COMPUTER NETWORKS LABORATORY

B.E., VII Semester, Electronics & Communication Engineering

[As per Choice Based Credit System (CBCS) scheme]

Course objectives: This course will enable students to:

1. Choose suitable tools to model a network and understand the protocols at various OSI reference levels.
2. Design a suitable network and simulate using a Network simulator tool.
3. Simulate the networking concepts and protocols using C/C++ programming.
4. Model the networks for different configurations and analyze the results.

Laboratory Experiments

PART-A: Simulation experiments using NS2/ NS3/ OPNET/ NCTUNS/ NetSim/QualNet or any other equivalent tool

1. Implement a point to point network with four nodes and duplex links between them. Analyze the network performance by setting the queue size and varying the bandwidth.

2. Implement a four node point to point network with links n0-n2, n1-n2 and n2-n3. Apply TCP agent between n0-n3 and UDP between n1-n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP/UDP.

3. Implement Ethernet LAN using n (6-10) nodes. Compare the throughput by changing the error rate and data rate.

4. Implement Ethernet LAN using n nodes and assign multiple traffic to the nodes and obtain congestion window for different sources/ destinations.

5. Implement ESS with transmission nodes in Wireless LAN and obtain the performance parameters.

6. Implementation of Link state routing algorithm.

PART-B: Implement the following in C/C++

1. Write a program for a HLDC frame to perform the following i) Bit Stuffing ii) Character Stuffing.

2. Write a program for distance vector algorithm to find suitable path for transmission.

3. Implement Dijkstra's algorithm to compute the shortest routing path.

4. For the given data, use CRC-CCITT polynomial to obtain CRC code. Verify the program for the cases(a) Without error (b) With error

5. Implementation of Stop and Wait Protocol and Sliding Window Protocol

6. Write a program for congestion control using Leaky Bucket Algorithm.

Course Outcomes (COs)

CO	Description
CO1:	Choose suitable tools to model a network
CO2:	Using the network simulator for learning and practice of networking algorithms
CO3:	Illustrate the operations of network protocols and algorithm using C programming
CO4:	Simulate the network with different configurations to measure the performance parameters
CO5:	Implement the data link and routing protocol using C Programming

Introduction to Computer Networks Lab

The purpose of this is to acquaint the students with an overview of the Computer Networks from the perspective how the information is transferred from source to destination and different layers in networks. This course provides a basis for u. They can understand how the data transferred from source to destination. They can come to know that how the routing algorithms worked out in network layer understanding the networking techniques that can take place in computer. A computer network is made of two distinct subsets of components

Distributed applications are programs running on interconnected computers; a web server, a remote login server, an e-mail exchanger are examples. This is the visible part of what people call “the Internet”. In this lecture we will study the simplest aspects of distributed applications. More sophisticated aspects are the object of lectures called “Distributed Systems” and “Information Systems”. The network infrastructure is the collection of systems which are required for the interconnection of computers running the distributed applications. It is the main focus of this lecture. The network infrastructure problem has itself two aspects: Distance: interconnect remote systems that are too far apart for a direct cable connection Meshing: interconnect systems together; even in the case of systems close to each other, it is not possible in non-trivial cases to put cables from all systems to all systems (combinatorial explosion, cable salad management problem s etc.).

Recommended System/Software Requirements

Intel based desktop PC with minimum of 2.6GHZ or faster processor with at least 1 GB RAM and 40 GB free disk space and LAN connected.

Operating system: Linux (Redhat Linux)

Software : Programming on Linux using GNU Compiler Collection: gcc

NS-2 Simulator

NS2 is an open-source object oriented, discrete, event-driven network simulator written in C++ and Tool Command Language (Tcl) with Object Oriented extensions (OTcl). It implements network protocols for network simulations. NS-2 includes a tool for viewing the simulation results, called network animator (NAM). NAM is a Tool Command Language/Toolkit (Tcl/Tk) based animation tool for viewing network simulation results and real-world packet trace data.

The first step in using NAM is to produce the trace file. The trace file should contain topology information, for example, nodes, links, as well as packet traces. Usually, the trace file is generated by ns. During an ns simulation, the user can produce topology configurations, layout information, and packet traces using tracing events in ns. When the trace file is generated, it is ready to be animated by NAM. Upon start-up NAM will read the trace file, create topology, pop-up a window, do layout if necessary, and then pause at the time of the first packet in the trace file. Though its user interface, NAM provides control over many aspects of animation.

To model a network simulation using NS-2, it is necessary to write a Tcl script describing the topology (nodes, agents, applications, etc). NS2 provide users with an executable command "ns" which takes one input argument which is the name of a Tcl simulation scripting file. A simulation trace file is then created which can be used to plot graphs and/or to create animations. AWK is a scripting language tool used for manipulating data and generating reports. It searches one or more files to see if they contain lines that matches with the specified patterns and performs the associated actions.

The Trace file contains 12 columns:Event type, Event time, From Node, To Node, Packet Type, Packet Size, *Flags (indicated by --------), Flow ID, Source address, Destination address, Sequence ID, Packet ID*

To visualize the result, use a NAM. You can either start a NAM with the command 'nam <nam-file>' where '<nam-file>' is the name of a NAM trace file that was generated by ns or we can executeit directly out of the Tcl simulation script for the simulation which we want to visualize.

Study of NS-2 Simulator

NS2 plays a very important role as a simulation tool for implementing different protocol like the physical layer protocol (ETHERNET, DSL, MAC etc.), routing protocols (RIP, IGRP), Transport layer protocol (TCP, UDP) and Network protocols(IP) of different types of network. NS2 is widely used tool to simulate the behavior of wired and wireless networks.

NS2 supports for simulation of both wired and wireless network and is best suitable for the projects related to networking as it works on packet transmission scenarios i.e. how packets can be transmitted or received from source to destination which offers packet level inspection i.e. how it is dropped, percentage of packet dropped and why it is dropped using various traffic generators which generates traffic using different parameters.

In general, NS2 provides users with a way of specifying different network protocols and simulating their corresponding behaviors.Various traffic generators are present for generating traffic at source nodes in wired network simulation, for example, CBR, VBR, and Exponential etc. NAM stands for Network

AniMator window is a visualization tool used in NS2 to provide outputs on a Window screen which shows the network scenarios like at what time and at how much rate data packets starts dropping etc.

For detailed study of NS-2, please refer to the ns manual. The ns manual can be downloaded from

http://www.isi.edu/nsnam/ns/doc/ns-doc.pdf

For getting the NS-2 software and installing the package, please refer the ns manual.

Detailed notes on NS2, NAM and Xgraph are given at the end for better understanding. Read these documents before taking up the NS2 simulation programs in Part-A of Computer Networks Lab.

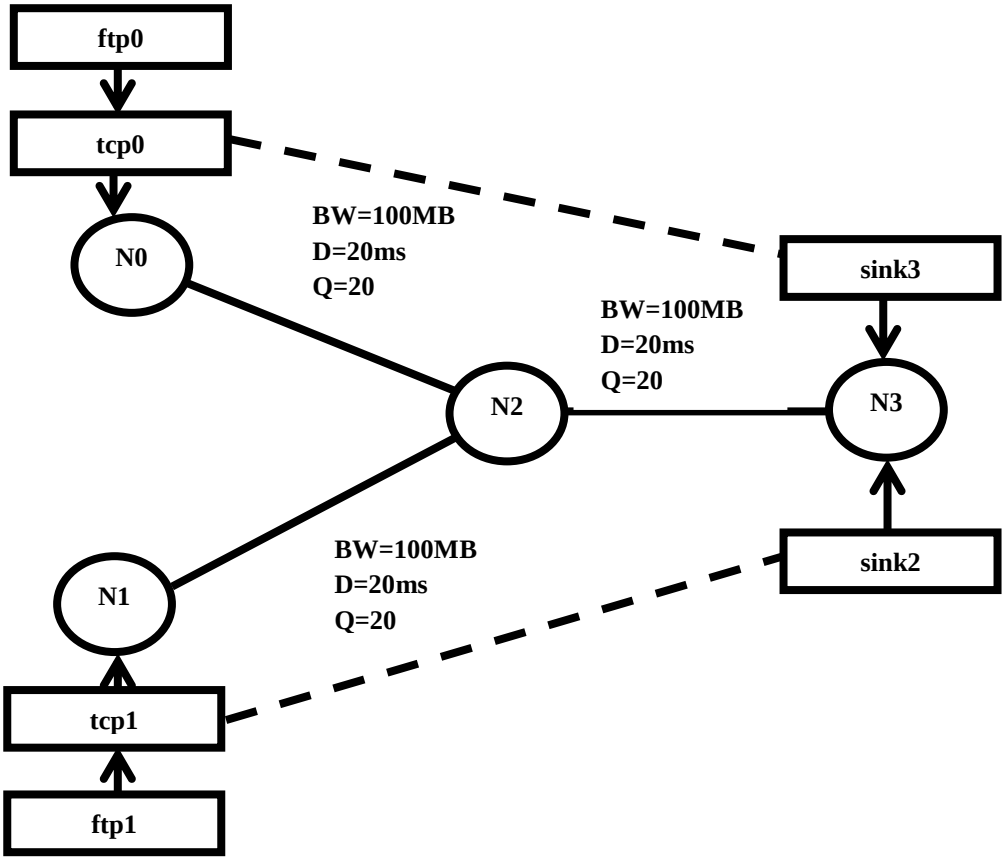
Steps to setup simulation

1. Initialize the simulator
2. Define files for output (tracing)
3. Setup the topology
4. Setup the agents
5. Setup traffic between the nodes
6. Start the simulation
7. Analyze the trace files to compute the parameters of interest.

Note: The NAM topology, Trace file and output plots (graphs) are shown at end of each part-Aexperiment as a reference.

Part-A

1. Implement a point to point network with four nodes and duplex links between them. Analyse the network performance by setting the queue size and varying the bandwidth.



- **N0, N1, N2, N3 → Nodes (N0 and N1 are Source Nodes, N2-Intermediate Node, N3-Destination Node)**
- **tcp0, tcp1 → Transmission Control Protocol (TCP)**
- **ftp0, ftp1 → File Transfer Protocol (FTP)**
- **sink0, sink1 → TCP Agent connected to 'N3' and it reacts to TCP Packets**
- **BW → Bandwidth of the Link**
- **D → Delay of the Link**
- **Q → Queue Size of the Link**

Note:

A small wired network topology consisting of four nodes, initially the bandwidth, delay and queue size of all three duplex links remains same, later the bandwidth is varied to **0.1, 0.01, 0.001** and **500MB** and queue size to **3, 5, 10, 20**. FTP is an application layer protocol, which generates packets and transmits it through transport layer (TCP Protocol). Control and data packets are routed from **N0-N2-N3** and **N1-N2-**

N3. First, the control packets are transmitted between source and destination node to check the connectivity. The above network topology can also be created using NSG2.1 Tool.

```
#=================================
#     Simulation parameters setup
#=================================
set val(stop)   50.0                ;# time of simulation end
#=================================
#       Initialization
#=================================
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
set tracefile [open lab1.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open lab1.nam w]
$ns namtrace-all $namfile
#=================================
#       Nodes Definition
#=================================
#Create 4 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#=================================
#       Links Definition
#=================================
#Create links between nodes
$ns duplex-link $n0 $n2 500.0Mb 20ms DropTail
$ns queue-limit $n0 $n2 20
$ns duplex-link $n1 $n2 500.0Mb 20ms DropTail
$ns queue-limit $n1 $n2 20
$ns duplex-link $n2 $n3 500.0Mb 20ms DropTail
$ns queue-limit $n2 $n3 20
#Above Highlighted Bandwidth and Queue Size will be changed
#Give node position (for NAM)
$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
#=================================
#       Agents Definition
#=================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink2 [new Agent/TCPSink]
```

```
$ns attach-agent $n3 $sink2
$ns connect $tcp0 $sink2
$tcp0 set packetSize_ 1500
#Setup a TCP connection
set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
$ns connect $tcp1 $sink3
$tcp1 set packetSize_ 1500
#=================================
#       Applications Definition
#=================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 23.0 "$ftp0 stop"

#Setup a FTP Application over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 24.0 "$ftp1 start"
$ns at 48.0 "$ftp1 stop"
#=================================
#       Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
   global ns tracefile namfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec nam lab1.nam &
   exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Awk Script(lab1.awk)

```
BEGIN{
a=0
b=0
}
{
if($1=="r"&&$3=="2"&&$4=="3"&&$5=="tcp"&&$6=="1540")
{
```

```
a++;
}
if($1=="d"&&$3=="2"&&$4=="3"&&$5=="tcp"&&$6=="1540")
{
b++;
}
}
END{
printf("\n total number of data packets received at Node 3: %d\n", a++);
printf("\n total number of packets dropped between Node 2 and Node 3: %d\n", b++);
}
```

Steps for Execution

Step 1: Type the above program in text editor and save it with the filename and extension ".tcl"

For Example, lab1.tcl

- **tcl- Tool Command Language**

Step 2: Run the program in terminal window

[root@localhost ~]# ns lab1.tcl

- **ns-Network Simulator**

Step 3: Once the simulation is completed, run awk file to observe the output

[root@localhost ~]# awk -f lab1.awk lab1.tr

- **tr- trace file**

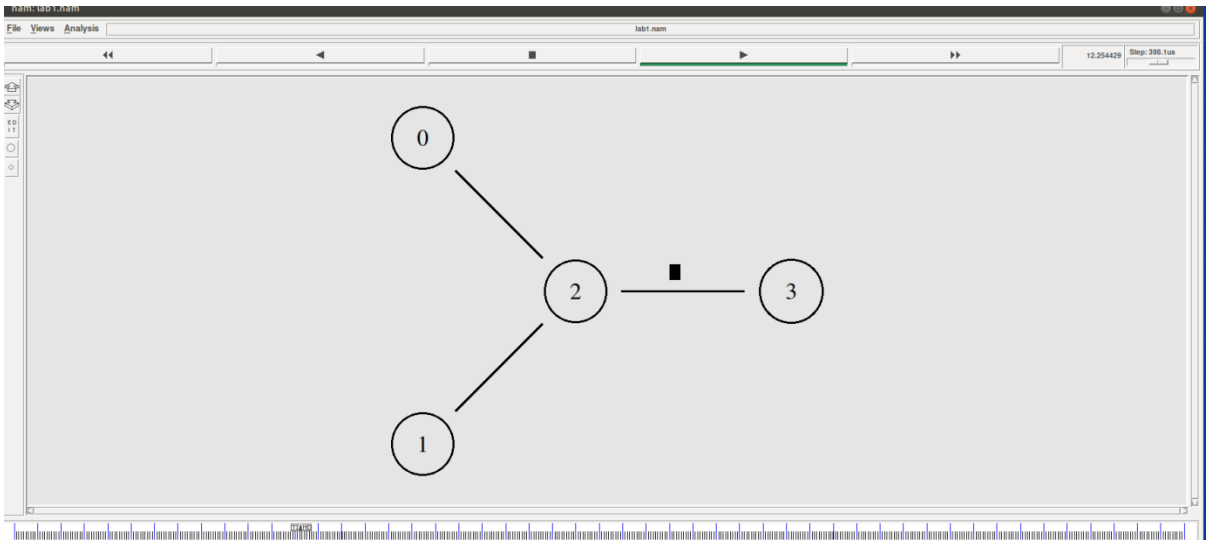
Simulation Result

Serial No.	Bandwidth (Mb) between N0-N1, N1-N2, N2-N3	Queue Size between N0-N1, N1-N2, N2-N3	Packet Received at Node 3	Packet Dropped at Node 3
1	500/500/500	20/20/20	11360	0
2	500/500/0.1	10/10/10	385	24
3	500/500/0.01	5/5/5	39	4
4	500/500/0.001	3/3/3	3	3

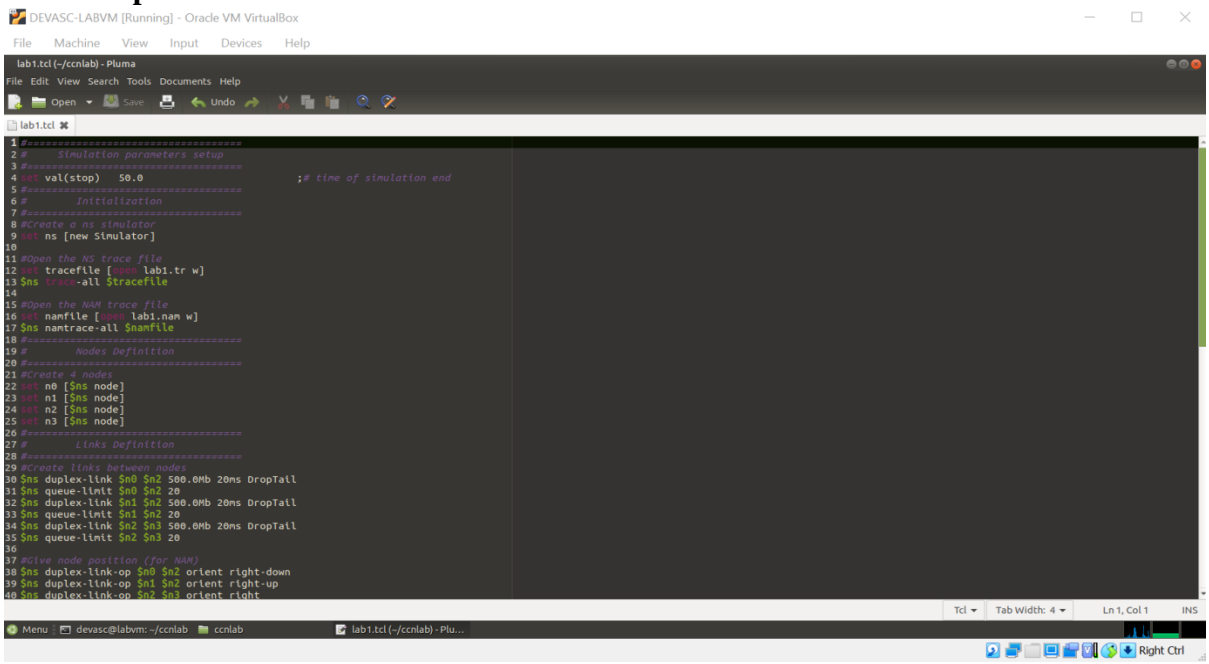
Terminal

