# 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 (2020CA054)
Vinay Kumar(2020CA103)

**Submitted To**

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

# 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.**

# Introduction of NS-2

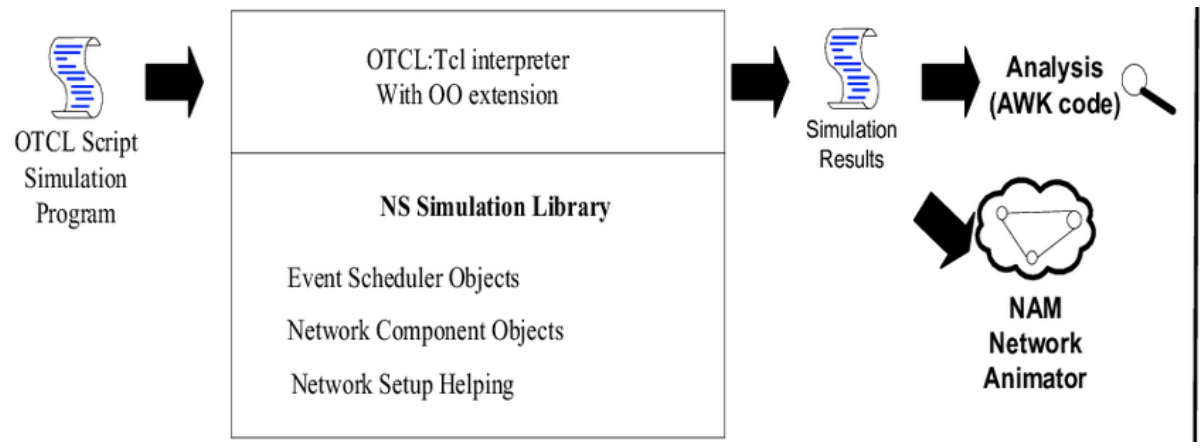
**Simulation** is the process of learning by doing. Whenever there is something new in the world, we try to analyze it first by examining it and in the process get to learn a lot of things. This entire course is called Simulation.

**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 communications 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 charactersitics for telephony and also native voice applications.

It also offer suppprt for timing sensitive traffic.

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

# Tcl script in NS-2

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-trace
    close $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
```

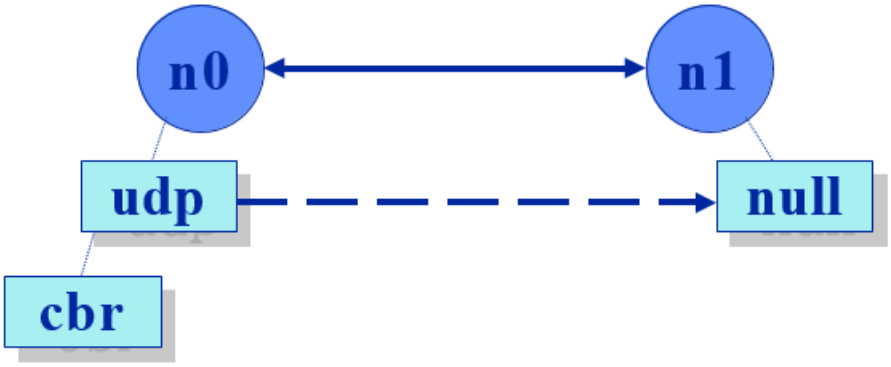
This line tells the simulator object to connect the nodes n0 and n1 with a duplex link with the bandwidth 1Megabit, a delay of 10ms and a DropTail queue.

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 n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to udp0
set 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"'.

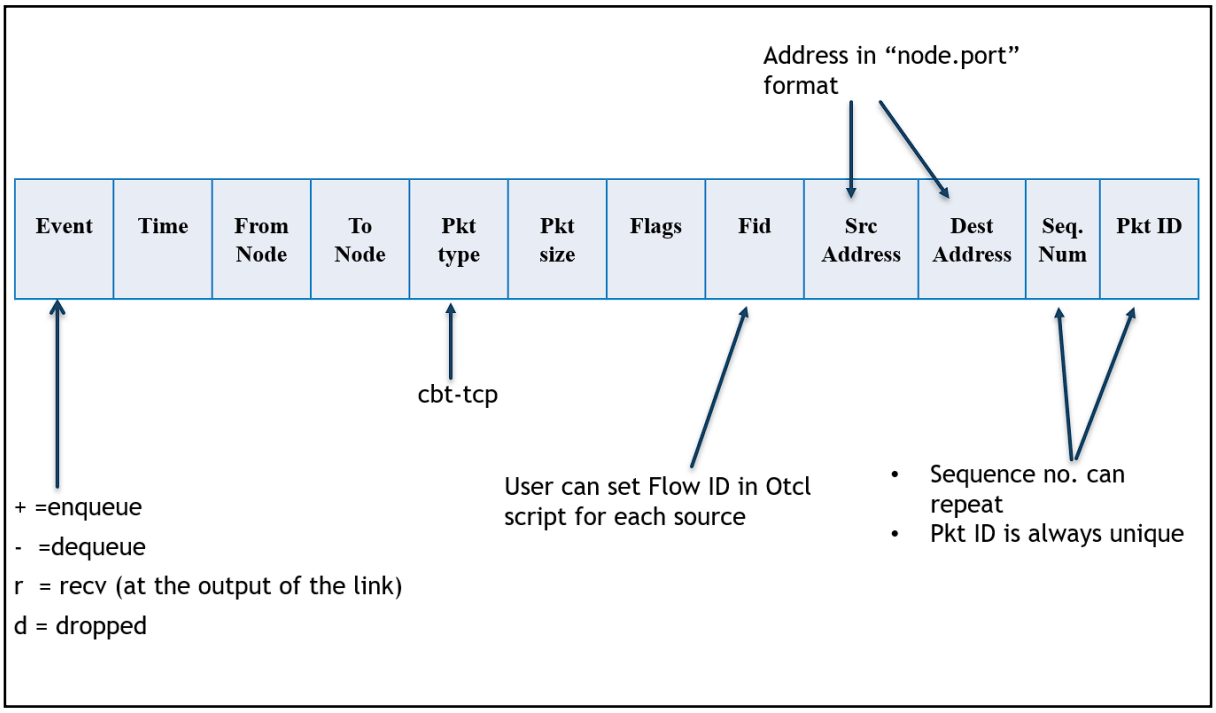
```
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

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 analyse the various 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.

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

### Average 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.

### 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 shownAverage 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.

