Subject: Computer Networks Lab
Subject Code: 15ECL68
Semester: 6th B.E(ECE) (CBCS)

Prepared by

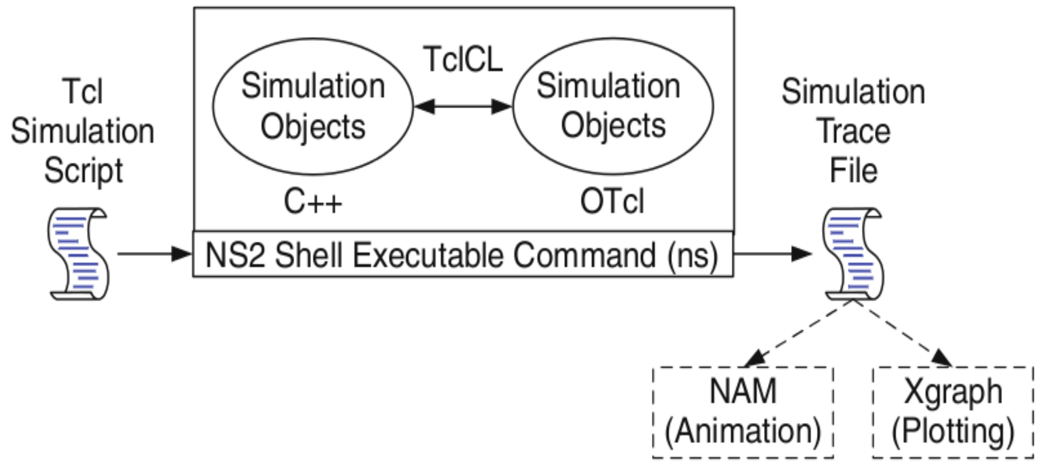
Mr. Nataraju A.B. M.E., (PhD)
Assistant Professor,
Department of Electronics & Communication Engineering,
Acharya Institute of Technology,
Dr. Sarvepalli Radhakrishnan Road, Bangalore – 560 107
Phone: +91-98455-95744

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 behaviors.

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)]this
        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 simulation

NS Simulator Preliminaries.

1. Initialization and termination aspects of the ns simulator.
2. Definition of network nodes, links, queues and topology.
3. Definition of agents and of applications.
4. The nam visualization tool.
5. Tracing and random variables.

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 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 begin 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 exit because something failed.

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
```

The node is created which is printed by the variable n0. When we 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:

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 behavior 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

The below 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.

```
set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set packetsize_  100

$cbr set rate_  0.01Mb

$cbr set random_  false
```

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"

$ns at 124.5 "$cbr 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 scippting tool

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

THE –f OPTION: STORING awk PROGRAMS IN A FILE

You should holds large awk programs in separate file 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 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=”|”}

***<u>Experiment No</u>**: 1* **THREE NODE POINT TO POINT NETWORK**

<u>Aim</u>: *Simulate a three node point to point network with duplex links between them. Set queue size and vary the bandwidth and find **number of packets dropped**.*

<u>Program</u>:

```
#Creating an instance of Simulator Object to setup and use the network setup.
set myNS [new Simulator]

#Setting up files for tracing & Network Animation….
set trfile [open out.tr w]
set namfile [open out.nam w]

# Tracing files using their commands
# log all the events to trace file
$myNS trace-all $trfile

# log all the events to Animation file
$myNS namtrace-all $namfile

#Creating NODES
set n0 [$myNS node]
set n1 [$myNS node]
set n2 [$myNS node]
set n3 [$myNS node]

#Label the nodes
$n0 label "TCP Source"
$n1 label "UDP Source"
$n2 label "Router"
$n3 label "Sink/Null"

#Define different colors for data flow
$myNS color 1 blue
$myNS color 2 red

#Creating LINKS between different nodes

$myNS duplex-link $n0 $n2 2Mb 10ms DropTail
$myNS duplex-link $n1 $n2 2Mb 10ms DropTail
$myNS duplex-link $n2 $n3 1.0Mb 20ms DropTail

# Set the queue limit after which the packets are going to be dropped during
# congestion situation.

$myNS queue-limit $n2 $n3 40

#Node position for Animation…
$myNS duplex-link-op $n0 $n2 orient right-down
$myNS duplex-link-op $n1 $n2 orient right-up
$myNS duplex-link-op $n2 $n3 orient right

#Create and attach UDP agent over nodes… (Sender)
set udp [new Agent/UDP]
$myNS attach-agent $n1 $udp

#Create and attach NULL agent over nodes… (Receiver)
set null [new Agent/Null]
$myNS attach-agent $n3 $null
```

```
#Logically associate the UDP sender and receiver
$myNS connect $udp $null
$udp set fid_ 0

#Create and attach TCP agent over nodes… (Sender)
set tcp [new Agent/TCP]
$myNS attach-agent $n0 $tcp

#Create and attach TCPSink agent over nodes… (Receiver)
set sink [new Agent/TCPSink]
$myNS attach-agent $n3 $sink

#Logically associate the TCP sender and receiver
$myNS connect $tcp $sink
$tcp set fid_ 1

#create a CBR application… and attach with UDP transport
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp

#create an FTP application… and attach with TCP transport
set ftp [new Application/FTP]
$ftp attach-agent $tcp

$myNS at 0.1 "$cbr start"
$myNS at 1.0 "$ftp start"
$myNS at 4.0 "$ftp stop"
$myNS at 4.5 "$cbr stop"
$myNS at 5.0 "finish"


#Closing trace file and starting NAM
proc finish { } {
        global myNS trfile namfile
        $myNS flush-trace
        close $trfile
        close $namfile
        # exec nam out.nam &
        # exec awk -f n1.awk out.tr &
        exit 0
}

# RUN the simulation....
$myNS run
```

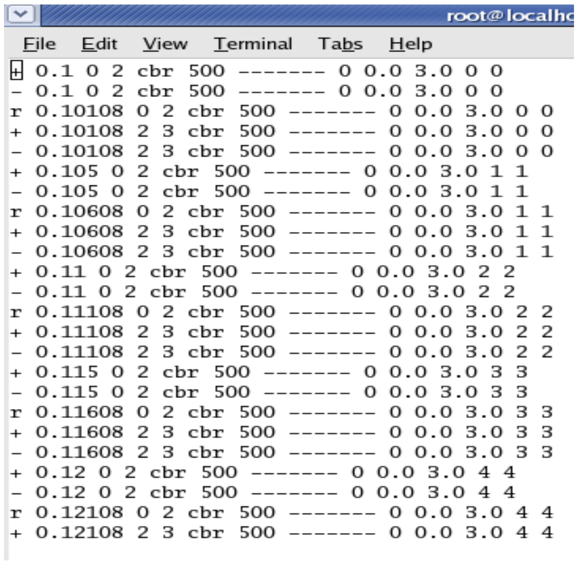
AWK File **// Post processing the logs to obtain statistics…**

```
BEGIN{
      cbrPkt=0;
      tcpPkt=0;
}

{
      # identify the dropped CBR packets
      if(($1 == "d")&&($5 == "cbr")) {
            cbrPkt = cbrPkt + 1;
      }

      # identify the dropped TCP packets
      if(($1 == "d")&&($5 == "tcp")) {
            tcpPkt = tcpPkt + 1;
      }
}

END {
      printf "No. of CBR Packets Dropped %d\n", cbrPkt;
      printf "No. of TCP Packets Dropped %d\n", tcpPkt;
}
```

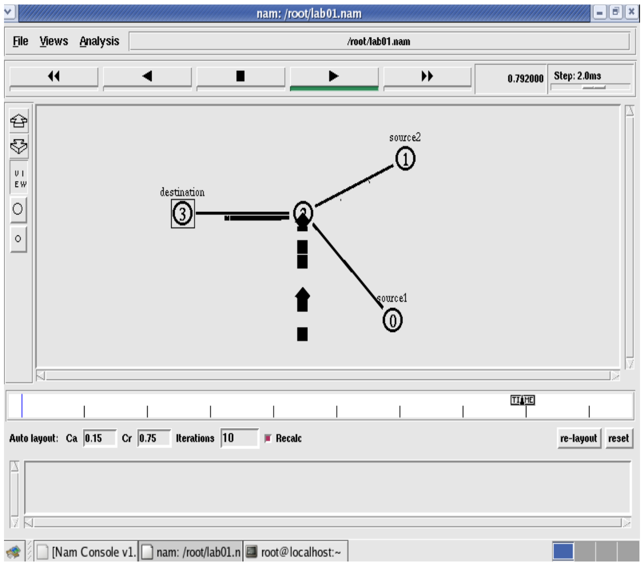
Steps for execution:

- *Open vi editor and type program. Program name should have the extension “* ***.tcl*** *”*
  [root@localhost ~]# vi lab1.tcl
- *Save the program by pressing* ***“ESC key”*** *first, followed by* ***“Shift and :”*** *keys simultaneously and type* ***“wq”*** *and press* ***Enter key****.*
- *Open vi editor and type* ***awk*** *program. Program name should have the extension “****.awk*** *”*
  [root@localhost ~]# vi lab1.awk
- *Save the program by pressing* ***“ESC key”*** *first, followed by* ***“Shift and :”*** *keys simultaneously and type* ***“wq”*** *and press* ***Enter key****.*
- *Run the simulation program*
  [root@localhost~]# ns lab1.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 begins.*
- *After simulation is completed run* ***awk file*** *to see the output ,*
  [root@localhost~]# awk –f lab1.awk lab1.tr
- *To see the trace file contents open the file as ,*
  [root@localhost~]# vi lab1.tr

Note: lab1 is file name



