which is then restored as those node's (2 and 3) traffic was closed after 10th second

Due to one UDP source the number of dropping packets are increases as UDP is a un reliable protocol.

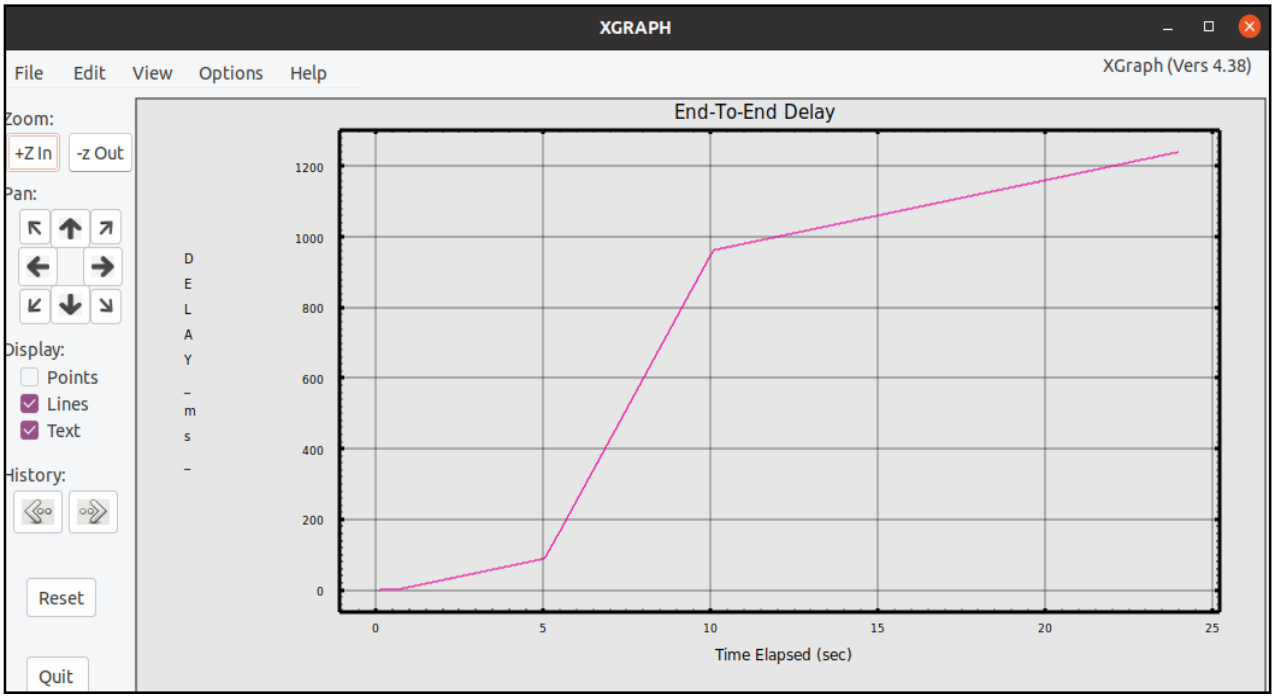
The graph of PLR is exactly opposite of PDR.



End-To-End Delay

Here,
x-axis represents time elapsed in simulation
y-axis represents the delay with time elapsed (in milliseconds)

The delay of the packets are been keep adding and the overall delay is shown in the graph