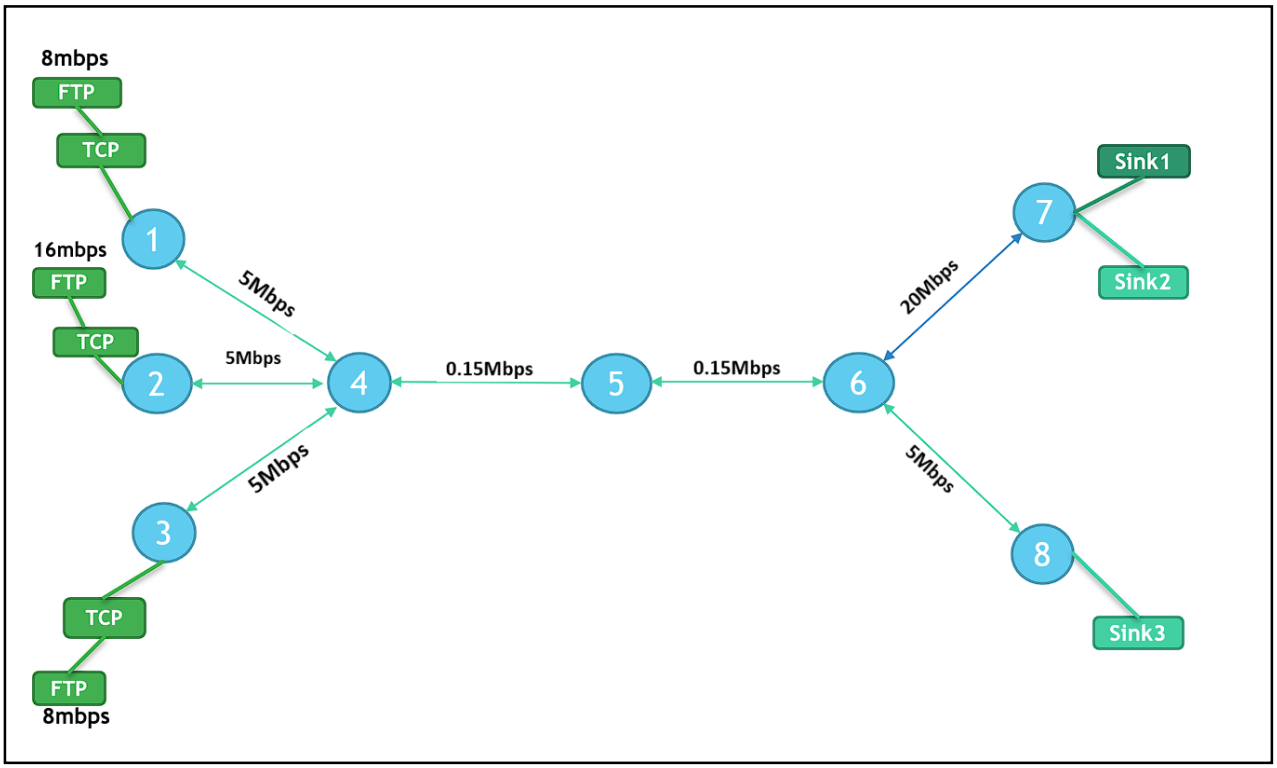
A data set consists of an ordered list of points of the form “directive X Y”. For directive “draw”, a line will be drawn between the previous point and the current point. Specifying a “move” directive tells xgraph not to draw a line between the points. “draw” is the default directive. The name of a data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

## Case – I




## 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:- "
```

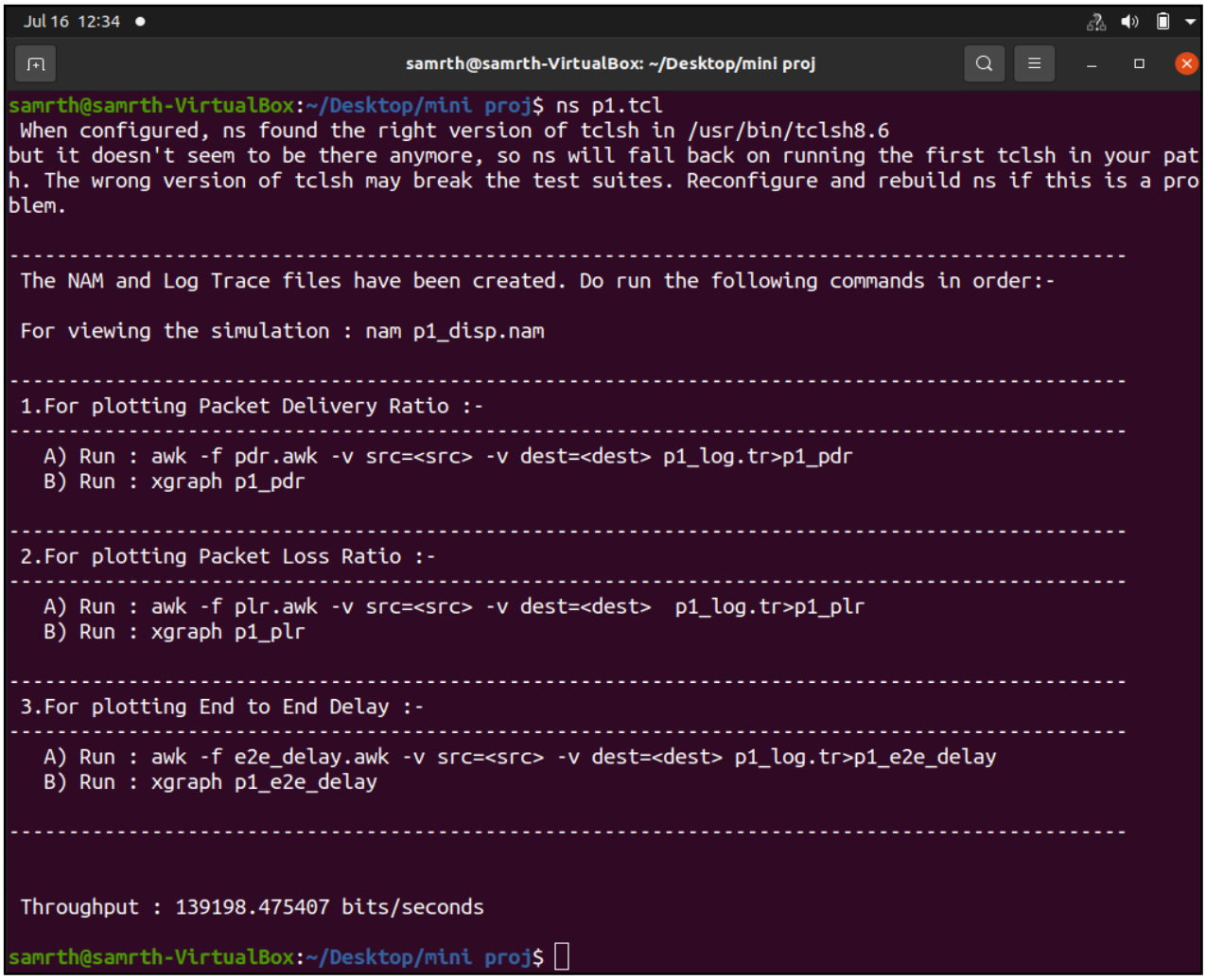
```
puts ""
puts " For viewing the simulation : nam p1_disp.nam"
puts ""

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
```

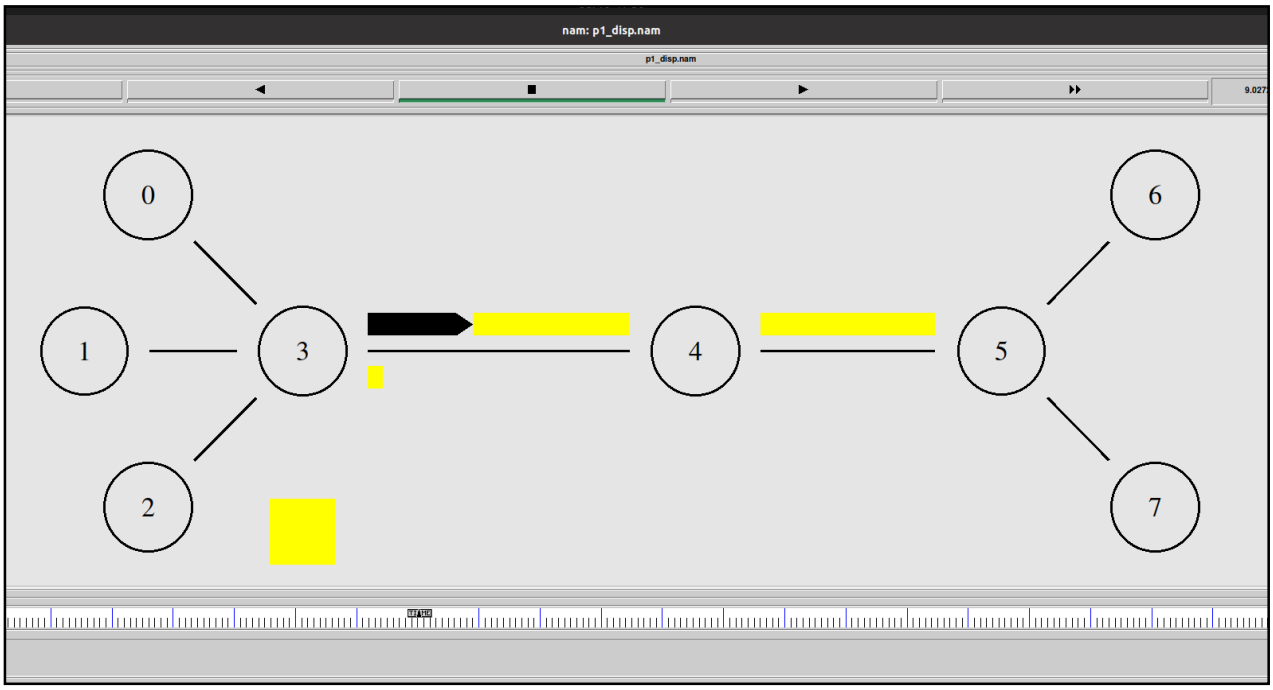
```
# Setup second TCP connection
set tcp2 [new Agent/TCP]
$tcp2 set class_ 2
$ns attach-agent $n2 $tcp2