```
+ 0.1 0 2 cbr 500 ------- 0 0.0 3.0 0 0
- 0.1 0 2 cbr 500 ------- 0 0.0 3.0 0 0
r 0.10108 0 2 cbr 500 ------- 0 0.0 3.0 0 0
+ 0.10108 2 3 cbr 500 ------- 0 0.0 3.0 0 0
- 0.10108 2 3 cbr 500 ------- 0 0.0 3.0 0 0
+ 0.105 0 2 cbr 500 ------- 0 0.0 3.0 1 1
- 0.105 0 2 cbr 500 ------- 0 0.0 3.0 1 1
r 0.10608 0 2 cbr 500 ------- 0 0.0 3.0 1 1
+ 0.10608 2 3 cbr 500 ------- 0 0.0 3.0 1 1
- 0.10608 2 3 cbr 500 ------- 0 0.0 3.0 1 1
+ 0.11 0 2 cbr 500 ------- 0 0.0 3.0 2 2
- 0.11 0 2 cbr 500 ------- 0 0.0 3.0 2 2
r 0.11108 0 2 cbr 500 ------- 0 0.0 3.0 2 2
+ 0.11108 2 3 cbr 500 ------- 0 0.0 3.0 2 2
- 0.11108 2 3 cbr 500 ------- 0 0.0 3.0 2 2
+ 0.115 0 2 cbr 500 ------- 0 0.0 3.0 3 3
- 0.115 0 2 cbr 500 ------- 0 0.0 3.0 3 3
r 0.11608 0 2 cbr 500 ------- 0 0.0 3.0 3 3
+ 0.11608 2 3 cbr 500 ------- 0 0.0 3.0 3 3
- 0.11608 2 3 cbr 500 ------- 0 0.0 3.0 3 3
+ 0.12 0 2 cbr 500 ------- 0 0.0 3.0 4 4
- 0.12 0 2 cbr 500 ------- 0 0.0 3.0 4 4
r 0.12108 0 2 cbr 500 ------- 0 0.0 3.0 4 4
+ 0.12108 2 3 cbr 500 ------- 0 0.0 3.0 4 4
```



Procedure for program execution:

Step-1: run the simulation scenario using command **$ ns <n1.tcl>** … here **n1.tcl** is the input file which has got he network configuration

Step-2 : Find out the statistics from the simulation using the command **$ awk –f <n1.awk> <out.tr>**

n1.awk has got the logic to identify the necessary statistics from the log / trace file out.tr

Step-3 : modify the que-limit for n2-n3 link and find out the dropped packets and populate the following table

Assume n0-n2 : 1Mb, n1-n2: 1Mb, n2-n3: 1Mb

Queue-limit	**TCP packets dropped**	**CBR / UDP packets dropped**
5		
10		
15		
20		
25		
30		

Step-3 : modify the link capacity for n2-n3 link and find out the dropped packets and populate the following table

Assume n0-n2 : 1Mb, n1-n2: 1Mb, Queue-Limit : 10

n2-n3 link BW (bps)	**TCP packets dropped**	**CBR / UDP packets dropped**
0.5M		
1.0M		
1.5M		
2.0M		
2.5M		

Experiment No: *2* **FOUR NODE POINT TO POINT NETWORK**

Aim: *Simulate a four node point to point network with the links connected as follows: n0 – n2, n1 – n2 and n2 – n3. Apply TCP agent between n0 – n3 and UDP agent between n1 – n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the* ***number of packets sent*** *by TCP / UDP.*

Program:

```
#Create a new Simulation Instance
set ns [new Simulator]

#Turn on the Trace and the animation files
set tf [open out.tr w]
set nf [open out.nam w]

$ns trace-all $tf
$ns namtrace-all $nf

#Define the finish procedure to perform at the end of the simulation
proc finish {} {
      global tf nf ns
      $ns flush-trace
      close $tf
      close $nf
      exec nam out.nam &
      # exec awk -f 2.awk out.tr &
      exit 0
}

#Create the nodes

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

#Label the nodes
$n0 label "TCP SOURCE"
$n1 label "UDP SOURCE"
$n2 label "ROUTER"
$n3 label "SINK"

#Set the color
$ns color 1 red
$ns color 2 blue

#Create the Topology
$ns duplex-link $n0 $n2 2Mb    10ms DropTail
$ns duplex-link $n1 $n2 2Mb    10ms DropTail
$ns duplex-link $n2 $n3 2.75Mb 20ms DropTail
$ns duplex-link $n2 $n4 2Mb    20ms DropTail
$ns duplex-link $n4 $n3 2Mb    20ms DropTail
```

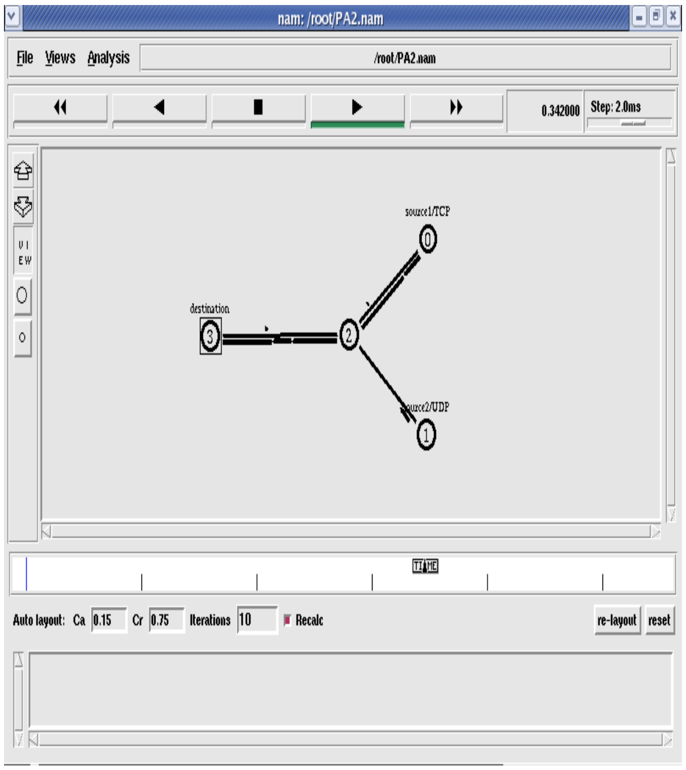
```
#Attach a Queue of size 20 Packets between the nodes n2 n3. This can be
varied … 30, 40, 50, …
$ns queue-limit $n2 $n3 20

#Make the Link Orientation (for NAM only)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#Create an UDP Agent and attach to the node n1
set udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0

#Create a CBR Traffic source and attach to the UDP Agent
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0

#Specify the Packet Size and interval, Thus the data rate specifications
can be set for the simulation.
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.002

#Create a Null Agent and attach to the node n3
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0

#Connect the CBR Traffic source to the Null agent
$ns connect $udp0 $null0

#Create a TCP agent and attach to the node n0
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0

#Create a FTP source and attach to the TCP agent
set ftp0 [new Application/FTP]

#Attach the FTP source to the TCP Agent
$ftp0 attach-agent $tcp0

#Create a TCPSink agent and attach to the node n3
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink

#Specify the Max file Size in Bytes & Packet Size
$ftp0 set maxPkts_ 1000
$ftp0 set packetSize_ 300

#Connect the TCP Agent with the TCP sink
$ns connect $tcp0 $sink

$udp0 set class_ 1
$tcp0 set class_ 2

#Schedule the Events of simulation…
$ns at 0.1 "$cbr0 start"
$ns at 1.0 "$ftp0 start"
$ns at 4.0 "$ftp0 stop"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
```

```
$ns run
```

AWK File: For post processing and results analysis….

```
BEGIN {
      tcpSend = 0;
      cbrSend = 0;
      tcpDrop = 0;
      cbrDrop = 0;
      tcpDropRatio = 0.0;
      cbrDropRatio = 0.0;
      tcpArrivalRatio = 0.0;
      cbrArrivalRatio = 0.0;
}

{
      src = $3;
      des = $4;
      type = $5;
      event = $1;
      if((src == "0") && (des == "2") && (event == "+")) {
            tcpSend = tcpSend + 1;
      }

      if((src == "1") && (des == "2") && (event == "+")) {
            cbrSend = cbrSend + 1;
      }

      if((event == "d") && (type == "tcp")) {
            tcpDrop = tcpDrop + 1;
      }

      if((event == "d") && (type == "cbr")) {
            cbrDrop = cbrDrop + 1;
      }

}

END {
      #To Caluculate TCP Arrival and Drop Ratio
      tcpArrivalRatio = (tcpSend - tcpDrop) / tcpSend;
      printf "TCP Packet Arrival Ratio = %f\n", tcpArrivalRatio;

      tcpDropRatio = tcpDrop / tcpSend;
      printf "TCP Packet Drop Ratio = %f\n", tcpDropRatio;

      #To Calculate CBR Arrival and Drop Ratio
      cbrArrivalRatio = (cbrSend - cbrDrop) / cbrSend;
      printf "CBR Packet Arrival Ratio = %f\n", cbrArrivalRatio;

      cbrDropRatio = cbrDrop / cbrSend;
      printf "CBR Packet Drop Ratio = %f\n", cbrDropRatio;

      # Following lines are not needed….
      #printf "Total # of TCP Packets Sent = %d\n", tcpSend;
      #printf "Total # of CBR Packets Send = %d\n", cbrSend;
      #printf "Total # of TCP Packets Dropped = %d\n", tcpDrop;
      #printf "Total # of CBR Packets Dropped = %d\n", cbrDrop;
}
```

Output:



Experiment No: **3 SETUP ETHERNET LAN with N-NODES**

Aim: *Simulate an Ethernet LAN using 'n' nodes,* ***change error rate*** *and* ***data rate*** *and compare throughput.*

Program:

```
#Create Simulator
set ns [new Simulator]

#Open trace and NAM trace file
set tf [open out.tr w]
$ns trace-all $tf

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

set n0 [$ns node]
set n1 [$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 [$ns node]
set n9 [$ns node]

$ns make-lan "$n0 $n1 $n2 $n3 $n4" 1Mb 10ms LL Queue/DropTail Mac/802_3

$ns make-lan "$n5 $n6 $n7 $n8 $n9" 1Mb 10ms LL Queue/DropTail Mac/802_3

$ns duplex-link    $n4 $n5 1Mb 30ms DropTail
$ns duplex-link-op $n4 $n5 orient right-down

$n0 label "TCP"
$n2 label "UDP"
$n7 label "SINK"
$n9 label "NULL"

$ns color 1 blue
$ns color 2 red

Set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
$tcp0 set class_ 2

Set sink0 [new Agent/TCPSink]
$ns attach-agent $n7 $sink0
```

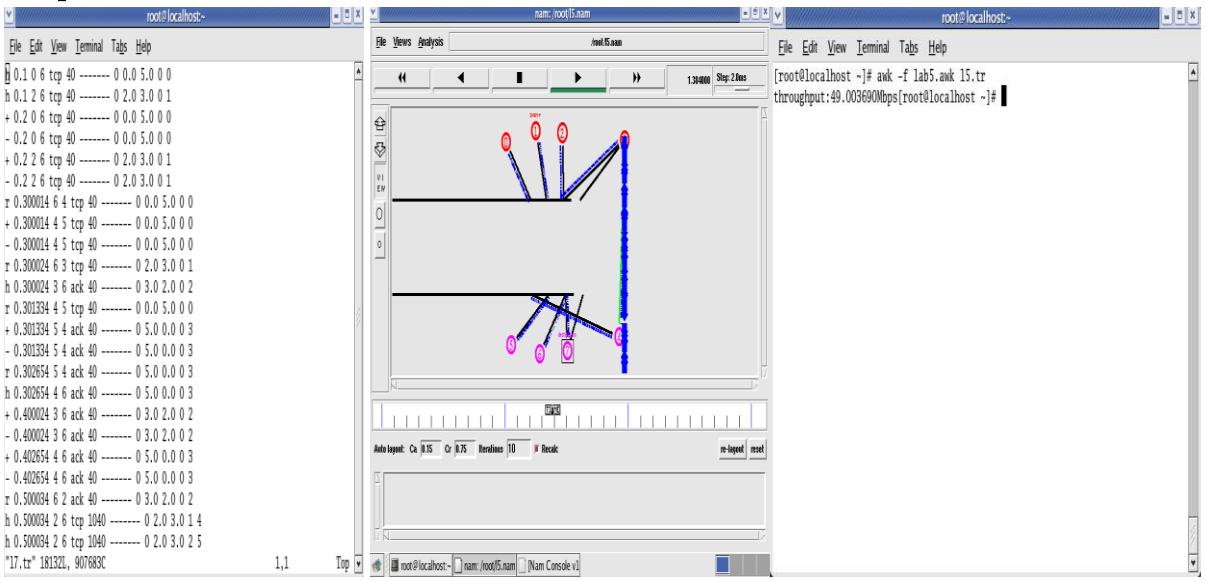
```
Set udp0 [new Agent/UDP]
$ns attach-agent $n2 $udp0
$tcp0 set class_ 1