Case-III

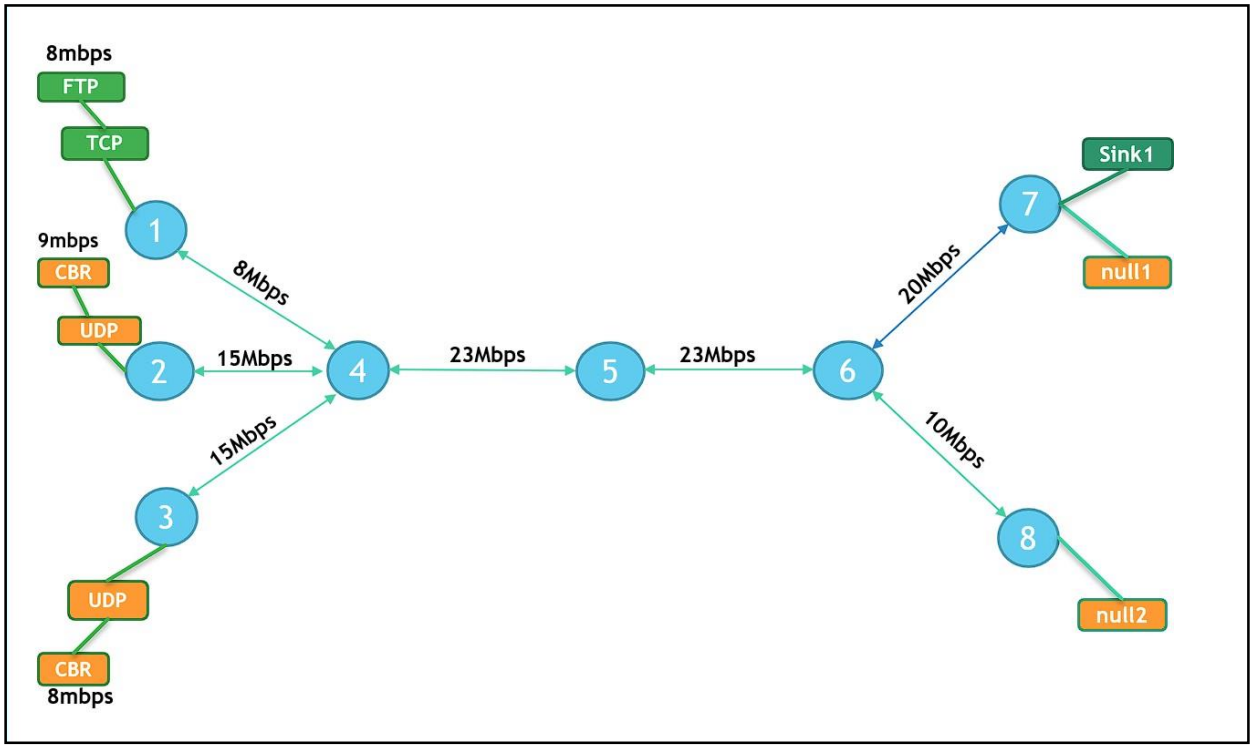
Node 1 is sending packets using TCP protocol with the help of FTP as traffic source generator whereas nodes 2 and 3 are sending packets using UDP protocol with help of CBR as traffic source generator

The bandwidth of each link is mentioned just above the link in each node-to-node connection

All the sources with their destinations

1. Node 1 is the source and node 7 is its destination with transmission speed of 8mbps
2. Node 2 is the source and node 7 is its destination with transmission speed of 9mbps
3. Node 3 is the source and node 8 is its destination with transmission speed of 8mbps

Here we have increased the bandwidth and queuing capacity of the common links.



TCL Script

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

# Define different colors
# for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
$ns color 3 Yellow

# Open the NAM trace file
set nf [open p3_disp.nam w]
$ns namtrace-all $nf

set tracefile [open p3_log.tr w]
$ns trace-all $tracefile

puts " "
puts    "-----------------------------------------------------------------------------------------
---- "
puts " The NAM and Log Trace files have been created. Do run the following
commands in order:- "
puts ""
puts " For viewing the simulation : nam p3_disp.nam"
puts ""
puts    "------------------------------------------------------------------------------------- "
puts " 1.For plotting Packet Delivery Ratio :-"
puts    "-------------------------------------------------------------------------------------"
puts " A) Run : awk -f pdr.awk -v src=<src> -v dest=<dest>
p3_log.tr>p3_pdr"
puts "  B) Run : xgraph p3_pdr"

puts ""
puts    "-------------------------------------------------------------------------------------"
puts " 2.For plotting Packet Loss Ratio :-"
puts    "-------------------------------------------------------------------------------------"
puts " A) Run : awk -f plr.awk -v src=<src> -v dest=<dest>
p3_log.tr>p3_plr"
puts "  B) Run : xgraph p3_plr"
puts ""
puts    "-------------------------------------------------------------------------------------"
puts " 3.For plotting End to End Delay :-"
puts    "-------------------------------------------------------------------------------------"
puts "  A) Run : awk -f e2e_delay.awk -v src=<src> -v dest=<dest>
p3_log.tr>p3_e2e_delay"
puts " B) Run : xgraph p3_e2e_delay"
puts ""
puts    "-------------------------------------------------------------------------------------"
puts " "
```

