

**HKBK COLLEGE OF ENGINEERING**  
(Affiliated to VTU, Belagavi and Approved by AICTE)  
**DEPARTMENT OF INFORMATION SCIENCE &  
ENGINEERING**  
**NBA Accredited Programme**



**LABORATORY MANUAL**  
**COMPUTER NETWORK LABORATORY**  
[As per Choice Based Credit System (CBCS) scheme]  
(Effective from the academic year 2021 -2022)  
**18CSL57**

**PREPARED BY**  
Prof. Priya J  
Prof. Bibiana Jennifer J



## **HKBK COLLEGE OF ENGINEERING**

**Nagawara, Bangaluru -560 045**

<https://hkbk.edu.in>

## **HKBK COLLEGE OF ENGINEERING**

(Affiliated to VTU, Belgaum and Approved by AICTE)

## **DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**

### **Mission and Vision of the Institution**

#### **Mission**

- To achieve academic excellence in science, engineering and technology through dedication to duty, innovation in teaching and faith in human values.
- To enable our students to develop into outstanding professional with high ethical standards to face the challenges of 21st century.
- To provide educational opportunities to the deprived and weaker section of the society to uplift their socio-economic status.

#### **Vision**

To empower students through wholesome education and enable the students to develop into highly qualified and trained professionals with ethics and emerge as responsible citizen with broad outlook to build a vibrant nation.

### **Mission and Vision of the ISE Department**

#### **Mission**

- To provide excellent technical knowledge and computing skills to make the graduates globally competitive with professional ethics.
- To involve in research activities and be committed to lifelong learning to make positive contributions to the society.

#### **Vision**

To advance the intellectual capacity of the nation and the international community by imparting knowledge to graduates who are globally recognized as innovators, entrepreneur and competent professionals.



**HKBK COLLEGE OF ENGINEERING**  
(Affiliated to VTU, Belgaum and Approved by AICTE)  
**DEPARTMENT OF INFORMATION SCIENCE &  
ENGINEERING**

**Program Educational Objectives**

<b>PEO-1</b>	To provide students with a strong foundation in engineering fundamentals and in the computer science and engineering to work in the global scenario.
<b>PEO-2</b>	To provide sound knowledge of programming and computing techniques and good communication and interpersonal skills so that they will be capable of analyzing, designing and building innovative software systems.
<b>PEO-3</b>	To equip students in the chosen field of engineering and related fields to enable him to work in multidisciplinary teams.
<b>PEO-4</b>	To inculcate in students professional, personal and ethical attitude to relate engineering issues to broader social context and become responsible citizen.
<b>PEO-5</b>	To provide students with an environment for life-long learning which allow them to successfully adapt to the evolving technologies throughout their professional carrier and face the global challenges.

**Program Outcomes**

<b>a.</b>	<b>Engineering Knowledge:</b> Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
<b>b.</b>	<b>Problem Analysis: Identify,</b> formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences
<b>c.</b>	<b>Design/ Development of Solutions:</b> Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.
<b>d.</b>	<b>Conduct investigations of complex problems</b> using research-based knowledge and research methods including design of experiments, analysis

	and interpretation of data and synthesis of information to provide valid conclusions.
<b>e.</b>	<b>Modern Tool Usage:</b> Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
<b>f.</b>	<b>The Engineer and Society:</b> Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.
<b>g.</b>	<b>Environment and Sustainability:</b> Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate knowledge of and need for sustainable development.
<b>h.</b>	<b>Ethics:</b> Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

<b>i.</b>	<b>Individual and Team Work:</b> Function effectively as an individual, and as a member or leader in diverse teams and in multi disciplinary settings.
<b>j.</b>	<b>Communication:</b> Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.
<b>k.</b>	<b>Life-long Learning:</b> Recognize the need for and have the preparation and ability to engage in independent and life- long learning in the broadest context of technological change.
<b>l.</b>	<b>Project Management and Finance:</b> Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
<b>Programme Specific Outcomes</b>	
<b>m.</b>	<b>Problem-Solving Skills:</b> An ability to investigate and solve a problem by analysis, interpretation of data, design and implementation through appropriate techniques, tools and skills.
<b>n.</b>	<b>Professional Skills:</b> An ability to apply algorithmic principles, computing skills and computer science theory in the modelling and design of computer based systems.

o.	<b>Entrepreneurial Ability:</b> An ability to apply design, development principles and management skills in the construction of software product of varying complexity to become an entrepreneur
----	--



**HKBK College of Engineering**  
**Department of Information Science and Engineering**  
**Bangalore-560045**  
**Computer Network Laboratory (18CSL57)**

**Course objectives:** This course will enable students to

- Demonstrate operation of network and its management commands
- Simulate and demonstrate the performance of GSM and CDMA
- Implement data link layer and transport layer protocols.

**Description (If any):** For the experiments below modify the topology and parameters set for the experiment and take multiple rounds of reading and analyze the results available in log files. Plot necessary graphs and conclude. Use NS2/NS3.

**Lab Experiments:**  
**PART A**

1. Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.
2. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.
4. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.
5. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.
6. Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.

**PART B**  
**Implement the following in Java:**

1. Write a program for error detecting code using CRC-CCITT (16- bits).
2. Write a program to find the shortest path between vertices using bellman-ford algorithm.
3. Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.
4. Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.
5. Write a program for simple RSA algorithm to encrypt and decrypt the data.
6. Write a program for congestion control using leaky bucket algorithm.

**Study Experiment / Project: NIL**

**Course outcomes:** The students should be able to:

- Analyze and Compare various networking protocols.
- Demonstrate the working of different concepts of networking.
- Implement, analyze and evaluate networking protocols in NS2 / NS3

**Conduction of Practical Examination:**

1. All laboratory experiments are to be included for practical examination.
2. Students are allowed to pick one experiment from part A and part B with lot.
3. Strictly follow the instructions as printed on the cover page of answer script
4. Marks distribution:

Procedure + Conduction + Viva: 100

**Part A: 8+35+7 =50**

**Part B: 8+35+7 =50**

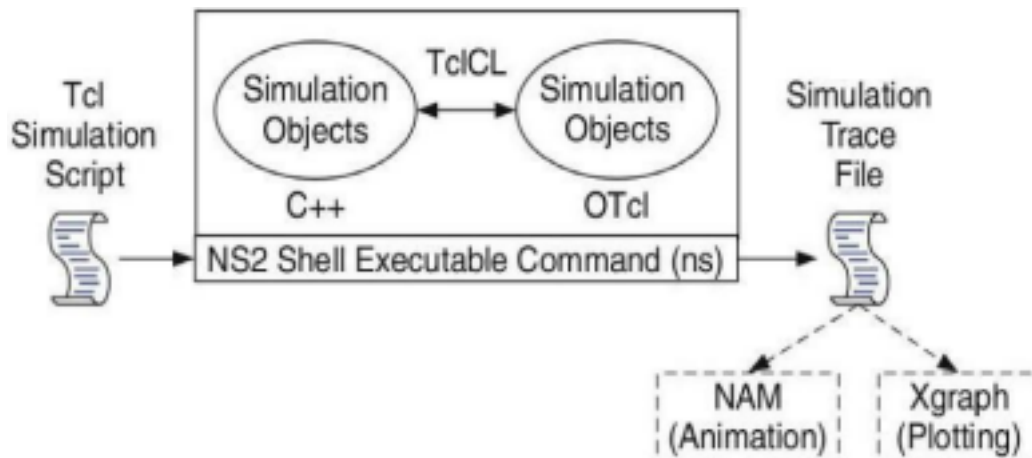
5. Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.

## **Part A**

### **Introduction to NS-2**

- Widely known as NS2, is simply an event driven simulation tool.
- 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.

#### **Basic Architecture of NS2**



#### **Tcl scripting**

- Tcl is a general purpose scripting language. [Interpreter]
- Tcl runs on most of the platforms such as Unix, Windows, and Mac.
- The strength of Tcl is its simplicity.
- It is not necessary to declare a data type for variable prior to the usage.

#### **Basics of TCL**

Syntax: command arg1 arg2 arg3

- **Hello World!**

```
puts stdout{Hello,
World!} Hello, World!
```

- **Variables Command Substitution**

```
set a 5 set len [string length foobar]
set b $a set len [expr [string length foobar] + 9]
```

- **Simple**

```
Arithmetic
expr 7.2 / 4
```

- **Procedures**

```
proc Diag {a b} {
set c [expr sqrt($a * $a + $b * $b)]
return $c }
puts "Diagonal of a 3, 4 right triangle is [Diag 3 4]"
Output: Diagonal of a 3, 4 right triangle is 5.0
```

- **Loops**

```
while{$i < $n} { for {set i 0} {$i < $n} {incr i} {
.....
} }
```

## **Wired TCL Script Components**

- Create the event scheduler
- Open new files & turn on the tracing
- Create the nodes
- Setup the links
- Configure the traffic type (e.g., TCP, UDP, etc)
- Set the time of traffic generation (e.g., CBR, FTP)
- Terminate the NS **Simulator**

## **Initialization and Termination of TCL Script in NS-2**

An ns simulation starts with the command

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

Which is thus the first line in the tcl script? This line declares a new variable as using the set command, you can call this variable as you wish, In general people declares it as ns because it is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the



reserved word new.

In order to have output files with data on the simulation (trace files) or files used for visualization (nam files), we need to create the files using “open” command:

### #Open the Trace file

```
set tracefile1 [open out.tr w]
$ns trace-all $tracefile1
```

### #Open the NAM trace file

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

The above creates a data trace file called “out.tr” and a nam visualization trace file called “out.nam”. Within the tcl script, these files are not called explicitly by their names, but instead by pointers that are declared above and called “tracefile1” and “namfile” respectively. Remark that they begins with a # symbol. The second line open the file “out.tr” to be used for writing, declared with the letter “w”. The third line uses a simulator method called trace-all that have as parameter the name of the file where the traces will go.

The last line tells the simulator to record all simulation traces in NAM input format. It also gives the file name that the trace will be written to later by the command \$ns flush-trace. In our case, this will be the file pointed at by the pointer “\$namfile”, i.e the file “out.tr”.

The termination of the program is done using a “finish” procedure.

### #Define a ‘finish’ procedure

```
Proc finish { } {
global ns tracefile1 namfile
$ns flush-trace
Close $tracefile1
Close $namfile
Exec nam out.nam &
Exit 0
```

The word proc declares a procedure in this case called **finish** and without arguments. The word **global** is used to tell that we are using variables declared outside the procedure. The simulator method “**flush-trace**” will dump the traces on the respective files. The tcl command “**close**” closes the trace files defined before and **exec** executes the nam program for visualization. The command **exit** will ends the application and return the number 0 as status to the system. Zero is the default for a clean exit. Other values can be used to say that is a exit because something fails.

At the end of ns program we should call the procedure “finish” and specify at what

time the termination should occur. For example,

**\$ns at 125.0 "finish"**

will be used to call **"finish"** at time 125sec. Indeed, the **at** method of the simulator allows us to schedule events explicitly.

The simulation can then begin using the command

**\$ns run**

### **Definition of a network of links and nodes**

The way to define a node is

**set n0 [\$ns node]**

We created a node that is printed by the variable n0. When we shall refer to that node in the script we shall thus write \$n0.