```
devasc@labvm: ~/ccnlab
File  Edit  View  Search  Terminal  Help
devasc@labvm:~$ ls
ccnlab    Documents  labs       out.nam     snap
Desktop   Downloads  myfile.sh  simple.tcl
devasc@labvm:~$ cd ccnlab/
devasc@labvm:~/ccnlab$ ns lab1.tcl
devasc@labvm:~/ccnlab$ Cannot connect to existing nam instance. Starting a new one...

devasc@labvm:~/ccnlab$ awk -f lab1.awk lab1.tr

 total number of data packets received at Node 3: 11360

 total number of packets dropped between Node 2 and Node 3: 0
devasc@labvm:~/ccnlab$
```

NAM File



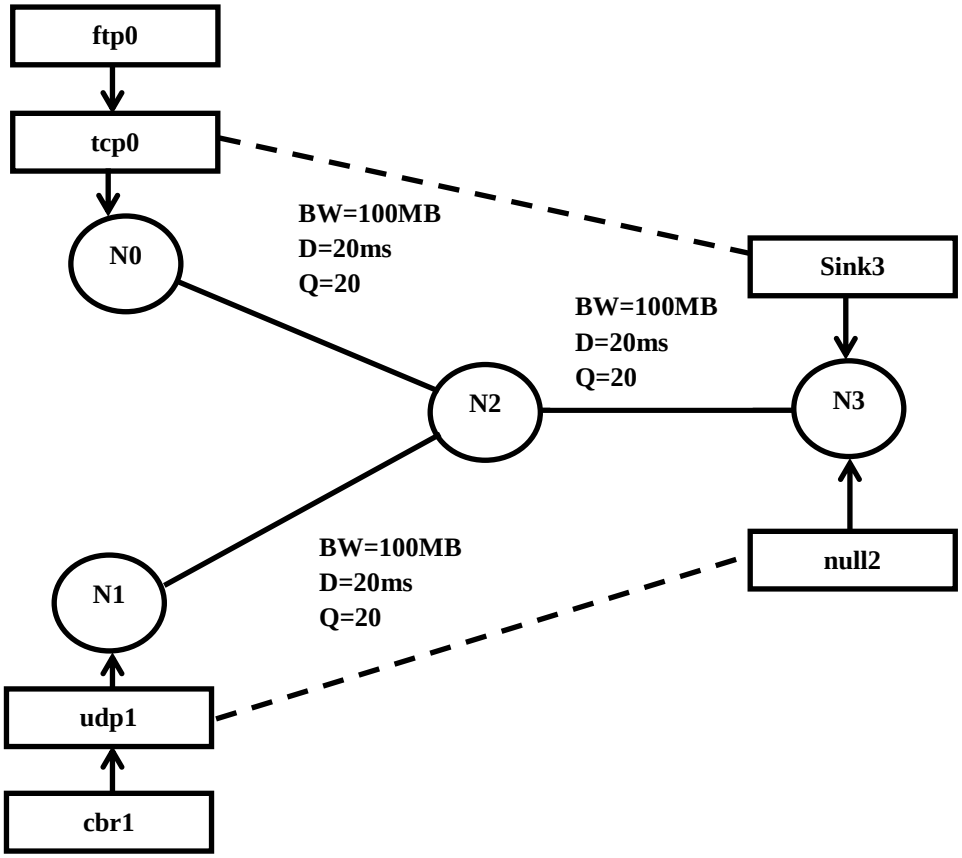
TCL Script File



Assignment Problems

1. **Implement a point to point network with six nodes and duplex links between them. Analyze the network performance by setting the queue size and varying start and stop time.**
2. **Implement a point to point network with five nodes and duplex links between them. Analyze the network performance by varying link propagation delay.**
3. **Implement a point to point network with seven nodes and duplex links between them. Analyze the network performance by varying link packet size.**
4. **Implement a point to point network with ten nodes and duplex links between them. Calculate the packet delivery ratio.**

2. Implement a four node point to point network with links N0- N2, N1- N2 and N2-N3. Apply TCP agent between N0-N3 and UDP between N1-N3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP/UDP.



- udp-User Datagram Protocol
- cbr-Constant Bit Rate,
- null-Agent

This network consists of 4 nodes (N0, N1, N2, N3) as shown in above figure. The duplex links between n0 and n2, n1 and n2, and n2 and n3 have 100 Mbps of bandwidth and 20 ms of delay. Each node uses a DropTail queue with queue size is 20. A "tcp" agent is attached to n0, and a connection is established to a tcp "sink" agent attached to n3. The packet size of a "tcp" agent will be of 1540Byte and udp agent will be 1000Byte. A tcp "sink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets. A "udp" agent that is attached to n1 is connected to a "null" agent attached to n3. A "null" agent just frees the packets received. A "ftp" and a "cbr" traffic generator are attached to "tcp" and "udp" agents respectively. The "cbr" is set to start at 1.0 sec and stop at 23.0 sec, and "ftp" is set to start at 24.0 sec and stop at 48.0 sec.

```
#=================================
#     Simulation parameters setup
```

```
#==================================
set val(stop)   50.0                ;# time of simulation end
#==================================
#       Initialization
#==================================
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
set tracefile [open lab2.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open lab2.nam w]
$ns namtrace-all $namfile
#==================================
#       Nodes Definition
#==================================
#Create 4 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#==================================
#       Links Definition
#==================================
#Createlinks between nodes
$ns duplex-link $n0 $n2 500.0Mb 20ms DropTail
$ns queue-limit $n0 $n2 20
$ns duplex-link $n2 $n3 500.0Mb 20ms DropTail
$ns queue-limit $n2 $n3 20
$ns duplex-link $n1 $n2 500.0Mb 20ms DropTail
$ns queue-limit $n1 $n2 20

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n1 $n2 orient right-up
#==================================
#       Agents Definition
#==================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
```

```
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
$ns connect $tcp0 $sink3
$tcp0 set packetSize_ 1000              # Change the Packet Size
$tcp0 set interval_ 0.1

#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set null2 [new Agent/Null]
$ns attach-agent $n3 $null2
$ns connect $udp1 $null2
$udp1 set packetSize_ 1500
$udp1 set interval_ 0.1
#=================================
#       Applications Definition
#=================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 20.0 "$ftp0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000    # Change the Packet Size
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$ns at 24.0 "$cbr1 start"
$ns at 48.0 "$cbr1 stop"
#=================================
#       Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam lab2.nam &
    exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
```

```
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Awk Script(lab2.awk)

```
BEGIN{
a=0
b=0
}
{
if($1=="r"&&$4=="2"&&$5=="tcp"&&$6=="1040")
{
a++;
}
if($1=="r"&&$4=="2"&&$5=="cbr"&&$6=="1000")
{
b++;
}

}
END{
printf("\n total number of TCP data packets sent between Node 0 and Node 2: %d\n", a++);
printf("\n total number of UDP data packets sent between Node 1 and Node 2: %d\n", b++);
}
```

Steps for Execution

Step 1: Type the above program in text editor and save it with the filename and extension ".tcl"

For Example, lab2.tcl

- **tcl- Tool Command Language**

Step 2: Run the program in terminal window

[root@localhost ~]# ns lab2.tcl

- **ns-Network Simulator**

Step 3: Once the simulation is completed, run awk file to observe the output

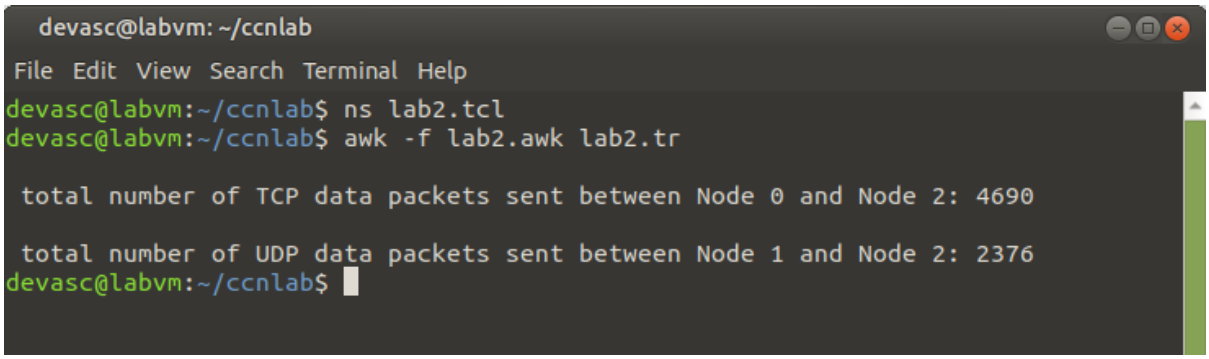
[root@localhost ~]# awk -f lab2.awk lab2.tr

- **tr- trace file**
-

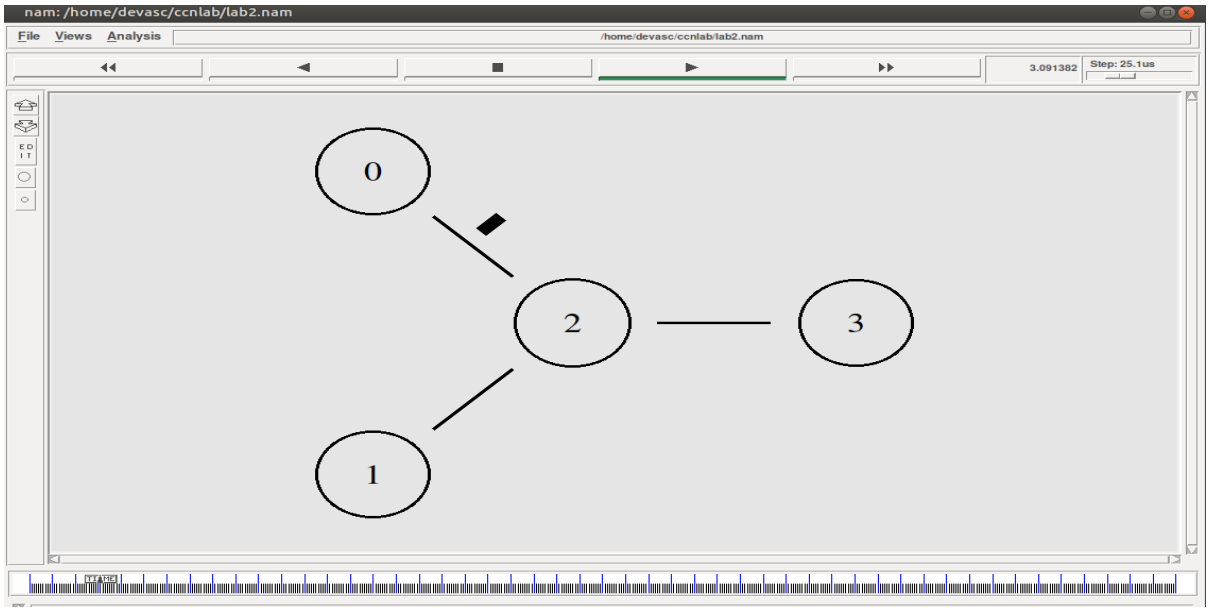
Simulation Result

Serial No.	Simulation Time (Sec)	Packet Size (Byte)	TCP Packet Received at Node 3	UDP Packet Received at Node 3
1	Start Time:1 Stop Time:9	TCP:1500 UDP:1500	1930	667
2	Start Time:1 Stop Time:20	TCP:1000 UDP:1000	4690	2376
3	Start Time:1 Stop Time:25	TCP:1300 UDP:1300	5930	2308

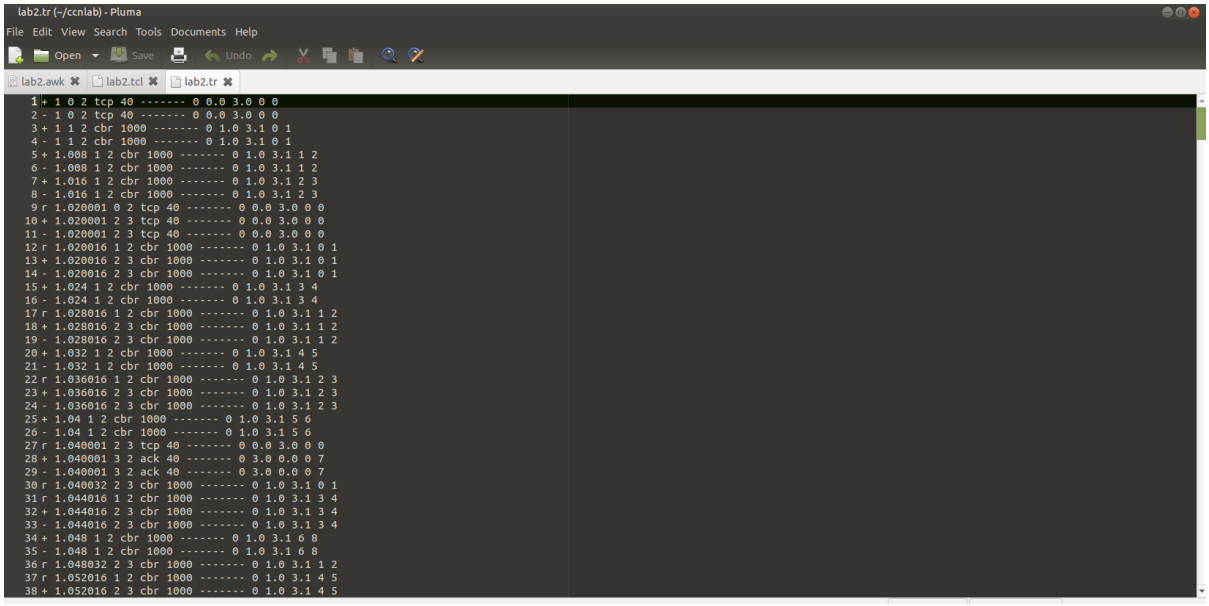
Terminal



NAM File



Trace File (lab2.tr)



Assignment Problems

1. Implement a five node point to point network with links N0- N2, N1- N2, N3-N2 and N2-N3. Apply TCP agent between N0-N3, N2-N3 and UDP between N1-N3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP/UDP.