```
# Define a 'finish' procedure
proc finish {} {
global ns nf
$ns flush-trace

# Close the NAM trace file
close $nf

exec awk -f throughput.awk -v dest=6 p3_log.tr &
after 300
puts "\n"
exit 0
}

#   Create   four
nodesset n1 [$ns
node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]

# Create links between the nodes
$ns duplex-link $n1 $n4 8Mb 10ms DropTail
$ns duplex-link $n2 $n4 15Mb 10ms DropTail
$ns duplex-link $n3 $n4 15Mb 10ms DropTail
$ns duplex-link $n4 $n5 23Mb 30ms RED
$ns duplex-link $n5 $n6 23Mb 20ms FQ
$ns duplex-link $n6 $n7 20Mb 10ms DropTail
$ns duplex-link $n6 $n8 5Mb 10ms DropTail

# Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n4 $n5 55
$ns queue-limit $n5 $n6 55

# Give node position (for NAM)
$ns duplex-link-op $n1 $n4 orient right-down
$ns duplex-link-op $n2 $n4 orient right
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n5 $n6 orient right
$ns duplex-link-op $n6 $n7 orient right-up
$ns duplex-link-op $n6 $n8 orient right-down
```

```
# Monitor the queue for link (n4-n5). (for NAM)
$ns duplex-link-op $n4 $n5 queuePos 0.5

# Setup first TCP connection
set tcp1 [new Agent/TCP]
$tcp1 set class_ 2
$ns attach-agent $n1 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $n7 $sink1
$ns connect $tcp1 $sink1
$tcp1 set fid_ 1

# Setup first FTP over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP

$ftp1 set packet_size_ 500
$ftp1 set rate_ 8mb
$ftp1 set random_ false

# Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n3 $udp1
set null [new Agent/Null]

$ns attach-agent $n7 $null
$ns connect $udp1 $null
$udp1 set fid_ 2

# Setup a CBR over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set type_ CBR
$cbr1 set packet_size_ 500
$cbr1 set rate_ 9mb
$cbr1 set random_ false

# Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n3 $udp2
set null [new Agent/Null]

$ns attach-agent $n7 $null
$ns connect $udp2 $null
$udp2 set fid_
```

```
# Setup a CBR over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$cbr2 set type_ CBR
$cbr2 set packet_size_ 500
$cbr2 set rate_ 8mb
$cbr2 set random_ false

# Schedule events for the FTP agents
$ns at 0.1 "$ftp1 start"
$ns at 24 "$ftp1 stop"
$ns at 5 "$cbr1 start"
$ns at 10.0 "$cbr1 stop"
$ns at 5.0 "$cbr2 start"
$ns at 10.0 "$cbr2 stop"


# Call the finish procedure after
# 5 seconds of simulation time

$ns at 24.0 "finish"