Once we define several nodes, we can define the links that connect them. An example of a definition of a link is:

**\$ns duplex-link \$n0 \$n2 10Mb 10ms DropTail**

Which means that \$n0 and \$n2 are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10Mb per sec for each direction. To define a directional link instead of a bi-directional one, we should replace "duplex-link" by "simplex-link".

In NS, an output queue of a node is implemented as a part of each link whose input is that node. The definition of the link then includes the way to handle overflow at that queue. In our case, if the buffer capacity of the output queue is exceeded then the last packet to arrive is dropped. Many alternative options exist, such as the RED (Random Early Discard) mechanism, the FQ (Fair Queuing), the DRR (Deficit Round Robin), the stochastic Fair Queuing (SFQ) and the CBQ (which including a priority and a round-robin scheduler).

In ns, an output queue of a node is implemented as a part of each link whose input is that node. We should also define the buffer capacity of the queue related to each link. An example would be:

**#set Queue Size of link (n0-n2) to 20**

**\$ns queue-limit \$n0 \$n2 20**

### **Agents and Applications**

We need to define routing (sources, destinations) the agents (protocols) the application that use them.

### **FTP over TCP**

TCP is a dynamic reliable congestion control protocol. It uses Acknowledgements created by the destination to know whether packets are well received.

There are number variants of the TCP protocol, such as Tahoe, Reno, NewReno, Vegas. The type of agent appears in the first line:

```
set tcp [new Agent/TCP]
```

The command **\$ns attach-agent \$n0 \$tcp** defines the source node of the tcp connection. The command

```
set sink [new Agent /TCPSink]
```

Defines the behaviour of the destination node of TCP and assigns to it a pointer called sink. **#Setup a UDP connection**

```
set udp [new Agent/UDP]  
$ns attach-agent $n1 $udp  
set null [new Agent/Null]  
$ns attach-agent $n5 $null  
$ns connect $udp $null  
$udp set fid_2
```

**#setup a CBR over UDP connection**

```
set cbr [new Application/Traffic/CBR]  
$cbr attach-agent $udp  
$cbr set packetSize_ 100  
$cbr set rate_ 0.01Mb  
$cbr set random_ false
```

Above shows the definition of a CBR application using a UDP agent

The command **\$ns attach-agent \$n4 \$sink** defines the destination node. The command **\$ns connect \$tcp \$sink** finally makes the TCP connection between the source and destination nodes.

TCP has many parameters with initial fixed defaults values that can be changed if mentioned explicitly. For example, the default TCP packet size has a size of 1000bytes. This can be changed to another value, say 552bytes, using the command **\$tcp set packetSize\_ 552**.

When we have several flows, we may wish to distinguish them so that we can identify them with different colors in the visualization part. This is done by the command **\$tcp set fid\_ 1** that assigns to the TCP connection a flow identification of

“1”.We shall later give the flow identification of “2” to the UDP connection.

### CBR over UDP

A UDP source and destination is defined in a similar way as in the case of TCP.

Instead of defining the rate in the command `$cbr set rate_ 0.01Mb`, one can define the time interval between transmission of packets using the command.

**`$cbr set interval_ 0.005`**

The packet size can be set to some value using

**`$cbr set packetSize_ <packet size>`**

### Scheduling Events

NS is a discrete event based simulation. The tcp script defines when event should occur. The initializing command `set ns [new Simulator]` creates an event scheduler, and events are then scheduled using the format:

**`$ns at <time> <event>`**

The scheduler is started when running ns that is through the command `$ns run`.

The beginning and end of the FTP and CBR application can be done through the following command

**`$ns at 0.1 “$cbr start”`**

**`$ns at 1.0 “$ftp start”`**

**`$ns at 124.0 “$ftp stop”`**

### Structure of Trace Files

When tracing into an output ASCII file, the trace is organized in 12 fields as follows in fig shown below, The meaning of the fields are:

Event	Time	From Node	To Node	PKT Type	PKT Size	Flags	Fid	Src Addr	Dest Addr	Seq NUM	Pkt id
-------	------	--------------	------------	-------------	-------------	-------	-----	-------------	--------------	------------	-----------

1. The first field is the event type. It is given by one of four possible symbols r, +, -, d which correspond respectively to receive (at the output of the link), enqueued, dequeued and dropped.
2. The second field gives the time at which the event occurs.
3. Gives the input node of the link at which the event occurs.

4. Gives the output node of the link at which the event occurs.
5. Gives the packet type (eg CBR or TCP)
6. Gives the packet size
7. Some flags
8. This is the flow id (fid) of IPv6 that a user can set for each flow at the input OTcl script one can further use this field for analysis purposes; it is also used when specifying stream color for the NAM display.
9. This is the source address given in the form of “node.port”.
10. This is the destination address, given in the same form.
11. This is the network layer protocol’s packet sequence number. Even though UDP implementations in a real network do not use sequence number, ns keeps track of UDP packet sequence number for analysis purposes
12. The last field shows the unique id of the packet.

## **XGRAPH**

The xgraph program draws a graph on an x-display given data read from either data file or from standard input if no files are specified. It can display upto 64 independent data sets using different colors and line styles for each set. It annotates the graph with a title, axis labels, grid lines or tick marks, grid labels and a legend.

### **Syntax:**

**Xgraph [options] file-name**

Options are listed here

**`/-bd <color> (Border)`**

This specifies the border color of the xgraph window.

**`/-bg <color> (Background)`**

This specifies the background color of the xgraph window.

**`/-fg<color> (Foreground)`**

This specifies the foreground color of the xgraph window.

**`/-lf <fontname> (LabelFont)`**

All axis labels and grid labels are drawn using this font.

**`/-t<string> (Title Text)`**

This string is centered at the top of the graph.

**`/-x <unit name> (XunitText)`**

This is the unit name for the x-axis. Its default is “X”.

**`/-y <unit name> (YunitText)`**

This is the unit name for the y-axis. Its default is “Y”.

### **Awk- An Advanced**

awk is a programmable, pattern-matching, and processing tool available in UNIX.

It works equally well with text and numbers.

awk is not just a command, but a programming language too. In other words, awk utility is a pattern scanning and processing language. It searches one or more files to see if they

contain lines that match specified patterns and then perform associated actions, such as writing the line to the standard output or incrementing a counter each time it finds a match.

Syntax: **awk option 'selection\_criteria {action}' file(s)**

Here, selection\_criteria filters input and select lines for the action component to act upon. The selection\_criteria is enclosed within single quotes and the action within the curly braces. Both the selection\_criteria and action forms an awk program.

**Example: \$ awk '/manager/ {print}' emp.lst**

### **Variables**

Awk allows the user to use variables of their choice. You can now print a serial number, using the variable count, and apply it to those directors drawing a salary exceeding 6700:

```
$ awk -F'|' '$3 == "director" && $6 > 6700 {  
count=count+1  
printf " %3f %20s %-12s %d\n", count,$2,$3,$6 }' empn.lst
```

### **THE -f OPTION: STORING awk PROGRAMS IN A FILE**

You should hold large awk programs in separate files and provide them with the awk extension for easier identification. Let's first store the previous program in the file empawk.awk:

```
$ cat empawk.awk
```

Observe that this time we haven't used quotes to enclose the awk program.

You can now use awk with the *-f filename* option to obtain the same output:

```
Awk -F'|' -f empawk.awk empn.lst
```

### **THE BEGIN AND END SECTIONS**

Awk statements are usually applied to all lines selected by the address, and if there are no addresses, then they are applied to every line of input. But, if you have to print something before processing the first line, for example, a heading, then the BEGIN section can be used gainfully. Similarly, the end section is useful in printing some totals after processing is over. The BEGIN and END sections are optional and take the form

**BEGIN {action}**

**END {action}**

These two sections, when present, are delimited by the body of the awk program. You can use them to print a suitable heading at the beginning and the average salary at the end. **BUILT-IN VARIABLES**

Awk has several built-in variables. They are all assigned automatically, though it is also possible for a user to reassign some of them. You have already used NR, which signifies the record number of the current line. We'll now have a brief look at some of the other variable. ***The FS Variable:*** as stated elsewhere, awk uses a contiguous string of spaces as the default field delimiter. FS redefines this field separator, which in the sample database happens to be the |. When used at all, it must occur in the BEGIN section so that the body of the program knows its value before it starts processing:

```
BEGIN {FS="|"} }
```

This is an alternative to the -F option which does the same thing.

***The OFS Variable:*** when you used the print statement with comma-separated arguments, each argument was separated from the other by a space. This is awk's default output field separator, and can be reassigned using the variable OFS in the BEGIN section:

```
BEGIN { OFS="~" }
```

When you reassign this variable with a ~ (tilde), awk will use this character for delimiting the print arguments. This is a useful variable for creating lines with delimited fields.

***The NF variable:*** NF comes in quite handy for cleaning up a database of lines that don't contain the right number of fields. By using it on a file, say emp.lst, you can locate those lines not having 6 fields, and which have crept in due to faulty data entry: **\$awk 'BEGIN {FS = "|"} NF!=6 {  
Print "Record No ", NR, "has", "fields"}' emp.lst**

***The FILENAME Variable:*** FILENAME stores the name of the current file being processed. Like grep and sed, awk can also handle multiple filenames in the command line. By default, awk doesn't print the filename, but you can instruct it to do so:

```
'$6<4000 {print FILENAME, $0}'
```

With FILENAME, you can devise logic that does different things depending on the file that is processed.

## PART-A

1. Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.

### l1.tcl

```
set ns [new Simulator]
set f [open l1.tr w]
$ns trace-all $f
set nf [open l1.nam w]
```

```
$ns namtrace-all $nf
```

```
proc finish {} {
    global f nf ns
    $ns flush-trace
    close $f
    close $nf
    exec nam l1.nam &
    exit 0
}
```

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$n0 label "Source"
$n2 label "Sink"
$ns color 1 red
$ns color 2 yellow
```

```
$ns duplex-link $n0 $n1 100Mb 10ms DropTail
$ns duplex-link $n1 $n2 10Mb 5ms DropTail
$ns queue-limit $n1 $n2 10
```

```
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
```



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

```
set n [new Agent/Null]
$ns attach-agent $n2 $n
```

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 5000
$cbr0 set interval_ 0.001
```

```
$ns connect $udp0 $n
```

```
$udp0 set class_ 1
```

```
$ns at 0.1 "$cbr0 start"
```

```
$ns at 1.8 "$cbr0 stop"
```