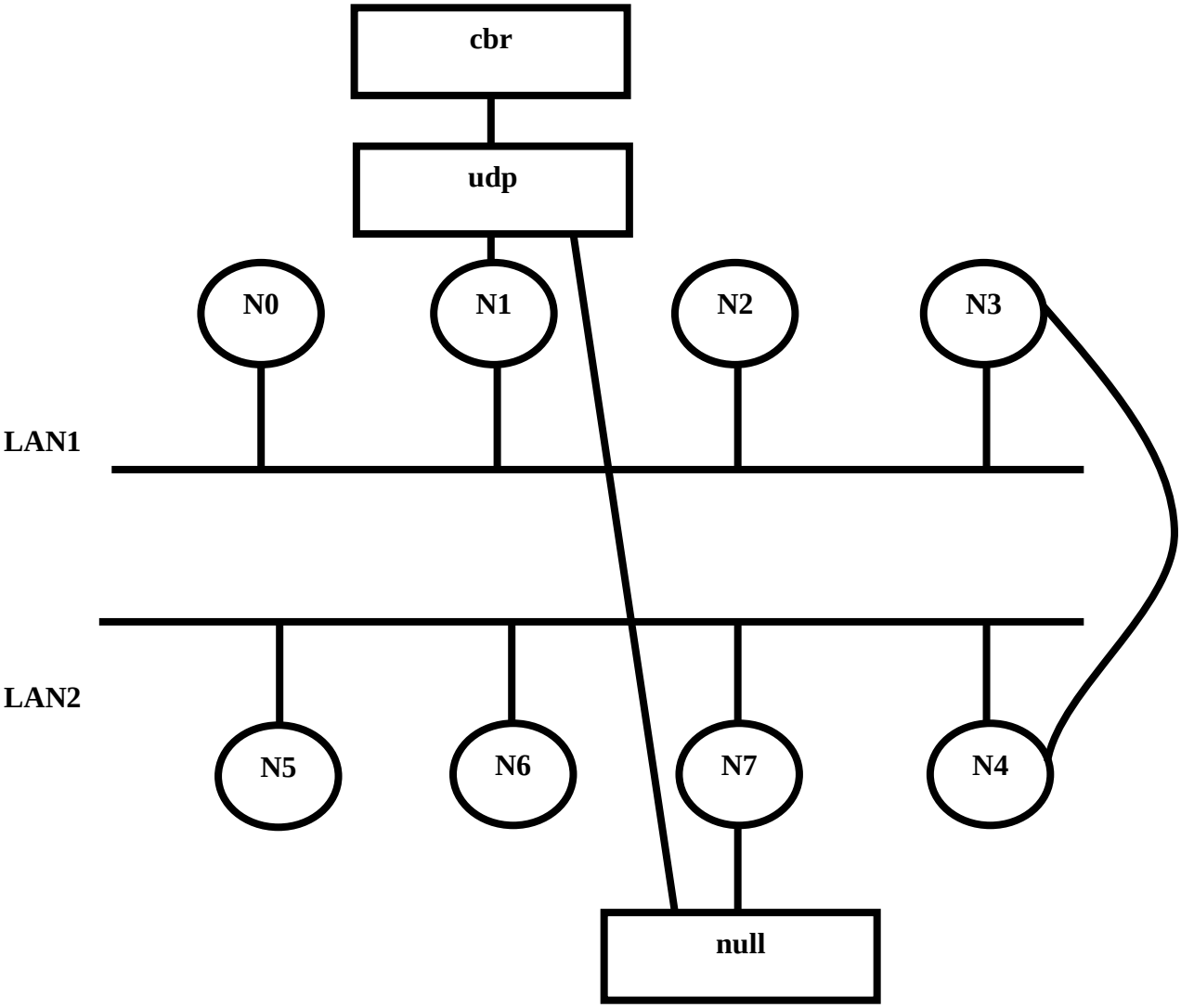
2. Implement a five node point to point network with links N0- N2, N1- N2, N3-N2 and N2-N3. Apply TCP agent between N0-N3, N2-N3 and N1-N3. Apply relevant applications over TCP agents changing the parameter and determine the number of packets sent by TCP.

3. Implement a five node point to point network with links N0- N2, N1- N2, N3-N2 and N2-N3. Apply UDP agent between N0-N3, N2-N3 and N1-N3. Apply relevant applications over UDP agents changing the parameter and determine the number of packets sent by UDP.

4. Implement a four node point to point network with links N0- N2, N1- N2 and N2-N3. Apply TCP agent between N0-N3 and N1-N3. Apply relevant applications over TCP agents changing the parameter and determine the number of packets sent by TCP.

5. Implement a four node point to point network with links N0- N2, N1- N2 and N2-N3. Apply UDP agent between N0-N3 and N1-N3. Apply relevant applications over UDP agents changing the parameter and determine the number of packets sent by UDP.

3. Implement Ethernet LAN using N (6-10) nodes. Compare the throughput by changing the error rate and data rate.



This network consists of 8 nodes (N0, N1, N2, N3, N4, N5, N6, N7) as shown in above figure. Four nodes (N0, N1, N2, N3) are attached to LAN1 and other four nodes are attached to LAN2, which is have 100 Mbps of bandwidth and 300 ms of delay and each node uses a DropTail queue. The agent "udp" is attached to N1, and a connection is established to a "null" agent attached to N7 and the packet size of udp agent will be 1000Byte. The packets are transmitted from node **N1 to LAN1-N3-N4-LAN2-N7** and an error model is introduced between the path of LAN1 and LAN2. The function of error model is to drop the packets that are transmitted to it and higher the error rate more the packets are dropped. Throughput is the performance parameter of the network which depends upon packet received, packet size and simulation time.

```
set ns [new Simulator]
set tf [open lab3.tr w]
$ns trace-all $tf
```

```
set nf [open lab3.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]

$ns make-lan "$n0 $n1 $n2 $n3" 100Mb 300ms LL Queue/DropTail Mac/802_3
$ns make-lan "$n4 $n5 $n6 $n7" 100Mb 300ms LL Queue/DropTail Mac/802_3

$ns duplex-link $n3 $n4 100Mb 300ms DropTail

# set error rate. Here ErrorModel is a class and it is single word and space should not be given between Error and Model
# lossmodel is a command and it is single word. Space should not be given between loss and model

set err [new ErrorModel]
$ns lossmodel $err $n3 $n4
$err set rate_ 0.1                    #Change the Error Rate 0.1, 0.3, 0.5,

# error rate should be changed for each output like 0.1,0.3,0.5…. */
set udp [new Agent/UDP]
$ns attach-agent  $n1 $udp
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set fid_ 0
$cbr set packetSize_ 1500
$cbr set interval_ 0.001              #Change the Data Rate 0.001, 0.01, 0.1,
set null [new Agent/Null]
$ns attach-agent $n7 $null

$ns connect $udp $null

proc finish { } {
global ns nf tf
$ns flush-trace
close $nf
close $tf
```

```
exec nam lab3.nam &
exit 0
}

$ns at 0.1 "$cbr start"
$ns at 3.0 "finish"
$ns run
```

Awk Script (lab3.awk)

```
BEGIN{
a=0
}
{
if($1=="r"&&$4=="7"&&$5=="cbr"&&$6=="1000")
{
a++;
}
}
END{
printf("\n total number of  data packets at Node 7: %d\n", a++);
}
```

Steps for Execution

Step 1: Type the above program in text editor and save it with the filename and extension ".tcl"

For Example, lab3.tcl

- **tcl- Tool Command Language**

Step 2: Run the program in terminal window

[root@localhost ~]# ns lab3.tcl

- **ns-Network Simulator**

Step 3: Once the simulation is completed, run awk file to observe the output

[root@localhost ~]# awk -f lab3.awk lab3.tr

- **tr- trace file**

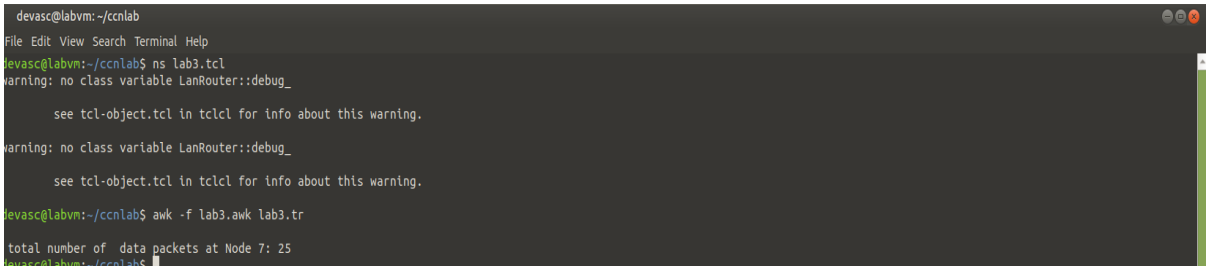
Simulation Result

Serial No.	Error Rate	Data Rate	Received at Node 7	Throughput (kbps)
1	0.1	0.001	68	22.67
2	0.3	0.01	32	10.67
3	0.5	0.1	25	8.34

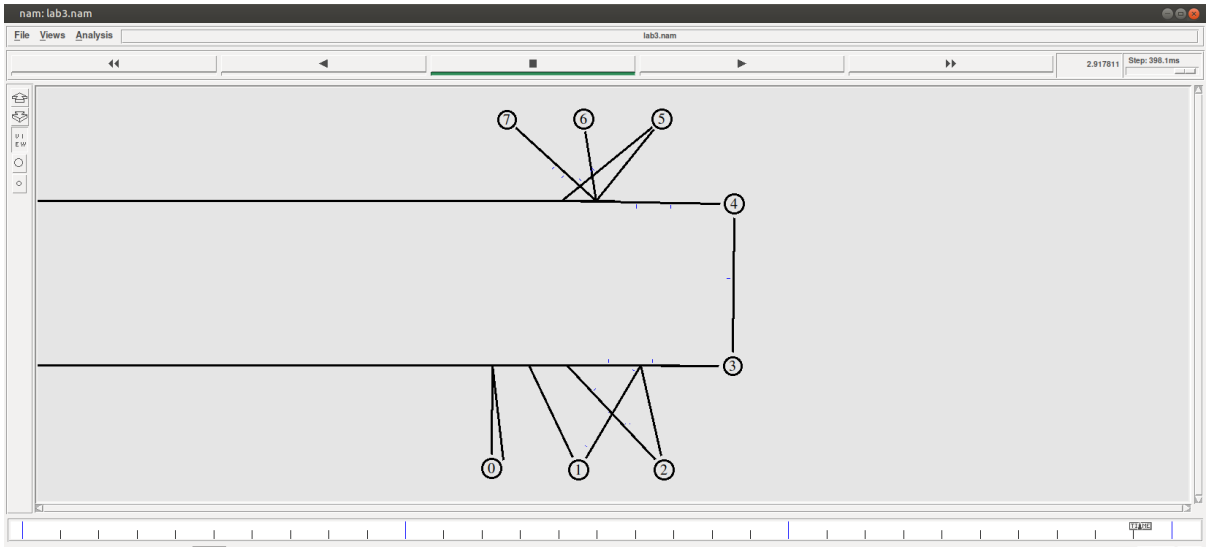
Note:

Packet Size: 1000Bytes
Simulation End Time: 3 Sec
Throughput=Packet received*Packet Size/Simulation End Time