# Run the simulation
$ns run
```

Terminal Output

This is the output of our tcl script as shown in the terminal below contains

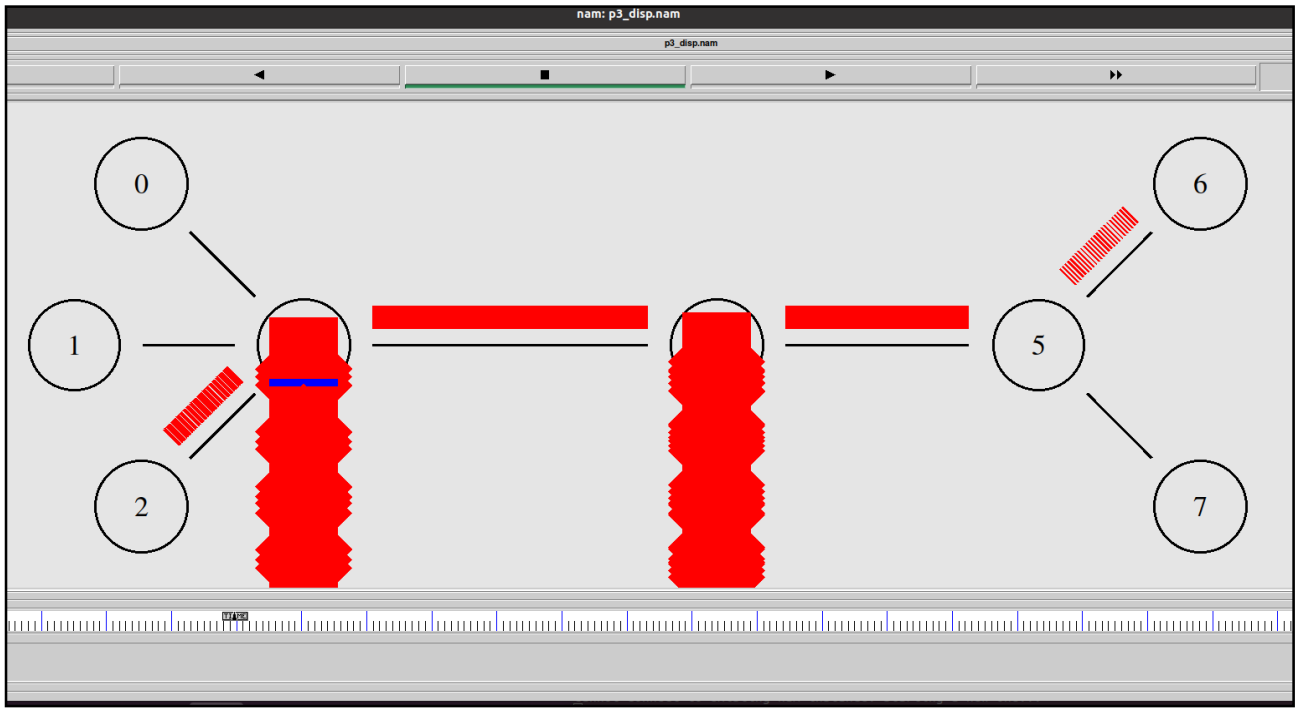
- The name of the trace file generated which can be used for further analyses (p3_log.tr)
- the name of the NAM file which can we used to see the visual representation of our topology (p3_disp.nam)
  To see nam output just type **nam p3_disp.nam** in the terminal
- Three factors which can be analyzed using the trace file with the help of awk script and xgraph are
  - Packer delivery ratio
  - Packet Loss Ratio
  - End to End Delay
- Each awk command has a src (source node) and a dest (destination node) variable which is used to analyze factor just between those two nodes.
  The output of the awk script is stored in a file which is then further used to plot the graphs.

  For example, if we want to calculate packet delivery ration from node 2 to node 7 then run
  awk -f pdr.awk -v src=2 -v dest=7 p1_log.tr>p1_pdr