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**

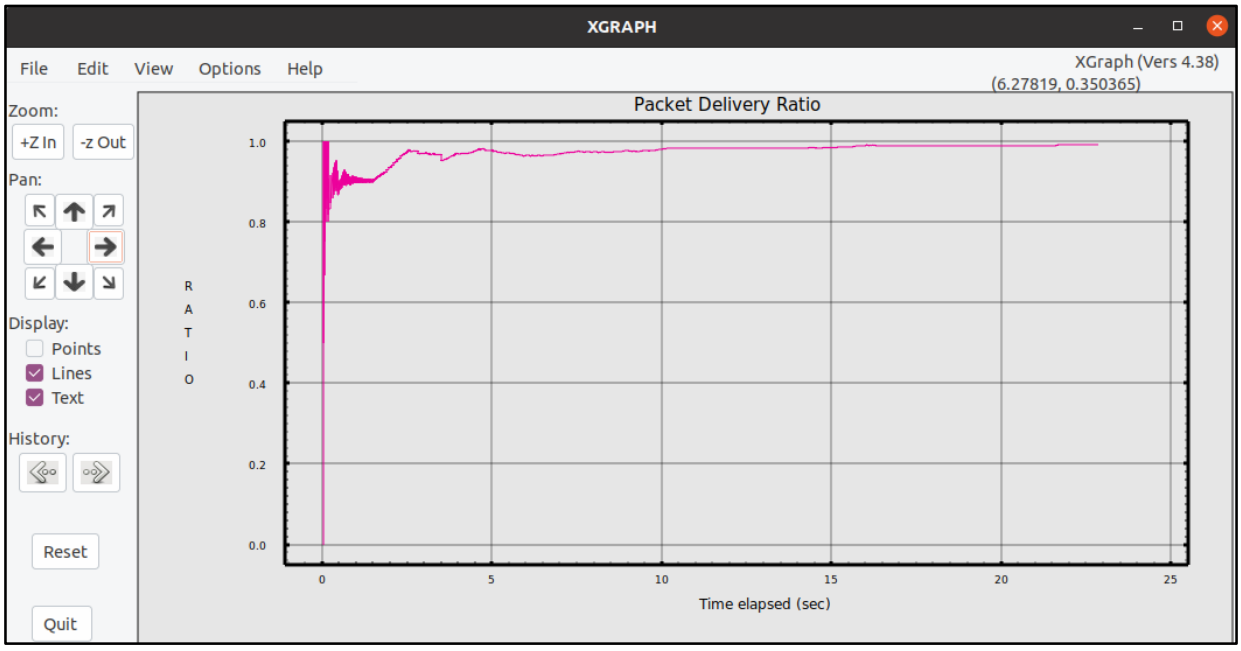
## Terminal Output



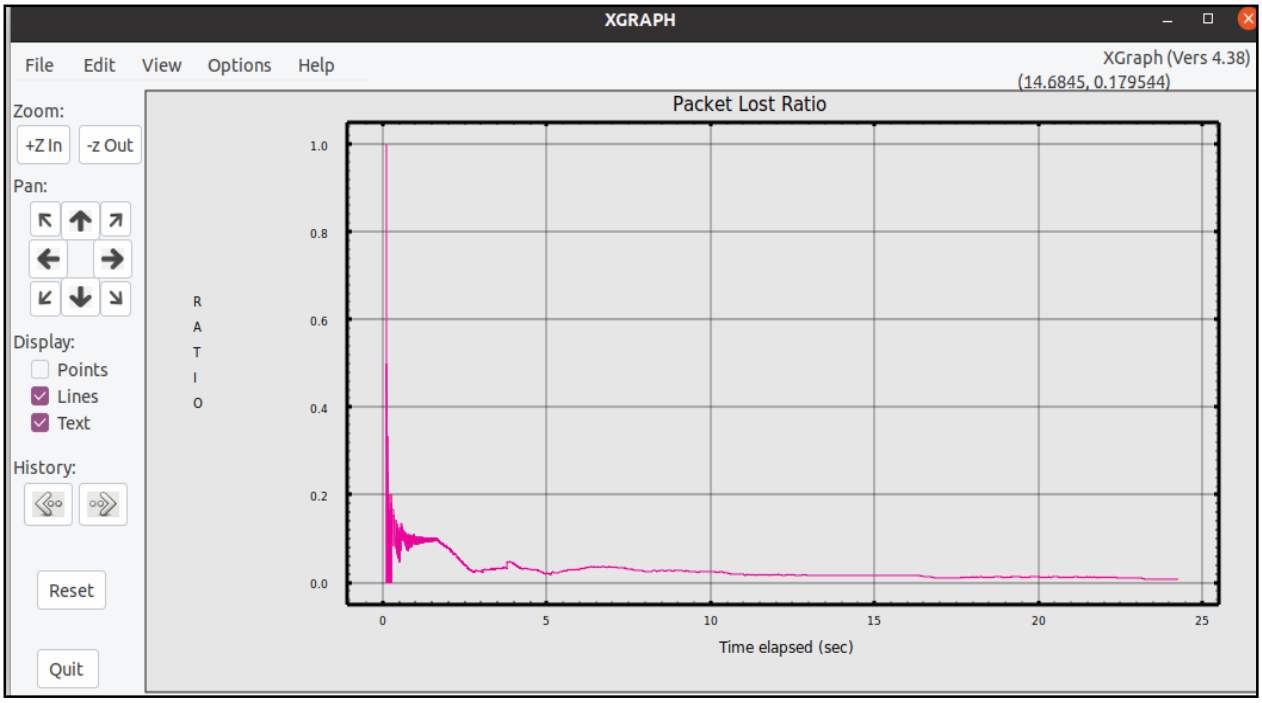
## NAM Output



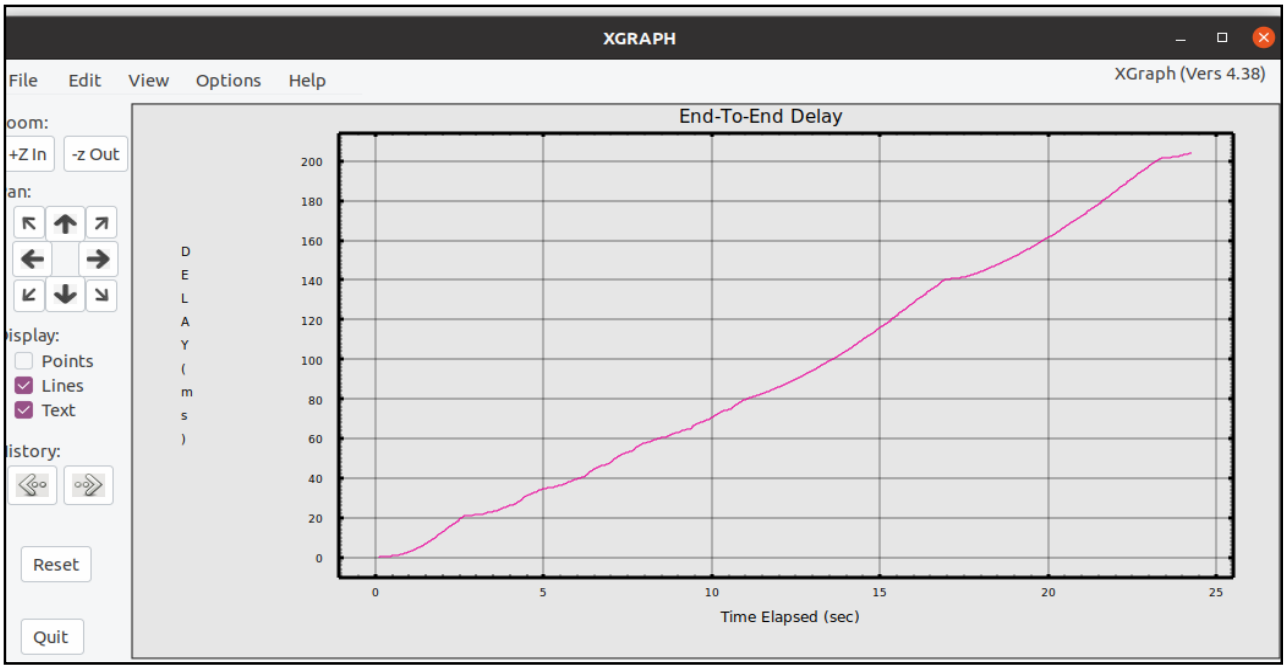

## Packet Delivery Ratio



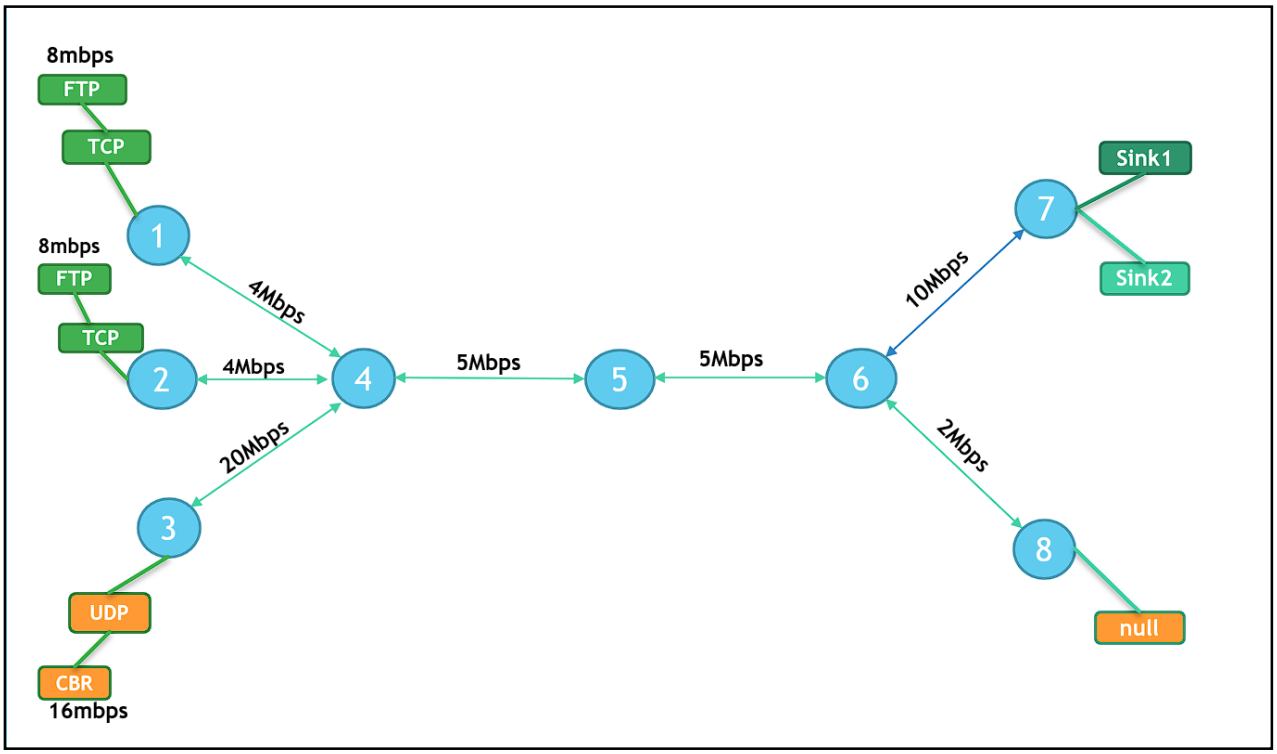
## Packet Lost Ratio




## End-To-End Delay



## Case-II




## 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 ""
```

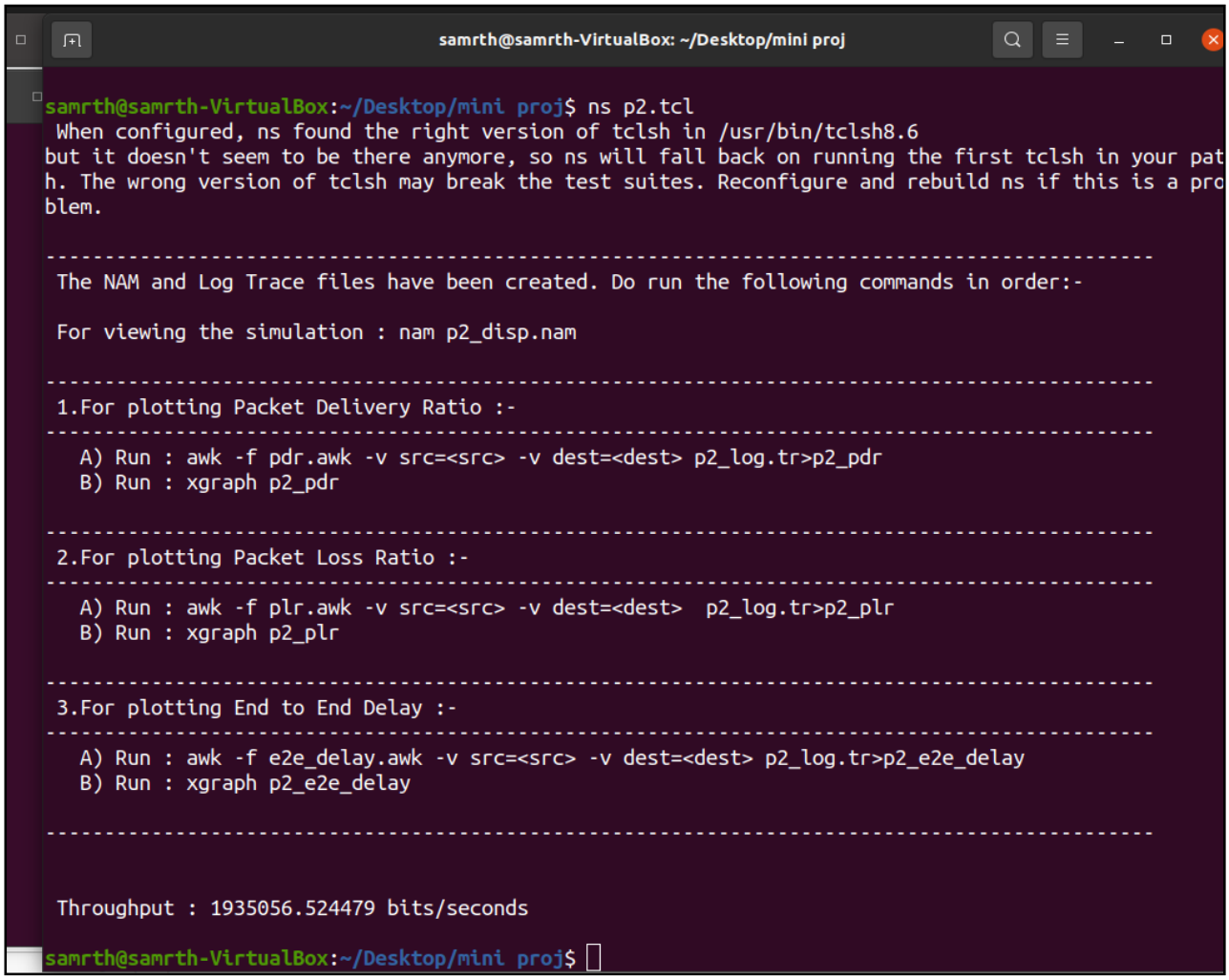
```