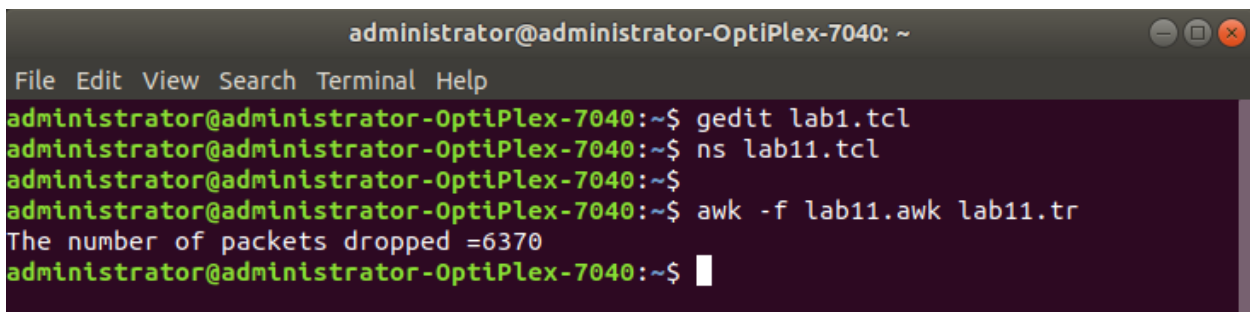
```
$ns at 2.0 "finish"
```

```
$ns run
```

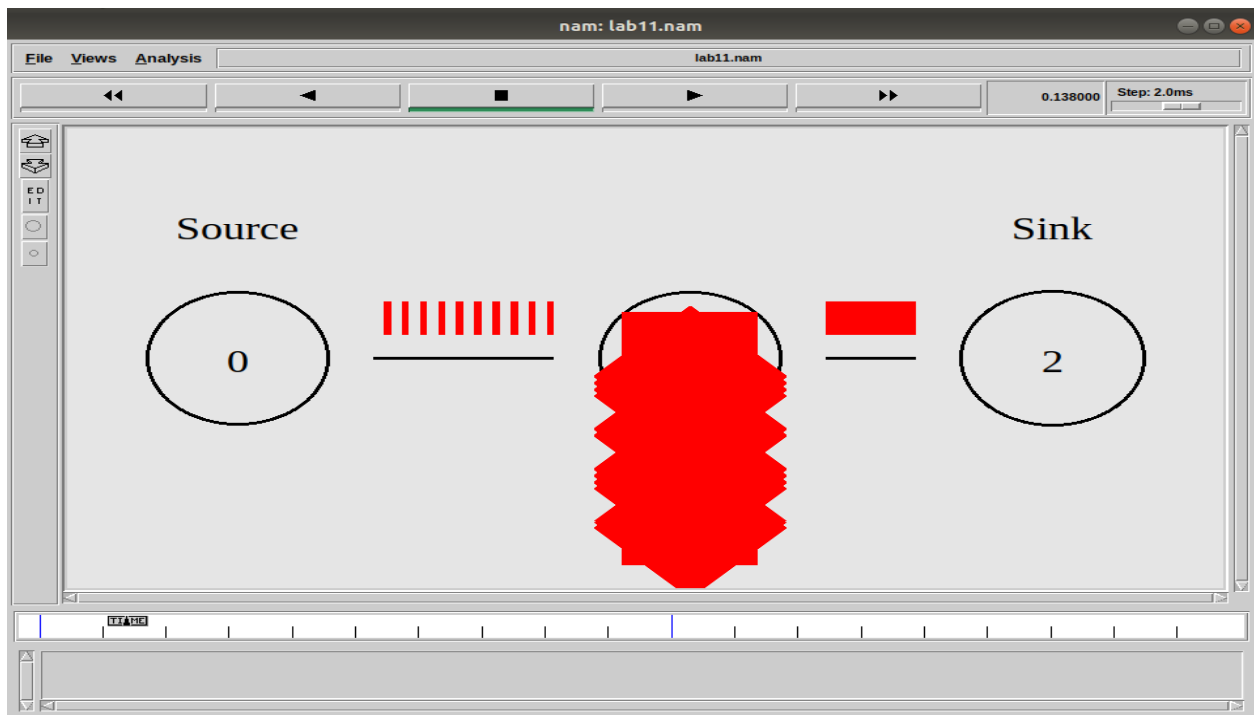
### l1.awk

```
BEGIN{ c=0;}
{
  if($1=="d")
  { c++;
  }
}
END{ printf("The number of packets dropped =%d\n",c); }
```

### Screenshot



```
administrator@administrator-OptiPlex-7040: ~
File Edit View Search Terminal Help
administrator@administrator-OptiPlex-7040:~$ gedit lab1.tcl
administrator@administrator-OptiPlex-7040:~$ ns lab11.tcl
administrator@administrator-OptiPlex-7040:~$
administrator@administrator-OptiPlex-7040:~$ awk -f lab11.awk lab11.tr
The number of packets dropped =6370
administrator@administrator-OptiPlex-7040:~$
```



```
+ 0.1 0 1 cbr 1000 ----- 1 0.0 2.0 0 0
- 0.1 0 1 cbr 1000 ----- 1 0.0 2.0 0 0
+ 0.1 0 1 cbr 1000 ----- 1 0.0 2.0 1 1
+ 0.1 0 1 cbr 1000 ----- 1 0.0 2.0 2 2
+ 0.1 0 1 cbr 1000 ----- 1 0.0 2.0 3 3
+ 0.1 0 1 cbr 1000 ----- 1 0.0 2.0 4 4
| 0 10000 0 1 cbr 1000 ----- 1 0 0 2 0 1 1
```

2. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

### l2.tcl

```
set ns [ new Simulator ]
set nf [ open l2.nam w ]
$ns namtrace-all $nf
set tf [ open l2.tr w ]
$ns trace-all $tf
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
```

```
$ns duplex-link $n0 $n4 1005Mb 1ms DropTail
$ns duplex-link $n1 $n4 50Mb 1ms DropTail
$ns duplex-link $n2 $n4 2000Mb 1ms DropTail
$ns duplex-link $n3 $n4 200Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 1ms DropTail
```

```
set p0 [new Agent/Ping]
$ns attach-agent $n0 $p0
$p0 set packetSize_ 50000
$p0 set interval_ 0.0001
```

```
set p1 [new Agent/Ping]
$ns attach-agent $n1 $p1
```

```
set p2 [new Agent/Ping]
$ns attach-agent $n2 $p2
$p2 set packetSize_ 30000
$p2 set interval_ 0.00001
```

```
set p3 [new Agent/Ping]
$ns attach-agent $n3 $p3
```

```
set p5 [new Agent/Ping]
```

**\$ns attach-agent \$n5 \$p5**

**\$ns queue-limit \$n0 \$n4 5**

**\$ns queue-limit \$n2 \$n4 3**

**\$ns queue-limit \$n4 \$n5 2**

```
Agent/Ping instproc recv {from rtt} {  
  $self instvar node_  
  puts "node [$node_id]received answer from $from with round  
  trip time $rtt msec"  
}  
$ns connect $p0 $p5  
$ns connect $p2 $p3  
proc finish { } {  
  global ns nf tf  
  $ns flush-trace  
  close $nf  
  close $tf  
  exec nam l2.nam &  
  exit 0  
}  
$ns at 0.1 "$p0 send"  
$ns at 0.2 "$p0 send"  
$ns at 0.3 "$p0 send"  
$ns at 0.4 "$p0 send"  
$ns at 0.5 "$p0 send"  
$ns at 0.6 "$p0 send"  
$ns at 0.7 "$p0 send"  
$ns at 0.8 "$p0 send"  
$ns at 0.9 "$p0 send"  
$ns at 1.0 "$p0 send"  
$ns at 0.1 "$p2 send"  
$ns at 0.2 "$p2 send"  
$ns at 0.3 "$p2 send"  
$ns at 0.4 "$p2 send"  
$ns at 0.5 "$p2 send"  
$ns at 0.6 "$p2 send"  
$ns at 0.7 "$p2 send"  
$ns at 0.8 "$p2 send"  
$ns at 0.9 "$p2 send"  
$ns at 1.0 "$p2 send"
```

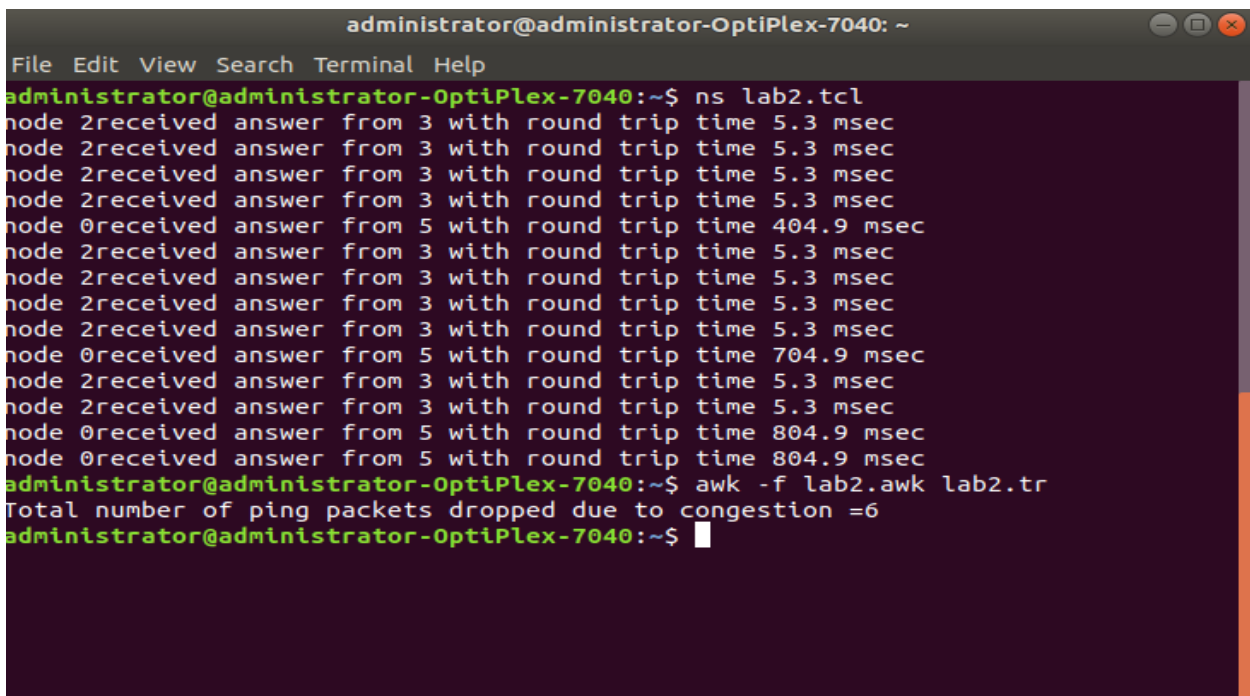
\$ns at 2.0 "finish"

\$ns run

## l2.awk

```
BEGIN{
drop=0;
}
{
if($1=="d" )
{
drop++;
}
}
END{
printf("Total number of %s packets dropped due to
congestion=%d\n",$5,drop);
}
```