  in the terminal after this p2_pdr file would be generated which is used to plot the graph by running **xgraph p3_pdr** in the terminal
- Throughput from the source to the destination

NAM Output

Here yellow and red color block represent flow of the packets using UDP protocol packets whereas blue color blocks represent TCP protocol packets

the red and blue colored squared block is a packet which is being dropped from the node 3



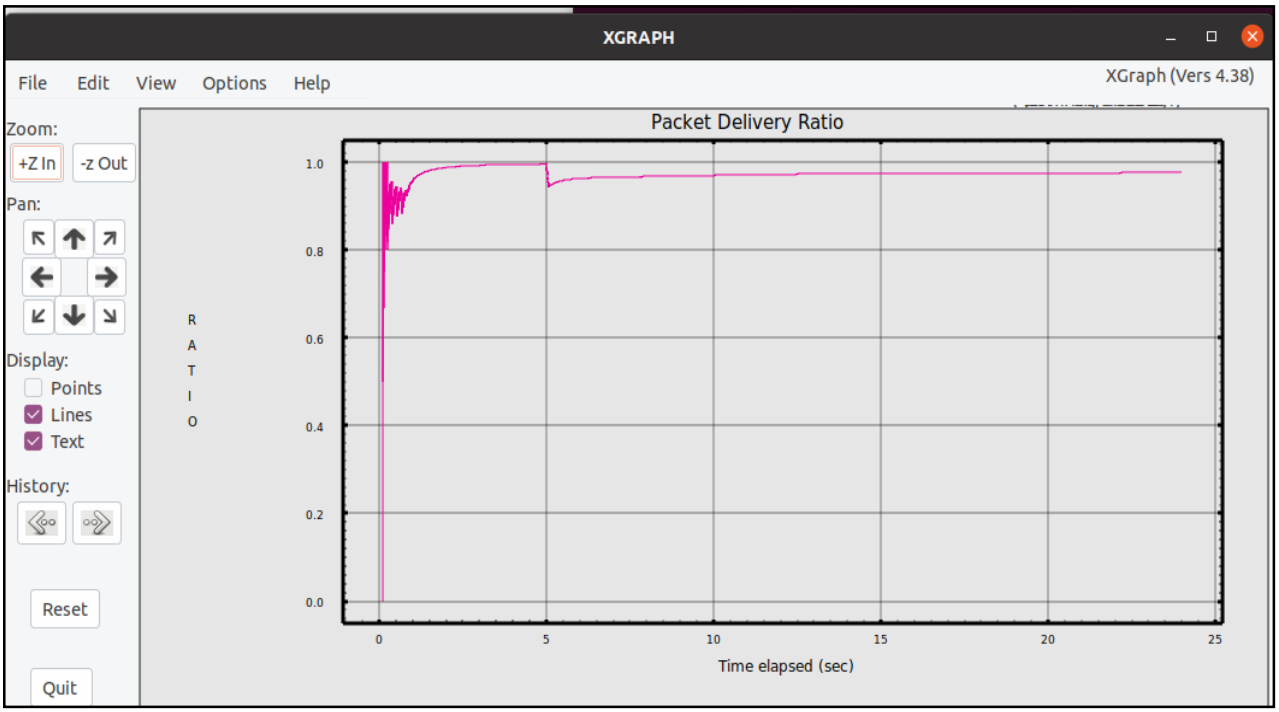
Packet Delivery Ratio

Here,

x-axis represents time elapsed in simulation
y-axis represents the ratio

since we know packet delivery is the ratio of number of packets sent from the source to the numbers of the packet received to the destination so its value would be always between 0-1

here Node 1 is sending packets for all the time but as soon as Node 2 and Node 3 start sending packets there is significant decrease in ratio after 5th second which is then restored as those node's (2 and 3) traffic was closed after 10th second



Packet Loss Ratio

Here,

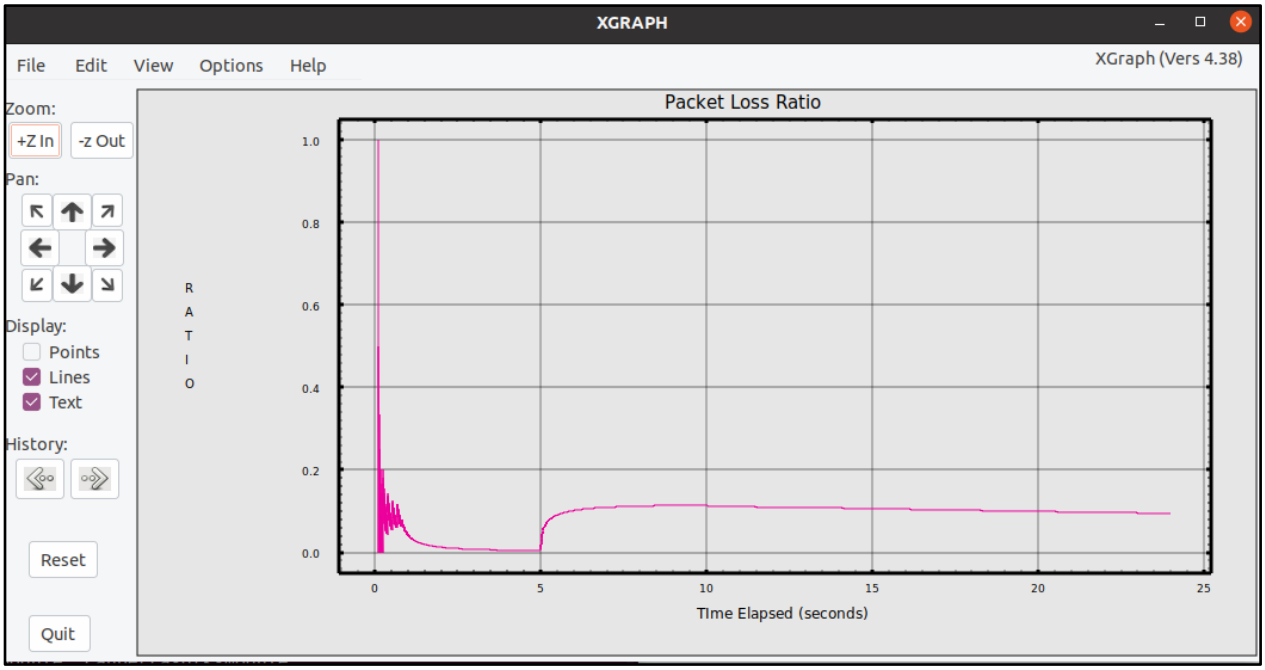