Set null0 [new Agent/Null]
$ns attach-agent $n9 $null0

$ns connect $tcp0 $sink0
$ns connect $udp0 $null0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005

#Attach CBR application over UDP transport (udp0)
$cbr0 attach-agent $udp0

Set ftp0 [new Application/FTP]
$ftp0 set packetSize_ 500

#Connect the TCP source and destination….
$ftp0 attach-agent $tcp0

#Schedule events
$ns at 0.1 "$cbr0 start"
$ns at 1.0 "$ftp0 start"
$ns at 9.0 "$ftp0 stop"
$ns at 9.5 "$cbr0 stop"
$ns at 10.0 "Finish"

#Finish Procedure
proc Finish {} {
      global ns tf nf
      #flush the trace data to trace file and close the files
      $ns flush-trace
      close $tf
      close $nf
      #Execute the nam animation file
      exec nam out.nam &
      Exit 0
}

#Set the type of errors which occurs over link between LANs
Set error [new ErrorModel]
$ns lossmodel $error $n4 $n5
$error set rate_ 0.50
$ns set datarate_5Mb
```

```
$ns run
```

AWK File:

```
## ANALYSIS OF SIMULATION RESULTS ####

BEGIN{
        totalPackets=0;         totalBytes=0;           time=0;
        tcpPktSize=0;           cbrPktSize=0;           Throughput=0.0;
        simTime=0;              startTime=0;            stopTime=0;
        cbrPktCnt=0;            tcpPktCnt=0;
}
{
     # $n indicates command line arguments

     # + 0.11 2 10 cbr 500 ------- 1 2.0 9.0 1 1
     # h 0.11 2 10 cbr 1000 ------- 1 2.0 9.0 2 2
     # + 0.11 2 10 cbr 1000 ------- 1 2.0 9.0 0 0
     # 1  2 3 4 5  6   7  8 9  10 11 12   --> $n

     event = $1
     time = $2
     appType = $5
     srcNode = $3
     dstNode = $4
     curPktSize = $6

     if((event == "+") && (srcNode == "4") && (dstNode == "5") && (appType == "tcp") ) {
          tcpPktCnt++;
          tcpPktSize = curPktSize;
     }

     if((event == "+") && (srcNode == "4") && (dstNode == "5") && (appType == "cbr") ) {
          cbrPktCnt++;
          cbrPktSize = curPktSize;
     }

     if (time < startTime) {
          startTime = time
     }

     if (time > stopTime) {
          stopTime = time
     }
}

END {
        totalBytes = (tcpPktCnt * tcpPktSize + cbrPktCnt * cbrPktSize);
        simTime = stopTime - startTime;
        Throughput = (totalBytes * 8.0)/(simTime*1000);

        printf(" Total Bytes sent (node 4 to 5)  = %.3f KB\n", totalBytes);
        printf(" The Simulation Time is = %.3f Sec\n", simTime);
        printf(" The Throughtput    = %d Kbps\n", Throughput);
        printf("\n\n");
}

####
# END  ####
```

Output:



```
h 0.1 0 6 tcp 40 ------- 0 0.0 5.0 0 0
h 0.1 2 6 tcp 40 ------- 0 2.0 3.0 0 1
+ 0.2 0 6 tcp 40 ------- 0 0.0 5.0 0 0
- 0.2 0 6 tcp 40 ------- 0 0.0 5.0 0 0
+ 0.2 2 6 tcp 40 ------- 0 2.0 3.0 0 1
- 0.2 2 6 tcp 40 ------- 0 2.0 3.0 0 1
r 0.300014 6 4 tcp 40 ------- 0 0.0 5.0 0 0
+ 0.300014 4 5 tcp 40 ------- 0 0.0 5.0 0 0
- 0.300014 4 5 tcp 40 ------- 0 0.0 5.0 0 0
r 0.300024 6 3 tcp 40 ------- 0 2.0 3.0 0 1
h 0.300024 3 6 ack 40 ------- 0 3.0 2.0 0 2
r 0.301334 4 5 tcp 40 ------- 0 0.0 5.0 0 0
+ 0.301334 5 4 ack 40 ------- 0 5.0 0.0 0 3
- 0.301334 5 4 ack 40 ------- 0 5.0 0.0 0 3
r 0.302654 5 4 ack 40 ------- 0 5.0 0.0 0 3
h 0.302654 4 6 ack 40 ------- 0 5.0 0.0 0 3
+ 0.400024 3 6 ack 40 ------- 0 3.0 2.0 0 2
- 0.400024 3 6 ack 40 ------- 0 3.0 2.0 0 2
+ 0.402654 4 6 ack 40 ------- 0 5.0 0.0 0 3
- 0.402654 4 6 ack 40 ------- 0 5.0 0.0 0 3
r 0.500034 6 2 ack 40 ------- 0 3.0 2.0 0 2
h 0.500034 2 6 tcp 1040 ------- 0 2.0 3.0 1 4
h 0.500034 2 6 tcp 1040 ------- 0 2.0 3.0 2 5
"17.tr" 18132L, 907683C                         1,1          Top
```

```
[root@localhost ~]# awk -f lab5.awk 15.tr
throughput:49.003690Mbps[root@localhost ~]#
```

Experiment No: 4 **<u>ETHERNET</u> LAN USING N-NODES WITH MULTIPLE TRAFFIC**

Aim: *Simulate an Ethernet LAN using 'n' nodes and set multiple traffic nodes and plot* ***congestion window*** *for different source / destination.*

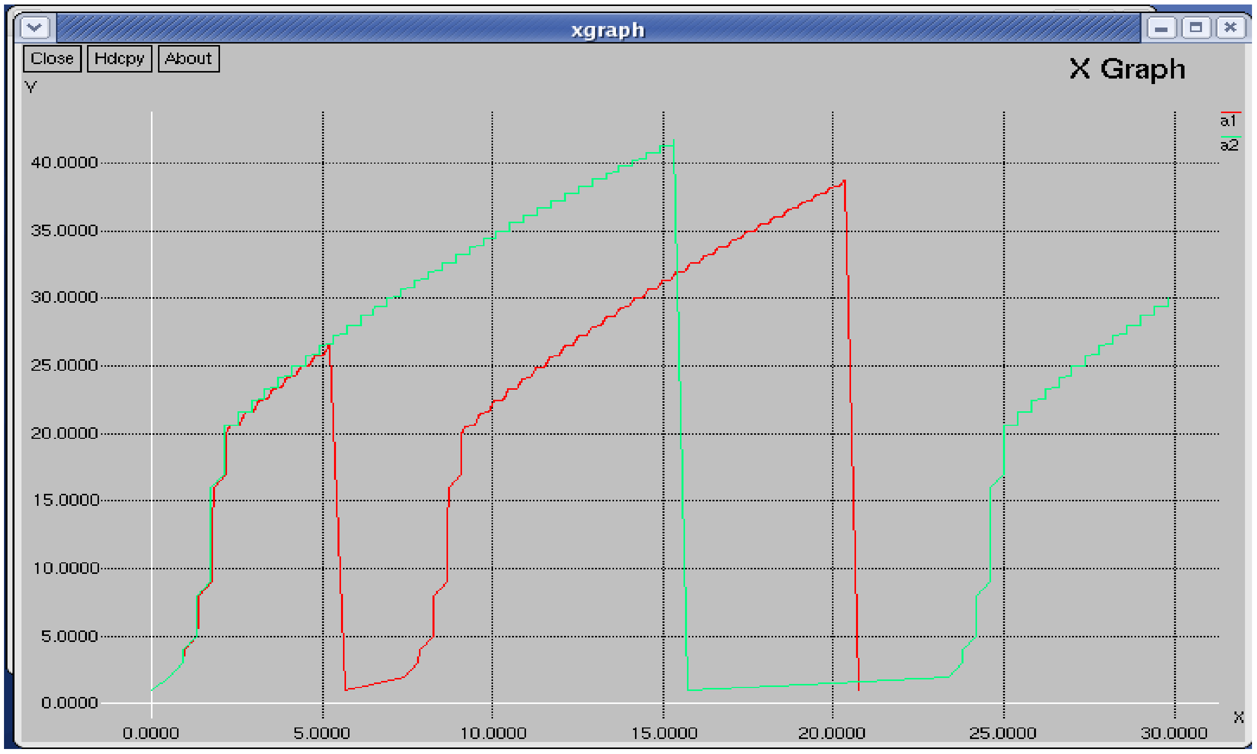
<u>Program:</u>

```
# Simulate an Ethernet LAN using 'n' nodes
# and set multiple traffic nodes and plot congestion
# window for different source / destination.