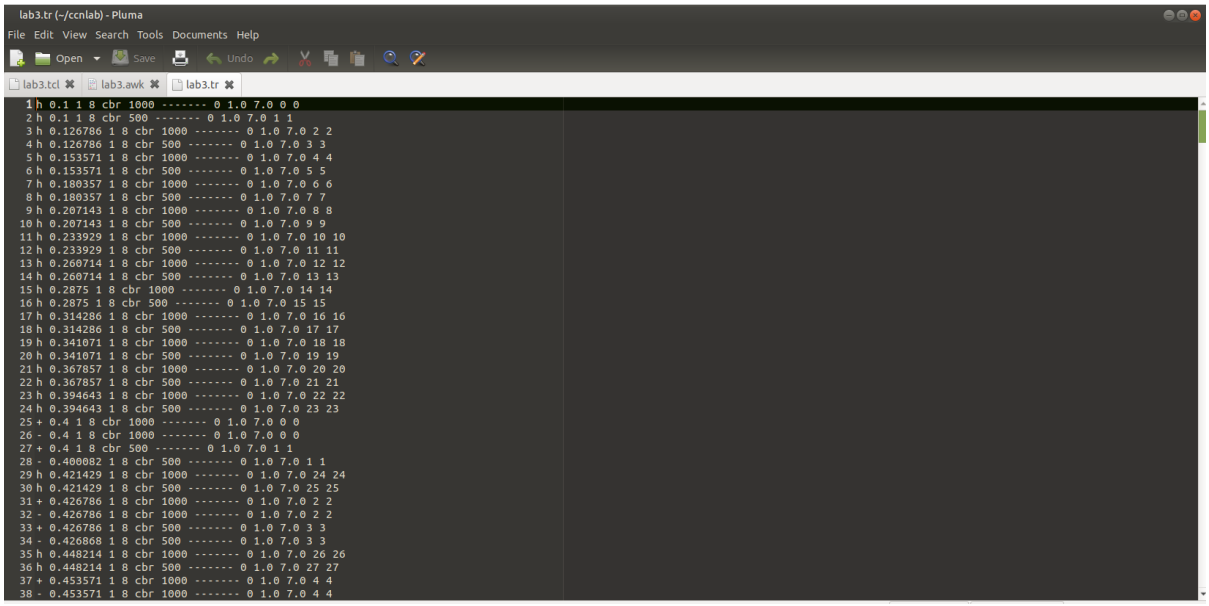
Terminal



NAM File



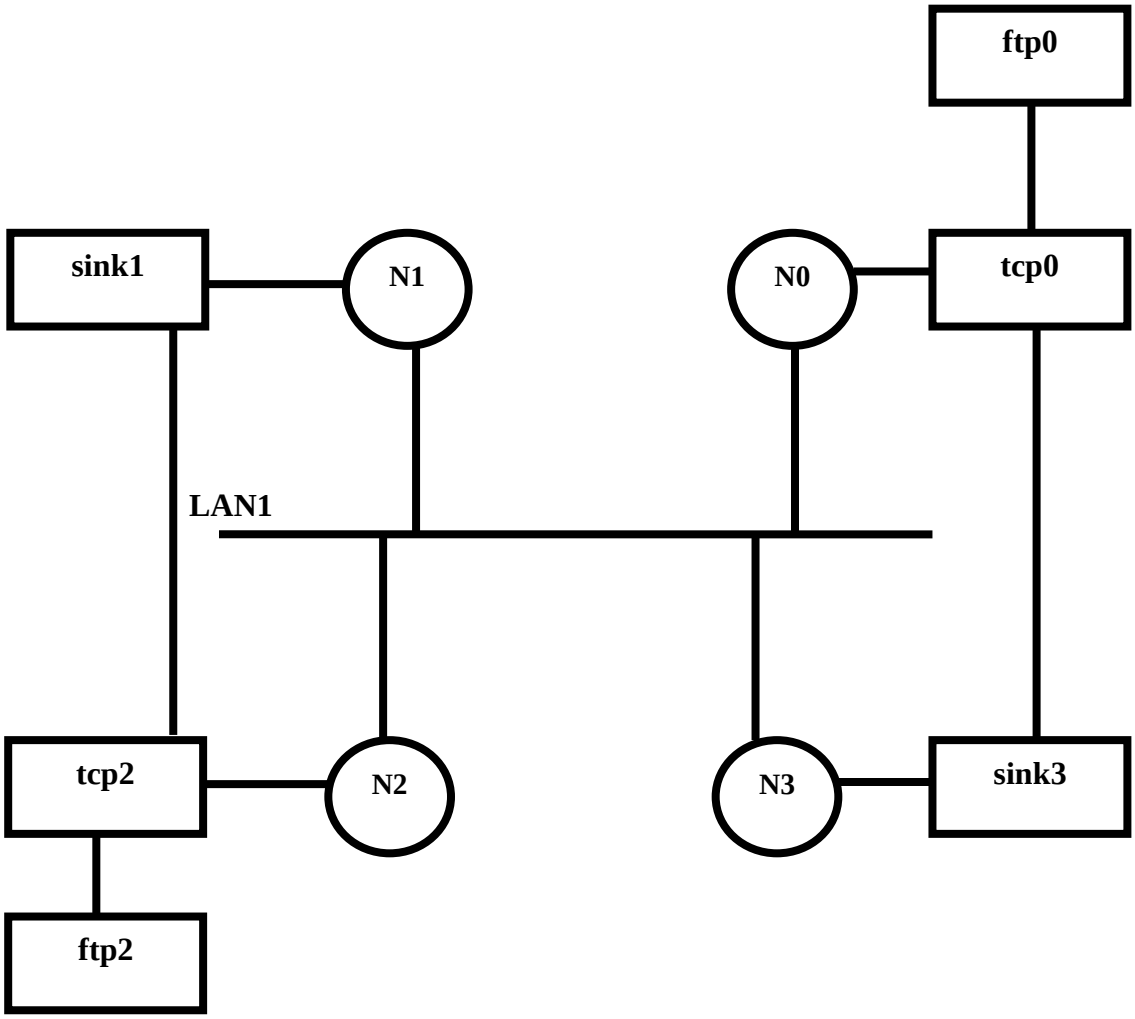
Trace File



Assignment Problems

1. **Implement Ethernet LAN using N (6-10) nodes. Compare the packet loss by changing the error rate and data rate.**
2. **Implement Ethernet LAN using N (6-10) nodes. Compare the packet delivery ratio by changing the error rate and data rate.**
3. **Implement Ethernet LAN using N (6-10) nodes. Compare the packet sent by changing the error rate and data rate.**
4. **Implement Ethernet LAN using N (6-10) nodes. Compare the packet received by changing the error rate and data rate.**
5. **Implement Ethernet LAN using N (6-10) nodes. Compare the throughput by changing the bandwidth.**

4. Implement Ethernet LAN using N nodes and assign multiple traffic to the nodes and obtain congestion window for different sources/ destinations



This network consists of 4 nodes (N0, N1, N2, N3) connected to single LAN1 and, node N0 and N2 are connected to traffic generator (ftp) act as source. Node N1 and N3 are connected to TCP agent sink, works as destination. Source node N2 and N0 sends the packets to N1 and N3 through common channel LAN1. After the execution of this program, we will get two output files such as file1.tr and file2.tr; these are attached to node N0 and N2. File1.tr and file2.tr consists of time period, congestion window size, source and destination address and port number. Congestion window size keep on increasing continuously till successful transmission and it resets window size to one, when source node start getting negative acknowledge packets.

```
set ns [new Simulator]
set tf [open lab4.tr w]
$ns trace-all $tf
set nf [open lab4.nam w]
$ns namtrace-all $nf
set n0 [$ns node]
set n1 [$ns node]
```

```
set n2 [$ns node]
set n3 [$ns node]
$ns make-lan "$n0 $n1 $n2 $n3" 10mb 10ms LL Queue/DropTail Mac/802_3
set tcp0 [new Agent/TCP/Reno]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
$ns connect $tcp0 $sink3
set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
set sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $sink1
$ns connect $tcp2 $sink1
######To trace the congestion window##########
set file1 [open file1.tr w]
$tcp0 attach $file1
$tcp0 trace cwnd_
set file2 [open file2.tr w]
$tcp2 attach $file2
$tcp2 trace cwnd_
proc finish { } {
global nf tf ns
$ns flush-trace
exec nam lab4.nam &
close $nf
close $tf
exit 0
}
$ns at 0.1 "$ftp0 start"
$ns at 4.5 "$ftp0 stop"
$ns at 1.5 "$ftp2 start"
$ns at 4 "$ftp2 stop"
$ns at 5.0 "finish"
$ns run
```

Steps for Execution

Step 1: Type the above program in text editor and save it with the filename and extension ".tcl"

For Example, lab4.tcl

- **tcl- Tool Command Language**

Step 2: Run the program in terminal window

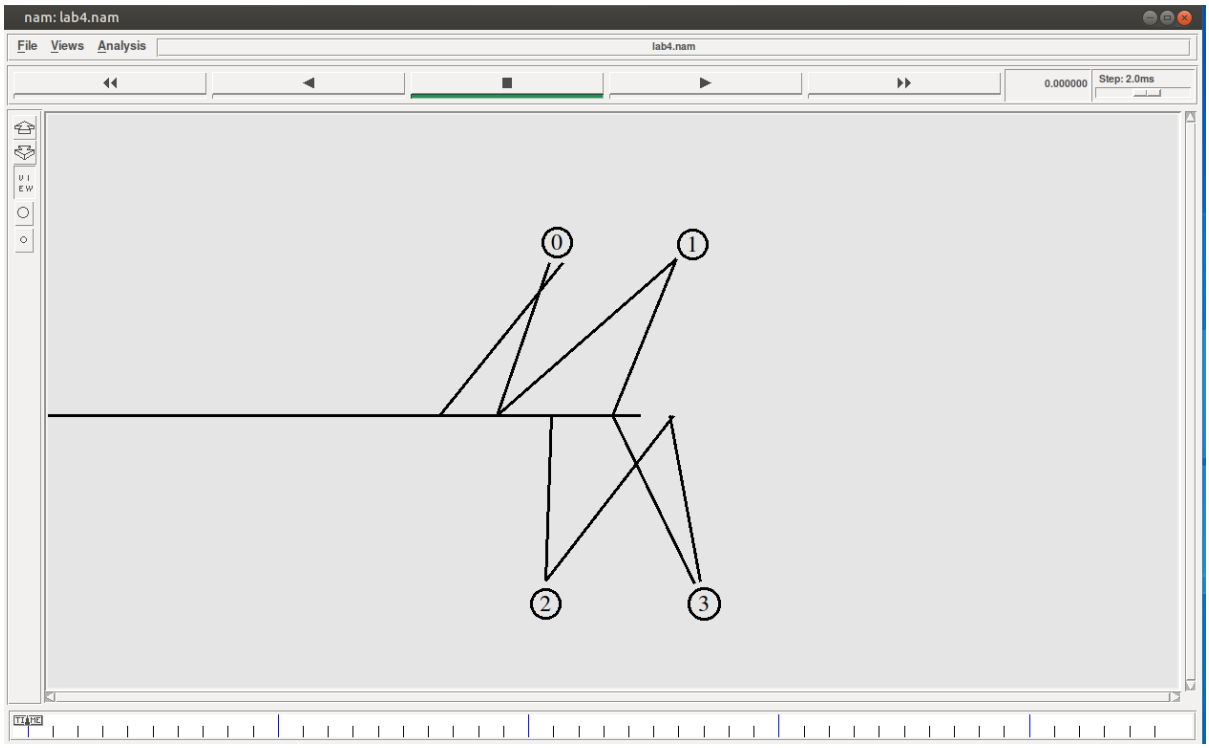
[root@localhost ~]# ns lab4.tcl

- **ns-Network Simulator**

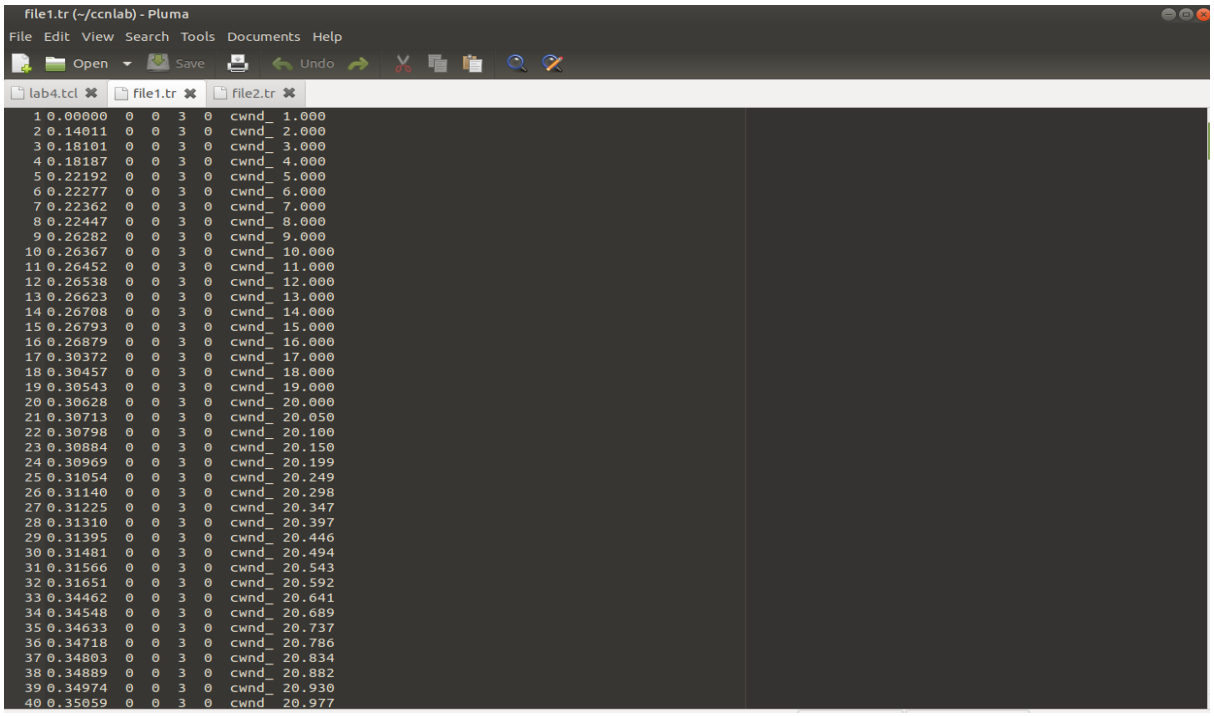
Step 3: Once the simulation is completed, run awk file to observe the output

[root@localhost ~]# awk -f lab4.awk lab4.tr

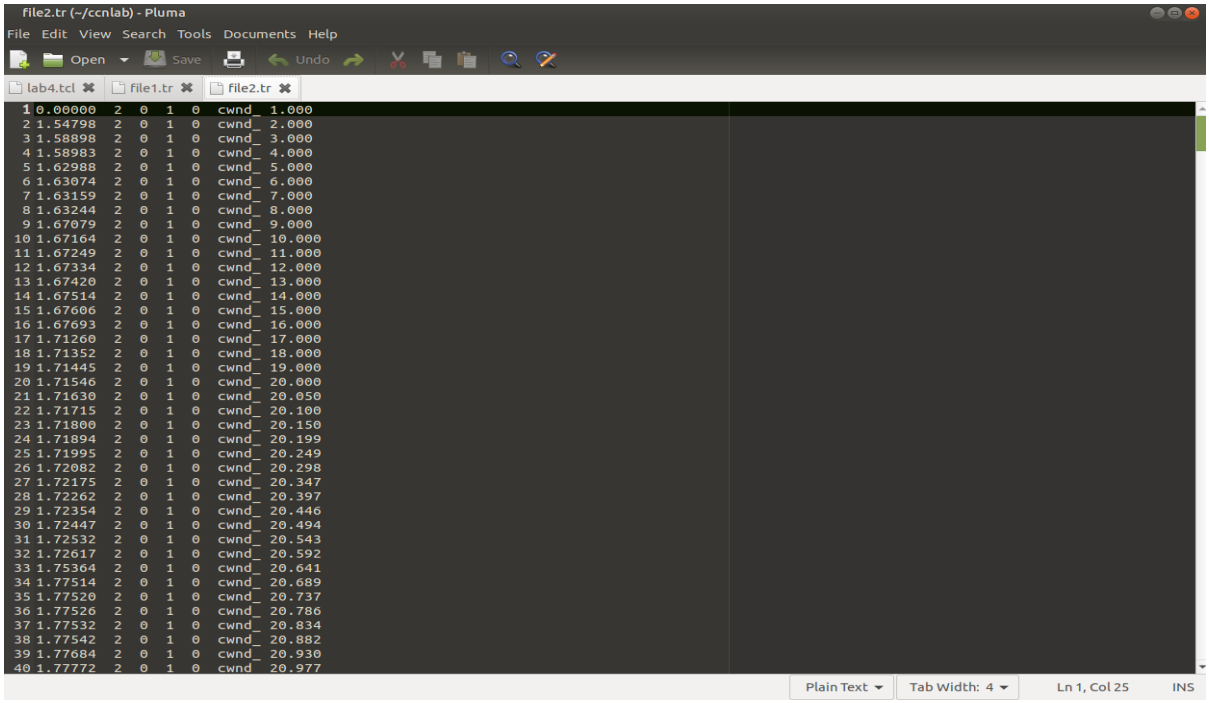
Network Animator



Trace File (file1.tr)