x-axis represents time elapsed in simulation
y-axis represents the ratio

Since we know packet loss is the ratio of number of packets lost from the source to the numbers of the packet received to the destination so its value would be always between 0-1.

Here, Node 1 is sending packets for all the time but as soon as Node 2 and Node 3 start sending packets there is some increase in ratio after 5th second which is then restored as those node's (2 and 3) traffic was closed after 10th second.

Due to two UDP source the number of dropping packets are increases as UDP is a un reliable protocol.

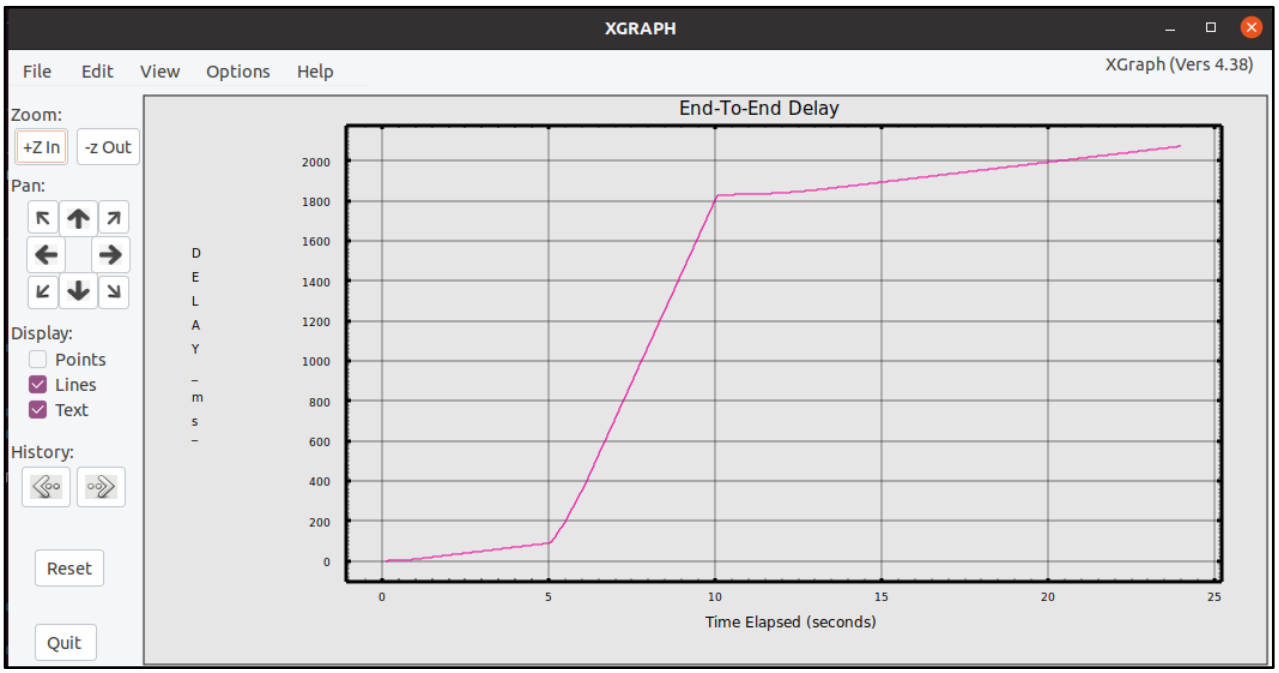
The graph of PLR is exactly opposite of PDR.



End-To-End Delay

Here,
x-axis represents time elapsed in simulation
y-axis represents the delay with time elapsed (in milliseconds)

The delay of the packets are been keep adding and the overall delay is shown in the graph
here there is a sudden increase in the delay at 5th because we have started node 2 and node 3 traffic at 5th second and stopped at 10th second



AWK scripts