#Create Simulator
set myNS [new Simulator]

#Open trace and NAM trace file
set myTraceFile [open out.tr w]
$myNS trace-all $myTraceFile

set myNamFile [open out.nam w]
$myNS namtrace-all $myNamFile

set n0 [$myNS node]
set n1 [$myNS node]
set n2 [$myNS node]
set n3 [$myNS node]
set n4 [$myNS node]
set n5 [$myNS node]
set n6 [$myNS node]
set n7 [$myNS node]

# Label the nodes
$n0 label "SRC-1"
$n7 label "DST-1"

$n4 label "SRC-2"
$n3 label "DST-2"

# Create a LAN of nodes
$myNS make-lan "$n0 $n1 $n2 $n3 $n4 $n5 $n6 $n7" 3Mb 30ms LL Queue/DropTail Mac/802_3

# Setup a pair of TCP sender and receiver (sender @ n0 , receiver @ n7)
set tcp1 [new Agent/TCP]
$myNS attach-agent $n0 $tcp1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

set sink1 [new Agent/TCPSink]
$myNS attach-agent $n7 $sink1

# Connect the TCP Source-1 and SINK-1
$myNS connect $tcp1 $sink1

# Setup another pair of TCP sender and receiver (sender @ n4, receiver @ n3)
set tcp2 [new Agent/TCP]
$myNS attach-agent $n4 $tcp2

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
```

```
set sink2 [new Agent/TCPSink]
$myNS attach-agent $n3 $sink2

# Connect the TCP Source-2 and SINK-2
$myNS connect $tcp2 $sink2

$ftp1 set type_ FTP
$ftp2 set type_ FTP

$tcp1 set class_ 1
$tcp2 set class_ 2

set outFile1 [open congestion1.xg w]
set outFile2 [open congestion2.xg w]

# Populate the TCP window values and write to a file for later plotting
# procedure to plot the congestion window
proc plotWindow {tcpSource outfile} {
      global myNS
      set curTime [$myNS now]
      set curCwnd [$tcpSource set cwnd_]

      # the data is recorded in a file called congestion.xg (this can be
      # plotted  using xgraph or gnuplot.
      # this example uses xgraph to plot the cwnd_

      puts $outfile "$curTime $curCwnd"

      $myNS at [expr $curTime+0.01] "plotWindow $tcpSource $outfile"
}

$myNS color 1 "red"
$myNS color 2 "green"

# What to do at the end of simulation ???
proc finish {} {
      global myNS myTraceFile myNamFile outFile1 outFile2
      $myNS flush-trace
      close $myTraceFile
      close $myNamFile

      # Plot a graph of Congestion window vation over time…
      exec xgraph congestion1.xg congestion2.xg -geometry 400x400 &

      exit 0
}

$myNS at 0.0 "plotWindow $tcp1 $outFile1"
$myNS at 0.0 "plotWindow $tcp2 $outFile2"

$myNS at 0.1 "$ftp1 start"
$myNS at 0.1 "$ftp2 start"
$myNS at 49.5 "$ftp1 stop"
$myNS at 49.5 "$ftp2 stop"
$myNS at 50.0 "finish"

# Start the simulation....
$myNS run

##..... END .....##
```

```
#exec xgraph congestion1.xg -geometry 400x400 &
#exec xgraph congestion2.xg -geometry 400x400 &
```

Output:



Experiment No: 5 **SIMPLE ESS WITH WIRELESS LAN**

Aim: *Simulate simple ESS and with transmitting nodes in wireless LAN by simulation and determine the performance with respect to transmission of packets.*

Program:

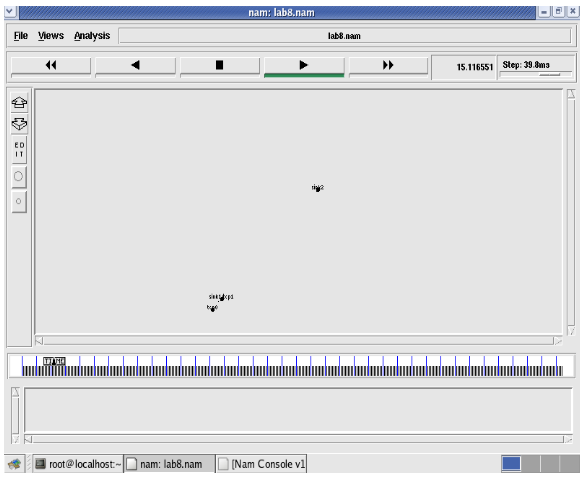
```
set ns [new Simulator]
set tf [open out.tr w]
$ns trace-all $tf

#To specify the node topology within which node is allowed to move…
set topo [new Topography]
$topo load_flatgrid 1000 1000
set nf [open out.nam w]

$ns namtrace-all-wireless $nf 1000 1000

# Configure various parameters (Layer-1,2,3) of a wireless nodes
$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 a create a matrix to store connectivity information of the
topology, node position, connection with diff nodes…
Create-god 3

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$n0 label "tcp0"
$n1 label "sink1/tcp1"
$n2 label "sink2‟
$n3 label “sink2/tcp2”

$n0 set X_ 70
$n0 set Y_ 70
$n0 set Z_ 0
```

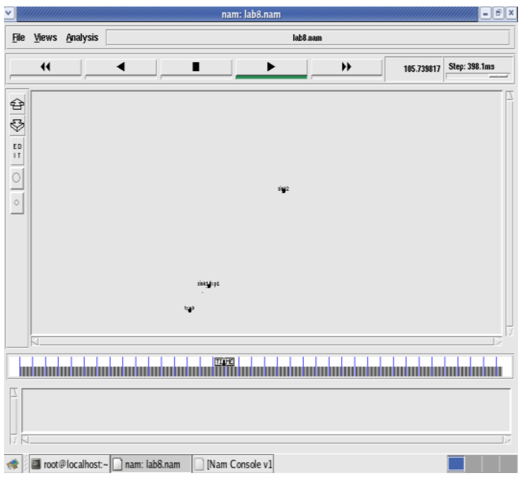
```
$n1 set X_ 120
$n1 set Y_ 120
$n1 set Z_ 0

$n2 set X_ 500
$n2 set Y_ 500
$n2 set Z_ 0

$ns at 0.1 "$n0 setdest 70 70 10"
$ns at 0.1 "$n1 setdest 120 120 20"
$ns at 0.1 "$n2 setdest 500 500 30"

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 $n2 $tcp1
set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1
set sink2 [new Agent/TCPSink]
$ns attach-agent $n3 $sink2
$ns connect $tcp1 $sink2

$ns at 5 "$ftp0 start"
$ns at 5 "$ftp1 start"
$ns at 120 "$n1 setdest 600 600 20‟

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

$ns at 400 "finish‟
$ns run
```

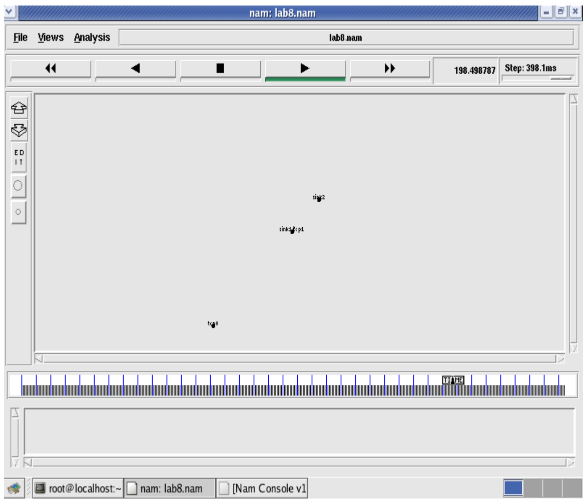
Awk File:

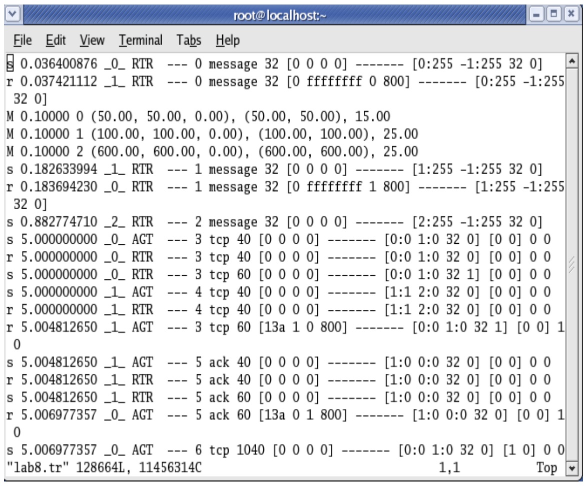
```
BEGIN{
      count1=0
      count2=0
      pack1=0
      pack2=0
      time1=0
      time2=0
}
{
      # Count number of packets received by Node-1 from Node-0
      if($1 == "r"&& $3 == "_1_" && $4 == "AGT")  {
            count1++
            pack1=pack1+$8
            time1=$2
      }
      # Count number of packets received by Node-3 from Node-2
      if($1 == "r" && $3 == "_3_" && $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 n2 to n3: %f Mbps",
      ((count2*pack2*8)/(time2*1000000)));
}
```

Output:

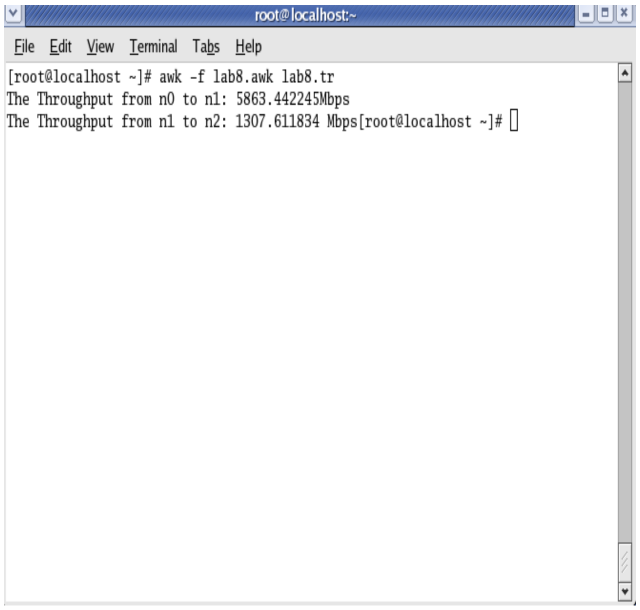








```
s 0.036400876 _0_ RTR  --- 0 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]
r 0.037421112 _1_ RTR  --- 0 message 32 [0 ffffffff 0 800] ------- [0:255 -1:255
 32 0]
M 0.10000 0 (50.00, 50.00, 0.00), (50.00, 50.00), 15.00
M 0.10000 1 (100.00, 100.00, 0.00), (100.00, 100.00), 25.00
M 0.10000 2 (600.00, 600.00, 0.00), (600.00, 600.00), 25.00
s 0.182633994 _1_ RTR  --- 1 message 32 [0 0 0 0] ------- [1:255 -1:255 32 0]
r 0.183694230 _0_ RTR  --- 1 message 32 [0 ffffffff 1 800] ------- [1:255 -1:255
 32 0]
s 0.882774710 _2_ RTR  --- 2 message 32 [0 0 0 0] ------- [2:255 -1:255 32 0]
s 5.000000000 _0_ AGT  --- 3 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 5.000000000 _0_ RTR  --- 3 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 5.000000000 _0_ RTR  --- 3 tcp 60 [0 0 0 0] ------- [0:0 1:0 32 1] [0 0] 0 0
s 5.000000000 _1_ AGT  --- 4 tcp 40 [0 0 0 0] ------- [1:1 2:0 32 0] [0 0] 0 0
r 5.000000000 _1_ RTR  --- 4 tcp 40 [0 0 0 0] ------- [1:1 2:0 32 0] [0 0] 0 0
r 5.004812650 _1_ AGT  --- 3 tcp 60 [13a 1 0 800] ------- [0:0 1:0 32 1] [0 0] 1
 0
s 5.004812650 _1_ AGT  --- 5 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0] [0 0] 0 0
r 5.004812650 _1_ RTR  --- 5 ack 40 [0 0 0 0] ------- [1:0 0:0 32 0] [0 0] 0 0
s 5.004812650 _1_ RTR  --- 5 ack 60 [0 0 0 0] ------- [1:0 0:0 32 0] [0 0] 0 0
r 5.006977357 _0_ AGT  --- 5 ack 60 [13a 0 1 800] ------- [1:0 0:0 32 0] [0 0] 1
 0
s 5.006977357 _0_ AGT  --- 6 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32 0] [1 0] 0 0
"lab8.tr" 128664L, 11456314C                                  1,1          Top
```