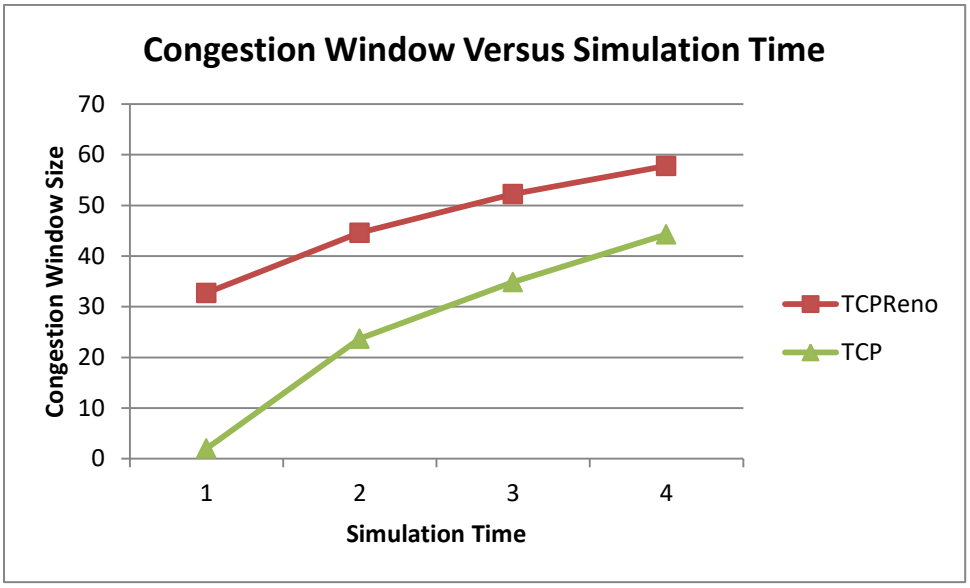


Trace File (file2.tr)



Time	TCPReno	TCP
1	32.72	2
2	44.59	23.70
3	52.22	34.85
4	57.78	44.26

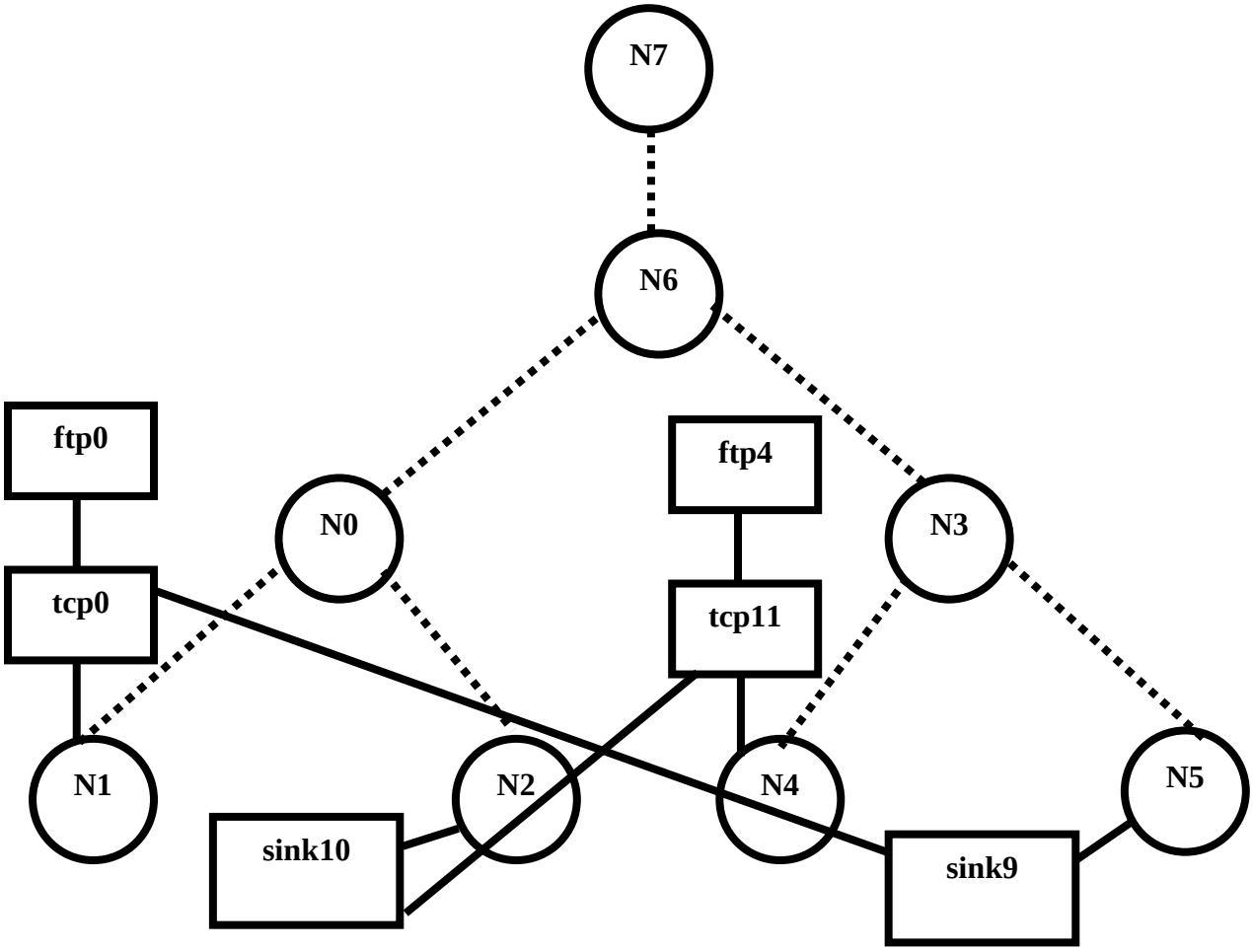
Table 1: Above values are taken from file1.tr and file2.tr



Assignment Problem

1. **Implement Ethernet LAN using N nodes and assign three traffic to the nodes and obtain congestion window for different sources/ destinations**
2. **Implement two Ethernet LAN using 4 nodes and assign multiple traffic to the nodes and obtain congestion window for different sources/ destinations**
3. **Implement Ethernet LAN using 6 nodes and assign multiple traffic to the nodes and obtain congestion window for different sources/ destinations**

4. **Implement ESS with transmission nodes in Wireless LAN and obtain then Performance Parameters.**



The above network consists of 8 nodes, among these nodes N1 and N4 acts as source and, N2 and N5 as destination. Node N0 and N3 performs as access point 1 and 2, whereas N6 and N7 behaves as router and server/gateway. This network has 2 BSS group (Basic Service Set) i.e N0, N1, N2 and N3, N4, N5. Group of more than one BSS is called as ESS (Extended Service Set).

Node N1 sends tcp packets to N5 through **N1-N0-N6-N3-N5** similarly N4 sends tcp packets to N2 through the path **N4-N3-N6-N0-N2**.

```
#================================
#     Simulation parameters setup
#================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
```

```
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     8                          ;# number of mobilenodes
set val(rp)     AODV                       ;# routing protocol
set val(x)      2483                       ;# X dimension of topography
set val(y)      2506                       ;# Y dimension of topography
set val(stop)   6.0                        ;# time of simulation end


#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open lab5.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open lab5.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#===================================
#     Mobile node parameter setup
#===================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
```

```
            -routerTrace   ON \
            -macTrace      ON \
            -movementTrace ON
#================================
#       Nodes Definition
#================================
#Create 8 nodes
set n0 [$ns node]
$n0 set X_ 1752
$n0 set Y_ 1142
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 1590
$n1 set Y_ 962
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 1862
$n2 set Y_ 950
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 2202
$n3 set Y_ 1171
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 2135
$n4 set Y_ 982
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 2383
$n5 set Y_ 1029
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 1975
$n6 set Y_ 1202
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 1971
```

```
$n7 set Y_ 1406
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20

$ns at 0.0 "$n0 label AP1"
$ns at 0.0 "$n1 label Source1"
$ns at 0.0 "$n2 label Destination2"
$ns at 0.0 "$n3 label AP2"
$ns at 0.0 "$n4 label Source2"
$ns at 0.0 "$n5 label Destination1"
$ns at 0.0 "$n6 label Router"
$ns at 0.0 "$n7 label Server/Gateway"
#=================================
#       Agents Definition
#=================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n1 $tcp0
set sink9 [new Agent/TCPSink]
$ns attach-agent $n5 $sink9
$ns connect $tcp0 $sink9
$tcp0 set packetSize_ 1500

#Setup a TCP connection
set tcp11 [new Agent/TCP]
$ns attach-agent $n4 $tcp11
set sink10 [new Agent/TCPSink]
$ns attach-agent $n2 $sink10
$ns connect $tcp11 $sink10
$tcp11 set packetSize_ 1500
#=================================
#       Applications Definition
#=================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 3.0 "$ftp0 stop"

#Setup a FTP Application over TCP connection
set ftp4 [new Application/FTP]
$ftp4 attach-agent $tcp11
$ns at 3.5 "$ftp4 start"
$ns at 5.0 "$ftp4 stop"
```

```
#=================================
#       Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
   global ns tracefile namfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec nam lab5.nam &
   exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
   $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Awk Script (lab5.awk)

```
BEGIN{
source1=0
source2=0
source3=0
source4=0
}
{
if($1=="s"&&$3=="_1_"&&$4=="AGT"&&$7=="tcp"&&$8=="1540")
{
source1++;
}
if($1=="s"&&$3=="_4_"&&$4=="AGT"&&$7=="tcp"&&$8=="1540")
{
source2++;
}
if($1=="r"&&$3=="_2_"&&$4=="AGT"&&$7=="tcp"&&$8=="1540")
{
source3++;
}
if($1=="r"&&$3=="_5_"&&$4=="AGT"&&$7=="tcp"&&$8=="1540")
{
```

```
source4++;
}
}
END{
printf("\n total number of data packets Sent by Node 1: %d\n", source1++);
printf("\n total number of data packets Sent by Node 4: %d\n", source2++);
printf("\n total number of data packets Received by Node 2: %d\n", source3++);
printf("\n total number of data packets Received by Node 5: %d\n", source4++);
}
```

Steps for Execution

Step 1: Type the above program in text editor and save it with the filename and extension ".tcl"

For Example, lab5.tcl

- **tcl- Tool Command Language**

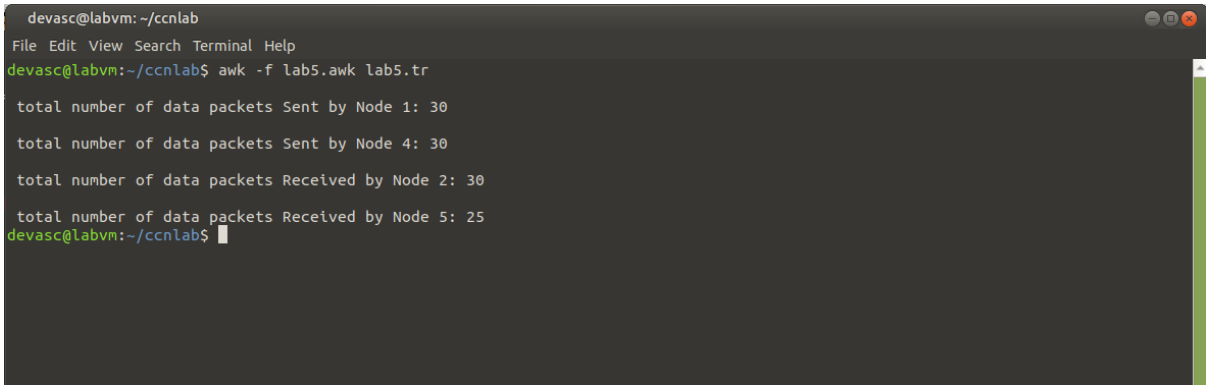
Step 2: Run the program in terminal window

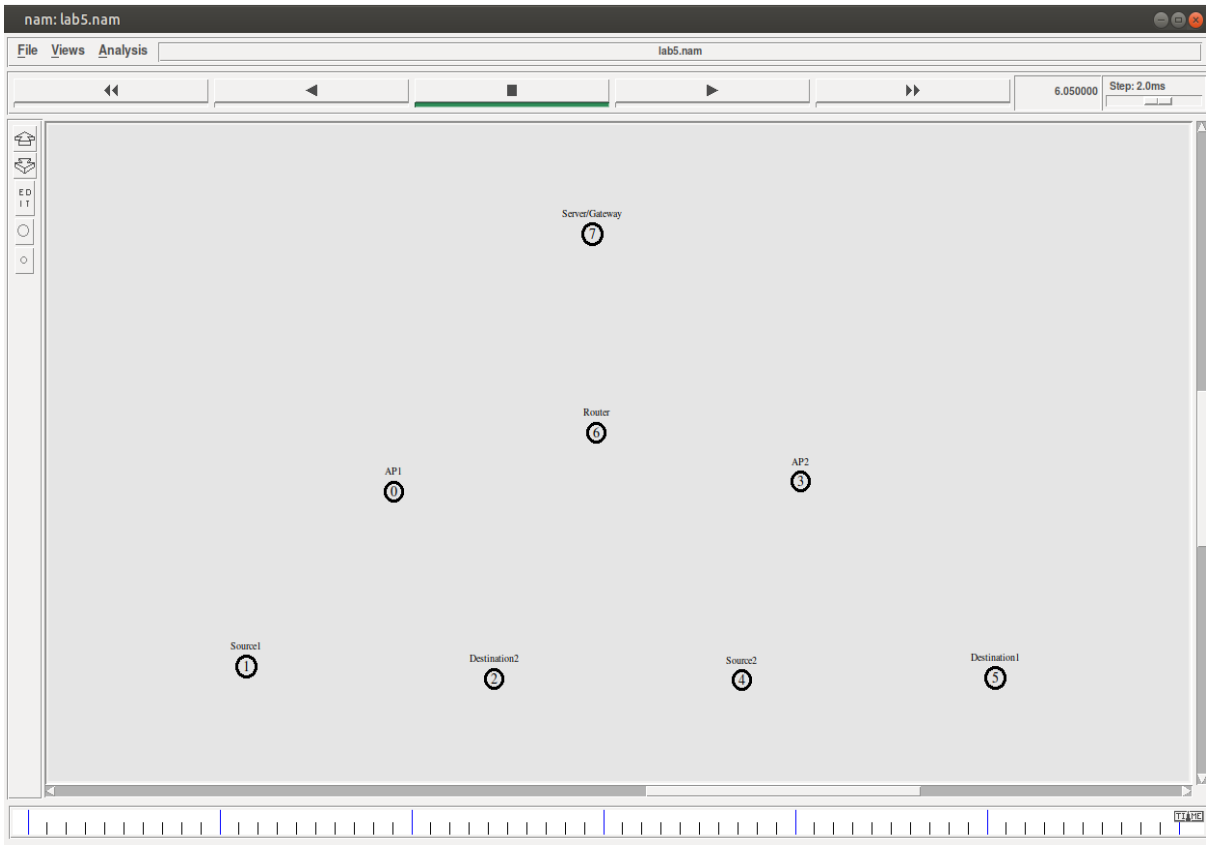
[root@localhost ~]# ns lab5.tcl

- **ns-Network Simulator**

Step 3: Once the simulation is completed, run awk file to observe the output

[root@localhost ~]# awk -f lab5.awk lab5.tr

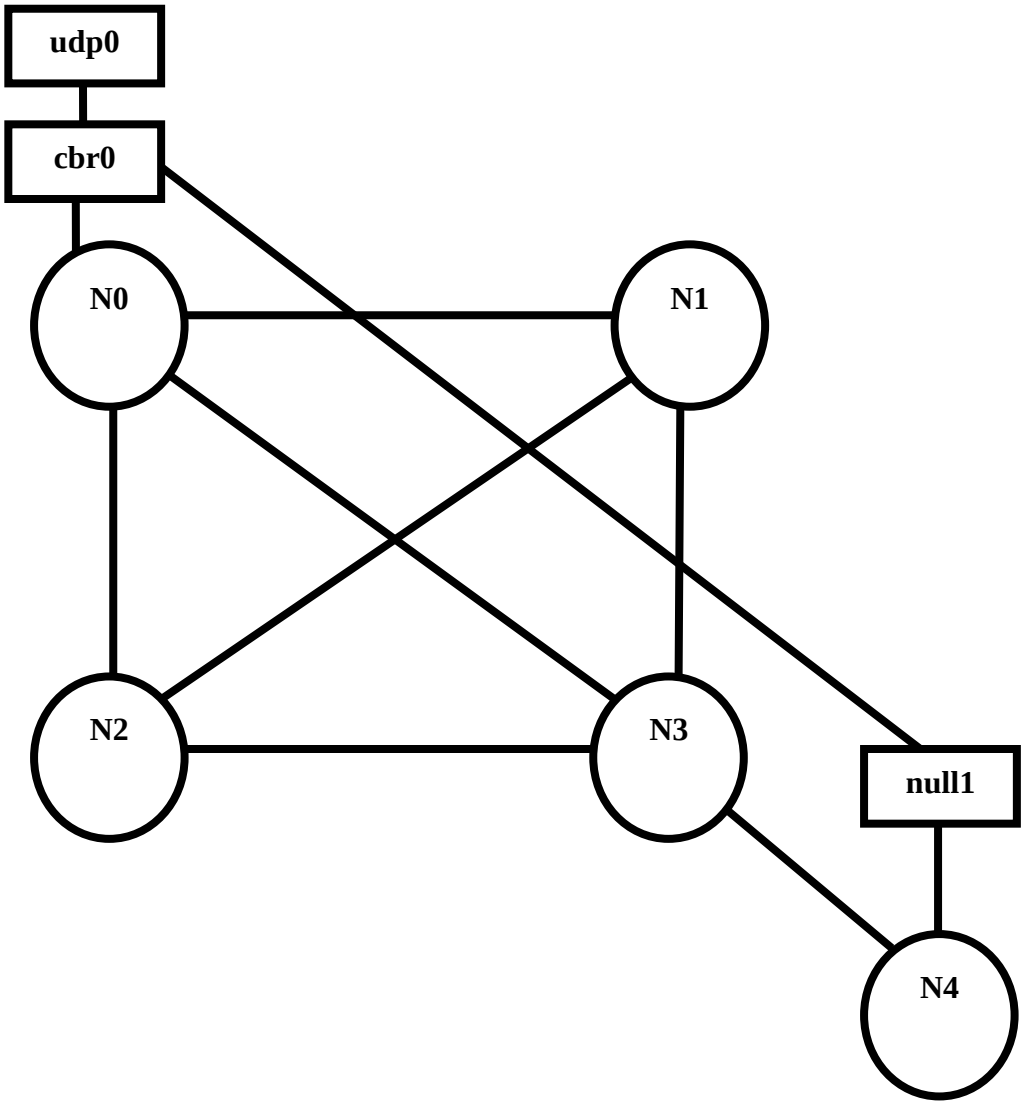




Assignment Problem

1. **Implement ESS with transmission nodes in Wireless LAN and obtain then Performance Parameter like packet delivery ratio.**
2. **Implement ESS with transmission nodes in Wireless LAN and obtain then Performance Parameter like packet loss.**
3. **Implement ESS with transmission nodes in Wireless LAN and obtain then Performance Parameter like packet delivery ratio.**
4. **Implement ESS with transmission nodes in Wireless LAN and obtain then Performance Parameter like packet delivery ratio and packet loss by change is bandwidth and queue size.**

5. Implementation of Link State Routing Algorithm.



The above network consists of 5 nodes in which node N0 and N4 acts as source and destination and other nodes behaves as intermediate nodes. There are five paths for sending cbr packets from N0 to N4 such as,

- N0-NI-N3-N4
- N0-N2-N3-N4
- N0-N3-N4
- N0-N1-N2-N3-N4
- N0-N2-N1-N3-N4

Among these entire paths, link state algorithm will find out the shortest path based on cost assigned to each link.

#==================================