## Screenshot

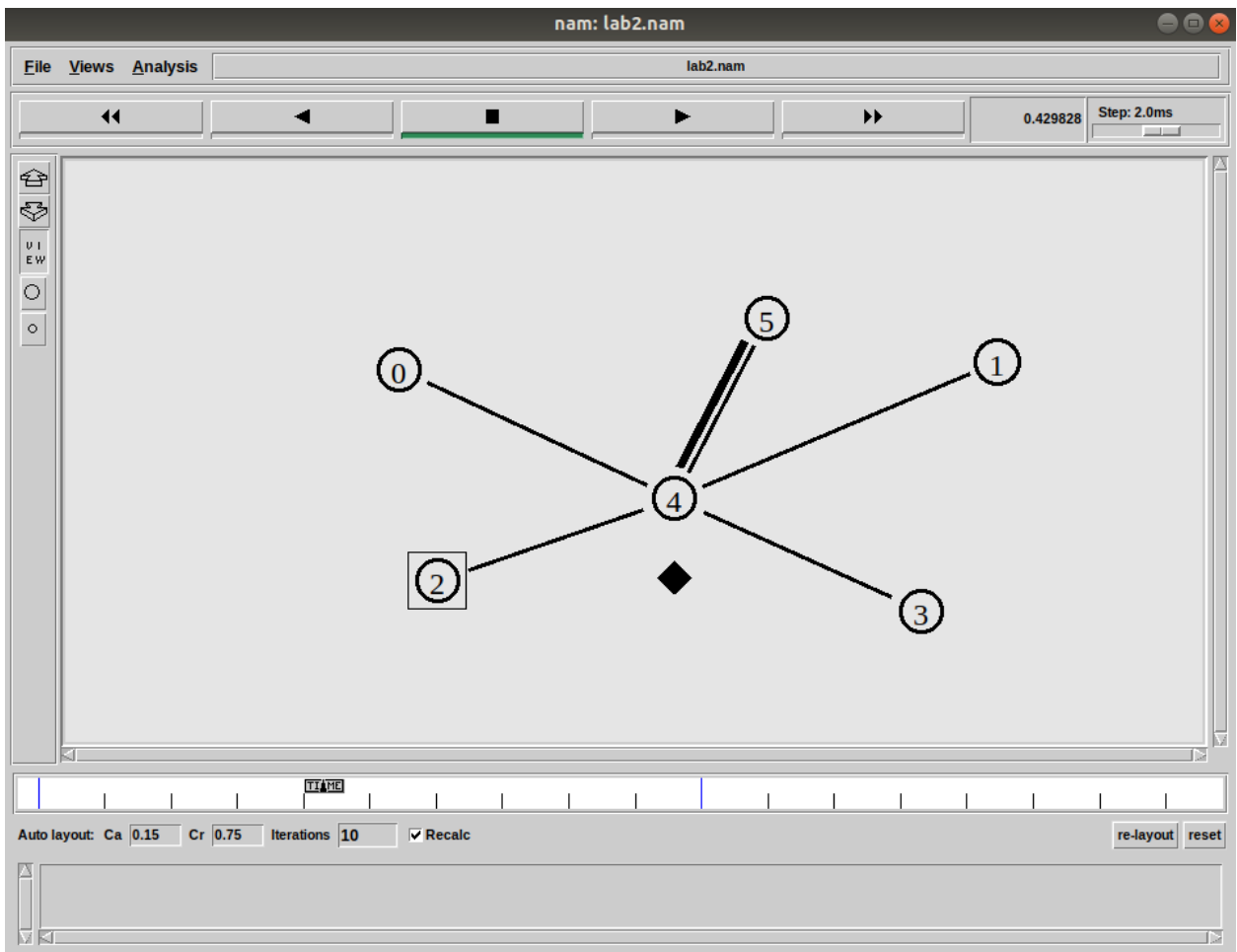


The screenshot shows a terminal window titled "administrator@administrator-OptiPlex-7040: ~". The terminal output displays the results of a network simulation using the 'ns' command. It shows several lines of output indicating received answers from nodes 2 and 0, with round trip times ranging from 5.3 msec to 804.9 msec. The output is followed by an 'awk' command that processes the data and prints the total number of ping packets dropped due to congestion, which is 6.

```
administrator@administrator-OptiPlex-7040: ~
File Edit View Search Terminal Help
administrator@administrator-OptiPlex-7040:~$ ns lab2.tcl
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 404.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 704.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 804.9 msec
node 0received answer from 5 with round trip time 804.9 msec
administrator@administrator-OptiPlex-7040:~$ awk -f lab2.awk lab2.tr
Total number of ping packets dropped due to congestion =6
administrator@administrator-OptiPlex-7040:~$
```

Open lab2.tr Save

```
+ 0.1 0 4 ping 50000 ----- 0 0.0 5.0 -1 0
- 0.1 0 4 ping 50000 ----- 0 0.0 5.0 -1 0
+ 0.1 2 4 ping 30000 ----- 0 2.0 3.0 -1 1
- 0.1 2 4 ping 30000 ----- 0 2.0 3.0 -1 1
r 0.10112 2 4 ping 30000 ----- 0 2.0 3.0 -1 1
+ 0.10112 4 3 ping 30000 ----- 0 2.0 3.0 -1 1
- 0.10112 4 3 ping 30000 ----- 0 2.0 3.0 -1 1
r 0.101398 0 4 ping 50000 ----- 0 0.0 5.0 -1 0
+ 0.101398 4 5 ping 50000 ----- 0 0.0 5.0 -1 0
- 0.101398 4 5 ping 50000 ----- 0 0.0 5.0 -1 0
r 0.10332 4 3 ping 30000 ----- 0 2.0 3.0 -1 1
```



### **3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination**

#### **l3.tcl**

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

```
set tf [open l3.tr w]  
$ns trace-all $tf
```

```
set nf [open l3.nam w]  
$ns namtrace-all $nf
```

```
set n0 [$ns node]  
$n0 color "magenta"  
$n0 label "src1"
```

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

```
set n2 [$ns node]  
$n2 color "magenta"  
$n2 label "src2"
```

```
set n3 [$ns node]  
$n3 color "blue"  
$n3 label "dest2"
```

```
set n4 [$ns node]
```

```
set n5 [$ns node]  
$n5 color "blue"  
$n5 label "dest1"
```

```
$ns make-lan "$n0 $n1 $n2 $n3 $n4" 100Mb 100ms LL Queue/DropTail Mac/802_3
```

```
$ns duplex-link $n4 $n5 1Mb 1ms DropTail
```

```
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
```

```
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.0001
```

```
set sink5 [new Agent/TCPSink]
$ns attach-agent $n5 $sink5
```

```
$ns connect $tcp0 $sink5
```

```
set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
```

```
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set packetSize_ 600
$ftp2 set interval_ 0.001
```

```
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
$ns connect $tcp2 $sink3
```

```
set file1 [open file1.tr w]
$tcp0 attach $file1
set file2 [open file2.tr w]
```

```
$tcp2 attach $file2
$tcp0 trace cwnd_
$tcp2 trace cwnd_
```



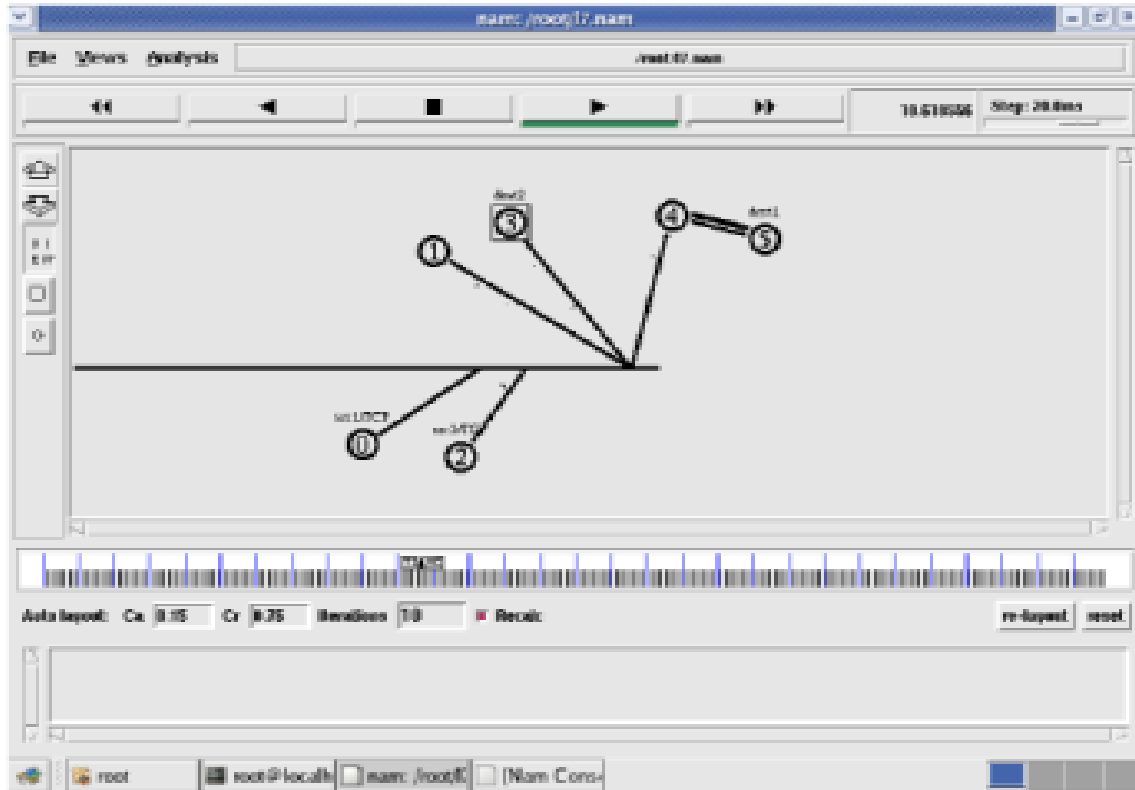
```
proc finish { } {  
  global ns nf tf  
  $ns flush-trace  
  close $tf  
  close $nf  
  exec nam l3.nam &  
  exit 0  
}
```

```
$ns at 0.1 "$ftp0 start"  
$ns at 5 "$ftp0 stop"  
$ns at 7 "$ftp0 start"  
$ns at 0.2 "$ftp2 start"  
$ns at 8 "$ftp2 stop"  
$ns at 14 "$ftp0 stop"  
$ns at 10 "$ftp2 start"  
$ns at 15 "$ftp2 stop"  
$ns at 16 "finish"
```

### **L3.awk (Awk file):**

```
BEGIN {  
  }  
  {  
    if($6=="cwnd_")  
      printf("%f\t%f\t\n",$1,$7);  
  }  
END {  
  }
```