puts "-------------------------------------------------------------------------------
---- "
puts " 1.For plotting Packet Delivery Ratio :-"
puts "-------------------------------------------------------------------------------
---- "
puts "   A) Run : awk -f pdr.awk -v src=<src> -v dest=<dest>
p2_log.tr>p2_pdr"
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
```

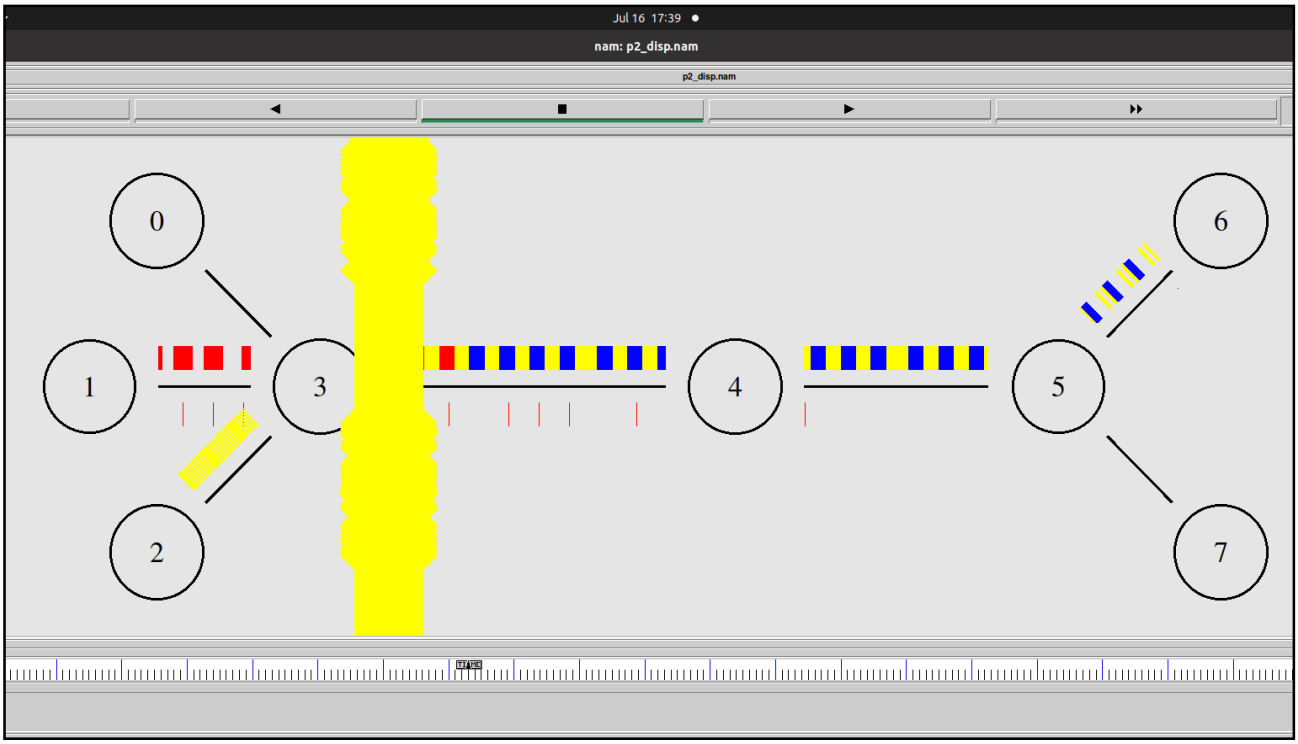
```
$ftp1 set packet_size_ 500
$ftp1 set rate_ 8mb
$ftp1 set random_ false