```
#     Simulation parameters setup
#==================================
set val(stop)   10.0                ;# time of simulation end
#==================================
#       Initialization
#==================================
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
set tracefile [open lab6.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open lab6.nam w]
$ns namtrace-all $namfile
#==================================
#       Nodes Definition
#==================================
#Create 5 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
#==================================
#       Links Definition
#==================================
#Createlinks between nodes
$ns duplex-link $n0 $n1 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n1 50
$ns duplex-link $n0 $n2 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n2 50
$ns duplex-link $n2 $n3 100.0Mb 10ms DropTail
$ns queue-limit $n2 $n3 50
$ns duplex-link $n1 $n3 100.0Mb 10ms DropTail
$ns queue-limit $n1 $n3 50
$ns duplex-link $n3 $n4 100.0Mb 10ms DropTail
$ns queue-limit $n3 $n4 50
$ns duplex-link $n0 $n3 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n3 50
$ns duplex-link $n1 $n2 100.0Mb 10ms DropTail
$ns queue-limit $n1 $n2 50
```

```
#Give node position (for NAM)
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n1 $n3 orient left-down
$ns duplex-link-op $n3 $n4 orient left-down
$ns duplex-link-op $n0 $n3 orient right-down
$ns duplex-link-op $n1 $n2 orient left-down

#Set the link costs. All link costs are symmetric

$ns cost $n0 $n1 2
$ns cost $n0 $n2 1
$ns cost $n0 $n3 3

$ns cost $n1 $n0 2
$ns cost $n1 $n2 2
$ns cost $n1 $n3 3

$ns cost $n2 $n1 2
$ns cost $n2 $n0 1
$ns cost $n2 $n3 1

$ns cost $n3 $n2 1
$ns cost $n3 $n1 3
$ns cost $n3 $n0 3
$ns cost $n3 $n4 2

$ns cost $n4 $n3 2


#===================================
#        Agents Definition
#===================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set null1 [new Agent/Null]
$ns attach-agent $n4 $null1
$ns connect $udp0 $null1
$udp0 set packetSize_ 1500
#===================================
#        Applications Definition
#===================================
```

```
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1500
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 5.0 "$cbr0 stop"
```

$ns rtproto LS

```
#=================================
#       Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
   global ns tracefile namfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec nam lab6.nam &
   exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Awk Script (lab6.awk)

```
BEGIN{
a=0
}
{
if($1=="r"&&$3=="3"&&$4=="4"&&$5=="cbr"&&$6=="1500")
{
a++;
}
}
END{
printf("\n total number of data packets received at Node 4: %d\n", a++);
}
```

Steps for Execution

Step 1: Type the above program in text editor and save it with the filename and extension ".tcl"

For Example, lab6.tcl

- **tcl- Tool Command Language**

Step 2: Run the program in terminal window

[root@localhost ~]# ns lab6.tcl

- **ns-Network Simulator**

Step 3: Once the simulation is completed, run awk file to observe the output

[root@localhost ~]# awk -f lab6.awk lab6.tr

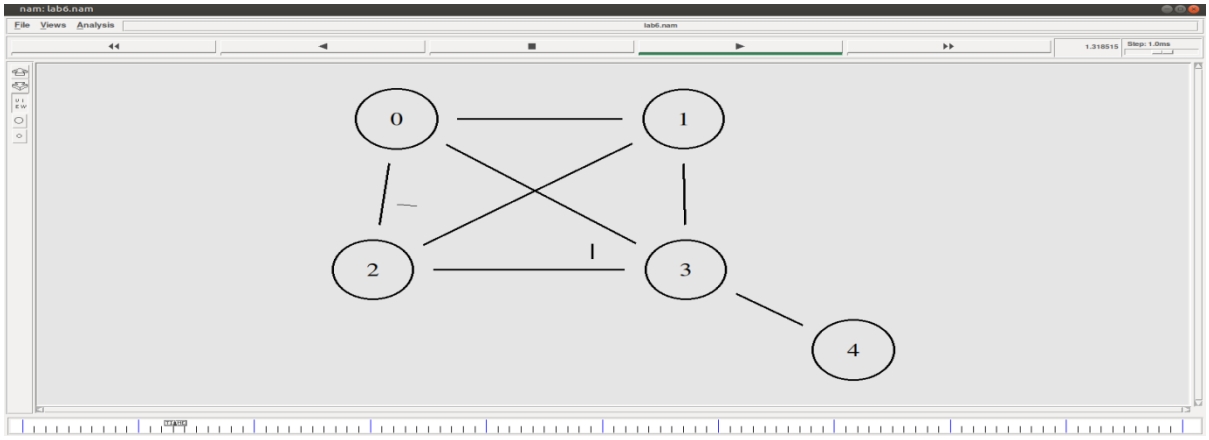
Simulation Results:

Total Number of Routing Paths: **4**

- N0-N1-N2-N3-N4: Total Cost is of **7**
- N0-N2- N1 -N3-N4 : Total Cost is of **8**
- N0-N2-N3-N4 : Total Cost is of **4**
- N0-N3-N4 : Total Cost is of **5**

Shortest according to Link State algorithm is **N0-N2-N3-N4** having Total Cost is of **4**
Total number of data packets received at Node 4: **334**

Network Animation (lab6.nam):



Assignment Problem

1. **Implementation of Link State Routing Algorithm by varying bandwidth and calculate throughput.**
2. **Implementation of Link State Routing Algorithm by varying queue size and calculate packet received.**
3. **Implementation of Link State Routing Algorithm by varying bandwidth and queue size, calculate packet sent.**

S

Part-B

Experiment No: 1 HDLC Frame

Aim: C Program for a HDLC frame to perform i) Bit stuffing ii) Character stuffing

1. BIT STUFFING

Aim: To write a program to implement bit stuffing for a given binary data. That is, stuff an extra '0' bit after continuous five 1's in the data..

Bit Stuffing

Simply, Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data. In a bit-oriented protocol, the data to send is a series of bits. In order to distinguish frames, most protocols use a bit pattern of 8-bit length (01111110) as flag at the beginning and end of each frame. Here also cause the problem of appearance of flag in the data part to deal with this an extra bit added. This method is called **bitstuffing**. In bit stuffing, if a 0 and five successive 1 bits are encountered, an extra 0 is added. The receiver node removes the extra-added zero.

CODING:
Program: bitstuf.c

```
// Program for Bit stuffing

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

int main()
{
        char a[100],b[100];
        int i,j,k,count,n,flag;

        strcpy(a,""); strcpy(b,"");

        for(;;){
                flag=0;
        printf("Enter input frame (0's & 1's only):");
        scanf("%s",a);
                n = strlen(a);
                for(i=0; i<n; i++){
                        if ((a[i]=='0') || (a[i]=='1' )){
                                continue;
                                }
                        else{
                                flag=1; break;
                        }
                }

                if(flag){
                        printf ("Invalid Input frame\n:");
```

```
                exit (0);
                }
        else{
                printf ("Valid Input frame\n:");
                printf ("Frame length = %d\n", n);  break;
            }
        }
    i=0; j=0; flag=0;
    while(i<n)
    {
        if (a[i]=='1')
        {
            b[j] = a[i]; count=1;
            for(k=i+1; a[k]=='1' &&  k<n  && count<5; k++)
            {
                j++;
                b[j]=a[k];
                count++;
                if(count==5)
                {
                    j++;
                    b[j]='0'; count=0;

                }
                i=k;
    }
        if(k<n){
            if(a[k]=='0'){
                j++;
                b[j] = a[k];
            }
            i=k;
          }
        else{   flag=1; break; }
    }
else{
        b[j] = a[i];
        i++;
        j++;
      }
  }
        j++;  b[i] = '\0';

        printf("\nAfter stuffing the frame is:");
        printf("%s\n",b);
        return 0;
}
```

Output:
Compile and run
[root@localhost]# **gcc bitstuf.c**
[root@localhost]# **./a.exe**

Enter Input frame (0's & 1's only):
00011111122222
Invalid Input frame

Enter Input frame (0's & 1's only):
100111110111111010101011110
Valid Input frame
Enter frame length:
25

After stuffing the frame is:
1001111100111111010101011110

Result:Implemented program for Bit stuffing and evaluated.

2 Character Stuffing

AIM: Implement the data link layer framing methods such as character, character stuffing.

Theory:

The character-oriented protocols are popular only with text data. Use reserved characters to indicate the start and end of a frame. For instance, use the two-character sequence DLE STX (Data-Link Escape Start of TeXt) to signal the beginning of a frame, and the sequence DLEETX (End of TeXt) to flag the frame's end.

The framing method gets around the problem of resynchronization after an error by having each frame starts with the ASCII character sequence DLESTX (Data Link Escape start of Text) and the sequence DLEETX (Data Link Escape End of Text). If the destination loses the track of frame boundaries all it has to do is look for DLESTX and DLEETX characters to figure out. The data link layer on the receiving end removes DLE before the data are given to the network layer. This technique is called byte stuffing or character stuffing.

Algorithm

1. Read the character stream from the user.
2. Measure the length of the entered sequence.
3. Read the source delimiter characters from the
4. Check for character sequence with source and destination delimiters of the given pattern.
5. If characters are in the sequence equal to delimiter characters, attach the same characters immediately after those characters.
6. Go to step-5 to repeat the process for the remaining.
7. Display the result.

program: bytestuf.c

// Program for Character stuffing

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

int main()
{
        char c[80],d[80],ed[10],sd[10];
        inti,j,m;

        strcpy(c,"");
        strcpy(sd,"dlestx");
        strcpy(ed,"dleetx");

        printf("\nEnter the characters to be stuffed:");
        scanf("%s", c);
        strcpy(d,sd);
```

```
        m = strlen(c);
        for (i=0, j=6; i<m+1; i++, j++)
        {
                if((c[i] == 'd') && (c[i+1] =='l') && (c[i+2] == 'e'))
                {
                d[j] = 'd'; j++;
                d[j] = 'l'; j++;
                d[j] = 'e'; j++;
                i = i+3;
                }
        d[j] = c[i];
         }
        j++; d[j]='\0';
        strcat (d,ed);
        printf("\n\nAfter stuffing, transmitted data: %s",d);
        return 0;
}
```

Compile and run

[root@localhost]# **gcc bytestuf.c**
[root@localhost]# **./a.exe**

Output:

Enter the characters to be stuffed: javed

After stuffing, transmitted data:dlestxjaveddleetx

Result: Thus the program for bit stuffing and character stuffing is executed

Experiment No: 2 Distance Vector Algorithm

Aim: C program for Distance vector algorithm to find suitable path for transmission. Basically obtain Routing table art each node using distance vector routing algorithm for given subnet.

Distance Vector Algorithm description:

The name distance vector is derived from the fact that routes are advertised as vectors of (distance, direction), where distance is defined in terms of a metric and direction is defined in terms of the next-hop router.

Distance vector algorithm operates by having each router maintain a table (i.e., vectors) giving best known distance to each destination and which line to get there. These tables are updated by exchanging the information with the neighbours.

In distance vector routing, each router maintains a routing table indexed by, and containing one entry for, each router in the subnet. This entry contains two parts: the preferred outgoing line to use for that destination and a distance to that destination. The router is assumed to know the “distance” to each of its neighbour.

Because each router depends on its neighbors for information, which the neighbors in turn may have learned from their neighbors, and so on, distance vector routing is sometimes facetiously referred to as "routing by rumor." The common Characteristics are

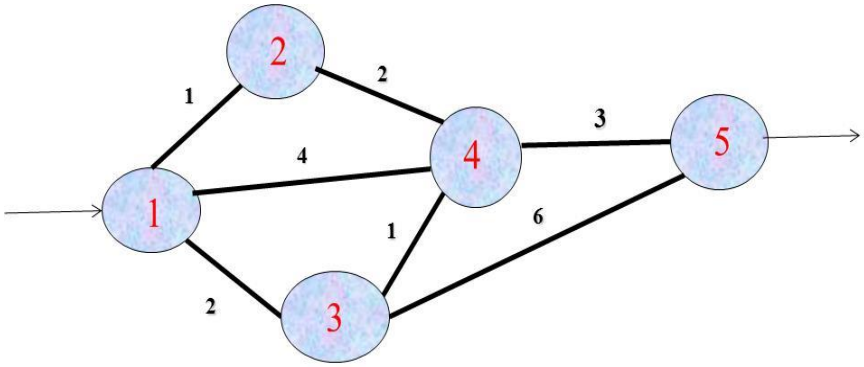
Periodic Updates

Periodic updates means that at the end of a certain time period, updates will be transmitted.

Neighbors

The starting assumption for distance-vector routing is that each node knows the cost of the link to each of its directly connected neighborsIn the context of routers, neighbors always mean routers sharing a common data link. Distance vector routing information may be, **Network ID, Cost and NextHop**. These three essentials need to form a Distance vector’s routing table.

Design



Algorithm

1. Start the program.
2. Read the number of nodes in the given network.
3. Read the distance matrix from the user. It represents cost distance of each node which connected directly.
4. By convention, the distance of the node to itself is assigned to zero and when a node is unreachable the distance is accepted as 999.
5. Store above values in a suitable variable and display the complete routing table.
6. Enter the source node and destination node to find the shortest.
7. Calculate the minimum distance by iteration. If the distance cost between the two nodes is smaller than the available cost, replace the existence distance with calculated distance.
8. Display the shortest path calculated and its cost between source node and destination node.
9. Go to step-7, to find other short route between other nodes or else goto next step.
10. End the program.

Program: dist_vector.c

// Program for Distance Vector Routing algorithm