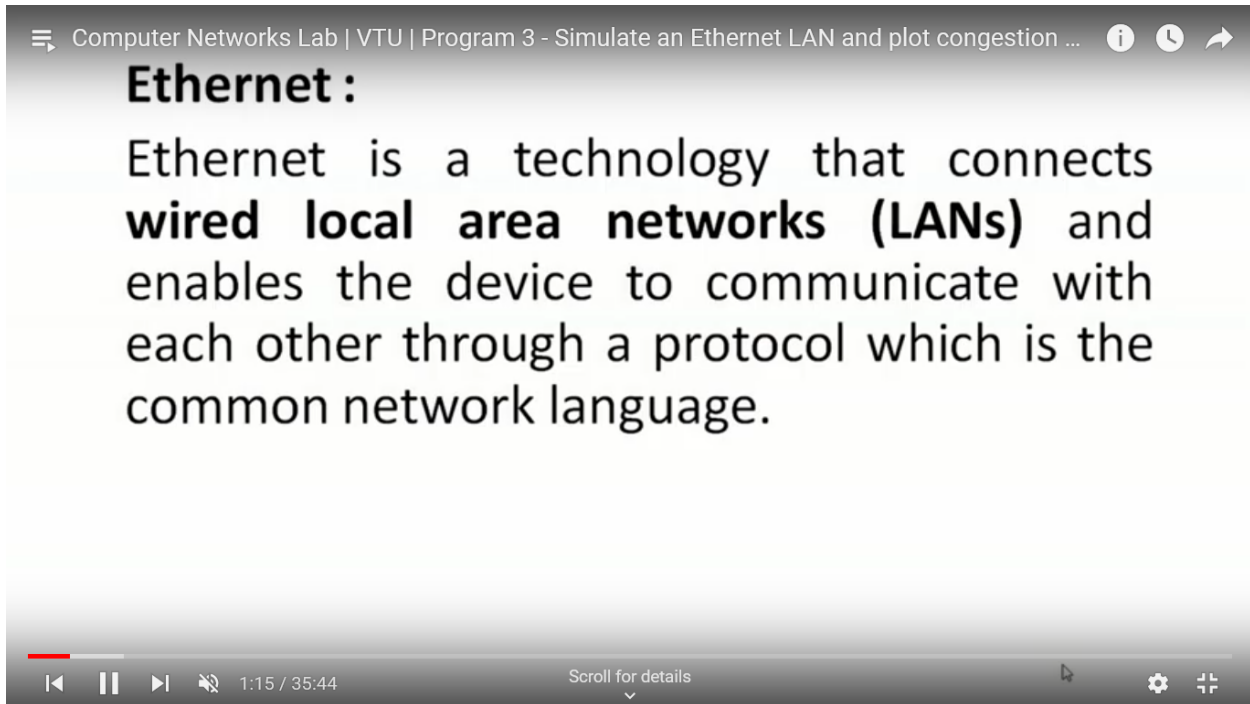
## Screenshot



### Steps For Execution:

- Open gedit editor and type program. Program name should have the extension “ .tcl ” `[root@localhost ~]# gedit l3.tcl`
- Save the program and close the file.
- Open gedit editor and type **awk** program. Program name should have the extension “**.awk** ” `[root@localhost ~]# gedit l3.awk`
- Save the program and close the file.
- Run the simulation program `[root@localhost~]# ns l3.tcl` ➤ Here “**ns**” indicates network simulator. We get the topology shown in the snapshot.
- Now press the play button in the simulation window and the simulation will begin.
- After simulation is completed run **awk** file to see the output ,  
`[root@localhost~]# awk -l3.awk file1.tr > a`  
`[root@localhost~]# awk -l3.awk file2.tr > b`  
`[root@localhost~]# xgraph a b`

➤ Here we are using the congestion window trace files i.e. **file1.tr** and **file2.tr** and we are redirecting the contents of those files to new files say **a** and **b** using **output redirection operator (>)**.



The screenshot shows a video player interface. At the top, the title bar reads "Computer Networks Lab | VTU | Program 3 - Simulate an Ethernet LAN and plot congestion ...". The main content area displays a slide with the heading "Ethernet :" and the text "Ethernet is a technology that connects **wired local area networks (LANs)** and enables the device to communicate with each other through a protocol which is the common network language." The video player controls at the bottom include a progress bar, play/pause button, volume icon, and a timestamp of "1:15 / 35:44". A "Scroll for details" link is also visible.

Computer Networks Lab | VTU | Program 3 - Simulate an Ethernet LAN and plot congestion ...

## Ethernet :

Ethernet is a technology that connects **wired local area networks (LANs)** and enables the device to communicate with each other through a protocol which is the common network language.

1:15 / 35:44

Scroll for details

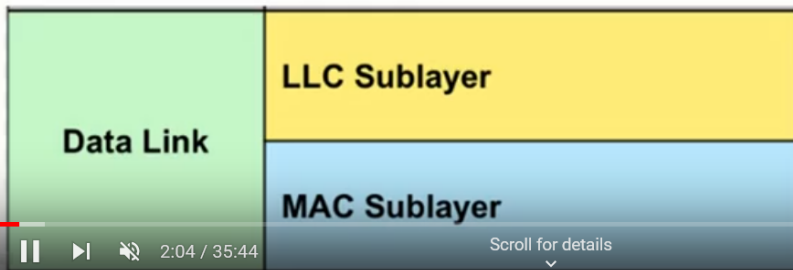
Ethernet is a wired LAN

It is a protocol present in Data link Layer

The data link layer consists of two sub layers:

Media Access Control (MAC)

Logical Link Control (LLC)



### **Congestion :**

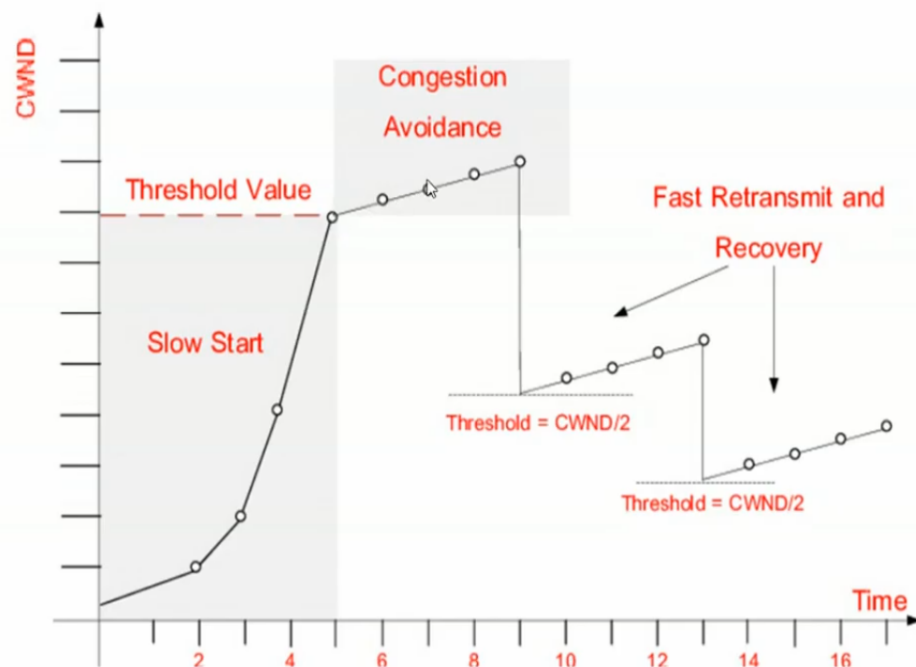
Network congestion is the reduced quality of service that occurs when a network node or link is carrying more data than it can handle.

## CWND and RWND

- Congestion Window (CWND) : cwnd is a TCP state variable that limits the amount of data the TCP can send into the network before receiving an ACK
- Receiver Window (RWND) : rwnd is a variable that advertises the amount of data that the destination side can receive.

Together, the two variables are used to regulate data flow in TCP connections, minimize congestion, and improve network performance.

## Slow Start and Congestion Avoidance in TCP



## Slow Start and Congestion Avoidance in TCP

- In slow start congestion control, TCP increases the window's size rapidly to reach the maximum transfer rate as fast as possible. This self-imposed window size increases as TCP confirms the network's ability to transmit the data without errors. However, this can only go up to a maximum advertised window (RWND).

**In this scenario, the sender uses two variables:**

- Congestion window with an initial value of one maximum segment size (MSS)
- The slow start threshold value (sssthresh) with an initial value equal to the receiver window.
- The congestion window increases exponentially during congestion control, and lineally during the congestion avoidance. Sslow start occurs when  $cwnd < sssthresh$  and congestion avoidance occurs when  $cwnd \geq sssthresh$
- Whenever a packet has been lost or we received 3 duplicate ACK, then the congestion has dropped to zero

4. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.

**p4.tcl**

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

```
set tf [open p4.tr w]
```

```
$ns trace-all $tf
```

```
set topo [new Topography]
```

```
$topo load_flatgrid 1000 1000
```

```
set nf [open p4.nam w]
```

```
$ns namtrace-all-wireless $nf 1000 1000
```

```
$ns node-config -adhocRouting DSDV \
```

```
    -llType LL \
```

```
    -macType Mac/802_11 \
```

```
    -ifqType Queue/DropTail \
```

```
    -ifqLen 50 \
```

```
    -phyType Phy/WirelessPhy \
```

```
    -channelType Channel/WirelessChannel \
```

```
    -propType Propagation/TwoRayGround \
```

```
    -antType Antenna/OmniAntenna \
```

```
    -topoInstance $topo \
```

**-agentTrace ON \**

**-routerTrace ON**

**create-god 3**

**set n0 [\$ns node]**

**set n1 [\$ns node]**

**set n2 [\$ns node]**

**\$n0 label "tcp0"**

**\$n1 label "sink1/tcp1"**

**\$n2 label "sink2"**

**\$n0 set X\_ 50**

**\$n0 set Y\_ 50**

**\$n0 set Z\_ 0**

**\$n1 set X\_ 100**

**\$n1 set Y\_ 100**

**\$n1 set Z\_ 0**

**\$n2 set X\_ 600**

**\$n2 set Y\_ 600**

**\$n2 set Z\_ 0**

**\$ns at 0.1 "\$n0 setdest 50 50 15"**

**\$ns at 0.1 "\$n1 setdest 100 100 25"**



**\$ns at 0.1 "\$n2 setdest 600 600 25"**

**set tcp0 [new Agent/TCP]**

**\$ns attach-agent \$n0 \$tcp0**

**set ftp0 [new Application/FTP]**

**\$ftp0 attach-agent \$tcp0**

**set sink1 [new Agent/TCPSink]**

**\$ns attach-agent \$n1 \$sink1**

**\$ns connect \$tcp0 \$sink1**

**set tcp1 [new Agent/TCP]**

**\$ns attach-agent \$n1 \$tcp1**

**set ftp1 [new Application/FTP]**

**\$ftp1 attach-agent \$tcp1**

**set sink2 [new Agent/TCPSink]**

**\$ns attach-agent \$n2 \$sink2**

**\$ns connect \$tcp1 \$sink2**

**\$ns at 5 "\$ftp0 start"**

**\$ns at 5 "\$ftp1 start"**

**\$ns at 100 "\$n1 setdest 550 550 15"**

```
$ns at 190 "$n1 setdest 70 70 15"
```

```
proc finish { } {
```

```
global ns nf tf
```

```
$ns flush-trace
```

```
exec nam p4.nam &
```

```
close $tf
```

```
exit 0
```

```
}
```

```
$ns at 250 "finish"
```

```
$ns run
```