# 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

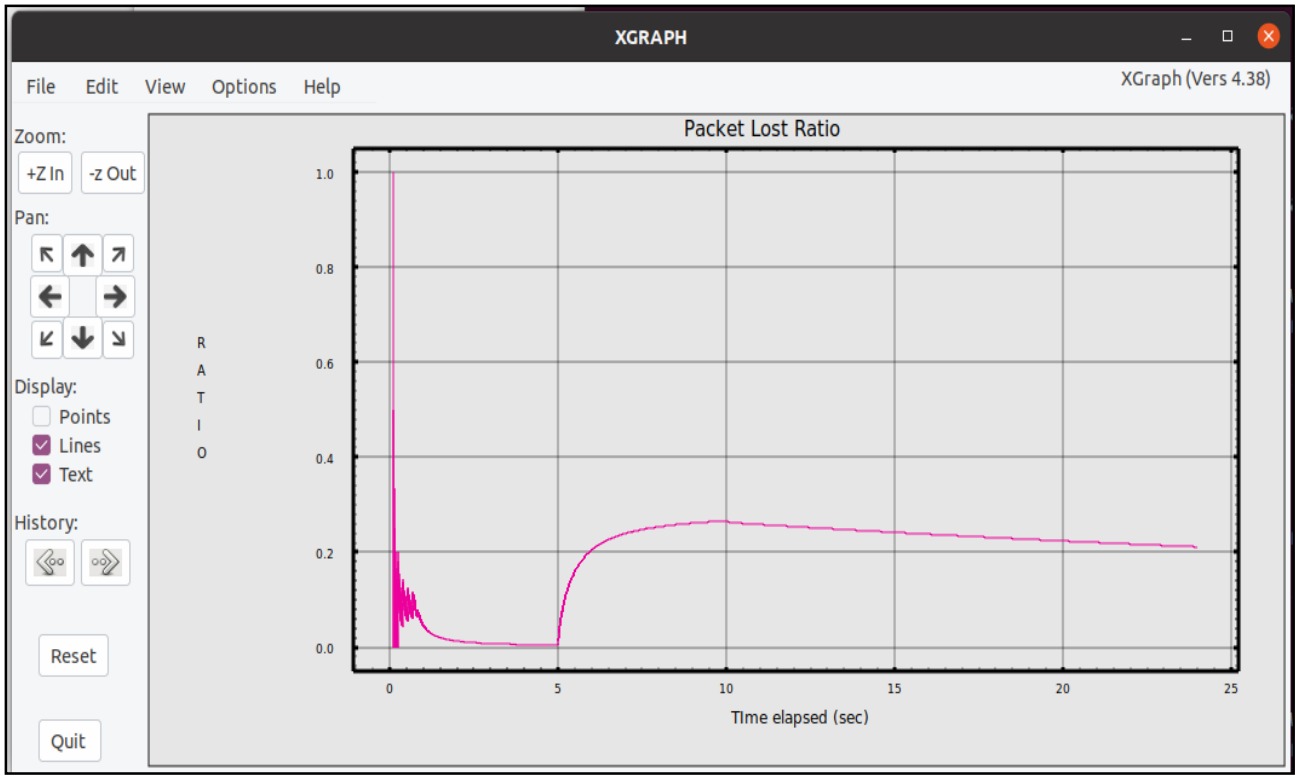


## NAM Output

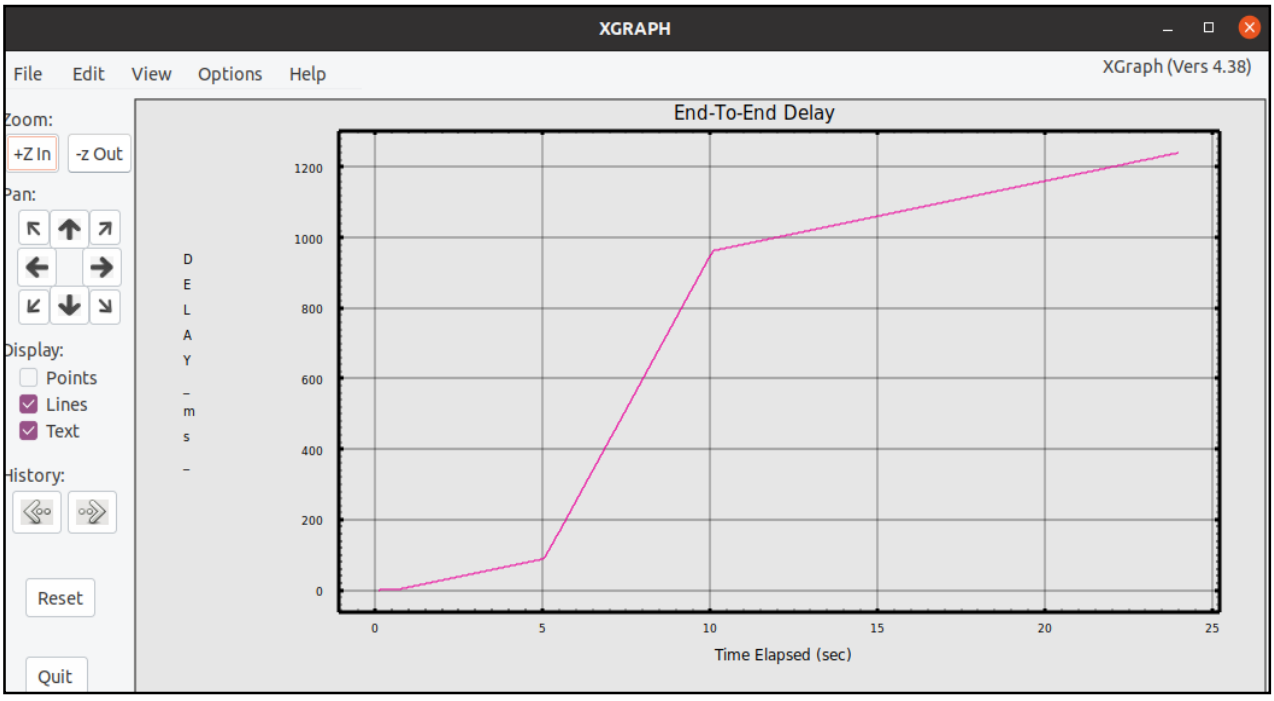



## Packet Delivery Ratio

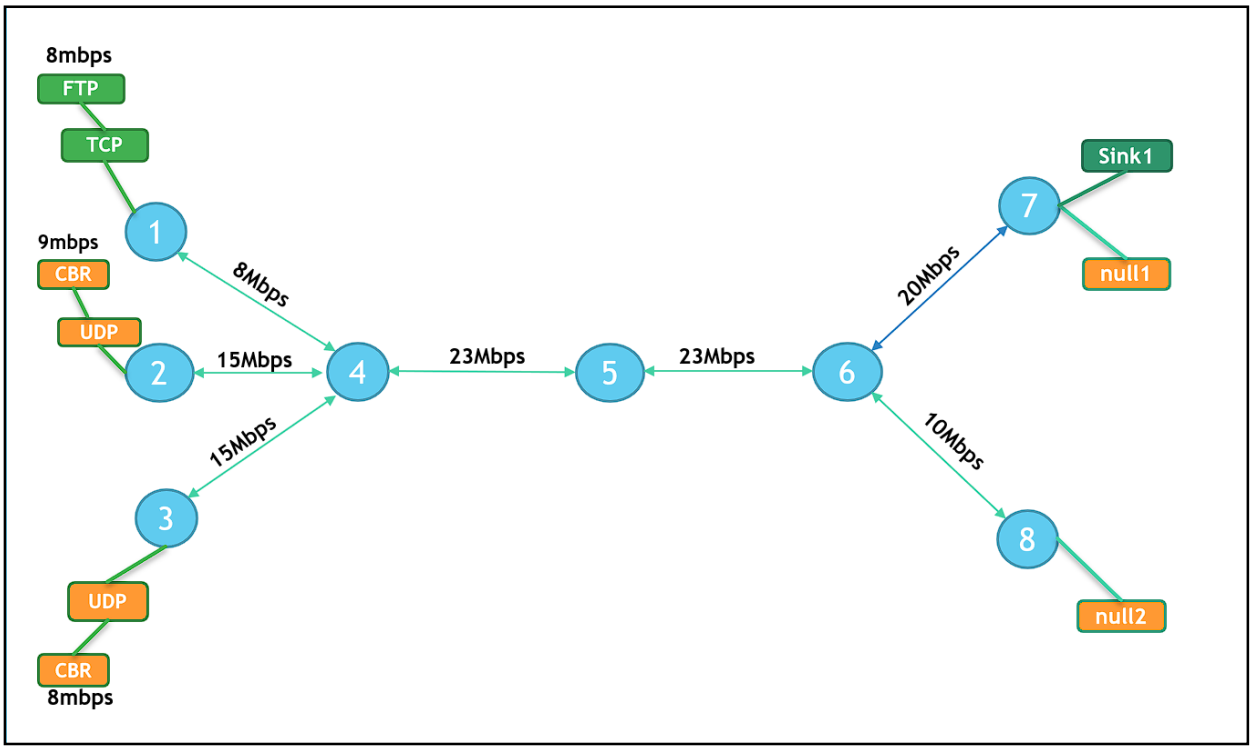