```
#include <stdio.h>
#include <stdlib.h>
#define nul 1000
#define nodes 10
int no=5;
struct node
{
        int a[nodes][3];
}
router[nodes];
void init(int r)
{
        int     i;

        for(i=1; i<=no; i++)
        {
                router[r].a[i][1]=i;
                router[r].a[i][2]=999;
                router[r].a[i][3]=nul;
        }
        router[r].a[r][2]=0;
        router[r].a[r][3]=r;
        }
```

```
void inp(int r)
{
        int i;
        printf("\n \t Enter distance to the node %d to other nodes", r);
        printf("\n \t Please enter 999 if there is no direct route\n");
        for(i=1; i<=no; i++)
        {
                if(i!=r){
                        printf("\n Enter distance to the node %d:", i);
                        scanf("%d", &router[r].a[i][2]);
                        router[r].a[i][3]=i;
                        }
        }
}
void display(int r)
{
        int      i;
        printf("\n\n The routing table for node %d is as follows", r);
        printf("\n\t\tDest\t\tNext Hop\t\tDist");
        for(i=1; i<=no; i++)
        {
                printf("\n\t\t%d\t%d \t\t%d",   router[r].a[i][1],router[r].a[i]
                                                        [3],router[r].a[i][2]);
        }
}
voiddv_algo(int r)
{
        inti,j,z;
        for(i=1; i<=no; i++)
        {
                if(router[r].a[i][2]!=999 && router[r].a[i][2]!=0)
                {
                        for(j=1; j<=no; j++)
                        {
                                z=router[r].a[i][2]+router[i].a[j][2];
                                if(router[r].a[j][2]>z)
                                {
                                        router[r].a[j][2]=z;
                                        router[r].a[j][3]=i;
                                }
                        }
                }
        }
}
```

```
void find(int x, int y)
{
	if(router[x].a[y][3]!=y)
	{
		find(x, router[x].a[y][3]);
		printf("%d-->",router[x].a[y][3]);
		find(router[x].a[y][3],y);
		return;
	}
}
int main()
{
	int	i,j,x,y,no;
	int	choice;

	no = 5;
	for(i=1; i<=no; i++)
	{
		init(i);
		inp(i);
	}
	printf("\n The configuration of the nodes after initialization is as follows:");
	for(i=1; i<=no; i++)
		display(i);

	for(j=1; j<=no; j++)
		for(i=1; i<=no; i++)
			dv_algo(i);
	printf("\n The configuration of the nodes after computation of path is as follows:");
	/*printf("\n\tDest\tNext Hop\tDist");*/
	for(i=1; i<=no; i++)
		display(i);

	while(1)
	{
		printf("\n\n Enter 1 to continue, 0 to quit:");
		scanf("%d",&choice);
		if(choice!=1)
			break;
		printf("\n Enter the nodes between which shortest path is to be found:");
		scanf("%d%d",&x,&y);
		printf("\n The shortest path is:");
		printf("%d--->",x);
```

```
            find(x,y);
            printf("%d",y);
            printf("\n The length of the shortest path is %d",router[x].a[y][2]);
            }
    return 0;
    }
```

Output:

Compile and run

[root@localhost]# **gcc dist_vector.c**
[root@localhost]# **./a.exe**

Enter distance to the node 1 to other nodes
Please enter 999 if there is no direct route
Enter distance to the node 2:1
Enter distance to the node 3:2
Enter distance to the node 4:4
Enter distance to the node 5:999

Enter distance to the node 2 to other nodes
Please enter 999 if there is no direct route
Enter distance to the node 1:1
Enter distance to the node 3:999
Enter distance to the node 4:2
Enter distance to the node 5:999

Enter distance to the node 3 to other nodes
Please enter 999 if there is no direct route
Enter distance to the node 1:2
Enter distance to the node 2:999
Enter distance to the node 4:1
Enter distance to the node 5:6

Enter distance to the node 4 to other nodes
Please enter 999 if there is no direct route
Enter distance to the node 1:4
Enter distance to the node 2:2
Enter distance to the node 3:1
Enter distance to the node 5:3

Enter distance to the node 5 to other nodes
Please enter 999 if there is no direct route
Enter distance to the node 1:999
Enter distance to the node 2:999
Enter distance to the node 3:6
Enter distance to the node 4:3

The configuration of nodes after computation of path is as follows:

The routing table for node 1 is as follows:

Dest	Next Hop	Dist
1	1	0
2	2	1
3	3	2
4	2	3
5	4	6

The routing table for node 2 is as follows:

1	1	1
2	2	0
3	1	3
4	4	2
5	4	5

The routing table for node 3 is as follows:

1	1	2
2	1	3
3	3	0
4	4	1
5	4	4

The routing table for node 4 is as follows:

1	2	3
2	2	2
3	3	1
4	4	0
5	5	3

The routing table for node 5 is as follows:

1	4	6
2	4	5
3	4	4
4	4	3
5	5	0

Enter 1 to continue, 0 to quit: 1
Enter the nodes between which shortest path is to be found: 1 5

The shortest path is : 1→2→4→5
The length of the shortest path is 6

Enter 1 to continue, 0 to quit: 1
Enter the nodes between which shortest path is to be found: 2 5

The shortest path is :2→4→5
The length of the shortest path is 5

Enter 1 to continue, 0 to quit: 1
Enter the nodes between which shortest path is to be found: 3 5

The shortest path is :3→4→5
The length of the shortest path is 4

Enter 1 to continue, 0 to quit: 0

Result: **Thus the program for Distance Vector Algorithm is executed.**

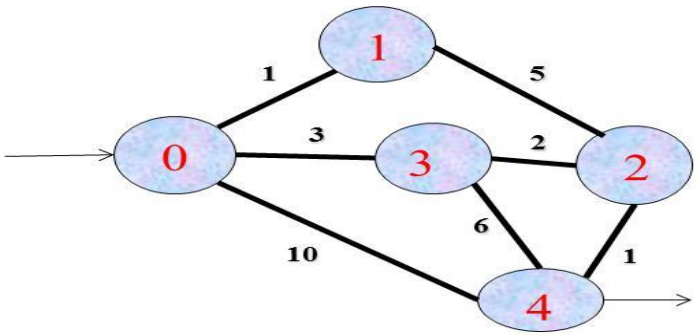
Experiment No: 3 Dijkstra's Algorithm

Theory: Dijkstra's Algorithm

Dijkstra's algorithm is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs, producing a shortest path tree. The Dijkstra algorithm follows four steps to discover what is called the shortest path tree(routing table) for each router: The algorithm begins to build the tree by identifying its roots. The root router's trees the router itself. The algorithm then attaches all nodes that can be reached from the root. The algorithm compares the tree's temporary arcs and identifies the arc with the lowest cumulative cost. This arc and the node to which it connects are now a permanent part of the shortest path tree. The algorithm examines the database and identifies every node that can be reached from its chosen node. These nodes and their arcs are added temporarily to the tree. The last two steps are repeated until every node in the network has become a permanent part of the tree.

Aim: Implement Dijkstra's algorithm to compute the shortest path through a given graph using C program

Design



Algorithm:

1. Assign to every node a tentative distance value: set it to zero for our initial node and infinity for all other nodes.

2. Set the initial node as current. Mark all other nodes unvisited. Create a set of all the unvisited nodes called the unvisited set.

3. For the current node, consider all of its neighbors and calculate their tentative distances. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbour B has length 2, then the distance to B (through A) will be 6 +2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.

4. When we are done considering all of the neighbours of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.

5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.

6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.

program: dijkstra.c
//Program for Dijkstra's Algorithm for shortest path

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

#define    INFINITY  9999
#define         MAX  10

void dijkstra( int G[MAX][MAX], int n, int startnode);
int main ()
{
        int   G[MAX][MAX ],i, j, n, u;
        printf("Enter no. of vertices:");
        scanf("%d",&n);
        printf ("\nEnter the adjacency matrix:\n");
        for(i=0; i<5; i++)
                for(j=0; j<5; j++)
                        scanf("%d", &G[i][j]);
        printf("Enter the starting node:");
        scanf("%d", &u);
        dijkstra(G,n,u);
        return 0;
}

void dijkstra(int G[MAX][MAX],int n, int startnode)
{
        int cost[MAX][MAX], distance[MAX], pred[MAX];
        int visited[MAX], count, mindistance, nextnode, i, j;

        /*pred[] stores the predecessor of each nod
        count gives the number of nodes seen so far and create the cost matrix */

        for(i=0; i<n; i++)
```

```
        for(j=0; j<n; j++)
                if(G[i][j]==0)
                        cost[i][j]=INFINITY;
                else
                        cost[i][j]=G[i][j];
//initialize pred[],distance[] and visited[]
for(i=0; i<n; i++)
{
        distance[i]=cost[startnode][i];
        pred[i]=startnode;
        visited[i]=0;
}
distance[startnode]=0;
visited[startnode]=1;
count=1;
while(count<n-1)
{
        mindistance=INFINITY;

        //next node gives the node at minimum distance
        for (i=0; i<n; i++)
                if((distance[i]<mindistance) && (!visited[i]))
                {
                        mindistance=distance[i];
                        nextnode=i;
                }

                //check if a better path exists through nextnode
                visited[nextnode]=1;
                for(i=0; i<n; i++)
                        if(!visited[i])
                                if((mindistance+cost[nextnode][i])<distance[i])
                                {
                                        distance[i]=nextnode;
                                }
count++;
}
//print the path and distance of each node
for(i=0; i<n; i++)
        if(i!=startnode)
        {
                printf("\nDistance of node%d=%d",i,distance[i]);
                printf("\nPath=%d", i);

                j=i;
                do{
                        j=pred[j];
                        printf("←%d", j);
                }while(j!=startnode);
        }
```

}

Output:

Compile and run

[root@localhost]# gcc dijkstra.c
[root@localhost]# ./a.exe

Enter no. of vertices:5
0 1 03 10
1 0 50 0
0 5 0 2 1
302 0 6
10 0 1 6 0

Enter thestarting node:0
Distance of node1=1
Path=1←0
Distance of node2=5
Path=2←3←0
Distance of node3=3
Path=3←0
Distance of node4=6
Path=4←2←3←0

Result: Thus the program implement Dijkstra‘s algorithm to compute the Shortest path through a graph is executed

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

Theory: CRC (Cyclic Redundancy Check)

The cyclic redundancy check, or CRC, is a technique for detecting errors in digital data, but not for making corrections when errors are detected. It is used primarily in data transmission.

This Cyclic Redundancy Check is the most powerful and easy to implement technique. CRC is based on binary division. In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number. At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted. A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.

Some CRC polynomials that are actually used

- CRC-8:
  x^8+x^2+x+1 Used in: 802.16(along with error correction)
- CRC-CCITT:
  $x^{16}+x^{12}+x^5+1$ Used in: HDLC, SDLC, PPP default

Performance:

CRC is a very effective error detection technique. If the divisor is chosen according to the previously mentioned rules, its performance can be summarized as follows:

• CRC can detect all single-bit errors

• CRC can detect all double-bit errors (three 1's)

• CRC can detect any odd number of errors (X+1)

• CRC can detect all burst errors of less than the degree of the polynomial.

• CRC detects most of the larger burst errors with a high probability.

• For example CRC-12 detects 99.97% of errors with a length 12 or more.

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

Algorithm

1. Enter the message to be transmitted
2. Append the message with 16(since it is 16-bit CRC) 0`s (i.e. if you input 5 digit message, the appeneded message should be 21-bits.)
3. XOR appended message and transmit it.(Here, you compare with an already existing string such as 10001000000100001 and replace the bits the same way XOR operation works)
4. Verify the message that is received is the same as the one sent.

program: crc.c

```
// Program for CRC Algorithm

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

intcrc(char *ip, char *op, char *poly, int mode)
{
int  i,j,k;
strcpy(op, ip);

if (mode) {
for ( i = 0; i < strlen(poly); i++)
strcat(op, "0");
   }
   /* Perform XOR on the msg with the selected polynomial */
for (j = 0; j < strlen(ip); j++) {
if (op[j] == '1') {
for (k = 0; k < strlen(poly); k++) {
if ((op[j + k] == '0') && (poly[k]=='0') || (op[j+k] == '1') && (poly[k] =='1'))
op[j + k] = '0';
else
op[j +k] = '1';
          }
      }
   }

 /* check for errors. return 0 if error detected */
for (j = 0; j < strlen(op); j++)
```

```
if (op[j]== '1')
return 1;
return 0;
}

int main()
{
inti,n,flag;
charip[50], op[50], recv[50];
char poly[] = "10001000000100001";

/*The Generator polynomial consists of 17 bits.*/
        flag=0;strcpy(ip,""); strcpy(op,"");
        printf("CRC-16 \n");
        printf("Enter the input message in binary:\n");
scanf("%s", &ip);
        n = strlen(ip);
        for(i=0; i<n; i++){
                if ((ip[i]=='0') ||(ip[i]=='1' )){
                        continue;
                }
                else{
                        flag=1; break;
                 }
          }

if(flag){
        printf ("Invalid input message\n:");
        exit (0);
        }
crc(ip, op, poly, 1);
printf("The transmitted message is: %s%s\n",ip,op/*+strlen(ip)*/);
printf("Enter the received message in binary\n");
scanf("%s", &recv);
if(!crc(recv, op, poly, 0))
printf("No error in data\n");
else
printf("Error in data transmission has occurred\n");

return 0;
}
```

Output:

Compile and run

[root@localhost]# **gcc crc.c**
[root@localhost]# **./a.exe**

Enter the input message in binary:
11111
The transmitted message is: 111111110001111011110

Enter the received message in binary:
11111
No error in data.

Enter the input message in binary:
11111
The transmitted message is: 111111110001111011110

Enter the received message in binary:
1111

Error in data transmission occurred.