**p4.awk**

```
BEGIN{
```

```
    count1=0
```

```
    count2=0
```

```
    pack1=0
```

```
    pack2=0
```

```
    time1=0
```

```
    time2=0
```

```
}
```

```
{
```

```
    if($1=="r"&& $3=="_1_" && $4=="AGT")
```

```
    {
```

```
count1++

pack1=pack1+$8

time1=$2

}

if($1=="r" && $3=="_2_" && $4=="AGT")

{

count2++

pack2=pack2+$8

time2=$2

}}

END{

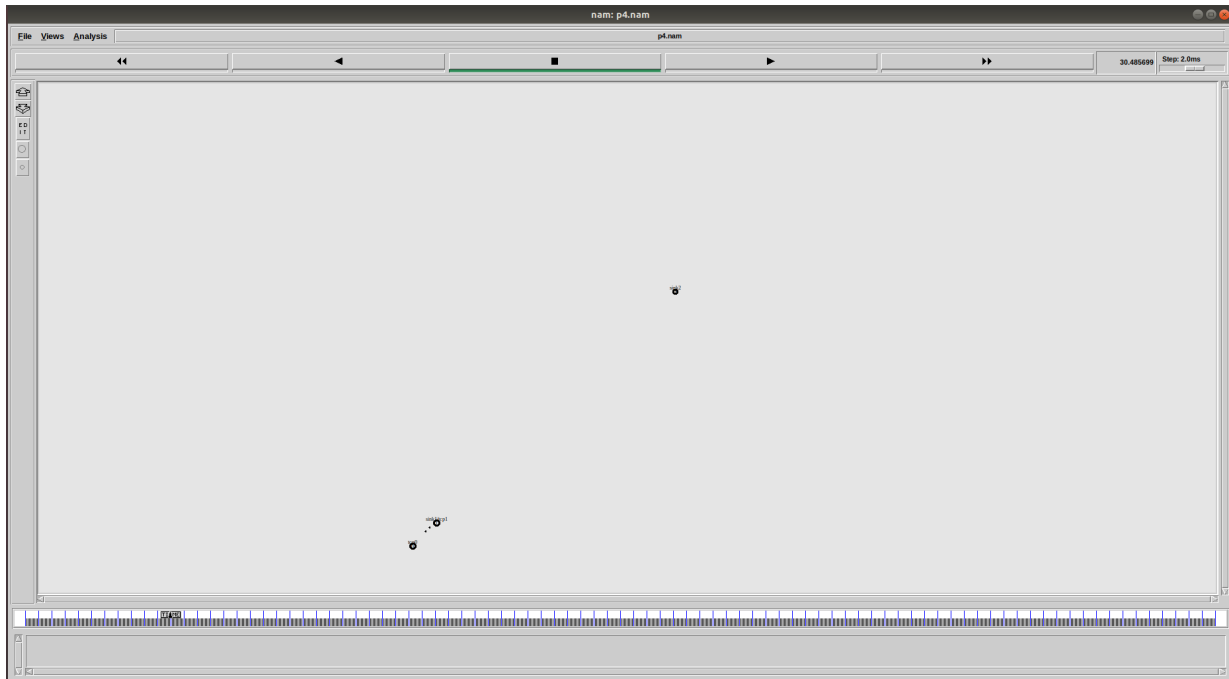
printf("The Throughput from n0 to n1: %f Mbps \n" ,(((count1*pack1*8)/(time1*1000000))));

printf("The Throughput from n1 to n2: %f Mbps \n" , (((count2*pack2*8)/(time2*1000000))));

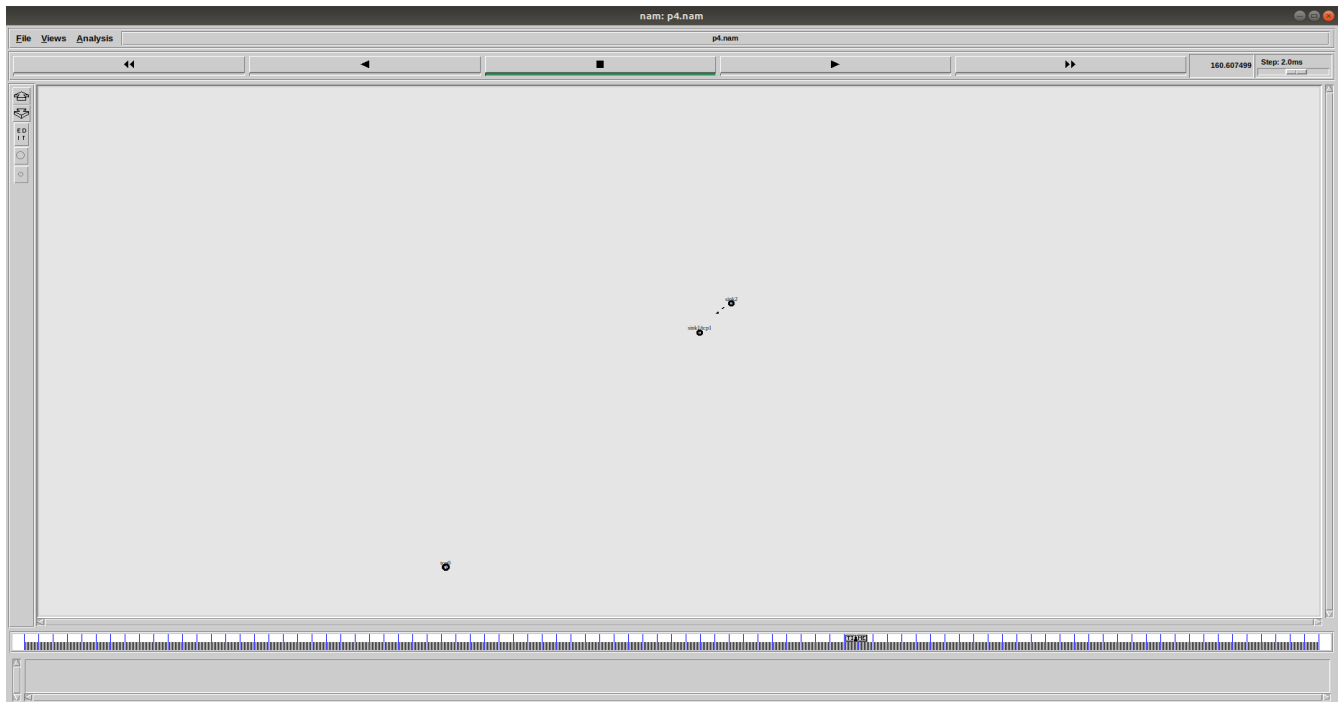
}
```

## SCREENSHOTS

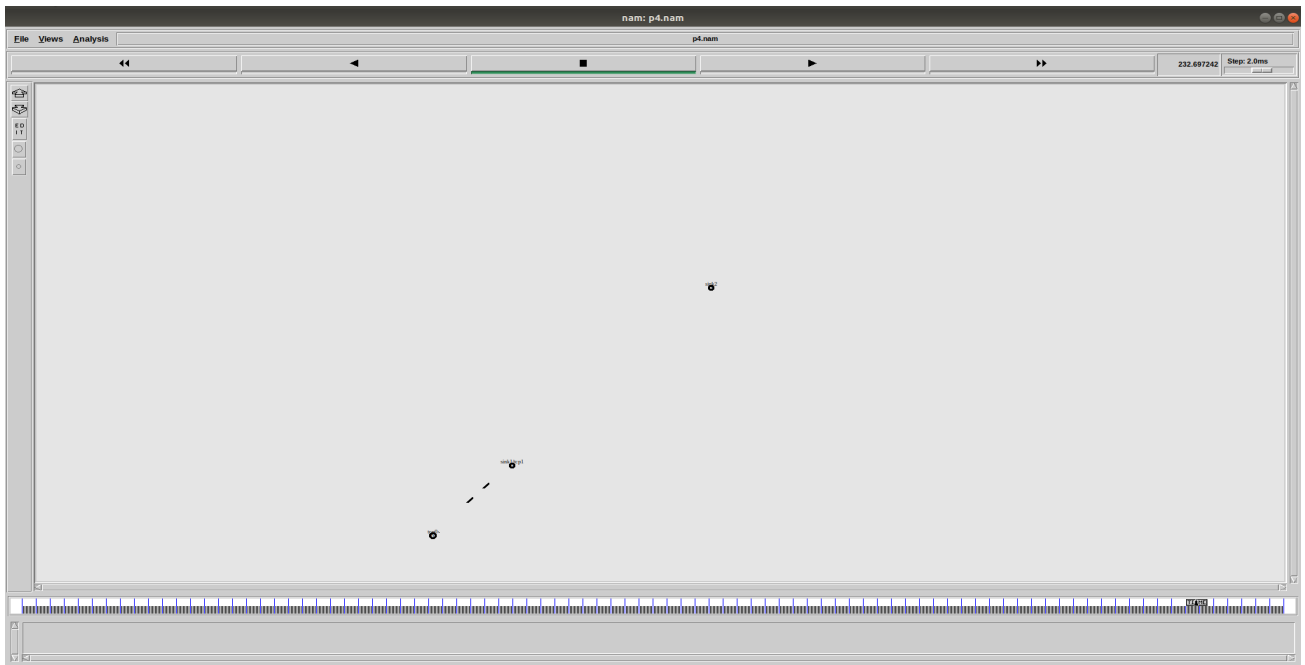
### 1. Node 2 coming to Node 1



### 2. Node 2 coming to Node 3



### 3. Node 2 coming to Node 1



```
administrator@administrator-OptiPlex-7040: ~  
File Edit View Search Terminal Help  
administrator@administrator-OptiPlex-7040:~$ gedit p4.tcl  
administrator@administrator-OptiPlex-7040:~$ ns p4.tcl  
warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl  
num_nodes is set 3  
INITIALIZE THE LIST xListHead  
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_  
highestAntennaZ_ = 1.5, distCST_ = 550.0  
SORTING LISTS ...DONE!  
administrator@administrator-OptiPlex-7040:~$ awk -f p4.awk p4.tr  
The Throughput from n0 to n1: 5863.442245Mbps  
The Throughput from n1 to n2: 1307.611834Mbps  
administrator@administrator-OptiPlex-7040:~$ gedit p4.tcl  
administrator@administrator-OptiPlex-7040:~$ gedit p4.awk  
administrator@administrator-OptiPlex-7040:~$ awk -f p4.awk p4.tr  
The Throughput from n0 to n1: 5863.442245 Mbps  
The Throughput from n1 to n2: 1307.611834 Mbps  
administrator@administrator-OptiPlex-7040:~$
```

**5. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.**

lab5.tcl

```
set opt(title) zero;
set opt(stop) 100;
set opt(ecn) 0 ;
```

```
set opt(type) gsm ;
set opt(secondDelay) 5;
```

```
set opt(minth) 30 ;
set opt(maxth) 0 ;
set opt(adaptive) 1 ;
```

```
set opt(flows) 0 ;
set opt(window) 30 ;
set opt(web) 2 ;
set opt(quiet) 0 ;
set opt(wrap) 100 ;
set opt(srcTrace) is ;
set opt(dstTrace) bs2 ;
set opt(gsmbuf) 10 ;
```

```
set bwDL(gsm) 9600
```

```
set bwUL(gsm) 9600
```

```
set propDL(gsm) .500
```

```
set propUL(gsm) .500
```

```
set buf(gsm) 10
```

```
set ns [new Simulator]
set tf [open out.tr k w]
$ns trace-all $tf
```

```
set nodes(is) [$ns node]
set nodes(ms) [$ns node]
set nodes(bs1) [$ns node]
set nodes(bs2) [$ns node]
set nodes(lp) [$ns node]
```

```
proc cell_topo {} {
```

```

global ns nodes
$ns duplex-link $nodes(lp) $nodes(bs1) 3Mbps 10ms DropTail
$ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED
$ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED
$ns duplex-link $nodes(bs2) $nodes(is) 3Mbps 50ms DropTail
puts "Cell Topology"
}