## Packet Loss Ratio




## End-To-End Delay



## Case-III




## 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 nodes

set n1 [$ns node]
```

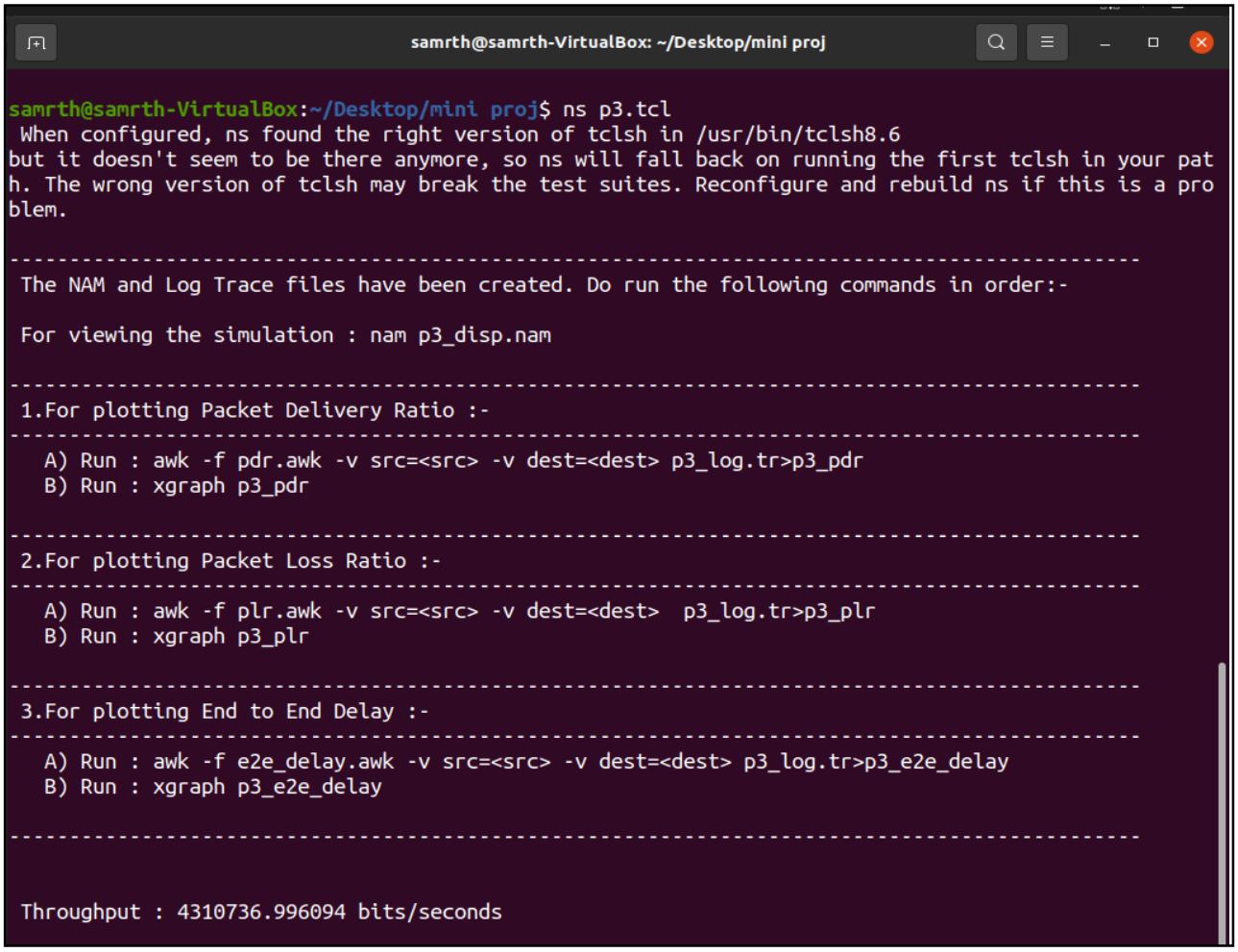
```
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]