```
[root@localhost ~]# awk -f lab8.awk lab8.tr
The Throughput from n0 to n1: 5863.442245Mbps
The Throughput from n1 to n2: 1307.611834 Mbps[root@localhost ~]#
```

Experiment No: 6 **Implementation of Link State routing algorithm.**

Aim: *Implementation of Link state routing algorithm*

Program:

```
#NS2 simulation using Link State routing protocol
set ns [new Simulator]

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

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

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

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n1 10Mb 10ms DropTail
$ns duplex-link $n1 $n3 10Mb 10ms DropTail
$ns duplex-link $n2 $n1 10Mb 10ms DropTail

$ns duplex-link-op $n0 $n1 orient right-down
$ns duplex-link-op $n1 $n3 orient right
$ns duplex-link-op $n2 $n1 orient right-up

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

set ftp [new Application/FTP]
$ftp attach-agent $tcp

set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink

set udp [new Agent/UDP]
$ns attach-agent $n2 $udp

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp

set null [new Agent/Null]
```

```
$ns attach-agent $n3 $null

$ns connect $tcp $sink
$ns connect $udp $null

$ns rtmodel-at 1.0 down $n1 $n3
$ns rtmodel-at 2.0 up $n1 $n3
```

`$ns rtproto LS`

```
$ns at 0.0 "$ftp start"
$ns at 0.0 "$cbr start"

$ns at 5.0 "finish"

$ns run

################# Alternative code ##################

# ——————-Routing Protocol————-#
$ns rtproto LS
# ——————-ESTABLISHING COMMUNICATION ————-#
#—————-TCP CONNECTION BETWEEN NODES——————#
$ns at 0.0 "Tranmission"
proc Tranmission {} {
      global C ROU R ns
      set now [$ns now]
      set time 10.0
      set x 3
      set y 4
      set tcp1 [$ns create-connection TCP $C($x) TCPSink $R($y) 1]
      $ns at $now "$ns trace-annotate \"Time: $now Pkt Transfer between
      Client($x) Receiver($y)..\""
      $tcp1 set class_ 1
      $tcp1 set maxcwnd_ 16
      $tcp1 set packetsize_ 4000
      $tcp1 set fid_ 1
      set ftp1 [$tcp1 attach-app FTP]
      $ftp1 set interval_ .005
      $ns at $now "$ftp1 start"
}

################# Alternative code ##################
```

Part B

Experiment No: 1 Write a program for a HLDC frame to perform the following. i) Bit stuffing ii) Character stuffing.

```
#include<stdio.h>
#include<stdlib.h>
#define MAXSIZE 100

int main()
{
 char *p,*q;
 char temp;
 char in[MAXSIZE];
 char stuff[MAXSIZE];
 char destuff[MAXSIZE];

 int count=0;

 printf("enter the input character string (0‘s & 1‘s only):\n");
 scanf("%s",in);

 p=in;
 q=stuff;

 while(*p != '\0') // Check for the end of the string….
 {
  if(*p == '0')   // Check for the letter 0
  {
   *q=*p;
   q++;
   p++;
  }
  else
  {
   while(*p == '1' && count!=5) // check for 5 consecutive 1’s
   {
    count++;
    *q=*p;
    q++;
    p++;
   }

   if(count == 5) // when 5 1’s found stuff an extra ‘0’
   {
    *q='0';
    q++;
   }
   count=0;
  }
 }
 *q='\0'; // Null terminate the stuff string
 printf("\nthe stuffed character string is");
 printf("\n%s", stuff);

 p=stuff;
 q=destuff;
 while(*p!='\0')  // Try de-stuffing till end of the data
 {
  if(*p == '0')   // What to do when ‘0’ is found ???
  {
   *q=*p;
   q++;
   p++;
  }
  else
  {
   while(*p == '1' && count!=5) // What to do when '1' is found 5 times ?
   {
    count++;
    *q=*p;
    q++;
    p++;
   }
   if(count == 5) // 5 1's found, remove the stuffed '0'
   {
    p++;
   }
   count=0;
  }
 }
 *q='\0';
 printf("\n the destuffed character string is");
 printf("\n %s \n",destuff);
 return 0;
}
```

Output:

```
enter the input character string (0's & 1's only):
1 0 1 0 1 1 1 1 1 1

the stuffed character string is
1 0 1 0 1 1 1 1 1 0 1

the destuffed character string is
1 0 1 0 1 1 1 1 1 1
```

Experiment No: 2 **Distance Vector Algorithm**

Aim: *C Program for Distance Vector Algorithm to find suitable path for transmission*

Distance Vector Algorithm is a decentralized routing algorithm that requires that each router simply inform its neighbors of its routing table. For each network path, the receiving routers pick the neighbor advertising the lowest cost, then add this entry into its routing table for re-advertisement. To find the shortest path, Distance Vector Algorithm is based on one of two basic algorithms: the Bellman-Ford and the Dijkstra algorithms.

Routers that use this algorithm have to maintain the distance tables (which is a one-dimension array -- "a vector"), which tell the distances and shortest path to sending packets to each node in the network. The information in the distance table is always upd by exchanging information with the neighboring nodes. The number of data in the table equals to that of all nodes in networks (excluded itself). The columns of table represent the directly attached neighbors whereas the rows represent all destinations in the network. Each data contains the path for sending packets to each destination in the network and distance/or time to transmit on that path (we call this as "cost"). The measurements in this algorithm are the number of hops, latency, the number of outgoing packets, etc.

The starting assumption for distance-vector routing is each node knows the cost of the link of each of its directly connected neighbors. Next, every node sends a configured message to its directly connected neighbors containing its own distance table. Now, every node can learn and up its distance table with cost and next hops for all nodes network. Repeat exchanging until no more information between the neighbors.

Consider a node A that is interested in routing to destination H via a directly attached neighbor J. Node A's distance table entry, Dx(Y,Z) is the sum of the cost of the direct-one hop link between A and J, c(A,J), plus neighboring J's currently known minimum-cost path (shortest path) from itself(J) to H. That is Dx(H,J) = c(A,J) + minw{Dj(H,w)} The minw is taken over all the J's.

This equation suggests that the form of neighbor-to-neighbor communication that will take place in the DV algorithm - each node must know the cost of each of its neighbors' minimum-cost path to each destination. Hence, whenever a node computes a new minimum cost to some destination, it must inform its neighbors of this new minimum cost.

Implementation Algorithm:

1. send my routing table to all my neighbors whenever my link table changes

2. when I get a routing table from a neighbor on port P with link metric M:
    a. add L to each of the neighbor's metrics
    b. for each entry (D, P', M') in the updated neighbor's table:
        i. if I do not have an entry for D, add (D, P, M') to my routing table
        ii. if I have an entry for D with metric M", add (D, P, M') to my routing table if M' < M"
3. if my routing table has changed, send all the new entries to all my neighbors

Program:

```
#include<stdio.h>
struct node
{
      int dist[20];
      int from[20];
}rt[10];

int main()
{
      int distance_matrix[20][20];
      int n,i,j,k,count=0,src,dest;
      printf("\nEnter the number of nodes : ");
      scanf("%d",&n);
      printf("\nEnter the cost/distance matrix :\n");
      for(i=0;i<n;i++)
            for(j=0;j<n;j++)
            {
                  scanf("%d", &distance_matrix[i][j]);
                  distance_matrix[i][i]=0;
                  rt[i].dist[j] = distance_matrix[i][j];
                  rt[i].from[j] = j;
            }
      do
      {
            count=0;
            for(i=0;i<n;i++)
                  for(j=0;j<n;j++)
                        for(k=0;k<n;k++)

            if(rt[i].dist[j]>distance_matrix[i][k]+rt[k].dist[j])
                              {

            rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
                                    rt[i].from[j]=k;
                                    count++;
                              }
      } while(count!=0);
      for(i=0;i<n;i++)
      {
            printf("\nRouting table for router %d
                  :\nDest\tNextHop\tDist\n",i+1);
            for(j=0;j<n;j++)
                  printf("%d\t%d\t%d\n",j+1,rt[i].from[j]+1,rt[i].dist[j]);
      }
      printf("\nEnter source and destination : ");
      scanf("%d%d",&src,&dest);
      src--;      dest--;
      printf("Shortest path : \n Via router : %d\n Shortest distance : %d \n",
            rt[src].from[dest]+1, rt[src].dist[dest]);
            return 0;
}
```

<u>Output:</u>

```
[root@localhost ]# cc prg3.c
[root@localhost ]# ./a.out

enter the value of no. of nodes
4
Enter the routing table :
   |a  b   c   d
-----------------------------------
a   |0  5  1   4
b   |5  0  6   2
c   |1  6  0   3
d   |4  2  3   0

table for router a
a:: 0 via a
b:: 5 via a
c:: 1 via a
d:: 4 via a

table for router b
a:: 5 via b
b:: 0 via b
c:: 5 via d
d:: 2 via b

table for router c
a:: 1 via c
b:: 5 via d
c:: 0 via c
d:: 3 via c
```

Experiment No: 3 **Implement Dijkstra's algorithm to compute the shortest routing path.**

Program:

```
#include "stdio.h"
#include "conio.h"
#define infinity 999

void dij(int n,int v,int cost[10][10],int dist[])
{
 int i,u,count,w,flag[10],min;
 for(i=1;i<=n;i++)
 flag[i]=0,dist[i]=cost[v][i];
 count=2;
 while(count<=n)
 {
 min=99;
 for(w=1;w<=n;w++)
  if(dist[w]<min && !flag[w])
  min=dist[w],u=w;
 flag[u]=1;
 count++;
 for(w=1;w<=n;w++)
  if((dist[u]+cost[u][w]<dist[w]) && !flag[w])
  dist[w]=dist[u]+cost[u][w];
 }
}

void main()
{
 int n,v,i,j,cost[10][10],dist[10];
 clrscr();
 printf("n Enter the number of nodes:");
 scanf("%d",&n);
 printf("n Enter the cost matrix:n");
 for(i=1;i<=n;i++)
 for(j=1;j<=n;j++)
 {
  scanf("%d",&cost[i][j]);
  if(cost[i][j] == 0)
  cost[i][j]=infinity;
 }
 printf("n Enter the source matrix:");
 scanf("%d",&v);
 dij(n,v,cost,dist);
 printf("n Shortest path:n");
 for(i=1;i<=n;i++)
 if(i!=v)
  printf("%d->%d,cost=%dn",v,i,dist[i]);
 getch();
}
```

Experiment No: 4 **Error Detecting Code Using CRC-CCITT (16-bit)**

Aim: *C Program for ERROR detecting code using CRC-CCITT (16bit).*

Whenever digital data is stored or interfaced, data corruption might occur. Since the beginning of computer science, developers have been thinking of ways to deal with this type of problem. For serial data they came up with the solution to attach a parity bit to each sent byte. This simple detection mechanism works if an odd number of bits in a byte changes, but an even number of false bits in one byte will not be detected by the parity check. To overcome this problem developers have searched for mathematical sound mechanisms to detect multiple false bits. The **CRC** calculation or *cyclic redundancy check* was the result of this. Nowadays CRC calculations are used in all types of communications. All packets sent over a network connection are checked with a CRC. Also each data block on your hard disk has a CRC value attached to it. Modern computer world cannot do without these CRC calculations. So let's see why they are so widely used. The answer is simple; they are powerful, detect many types of errors and are extremely fast to calculate especially when dedicated hardware chips are used.

The idea behind CRC calculation is to look at the data as one large binary number. This number is divided by a certain value and the remainder of the calculation is called the CRC. Dividing in the CRC calculation at first looks to cost a lot of computing power, but it can be performed very quickly if we use a method similar to the one learned at school. We will as an example calculate the remainder for the character 'm'—which is 1101101 in binary notation—by dividing it by 19 or 10011. Please note that 19 is an odd number. This is necessary as we will see further on. Please refer to your schoolbooks as the binary calculation method here is not very different from the decimal method you learned when you were young. It might only look a little bit strange. Also notations differ between countries, but the method is similar.

```
                    1 0 1 = 5
            -------------
1 0 0 1 1 / 1 1 0 1 1 0 1
            1 0 0 1 1 | |
            --------- | |
              1 0 0 0 0 |
              0 0 0 0 0 |
              --------- |
              1 0 0 0 0 1
                1 0 0 1 1
                ---------
                  1 1 1 0 = 14 = remainder
```

With decimal calculations you can quickly check that 109 divided by 19 gives a quotient of 5 with 14 as the remainder. But what we also see in the scheme is that every bit extra to check only costs one binary comparison and in 50% of the cases one binary subtraction. You can easily increase the number of bits of the test data string—for example to 56 bits if we use our example value "*Lammert*"—and the result can be calculated with 56 binary comparisons and an average of 28 binary subtractions. This can be implemented in hardware directly with only very few transistors involved. Also software algorithms can be very efficient.

All of the CRC formulas you will encounter are simply checksum algorithms based on modulo-2 binary division where we ignore carry bits and in effect the subtraction will be equal to an *exclusive or* operation. Though some differences exist in the specifics across different CRC formulas, the basic mathematical process is always the same:

- The message bits are appended with *c* zero bits; this *augmented message* is the dividend
- A predetermined *c+1*-bit binary sequence, called the *generator polynomial*, is the divisor
- The checksum is the *c*-bit remainder that results from the division operation

Table 1 lists some of the most commonly used generator polynomials for 16- and 32-bit CRCs. Remember that the width of the divisor is always one bit wider than the remainder. So, for example, you'd use a 17-bit generator polynomial whenever a 16-bit checksum is required.

	CRC-CCITT	**CRC-16**	**CRC-32**
Checksum Width	16 bits	16 bits	32 bits
Generator Polynomial	10001000000100001	11000000000000101	100000100110000010001110110110111

Table 1. International Standard CRC Polynomials

Error detection with CRC

Consider a message represented by the polynomial M(x)

Consider a *generating polynomial* G(x)

This is used to generate a CRC = C(x) to be appended to M(x).

Note this G(x) is prime.

Steps:

2. Multiply M(x) by highest power in G(x). i.e. Add So much zeros to M(x).
3. Divide the result by G(x). The remainder = C(x).
   Special case: This won't work if bitstring =all zeros. We don't allow such an M(x).But M(x) bitstring = 1 will work, for example. Can divide 1101 into 1000.
4. If: x div y gives remainder c
   that means: x = n y + c
   Hence (x-c) = n y
   (x-c) div y gives remainder 0

Here (x-c) = (x+c)

Hence (x+c) div y gives remainder 0

5. Transmit: T(x) = M(x) + C(x)
6. Receiver end: Receive T(x). Divide by G(x), should have remainder 0.

Note if G(x) has order n - highest power is x^n,

then G(x) will cover (n+1) bits

and the *remainder* will cover n bits.

i.e. Add n bits (Zeros) to message.

<u>Program:</u>

```
#include<stdio.h>
#include<string.h>

// mode=1 is for calculating CRC of original message and create final message
to be transmitted at sender's end

// mode=0 is for calculating CRC of received message at reciever's end

int crc(char *input, char *output, const char *gp, int mode)
{
      int j,k;
      strcpy(output,input); // copy original message to output
      if(mode)
      {
            for(j=1; j<strlen(gp); j++)
            // append as many zeroes as the width (position of the highest
            // 1 bit) of generator polynomial.
            strcat(output,"0");
      }

      // calculate final message to be transmitted and store it in
      // output - loop simulates CRC division and finding of remainder

      for(j=0; j<strlen(input); j++)
            if(*(output+j) == '1')
                  for(k=0; k<strlen(gp); k++)
                  {
                        // equivalent to performing XOR operation
                        if (((*(output+j+k) == '0') && (gp[k] == '0') ||
                           (*(output+j+k) == '1') && (gp[k] == '1')))
                              *(output+j+k)='0';
                        else
                              *(output+j+k)='1';
                  }
      // if any of the bits is 1, then output is not 0, hence there is error
      for(j=0; j<strlen(output); j++)
            if(output[j] == '1')
                  return 1;
      return 0; // error free
}

int main()
{
      char input[50],output[50],recv[50];
      const char gp[18]="10001000000100001"; // 16-bit CRC-CCITT
            specification G(x): x^16+x^12+x^5+1
      printf("\n Enter the input message in binary\n");
      scanf("%s",input);
      crc(input,output,gp,1);
      printf("\n The transmitted message is %s%s\n", input,
      output+strlen(input));
            printf("\n\n Enter the recevied message in binary \n");
            scanf("%s",recv);
            if(!crc(recv,output,gp,0))
                  printf("\n No error in data\n");
            else
                  printf("\n Error in data transmission has occurred\n");
            return 0;
      }
```

Output:

```
[root@localhost ]# cc prg1.c
[root@localhost ]# ./a.out
Enter the length of Data Frame :4
Enter the Message :1 0 1 1
Data to be transmitted : 1 0 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1
Enter the Reveived Data : 1 0 1 1 1 0 1 1 0 0 0 0 0 1 1 0 1 0 1 1
ERROR in Recived Data
Remender is : 0000000100000000
```

Experiment No: 6 Congestion Control Using Leaky Bucket Algorithm

Aim: *C Program for Congestion control using Leaky Bucket Algorithm*

The main concept of the leaky bucket algorithm is that the output data flow remains constant despite the variant input traffic, such as the water flow in a bucket with a small hole at the bottom. In case the bucket contains water (or packets) then the output flow follows a constant rate, while if the bucket is full any additional load will be lost because of spillover. In a similar way if the bucket is empty the output will be zero. From network perspective, leaky bucket consists of a finite queue (bucket) where all the incoming packets are stored in case there is space in the queue, otherwise the packets are discarded. In order to regulate the output flow, leaky bucket transmits one packet from the queue in a fixed time (e.g. at every clock tick). In the following figure we can notice the main rationale of leaky bucket algorithm, for both the two approaches (e.g. leaky bucket with water (a) and with packets (b)).

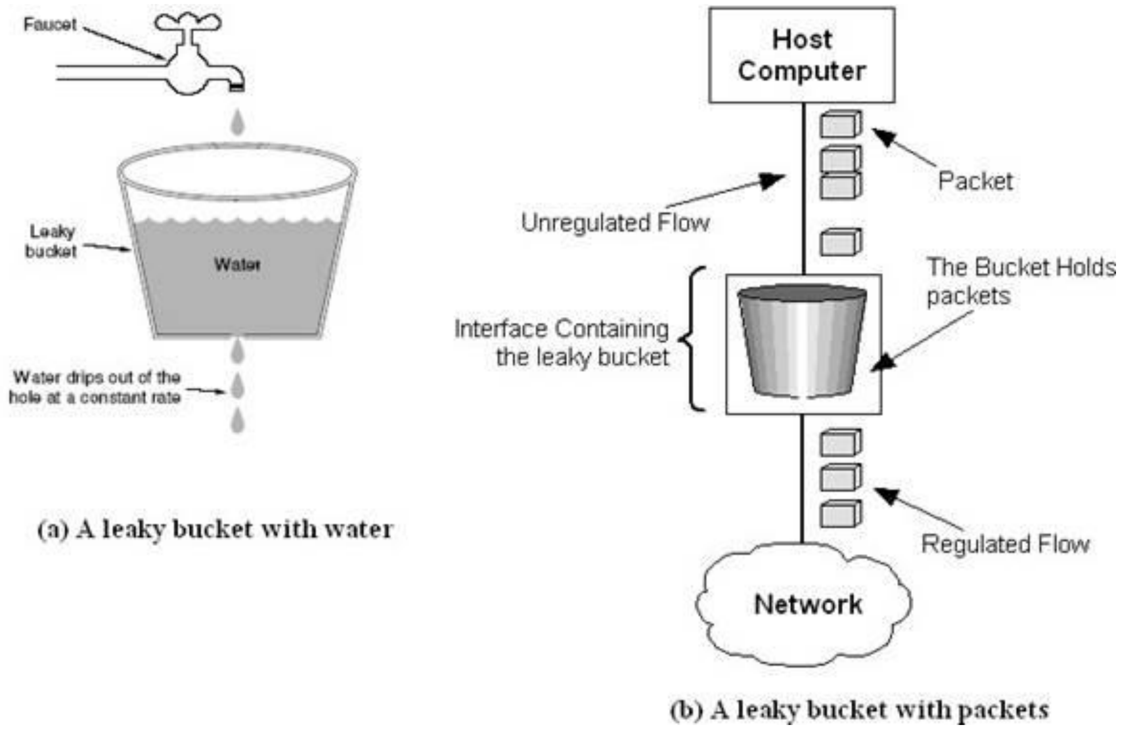

Faucet
Leaky bucket
Water
Water drips out of the hole at a constant rate
(a) A leaky bucket with water
Host Computer
Packet
Unregulated Flow
The Bucket Holds packets
Interface Containing the leaky bucket
Regulated Flow
Network
(b) A leaky bucket with packets

Figure: The leaky bucket traffic shaping algorithm

While leaky bucket eliminates completely bursty traffic by regulating the incoming data flow its main drawback is that it drops packets if the bucket is full. Also, it doesn't take into account the idle process of the sender which means that if the host doesn't transmit data for some time the bucket becomes empty without permitting the transmission of any packet.

Implementation Algorithm:

Steps:

1. Read The Data For Packets