```

```

proc set_link_params {t} {
    global ns nodes bwUL bwDL propUL propDL buf
    $ns bandwidth $nodes(bs1) $nodes(ms) $bwDL($t) simplex
    $ns bandwidth $nodes(ms) $nodes(bs1) $bwUL($t) simplex
    $ns bandwidth $nodes(bs2) $nodes(ms) $bwDL($t) simplex
    $ns bandwidth $nodes(ms) $nodes(bs2) $bwUL($t) simplex
    $ns delay $nodes(bs1) $nodes(ms) $propDL($t) simplex
    $ns delay $nodes(ms) $nodes(bs1) $propDL($t) simplex
    $ns delay $nodes(bs2) $nodes(ms) $propDL($t) simplex
    $ns delay $nodes(ms) $nodes(bs2) $propDL($t) simplex
    $ns queue-limit $nodes(bs1) $nodes(ms) $buf($t)
    $ns queue-limit $nodes(ms) $nodes(bs1) $buf($t)
    $ns queue-limit $nodes(bs2) $nodes(ms) $buf($t)
    $ns queue-limit $nodes(ms) $nodes(bs2) $buf($t)
}

```

```

Queue/RED set summarystats_ true
Queue/DropTail set summarystats_ true
Queue/RED set adaptive_ $opt(adaptive)
Queue/RED set q_weight_ 0.0
Queue/RED set thresh_ $opt(minth)
Queue/RED set maxthresh_ $opt(maxth)
Queue/DropTail set shrink_drops_ true
Agent/TCP set ecn_ $opt(ecn)
Agent/TCP set window_ $opt(window)
DelayLink set avoidReordering_ true

```

```

source web.tcl

```

```

switch $opt(type) {
    gsm -
    gprs -
    umts {cell_topo}
}
set_link_params $opt(type)
$ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]
$ns insert-delayer $nodes(bs1) $nodes(ms) [new Delayer]
$ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]

```

```

$ns insert-delayer $nodes(bs2) $nodes(ms) [new Delayer]

if {$opt(flows) == 0} {
    set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
    set ftp1 [[set tcp1] attach-app FTP]
    $ns at 0.8 "[set ftp1] start"
}
if {$opt(flows) > 0} {
    set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
    set ftp1 [[set tcp1] attach-app FTP]
    $tcp1 set window_ 100
    $ns at 0.0 "[set ftp1] start"

    $ns at 3.5 "[set ftp1] stop"
    set tcp2 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
    set ftp2 [[set tcp2] attach-app FTP]
    $tcp2 set window_ 3
    $ns at 1.0 "[set ftp2] start"
    $ns at 8.0 "[set ftp2] stop"
}

proc stop {} {
    global nodes opt nf
    set wrap $opt(wrap)
    set sid [$nodes($opt(srcTrace)) id]
    set did [$nodes($opt(dstTrace)) id]
    if {$opt(srcTrace) == "is"} {
        set a "-a out.tr"
    } else {
        set a "out.tr"
    }
    set GETRC "../..../bin/getrc"
    set RAW2XG "../..../bin/raw2xg"

    exec $GETRC -s $sid -d $did -f 0 out.tr | \
    $RAW2XG -s 0.01 -m $wrap -r > plot.xgr
    exec $GETRC -s $did -d $sid -f 0 out.tr | \
    $RAW2XG -a -s 0.01 -m $wrap >> plot.xgr

    exec $GETRC -s $sid -d $did -f 1 out.tr | \
    $RAW2XG -s 0.01 -m $wrap -r >> plot.xgr
    exec $GETRC -s $did -d $sid -f 1 out.tr | \
    $RAW2XG -s 0.01 -m $wrap -a >> plot.xgr

    exec ./xg2gp.awk plot.xgr
    if {!$opt(quiet)} {

```



```
        exec xgraph -bb -tk -nl -m -x time -y packets plot.xgr &
    }
    exit 0
}
$ns at $opt(stop) "stop"
$ns run
```

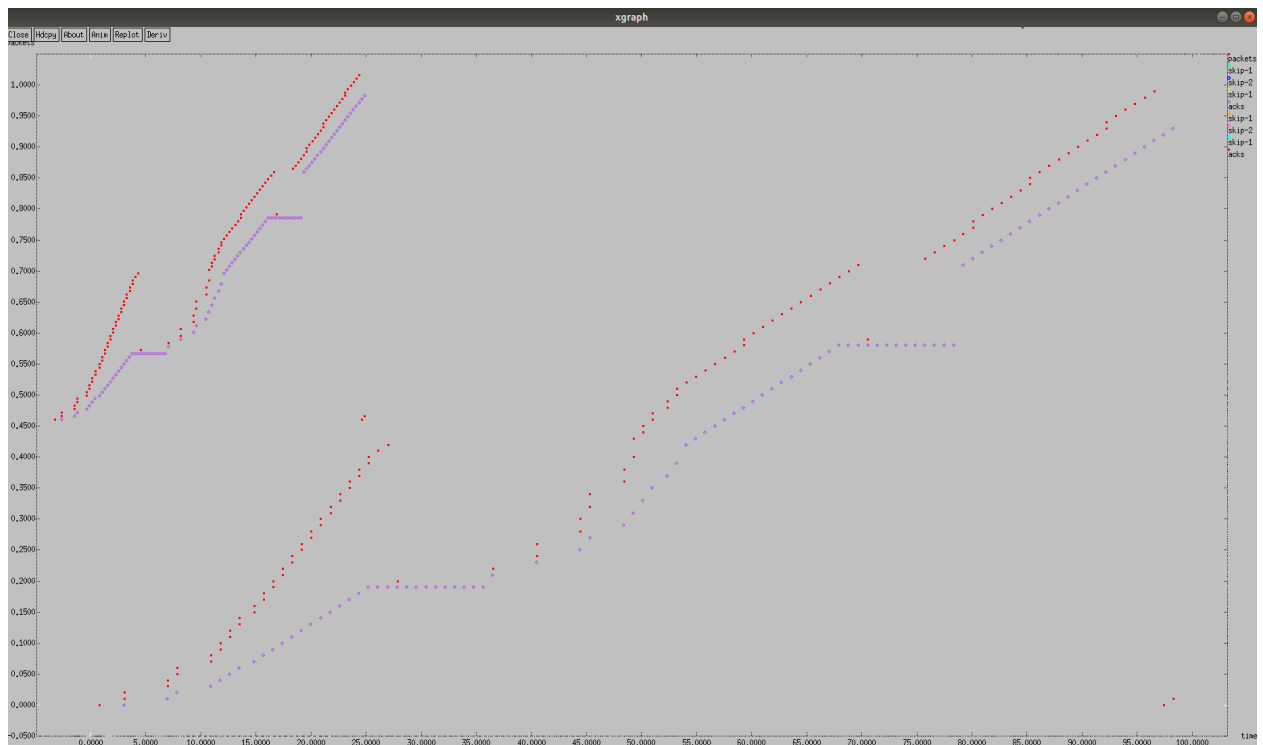
[ScreenShots](#)

## Terminal

File Edit View Search Terminal Help

```
administrator@administrator-OptiPlex-7040:~$ cd ns-allinone-2.35
administrator@administrator-OptiPlex-7040:~/ns-allinone-2.35$ cd ns-2.35
administrator@administrator-OptiPlex-7040:~/ns-allinone-2.35/ns-2.35$ cd tcl
administrator@administrator-OptiPlex-7040:~/ns-allinone-2.35/ns-2.35/tcl$ cd ex
administrator@administrator-OptiPlex-7040:~/ns-allinone-2.35/ns-2.35/tcl/ex$ cd wireless-scripts
administrator@administrator-OptiPlex-7040:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$ gedit lab5.tcl
administrator@administrator-OptiPlex-7040:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$ ns lab5.tcl
Cell Topology
administrator@administrator-OptiPlex-7040:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$ Parameter LabelFont: can't translate 'helvetica-10' into a font (defaulting to 'fixed')
Parameter TitleFont: can't translate 'helvetica-18' into a font (defaulting to 'fixed')

```



```
Open  out.tr  Save
~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts

1+ 0.0 0 3 tcp 40 ----- 0 0.0 4.0 0 0
2- 0.0 0 3 tcp 40 ----- 0 0.0 4.0 0 0
3r 0.850107 0 3 tcp 40 ----- 0 0.0 4.0 0 0
4+ 0.850107 3 1 tcp 40 ----- 0 0.0 4.0 0 0
5- 0.850107 3 1 tcp 40 ----- 0 0.0 4.0 0 0
6r 1.38344 3 1 tcp 40 ----- 0 0.0 4.0 0 0
7+ 1.38344 1 2 tcp 40 ----- 0 0.0 4.0 0 0
8- 1.38344 1 2 tcp 40 ----- 0 0.0 4.0 0 0
9r 1.916773 1 2 tcp 40 ----- 0 0.0 4.0 0 0
10+ 1.916773 2 4 tcp 40 ----- 0 0.0 4.0 0 0
11- 1.916773 2 4 tcp 40 ----- 0 0.0 4.0 0 0
12r 1.92688 2 4 tcp 40 ----- 0 0.0 4.0 0 0
13+ 1.92688 4 2 ack 40 ----- 0 4.0 0.0 0 1
14- 1.92688 4 2 ack 40 ----- 0 4.0 0.0 0 1
15r 1.936987 4 2 ack 40 ----- 0 4.0 0.0 0 1
16+ 1.936987 2 1 ack 40 ----- 0 4.0 0.0 0 1
17- 1.936987 2 1 ack 40 ----- 0 4.0 0.0 0 1
18r 2.47032 2 1 ack 40 ----- 0 4.0 0.0 0 1
```

6. Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.