# Create links between the nodes
$ns duplex-link $n1 $n4 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_ 2

# 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
```

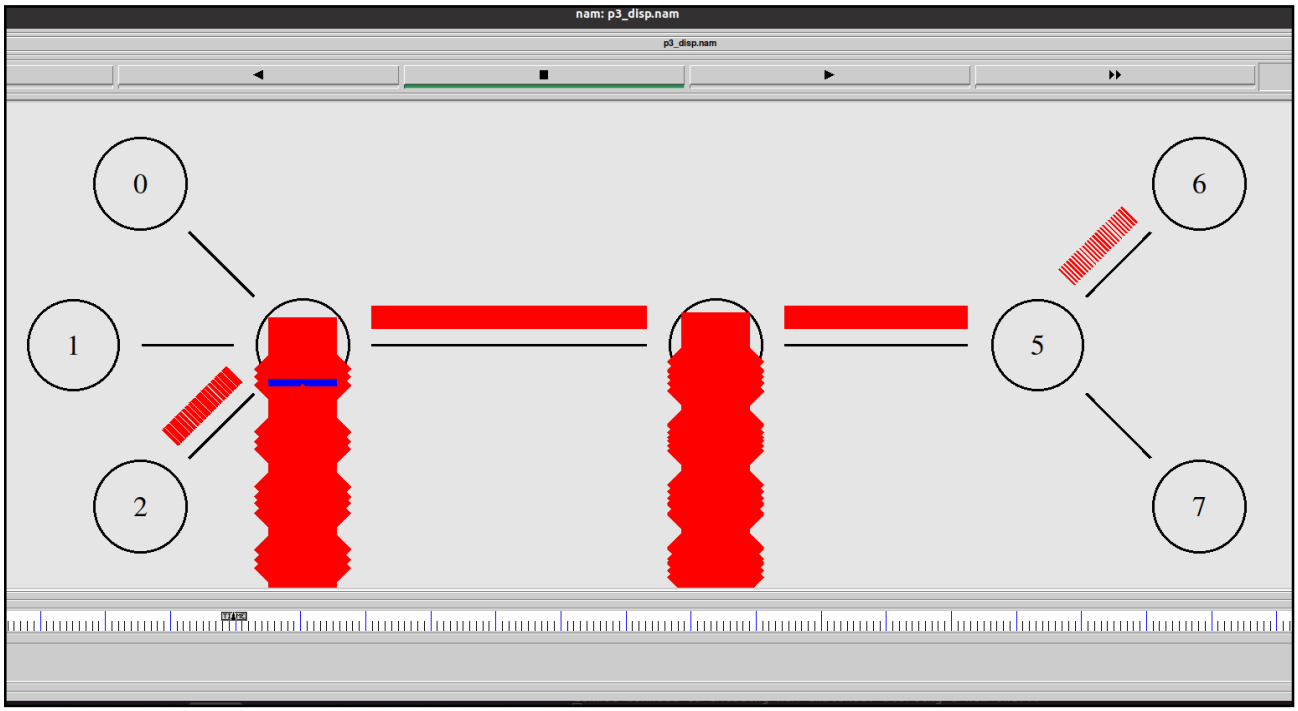
**$ns at 24.0 "finish"**

**# Run the simulation**
**$ns run**

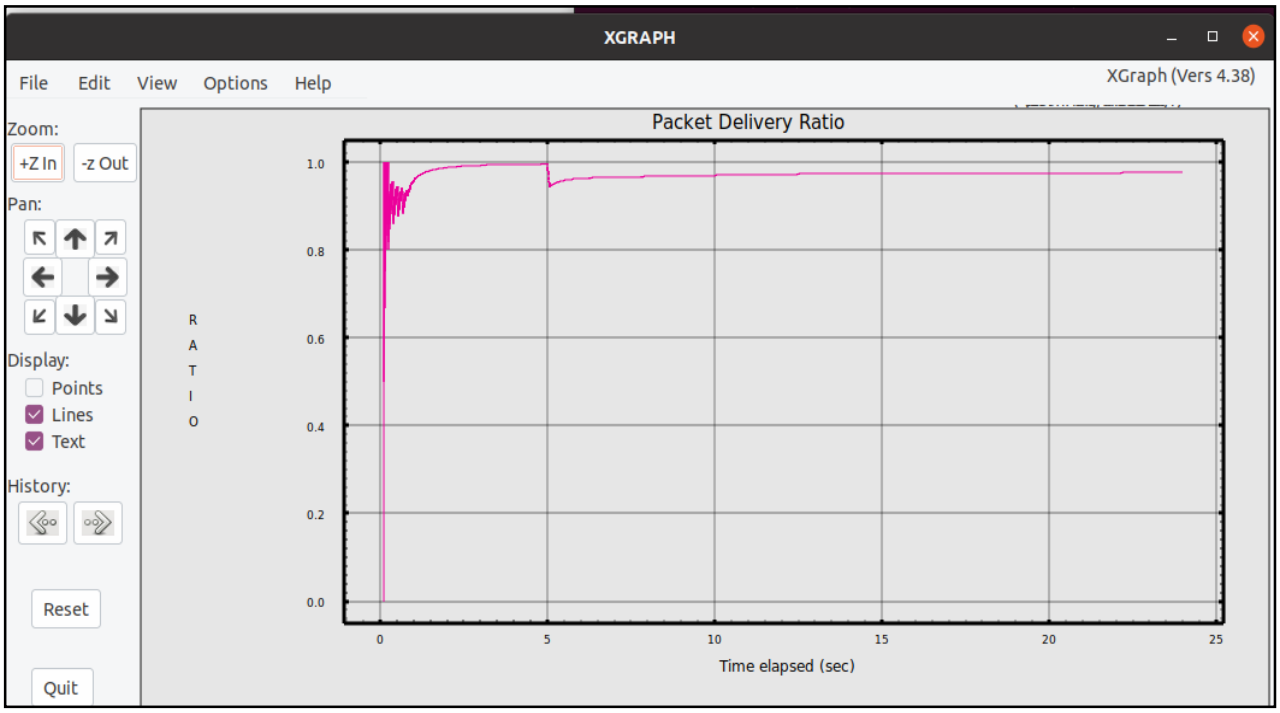