```
2. Read The Queue Size
3. Divide the Data into Packets
4. Assign the random Propagation delays for each packets to input into the bucket
   (input_packet).
5. wlile((Clock++ < 5*total_packets)and
     (out_packets < total_paclets))
     a. if (clock == input_packet)
          i. insert into Queue
     b. if (clock % 5 == 0 )
          i. Remove packet from Queue
6. Stop
```

Program:

```
#include<stdio.h>
int min(int x, int y)
{
  if(x<y)
    return x;
  else
    return y;
}

int main()
{
  int drop=0, mini, nsec, cap, count=0, i, inp[25], process;
      // note count is a local variable
  printf("Enter bucket size :");
  scanf("%d",&cap);// cap = size of bucket
  printf("Enter transmission rate :");
  scanf("%d", &process);//process= rate of transmission
  printf("Enter The duration of simulation in seconds : ");
  scanf("%d", &nsec);//nsec = duration of simulation in seconds
  for(i=0;i<nsec;i++)
  {
     printf("Enter packet size at %d sec : ",i+1);
     scanf("%d", &inp[i]);//inp = input packet size
  }
  printf("\nSecond|Packet Recieved|Packet Sent|Packet Left|Packet Dropped|\n");
      ///main stuff to remember
  printf("--------------------------------------------------------------\n");

  for(i=0;i<nsec;i++)
    {
    // initaializing the packet size to the first packet
    count+=inp[i];