Lab6.tcl

```
set opt(title) zero ;
set opt(stop) 100 ;
set opt(ecn) 0 ;
set opt(type) umts ;
set opt(secondDelay) 55 ;
```

```
set opt(minth) 30 ;
set opt(maxth) 0 ;
set opt(adaptive) 1 ;
```

```
set opt(flows) 0 ;
set opt(window) 30 ;
set opt(web) 2 ;
```

```
set opt(quiet) 0 ;
set opt(wrap) 100 ;
set opt(srcTrace) is ;
set opt(dstTrace) bs2 ;
set opt(umtsbuf) 10 ;
```

```
set bwDL(umts) 384000
```

```
set bwUL(umts) 64000
```

```
set propDL(umts) .150
```

```
set propUL(umts) .150
```

```
set buf(umts) 20
```

```
set ns [new Simulator]
set tf [open out.tr w]
```

```
$ns trace-all $tf
```

```
set nodes(is) [$ns node]  
set nodes(ms) [$ns node]  
set nodes(bs1) [$ns node]  
set nodes(bs2) [$ns node]  
set nodes(lp) [$ns node]
```

```
proc cell_topo {} {  
    global ns nodes  
    $ns duplex-link $nodes(lp) $nodes(bs1) 3Mbps 10ms DropTail  
    $ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED  
    $ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED  
    $ns duplex-link $nodes(bs2) $nodes(is) 3Mbps 50ms DropTail  
    puts "Cell Topology"  
}
```

```
proc set_link_params {t} {  
    global ns nodes bwUL bwDL propUL propDL buf  
    $ns bandwidth $nodes(bs1) $nodes(ms) $bwDL($t) simplex  
    $ns bandwidth $nodes(ms) $nodes(bs1) $bwUL($t) simplex  
    $ns delay $nodes(bs1) $nodes(ms) $propDL($t) simplex  
    $ns delay $nodes(ms) $nodes(bs1) $propDL($t) simplex  
    $ns queue-limit $nodes(bs1) $nodes(ms) $buf($t)  
    $ns queue-limit $nodes(ms) $nodes(bs1) $buf($t)  
    $ns bandwidth $nodes(bs2) $nodes(ms) $bwDL($t) simplex  
    $ns bandwidth $nodes(ms) $nodes(bs2) $bwUL($t) simplex  
    $ns delay $nodes(bs2) $nodes(ms) $propDL($t) simplex  
    $ns delay $nodes(ms) $nodes(bs2) $propDL($t) simplex  
    $ns queue-limit $nodes(bs2) $nodes(ms) $buf($t)  
    $ns queue-limit $nodes(ms) $nodes(bs2) $buf($t)  
}
```

```
Queue/RED set summarystats_ true  
Queue/DropTail set summarystats_ true  
Queue/RED set adaptive_ $opt(adaptive)  
Queue/RED set q_weight_ 0.0  
Queue/RED set thresh_ $opt(minth)  
Queue/RED set maxthresh_ $opt(maxth)
```

```

Queue/DropTail set shrink_drops_ true
Agent/TCP set ecn_ $opt(ecn)
Agent/TCP set window_ $opt(window)
DelayLink set avoidReordering_ true

source web.tcl

switch $opt(type) {
    umts {cell_topo}
}
set_link_params $opt(type)
$ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]
$ns insert-delayer $nodes(bs1) $nodes(ms) [new Delayer]
$ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]
$ns insert-delayer $nodes(bs2) $nodes(ms) [new Delayer]

if {$opt(flows) == 0} {
    set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1
$nodes(lp) 0]
    set ftp1 [[set tcp1] attach-app FTP]
    $ns at 0.8 "[set ftp1] start"
}
if {$opt(flows) > 0} {
    set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1
$nodes(lp) 0]
    set ftp1 [[set tcp1] attach-app FTP]
    $tcp1 set window_ 100
    $ns at 0.0 "[set ftp1] start"
    $ns at 3.5 "[set ftp1] stop"

    set tcp2 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1
$nodes(lp) 0]
    set ftp2 [[set tcp2] attach-app FTP]
    $tcp2 set window_ 3
    $ns at 1.0 "[set ftp2] start"
    $ns at 8.0 "[set ftp2] stop"
}
proc stop {} {
    global nodes opt nf

```

```

set wrap $opt(wrap)
set sid [$nodes($opt(srcTrace)) id]
set did [$nodes($opt(dstTrace)) id]
if {$opt(srcTrace) == "is"} {
    set a "-a out.tr"
} else {
    set a "out.tr"
}
set GETRC "../././bin/getrc"
set RAW2XG "../././bin/raw2xg"

exec $GETRC -s $sid -d $did -f 0 out.tr | \
$RAW2XG -s 0.01 -m $wrap -r > plot.xgr
exec $GETRC -s $did -d $sid -f 0 out.tr | \
$RAW2XG -a -s 0.01 -m $wrap >> plot.xgr

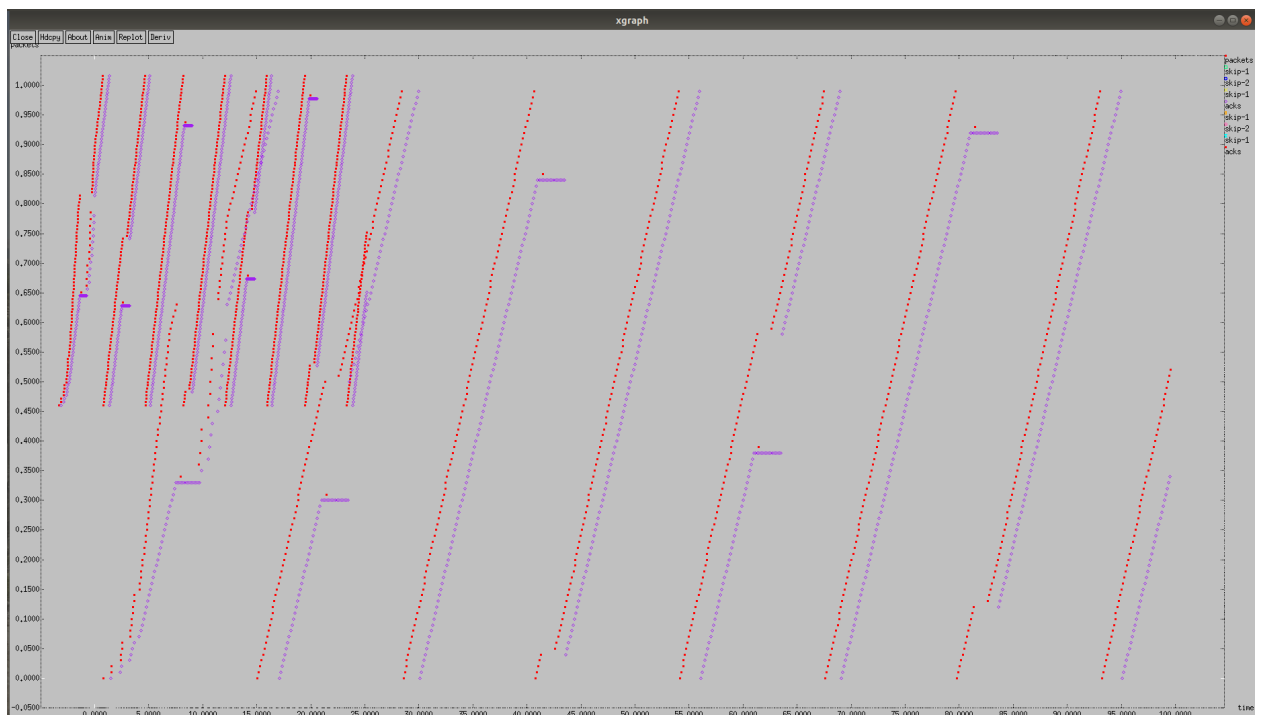
exec $GETRC -s $sid -d $did -f 1 out.tr | \
$RAW2XG -s 0.01 -m $wrap -r >> plot.xgr
exec $GETRC -s $did -d $sid -f 1 out.tr | \
$RAW2XG -s 0.01 -m $wrap -a >> plot.xgr

exec ./xg2gp.awk plot.xgr
if {!$opt(quiet)} {
    exec xgraph -bb -tk -nl -m -x time -y packets plot.xgr &
}
exit 0
}
$ns at $opt(stop) "stop"
$ns run

```

## SCREENSHOTS

```
Terminal
File Edit View Search Terminal Help
administrator@administrator-OptiPlex-7040:~$ cd ns-allinone-2.35
administrator@administrator-OptiPlex-7040:~/ns-allinone-2.35$ cd ns-2.35
administrator@administrator-OptiPlex-7040:~/ns-allinone-2.35/ns-2.35$ cd tcl
administrator@administrator-OptiPlex-7040:~/ns-allinone-2.35/ns-2.35/tcl$ cd ex
administrator@administrator-OptiPlex-7040:~/ns-allinone-2.35/ns-2.35/tcl/ex$ cd wireless-scripts
administrator@administrator-OptiPlex-7040:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$ ns lab6.tcl
tcl Topology
administrator@administrator-OptiPlex-7040:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$ Parameter LabelFont: can't translate 'helvetica-10' into a font (defaulting to 'fixed')
Parameter TitleFont: can't translate 'helvetica-18' into a font (defaulting to 'fixed')
```





```
Open  out.tr  Save
~/no allmone 2.35/no 2.35/tx/wireless-scripts

1 + 0.8 0 3 tcp 40 ----- 0 0.0 4.0 0 0
2 + 0.8 0 3 tcp 40 ----- 0 0.0 4.0 0 0
3 r 0.850107 0 3 tcp 40 ----- 0 0.0 4.0 0 0
4 + 0.850107 3 1 tcp 40 ----- 0 0.0 4.0 0 0
5 - 0.850107 3 1 tcp 40 ----- 0 0.0 4.0 0 0
6 r 1.000094 3 1 tcp 40 ----- 0 0.0 4.0 0 0
7 + 1.000094 1 2 tcp 40 ----- 0 0.0 4.0 0 0
8 - 1.000094 1 2 tcp 40 ----- 0 0.0 4.0 0 0
9 r 1.155594 1 2 tcp 40 ----- 0 0.0 4.0 0 0
10 + 1.155594 2 4 tcp 40 ----- 0 0.0 4.0 0 0
11 - 1.155594 2 4 tcp 40 ----- 0 0.0 4.0 0 0
12 r 1.166047 2 4 tcp 40 ----- 0 0.0 4.0 0 0
13 + 1.166047 4 2 ack 40 ----- 0 4.0 0.0 0 1
14 - 1.166047 4 2 ack 40 ----- 0 4.0 0.0 0 1
15 r 1.176153 4 2 ack 40 ----- 0 4.0 0.0 0 1
16 + 1.176153 2 1 ack 40 ----- 0 4.0 0.0 0 1
17 - 1.176153 2 1 ack 40 ----- 0 4.0 0.0 0 1
18 r 1.326987 2 1 ack 40 ----- 0 4.0 0.0 0 1
19 + 1.326987 1 3 ack 40 ----- 0 4.0 0.0 0 1
20 - 1.326987 1 3 ack 40 ----- 0 4.0 0.0 0 1
21 r 1.481987 1 3 ack 40 ----- 0 4.0 0.0 0 1
22 + 1.481987 3 0 ack 40 ----- 0 4.0 0.0 0 1
23 - 1.481987 3 0 ack 40 ----- 0 4.0 0.0 0 1
24 r 1.532093 3 0 ack 40 ----- 0 4.0 0.0 0 1
25 + 1.532093 0 3 tcp 1040 ----- 0 0.0 4.0 1 2
26 - 1.532093 0 3 tcp 1040 ----- 0 0.0 4.0 1 2
27 + 1.532093 0 3 tcp 1040 ----- 0 0.0 4.0 2 3
28 - 1.534867 0 3 tcp 1040 ----- 0 0.0 4.0 2 3
29 r 1.584867 0 3 tcp 1040 ----- 0 0.0 4.0 1 2
30 + 1.584867 3 1 tcp 1040 ----- 0 0.0 4.0 1 2
31 - 1.584867 3 1 tcp 1040 ----- 0 0.0 4.0 1 2
32 r 1.58764 0 3 tcp 1040 ----- 0 0.0 4.0 2 3
33 + 1.58764 3 1 tcp 1040 ----- 0 0.0 4.0 2 3
34 - 1.606533 3 1 tcp 1040 ----- 0 0.0 4.0 2 3
```