```
#packet delivery ratio

BEGIN{

        pktsend=0; #stores total number of pkts sent
        pktrec=0; #stores total number of pkts received
        ctime=0; #current time
}

{
        e=$1;
        #Trace File current destination
        tf_dest=$4;

        #Trace File current source
        tf_src=$3;
        ctime=$2

        if(e =="r" && tf_dest=dest){
                pktrec++;
        }

        if(e =="+" && tf_src=src){
                pktsend++;
        }

        if(pktsend>0){
                printf("%f %f \n", ctime, pktrec/pktsend);
        }
}

END{

}
```

To find out packet delivery ratio we are adding all the packets received and dividing them by number of packets sent with respect to time.

Packet delivery ratio awk script contains three variables
pktsend - which keeps the count of the packets sent by the source
pktrec - which keeps the count of the packets received by the destination
ctime - stores the current time

```
# End to End delay script

BEGIN{
        pksend=0;
        pkrec=0;
        ctime=0;
        tdelay=0;
}

{
        e=$1;
        #Trace File current destination
        tf_dest=$4;

        #Trace File current source
        tf_src=$3;
        ctime=$2

        if(e =="r" && tf_dest=dest){
                tdelay+= ctime - stime[$11];
        }

        if(e =="+" && tf_src=src){
                stime[$11]= ctime;
        }
        printf("%f %f \n", ctime, tdelay);
}

END{

}
```

Here we are maintaining an array to store start time of each packet and when the packet is received by the destination node its current time is subtracted from the start time which was already stored in the array which indeed gives its delay
Similarly delay time of every packet is calculated and added to overall delay (tdelay)

Variable used
pktsend which keeps the count of the packets sent by the source
pktrec which keeps the count of the packets received by the destination
ctime stores the current time
tdelay to store the overall delay

```
#packet loss ratio

BEGIN{
        pktsend=0;
        pktrec=0;
        ctime=0;
}

{
        e=$1;
        #Trace File current destination
        tf_dest=$4;

        #Trace File current source
        tf_src=$3;
        ctime=$2

        if(e =="r" && tf_dest=dest){
                pktrec++;
        }

        if(e =="+" && tf_src=src){
                pktsend++;
        }

        if(pktsend>0){
                printf("%f %f \n", ctime, (pktsend -pktrec)/pktsend);
        }
}

END{

}
```

To find out packet Loss ratio we are adding all the packets sent and subtracting them by number of packets received and finally dividing them with the number of packet sent with respect to time. Packet delivery ratio awk script contains three variables

pktsend which keeps the count of the packets sent by the source
pktrec which keeps the count of the packets received by the destination
ctime stores the current time

Conclusion

We have understood how protocols simulation done using NS2. In the process we learn to write the TCL scripts. And, how nam visualization file and trace files generated through TCL scripts. We are also making different types of topologies and anlysing them using parameters like end-to-end delay, throughput, packet delivery ratios and packet loss ratios. We have seen how parameters act differently in different cases. There we also see how by changing the protocols and their respective traffic generators their come changes in graphs. We also come to know about various new terms such as bandwidth, throughput, droptail, transmission delay, propagation delay etc. And by forming graphs using AWK scripts, we also get a good insight of shell scripts.

Future Works

• Will going to explore on wireless networks.

• Protocols related to wireless networks are-
DSR, AODV, AOMDV, DSDV, OLSR

• Also going to study about various other simulator like-
Cisco packet Tracer, GNS3 etc.

References

1. **https://sourceforge.net/projects/nsnam/**
2. **https://www.isi.edu/nsnam/ns/**
3. **https://www.nsnam.com/**
4. **https://www.xgraph.org/**