     // checking if the overall packet size is greater than the bucket size
     if(count>cap)
    {
      drop=count-cap;// if true this would lead to drop in packet
      count=cap; // count value is set to bucket size
    }
    printf("%d",i+1);// print the second
    printf("\t%d",inp[i]);// print the input size
    mini=min(count,process); //checking which is smaller the bucket size or
the rate of transmission
    printf("\t\t%d",mini);// printing the rate of transmission
    count=count-mini;//again seting the count value by deducting the
transmited value not the transmission rate
    printf("\t\t%d",count);//packets left in the bucket
    printf("\t\t%d\n",drop);// printing the pre calulated value of the drop
packets
    drop=0;// seting the drop packet to zero
    sleep(1);// sleeping for a second
  }
  // below here the line of code is used to send all the packect even thought
the ime has elapsed
  for(;count!=0;i++)
  {
    if(count>cap)
    {
      drop=count-cap;
      count=cap;
    }
    printf("%d",i+1);
    printf("\t0");
    mini=min(count,process);
    printf("\t\t%d",mini);
    count=count-mini;
    printf("\t\t%d",count);
    printf("\t\t%d\n",drop);
    sleep(1);
  }
}
```

VIVA QUESTION AND ANSWER

1) What is a Link?
A link refers to the connectivity between two devices. It includes the type of cables and protocols used in order for one device to be able to communicate with the other.
2) What are the layers of the OSI reference model?
There are 7 OSI layers: Physical Layer, Data Link Layer, Network Layer, Transport Layer, Session Layer, Presentation Layer and Application Layer.
3) What is backbone network?
A backbone network is a centralized infrastructure that is designed to distribute different routes and data to various networks. It also handles management of bandwidth and various channels.
4) What is a LAN?
LAN is short for Local Area Network. It refers to the connection between computers and other network devices that are located within a small physical location.
5) What is a node?
A node refers to a point or joint where a connection takes place. It can be computer or device that is part of a network. Two or more nodes are needed in order to form a network connection.
6) What are routers?
Routers can connect two or more network segments. These are intelligent network devices that store information in its routing table such as paths, hops and bottlenecks. With this info, they are able to determine the best path for data transfer. Routers operate at the OSI Network Layer.
7) What is point to point link?
It refers to a direct connection between two computers on a network. A point to point connection does not need any other network devices other than connecting a cable to the NIC cards of both computers.
8) What is anonymous FTP?
Anonymous FTP is a way of granting user access to files in public servers. Users that are allowed access to data in these servers do not need to identify themselves, but instead log in as an anonymous guest.
9) What is subnet mask?
A subnet mask is combined with an IP address in order to identify two parts: the extended network address and the host address. Like an IP address, a subnet mask is made up of 32 bits.
10) What is the maximum length allowed for a UTP cable?
A single segment of UTP cable has an allowable length of 90 to 100 meters. This limitation can be overcome by using repeaters and switches.
11) What is data encapsulation?
Data encapsulation is the process of breaking down information into smaller manageable chunks before it is transmitted across the network. It is also in this process that the source and destination addresses are attached into the headers, along with parity checks.
12) Describe Network Topology
Network Topology refers to the layout of a computer network. It shows how devices and cables are physically laid out, as well as how they connect to one another.
13) What is VPN?

VPN means Virtual Private Network, a technology that allows a secure tunnel to be created across a network such as the Internet. For example, VPNs allow you to establish a secure dialup connection to a remote server.

14) Briefly describe NAT.

NAT is Network Address Translation. This is a protocol that provides a way for multiple computers on a common network to share single connection to the Internet.

15) What is the job of the Network Layer under the OSI reference model?

The Network layer is responsible for data routing, packet switching and control of network congestion. Routers operate under this layer.

16) How does a network topology affect your decision in setting up a network?

Network topology dictates what media you must use to interconnect devices. It also serves as basis on what materials, connector and terminations that is applicable for the setup.

17) What is RIP?

RIP, short for Routing Information Protocol is used by routers to send data from one network to another. It efficiently manages routing data by broadcasting its routing table to all other routers within the network. It determines the network distance in units of hops.

18) What are different ways of securing a computer network?

There are several ways to do this. Install reliable and updated anti-virus program on all computers. Make sure firewalls are setup and configured properly. User authentication will also help a lot. All of these combined would make a highly secured network.

19) What is NIC?

NIC is short for Network Interface Card. This is a peripheral card that is attached to a PC in order to connect to a network. Every NIC has its own MAC address that identifies the PC on the network.

20) What is WAN?

WAN stands for Wide Area Network. It is an interconnection of computers and devices that are geographically dispersed. It connects networks that are located in different regions and countries.

21) What is the importance of the OSI Physical Layer?

The physical layer does the conversion from data bits to electrical signal, and vice versa. This is where network devices and cable types are considered and setup.

22) How many layers are there under TCP/IP?

There are four layers: the Network Layer, Internet Layer, Transport Layer and Application Layer.

23) What are proxy servers and how do they protect computer networks?

Proxy servers primarily prevent external users who identifying the IP addresses of an internal network. Without knowledge of the correct IP address, even the physical location of the network cannot be identified. Proxy servers can make a network virtually invisible to external users.

24) What is the function of the OSI Session Layer?

This layer provides the protocols and means for two devices on the network to communicate with each other by holding a session. This includes setting up the session, managing information exchange during the session, and tear-down process upon termination of the session.

25) What is the importance of implementing a Fault Tolerance System? Are there limitations?

A fault tolerance system ensures continuous data availability. This is done by eliminating a single point of failure. However, this type of system would not be able to protect data in some cases, such as in accidental deletions.

26) What does 10Base-T mean?

The 10 refers to the data transfer rate, in this case is 10Mbps. The word Base refers to base band, as oppose to broad band. T means twisted pair, which is the cable used for that network.

27) What is a private IP address?

Private IP addresses are assigned for use on intranets. These addresses are used for internal networks and are not routable on external public networks. These ensures that no conflicts are present among internal networks while at the same time the same range of private IP addresses are reusable for multiple intranets since they do not "see" each other.

28) What is NOS?

NOS, or Network Operating System, is specialized software whose main task is to provide network connectivity to a computer in order for it to be able to communicate with other computers and connected devices.

29) What is DoS?

DoS, or Denial-of-Service attack, is an attempt to prevent users from being able to access the internet or any other network services. Such attacks may come in different forms and are done by a group of perpetuators. One common method of doing this is to overload the system server so it cannot anymore process legitimate traffic and will be forced to reset.

30) What is OSI and what role does it play in computer networks?

OSI (Open Systems Interconnect) serves as a reference model for data communication. It is made up of 7 layers, with each layer defining a particular aspect on how network devices connect and communicate with one another. One layer may deal with the physical media used, while another layer dictates how data is actually transmitted across the network.

31) What is the purpose of cables being shielded and having twisted pairs?

The main purpose of this is to prevent crosstalk. Crosstalks are electromagnetic interferences or noise that can affect data being transmitted across cables.

32) What is the advantage of address sharing?

By using address translation instead of routing, address sharing provides an inherent security benefit. That's because host PCs on the Internet can only see the public IP address of the external interface on the computer that provides address translation and not the private IP addresses on the internal network.

33) What are MAC addresses?

MAC, or Media Access Control, uniquely identifies a device on the network. It is also known as physical address or Ethernet address. A MAC address is made up of 6-byte parts.

34) What is the equivalent layer or layers of the TCP/IP Application layer in terms of OSI reference model?

The TCP/IP Application layer actually has three counterparts on the OSI model: the Session layer, Presentation Layer and Application Layer.

35) How can you identify the IP class of a given IP address?

By looking at the first octet of any given IP address, you can identify whether it's Class A, B or C. If the first octet begins with a 0 bit, that address is Class A. If it begins with bits 10 then that address is a Class B address. If it begins with 110, then it's a Class C network.

36) What is the main purpose of OSPF?

OSPF, or Open Shortest Path First, is a link-state routing protocol that uses routing tables to determine the best possible path for data exchange.

37) What are firewalls?
Firewalls serve to protect an internal network from external attacks. These external threats can be hackers who want to steal data or computer viruses that can wipe out data in an instant. It also prevents other users from external networks from gaining access to the private network.
38) Describe star topology
Star topology consists of a central hub that connects to nodes. This is one of the easiest to setup and maintain.
39) What are gateways?
Gateways provide connectivity between two or more network segments. It is usually a computer that runs the gateway software and provides translation services. This translation is a key in allowing different systems to communicate on the network.
40) What is the disadvantage of a star topology?
One major disadvantage of star topology is that once the central hub or switch get damaged, the entire network becomes unusable.
41) What is SLIP?
SLIP, or Serial Line Interface Protocol, is actually an old protocol developed during the early UNIX days. This is one of the protocols that are used for remote access.
42) Give some examples of private network addresses.
10.0.0.0 with a subnet mask of 255.0.0.0
172.16.0.0 with subnet mask of 255.240.0.0
192.168.0.0 with subnet mask of 255.255.0.0
43) What is tracert?
Tracert is a Windows utility program that can used to trace the route taken by data from the router to the destination network. It also shows the number of hops taken during the entire transmission route.
44) What are the functions of a network administrator?
A network administrator has many responsibilities that can be summarize into 3 key functions: installation of a network, configuration of network settings, and maintenance/troubleshooting of networks.
45) Describe at one disadvantage of a peer to peer network.
When you are accessing the resources that are shared by one of the workstations on the network, that workstation takes a performance hit.
46) What is Hybrid Network?
A hybrid network is a network setup that makes use of both client-server and peer-to-peer architecture.
47) What is DHCP?
DHCP is short for Dynamic Host Configuration Protocol. Its main task is to automatically assign an IP address to devices across the network. It first checks for the next available address not yet taken by any device, then assigns this to a network device.
48) What is the main job of the ARP?
The main task of ARP or Address Resolution Protocol is to map a known IP address to a MAC layer address.
49) What is TCP/IP?
TCP/IP is short for Transmission Control Protocol / Internet Protocol. This is a set of protocol layers that is designed to make data exchange possible on different types of computer networks, also known as heterogeneous network.
50) How can you manage a network using a router?

Routers have built in console that lets you configure different settings, like security and data logging. You can assign restrictions to computers, such as what resources it is allowed access, or what particular time of the day they can browse the internet. You can even put restrictions on what websites are not viewable across the entire network.

51) What protocol can be applied when you want to transfer files between different platforms, such between UNIX systems and Windows servers?

Use FTP (File Transfer Protocol) for file transfers between such different servers. This is possible because FTP is platform independent.

52) What is the use of a default gateway?

Default gateways provide means for the local networks to connect to the external network. The default gateway for connecting to the external network is usually the address of the external router port.

53) One way of securing a network is through the use of passwords. What can be considered as good passwords?

Good passwords are made up of not just letters, but by combining letters and numbers. A password that combines uppercase and lowercase letters is favorable than one that uses all upper case or all lower case letters. Passwords must be not words that can easily be guessed by hackers, such as dates, names, favorites, etc. Longer passwords are also better than short ones.

54) What is the proper termination rate for UTP cables?

The proper termination for unshielded twisted pair network cable is 100 ohms.

55) What is netstat?

Netstat is a command line utility program. It provides useful information about the current TCP/IP settings of a connection.

56) What is the number of network IDs in a Class C network?

For a Class C network, the number of usable Network ID bits is 21. The number of possible network IDs is 2 raised to 21 or 2,097,152. The number of host IDs per network ID is 2 raised to 8 minus 2, or 254.

57) What happens when you use cables longer than the prescribed length?

Cables that are too long would result in signal loss. This means that data transmission and reception would be affected, because the signal degrades over length.

58) What common software problems can lead to network defects?

Software related problems can be any or a combination of the following:
- client server problems
- application conflicts
- error in configuration
- protocol mismatch
- security issues
- user policy and rights issues

59) What is ICMP?

ICMP is Internet Control Message Protocol. It provides messaging and communication for protocols within the TCP/IP stack. This is also the protocol that manages error messages that are used by network tools such as PING.

60) What is Ping?

Ping is a utility program that allows you to check connectivity between network devices on the network. You can ping a device by using its IP address or device name, such as a computer name.

61) What is peer to peer?

Peer to peer are networks that does not reply on a server. All PCs on this network act as individual workstations.

62) What is DNS?

DNS is Domain Name System. The main function of this network service is to provide host names to TCP/IP address resolution.

63) What advantages does fiber optics have over other media?

One major advantage of fiber optics is that is it less susceptible to electrical interference. It also supports higher bandwidth, meaning more data can be transmitted and received. Signal degrading is also very minimal over long distances.

64) What is the difference between a hub and a switch?

A hub acts as a multiport repeater. However, as more and more devices connect to it, it would not be able to efficiently manage the volume of traffic that passes through it. A switch provides a better alternative that can improve the performance especially when high traffic volume is expected across all ports.

65) What are the different network protocols that are supported by Windows RRAS services?

There are three main network protocols supported: NetBEUI, TCP/IP, and IPX.

66) What are the maximum networks and hosts in a class A, B and C network?

For Class A, there are 126 possible networks and 16,777,214 hosts
For Class B, there are 16,384 possible networks and 65,534 hosts
For Class C, there are 2,097,152 possible networks and 254 hosts

67) What is the standard color sequence of a straight-through cable?

orange/white, orange, green/white, blue, blue/white, green, brown/white, brown.

68) What protocols fall under the Application layer of the TCP/IP stack?

The following are the protocols under TCP/IP Application layer: FTP, TFTP, Telnet and SMTP.

69) You need to connect two computers for file sharing. Is it possible to do this without using a hub or router?

Yes, you can connect two computers together using only one cable. A crossover type cable can be use in this scenario. In this setup, the data transmit pin of one cable is connected to the data receive pin of the other cable, and vice versa.

70) What is ipconfig?

Ipconfig is a utility program that is commonly used to identify the addresses information of a computer on a network. It can show the physical address as well as the IP address.

71) What is the difference between a straight-through and crossover cable?

A straight-through cable is used to connect computers to a switch, hub or router. A crossover cable is used to connect two similar devices together, such as a PC to PC or Hub to hub.

72) What is client/server?

Client/server is a type of network wherein one or more computers act as servers. Servers provide a centralized repository of resources such as printers and files. Clients refers to workstation that access the server.

73) Describe networking.

Networking refers to the inter connection between computers and peripherals for data communication. Networking can be done using wired cabling or through wireless link.

74) When you move the NIC cards from one PC to another PC, does the MAC address gets transferred as well?

Yes, that's because MAC addresses are hard-wired into the NIC circuitry, not the PC. This also means that a PC can have a different MAC address when the NIC card was replace by another one.

75) Explain clustering support

Clustering support refers to the ability of a network operating system to connect multiple servers in a fault-tolerant group. The main purpose of this is the in the event that one server fails, all processing will continue on with the next server in the cluster.

76) In a network that contains two servers and twenty workstations, where is the best place to install an Anti-virus program?

An anti-virus program must be installed on all servers and workstations to ensure protection. That's because individual users can access any workstation and introduce a computer virus when plugging in their removable hard drives or flash drives.

77) Describe Ethernet.

Ethernet is one of the popular networking technologies used these days. It was developed during the early 1970s and is based on specifications as stated in the IEEE. Ethernet is used in local area networks.

78) What are some drawbacks of implementing a ring topology?

In case one workstation on the network suffers a malfunction, it can bring down the entire network. Another drawback is that when there are adjustments and reconfigurations needed to be performed on a particular part of the network, the entire network has to be temporarily brought down as well.

79) What is the difference between CSMA/CD and CSMA/CA?

CSMA/CD, or Collision Detect, retransmits data frames whenever a collision occurred. CSMA/CA, or Collision Avoidance, will first broadcast intent to send prior to data transmission.

80) What is SMTP?

SMTP is short for Simple Mail Transfer Protocol. This protocol deals with all Internal mail, and provides the necessary mail delivery services on the TCP/IP protocol stack.

81) What is multicast routing?

Multicast routing is a targeted form of broadcasting that sends message to a selected group of user, instead of sending it to all users on a subnet.

82) What is the importance of Encryption on a network?

Encryption is the process of translating information into a code that is unreadable by the user. It is then translated back or decrypted back to its normal readable format using a secret key or password. Encryption help ensure that information that is intercepted halfway would remain unreadable because the user has to have the correct password or key for it.

83) How are IP addresses arranged and displayed?

IP addresses are displayed as a series of four decimal numbers that are separated by period or dots. Another term for this arrangement is the dotted decimal format. An example is 192.168.101.2

84) Explain the importance of authentication.

Authentication is the process of verifying a user's credentials before he can log into the network. It is normally performed using a username and password. This provides a secure means of limiting the access from unwanted intruders on the network.

85) What do mean by tunnel mode?

This is a mode of data exchange wherein two communicating computers do not use IPSec themselves. Instead, the gateway that is connecting their LANs to the transit network creates a virtual tunnel that uses the IPSec protocol to secure all communication that passes through it.

86) What are the different technologies involved in establishing WAN links?

Analog connections - using conventional telephone lines; Digital connections - using digitalgrade telephone lines; switched connections - using multiple sets of links between sender and receiver to move data.

87) What is one advantage of mesh topology?

In the event that one link fails, there will always be another available. Mesh topology is actually one of the most fault-tolerant network topology.

88) When troubleshooting computer network problems, what common hardware-related problems can occur?

A large percentage of a network is made up of hardware. Problems in these areas can range from malfunctioning hard drives, broken NICs and even hardware startups. Incorrectly hardware configuration is also one of those culprits to look into.

89) What can be done to fix signal attenuation problems?

A common way of dealing with such a problem is to use repeaters and hub, because it will help regenerate the signal and therefore prevent signal loss. Checking if cables are properly terminated is also a must.

90) How does dynamic host configuration protocol aid in network administration?

Instead of having to visit each client computer to configure a static IP address, the network administrator can apply dynamic host configuration protocol to create a pool of IP addresses known as scopes that can be dynamically assigned to clients.

91) Explain profile in terms of networking concept?

Profiles are the configuration settings made for each user. A profile may be created that puts a user in a group, for example.

92) What is sneakernet?

Sneakernet is believed to be the earliest form of networking wherein data is physically transported using removable media, such as disk, tapes.

93) What is the role of IEEE in computer networking?

IEEE, or the Institute of Electrical and Electronics Engineers, is an organization composed of engineers that issues and manages standards for electrical and electronic devices. This includes networking devices, network interfaces, cablings and connectors.

94) What protocols fall under the TCP/IP Internet Layer?

There are 4 protocols that are being managed by this layer. These are ICMP, IGMP, IP and ARP.

95) When it comes to networking, what are rights?

Rights refer to the authorized permission to perform specific actions on the network. Each user on the network can be assigned individual rights, depending on what must be allowed for that user.

96) What is one basic requirement for establishing VLANs?

A VLAN requires dedicated equipment on each end of the connection that allows messages entering the Internet to be encrypted, as well as for authenticating users.

97) What is IPv6?

IPv6 , or Internet Protocol version 6, was developed to replace IPv4. At present, IPv4 is being used to control internet traffic, butis expected to get saturated in the near future. IPv6 was designed to overcome this limitation.

98) What is mesh topology?

Mesh topology is a setup wherein each device is connected directly to every other device on the network. Consequently, it requires that each device have at least two network connections.

REFERENCES

[1]. Marc Greis, “Tutorial of network simulator "ns"”, https://www.isi.edu/nsnam/ns/tutorial/, accessed on 20/03/2018

[2]. Kevin Fall, Kannan Varadhan, "The ns Manual", The VINT Project, March 14, 2008

[3]. http://myns2prac.blogspot.in/2012/12/ns2-simulation-using-link-state-routing.html

[4]. Behrouz A. "Forouzan, Data Communications and Networking", McGraw-Hill publication

[5]. https://www.isi.edu/nsnam/ns/doc/node1.html

[6]. https://www.isi.edu/nsnam/ns/doc/node195.html

[7].