## Terminal Output



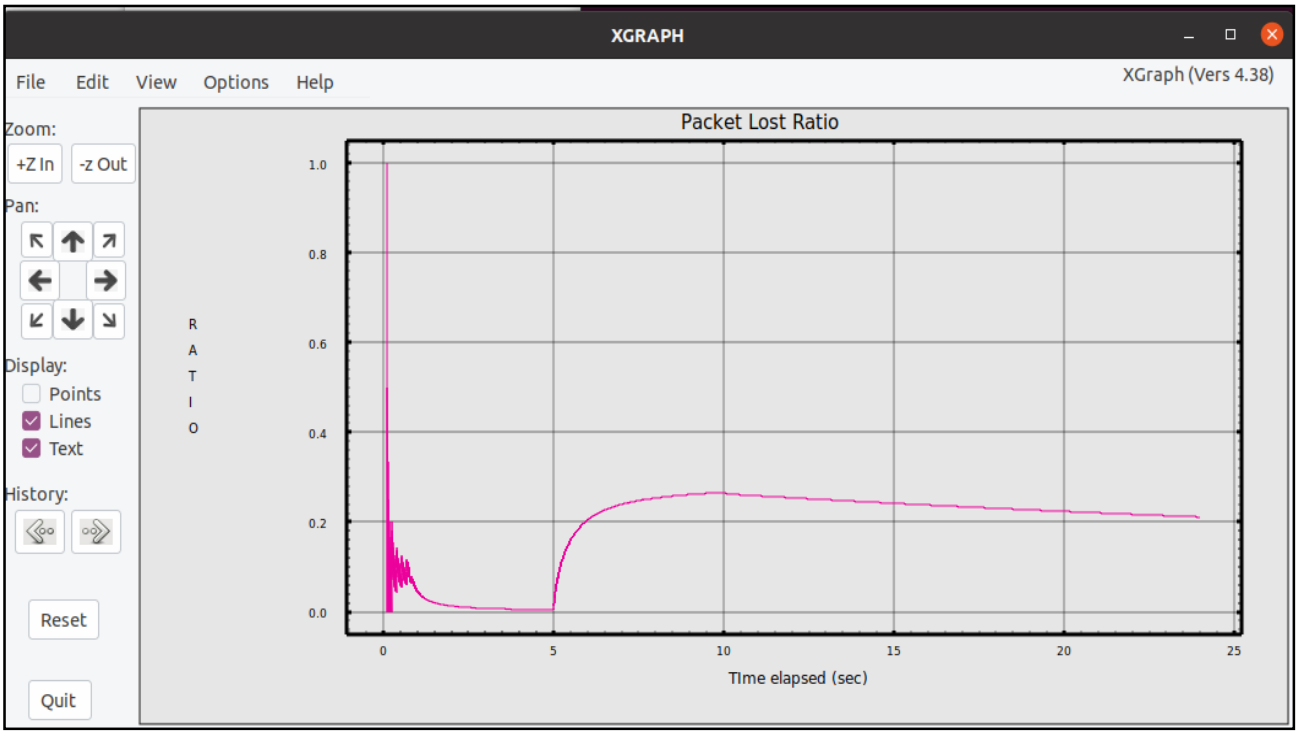
## NAM Output



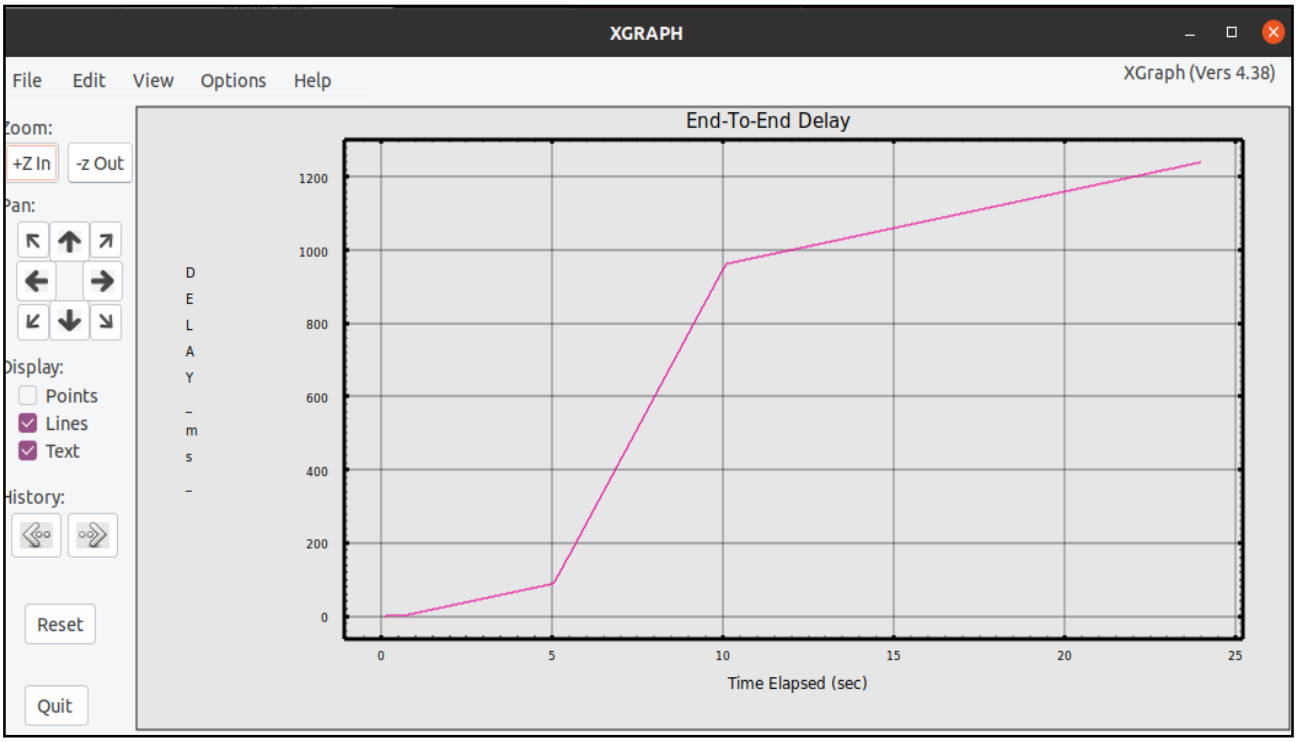

## Packet Delivery Ratio



## Packet Loss Ratio




## End-To-End Delay



## Future Works

## References


1) https://nodejs.org/en/
2) https://www.mongodb.com/
3) http://www.passportjs.org/
4) https://expressjs.com/
5) https://www.npmjs.com/