[root@localhost]

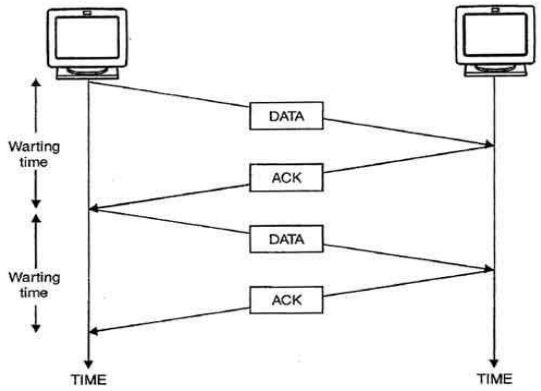
Result:Thus the program for cyclic redundancy check is executed.

Experiment No: 5 Implementation of stop and wait protocol and Sliding Window Protocol

Aim: *C Program for stop and wait protocol and Sliding Window Protocol*

1. STOP AND WAIT PROTOCOL

AIM: Implementation of Stop and Wait protocol



Stop & Wait Method.

Stop and wait is the fundamental technique to provide reliable transfer under unreliable packet delivery system. After transmitting one packet, the sender waits for an acknowledgment (ACK) from the receiver before transmitting the next one. In this way, the sender can recognize that the previous packet is transmitted successfully and we could say "stop-n-wait" guarantees reliable transfer between nodes. To support this feature, the sender keeps a record of each packet it sends. Also, to avoid confusion caused by delayed or duplicated ACKs, "stop-n-wait" send each packet with unique sequence numbers and receive that numbers in each ACK. If the sender doesn't receive ACK for previous sent packet after a certain period of time, the sender times out and retransmit that packet again. There are two cases when the sender doesn't receive ACK; one is when the ACK is lost and the other is when the frame itself is not transmitted. To support this feature, the sender keeps timer per each packet.

The main disadvantage of this method is that it is inefficient. It makes the transmission process slow. In this method single frame travels from source to destination and single acknowledgment travels from destination to source. As a result each frame sent and received uses the entire time needed to traverse the link. Moreover, if two devices are distance apart, a lot of time is wasted waiting for ACKs that leads to increase in total transmission time.

ALGORITHM

Step 1: Start the program
Step 2: Generate random that gives the total number of frames to be transmitted.
Step 3: Transmit the first frame
Step 4: Receive the acknowledgement for that frame
Step 5: Transmit the next frame
Step 6: Find the remaining frames to be sent
Step 7: If an acknowledgement is not received for a particular frame retransmit that frame alone again.
Step 8: Repeat the steps 5 to 7 till the number of remaining frames to be sent becomes zero.
Step 9: Stop the program.

program: stopnwait.c

// Program for Stop and waitprotocol

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

int main()
{
int   i,j,noframes;
int    x,x1,x2;

      x1=10; i=1;  j=1;
      printf("\n Enter number of frames\t:");
      scanf("%d",&noframes);
      while(noframes>0)
      {
            printf("\n Sending frame %d", i);
            x = rand()%15;
            if(x%5 == 0){
            for(x2=1; x2<2; x2++)
            {
                  printf("\n Waiting for %d seconds\n",x2);
                  sleep(x2);
            }
      printf(" Sending frame %d\n",i);
      x = rand()%10;
      }
      printf("\n Acknowledgement received for frame %d\n",j);
      noframes = noframes-1;
      i++;
      j++;
      }
      printf("\n End of stop and wait protocol");
      return 0;
}
```

Output:

Compile and run
[root@localhost]# gcc stopnwait.c
[root@localhost]# ./a.exe

No of frames is 6
Sending frame 1
Acknowledgement for frame 1
Sending frame 2
Acknowledgement for frame 2
Sending frame 3
Acknowledgement for frame 3
Sending frame 4
Acknowledgement for frame 4
Sending frame 5
Waiting for 1 second
Sending frame 5
Acknowledgement for frame 5
Sending frame 6
Waiting for 1 second
Sending frame 6
Acknowledgement for frame 6
End of stop and wait protocol

[root@localhost]#

Result:Implemented program for Stop and wait protocol and evaluated.

2. SLIDING WINDOW PROTOCOL
Aim: - Simulation of Sliding Window Protocol.

Theory:

In computer networks sliding window protocol is a method to transmit data on a network. Sliding window protocol is applied on the Data Link Layer of OSI model. At data link layer data is in the form of frames. In Networking, Window simply means a buffer which has data frames that needs to be transmitted.

Both sender and receiver agree on some window size. If window size=w then after sending w frames sender waits for the acknowledgement (ack) of the first frame.

As soon as sender receives the acknowledgement of a frame it is replaced by the next frames to be transmitted by the sender. If receiver sends a collective or cumulative acknowledgement to sender then it understands that more than one frames are properly received, for eg:- if ack of frame 3 is received it understands that frame 1 and frame 2 are received properly.

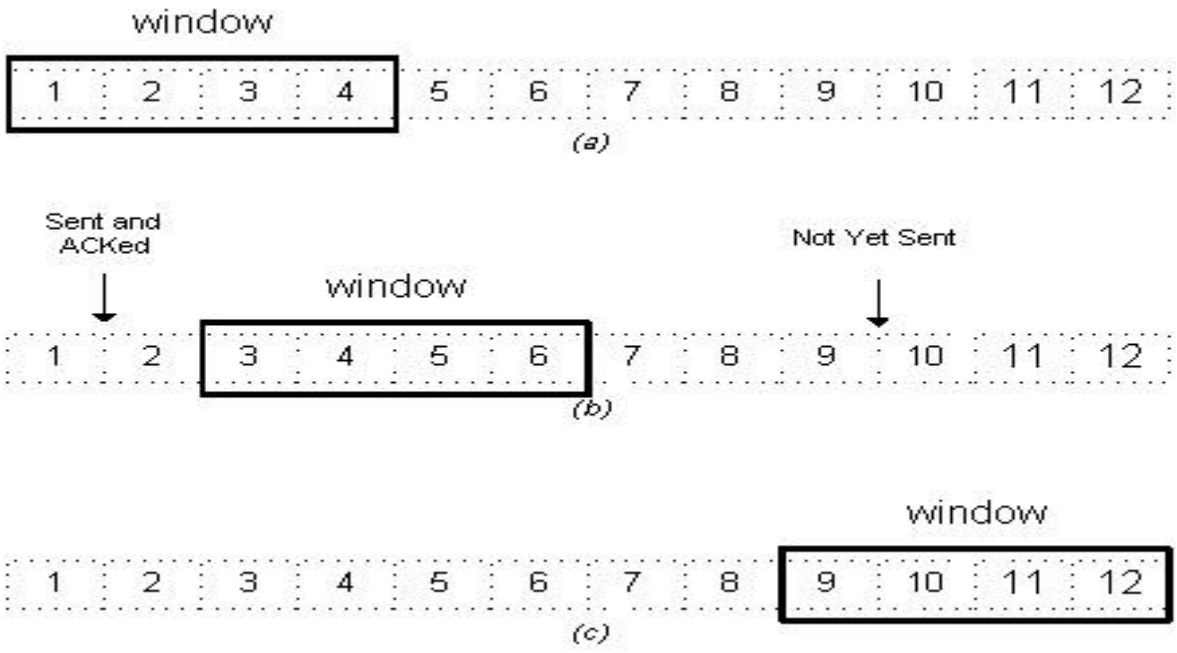


Figure: Image source

In sliding window protocol the receiver has to have some memory to compensate any loss in transmission or if the frames are received unordered.

Efficiency of Sliding Window Protocol

$\eta = (W*t_x)/(t_x+2t_p)$

W = Window Size

t_x = Transmission time
t_p = Propagation delay
Sliding window works in full duplex mode

It is of two types:-

1. Selective Repeat: Sender transmits only that frame which is erroneous or is lost.
2. Go back n: Sender transmits all frames present in the window that occurs after the error bit including error bit also.

Algorithm:

1. Start the program

2. Read the window size

3. Read number of frames to transmit

4. Read randomly selected frames to transmit

5. Transfer the packet until it reaches the maximum defined size.

6. Reduce the window size and repeat the above two steps until packets in.

7. Stop the program

Sliding Window Protocol Program in C

program: slidingwindow.c

```
#include  <stdio.h>
#include  <stdlib.h>
#include <string.h>
int main()
{
int w,i,f,frames[50];

        printf("Enter window size: ");
        scanf("%d",&w);

        printf("\nEnter number of frames to transmit: ");
        scanf("%d", &f);

        printf("\nEnter %d frames: ",f);

        for(i=1; i<=f; i++)
                scanf("%d",&frames[i]);

        printf("\nWith sliding window protocol the frames will be sent in the following manner
        (assuming no corruption of frames)\n\n");
        printf("After sending %d frames at each stage sender waits for acknowledgement sent by
        the receiver\n\n",w);

        for(i=1;i<=f;i++)
         {
                if(i%w==0)
                {
                        printf("%d\n",frames[i]);
                        printf("Acknowledgement of above frames sent is received by sender\n\n");
                }
                else
                printf("%d ",frames[i]);
        }

        if(f%w!=0)
                printf("\nAcknowledgement of above frames sent is received by sender\n");

        return 0;
}
```

Output

Compile and run

```
[root@localhost ]# gcc slidingwindow.c
[root@localhost ]# ./a.exe
```

Output:

Case-1: For Window Size < No. of Frames

Enter window size: 5
Enter number of frames to transmit: 10

Enter 10 frames: 1 2 3 4 5 6 7 8 9 10

With sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)

After sending 5 frames at each stage sender waits for acknowledgement sent by the receiver

1 2 3 4 5

Acknowledgement of above frames sent is received by sender

6 7 8 9 10

Acknowledgement of above frames sent is received by sender

Case-2: For Window Size = No. of Frames

Enter window size: 5
Enter number of frames to transmit: 5

Enter 5 frames: 1 2 3 4 5

With sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)

After sending 5 frames at each stage sender waits for acknowledgement sent by the receiver

1 2 3 4 5

Acknowledgement of above frames sent is received by sender

Case-3: For Window Size > No. of Frames

Enter window size: 5
Enter number of frames to transmit: 3

Enter 3 frames: 1 2 3

With sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)

After sending 3 frames at each stage sender waits for acknowledgement sent by the receiver

1 2 3

Acknowledgement of above frames sent is received by sender

Experiment No: 6 Congestion Control Using Leaky Bucket Algorithm

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

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.

Each host is connected to the network by an interface containing a leaky bucket that is a finite internal queue where all the incoming packets are stored in case there is space in the queue, otherwise the packets are discarded. This arrangement can be built into the hardware interface or simulated by the host operating system. The host is allowed to put one packet per clock into the network. This mechanism turns an uneven flow of packet from the user process inside the host into an even flow of packets onto the network, smoothing out bursts and greatly reducing the chances of congestion.

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)).

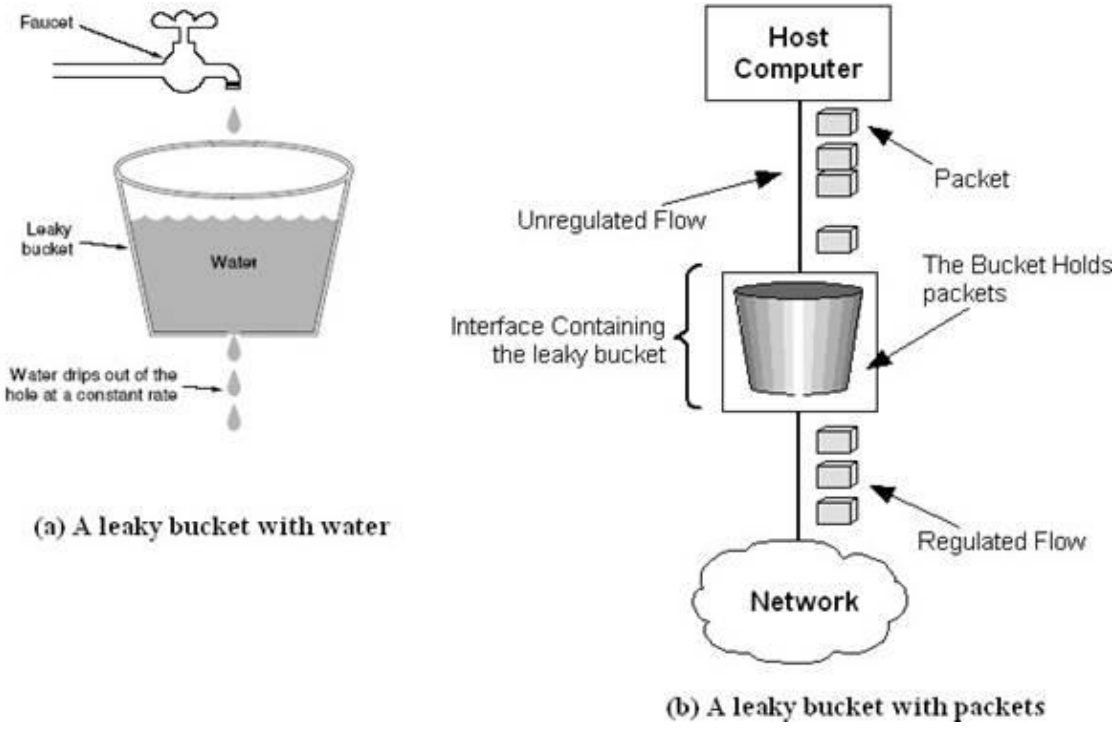


Figure: The leaky bucket traffic shaping algorithm

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.

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 packet to input into the bucket (input_packet).**
5. **while((Clock++<5*total_packets) and (out_packets<total_packets))**
    a. **if (clock == input_packet)**
        i. **insert into Queue**
    b. **if (clock % 5 == 0)**
        i. **Remove packet from Queue**
6. **end**

program: leakybucket.c

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

int main()
{
int drop=0, count=0, inp[25];
int   mini, nsec, cap, i, process;

        printf("\n Enter the Bucket Size: ");
        scanf("%d",&cap);
        printf("\n Enter the Operation Rate: ");
        scanf("%d",&process);
        printf("\n Enter the no. of Seconds you want to Stimulate: ");
        scanf("%d",&nsec);
        for(i=0;i<nsec;i++)
        {
                printf("\n Enter the Size of the Packet entering at %d sec: ",i+1);
                scanf("%d",&inp[i]);
        }
        printf("\nSecond|PacketRecieved|PacketSent|PacketLeft|Packet Dropped|\n");
        printf("-------------------------------------------------------------\n");
        for(i=0;i<nsec;i++)
        {
                count+=inp[i];
                if(count>cap)
                {
                        drop=count-cap;
                        count=cap;
                 }
        printf("%d",i+1);
        printf("\t%d",inp[i]);
        mini=min(count,process);
        printf("\t\t%d",mini);
        count=count-mini;
```

```
        printf("\t\t%d",count);
        printf("\t\t%d\n",drop);
        drop=0;
        }
        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);
        }
}
```

Output:
Compile and run

```
[root@localhost ]# gcc leakybucket.c
[root@localhost ]#  ./a.exe

Enter the Bucket Size: 5
Enter the Operation Rate: 2
Enter the no. of Seconds you want to Stimulate: 3
Enter the Size of the packet entering at 1 sec: 5
Enter the Size of the packet entering at 2 sec: 4
Enter the Size of the packet entering at 3 sec: 3

Second|PacketRecieved|PacketSent|PacketLeft|Packet Dropped|
-------------------------------------------------------------------------------
    1           5           2           3       0
    2           4           2           3       2
    3           3           2           3       1
    4           0           2           1       0
    5           0           1           0       0
[root@localhost ]#
```

Result: Implemented program for Leaky bucket algorithm and evaluated.

NS-2 Detailed information

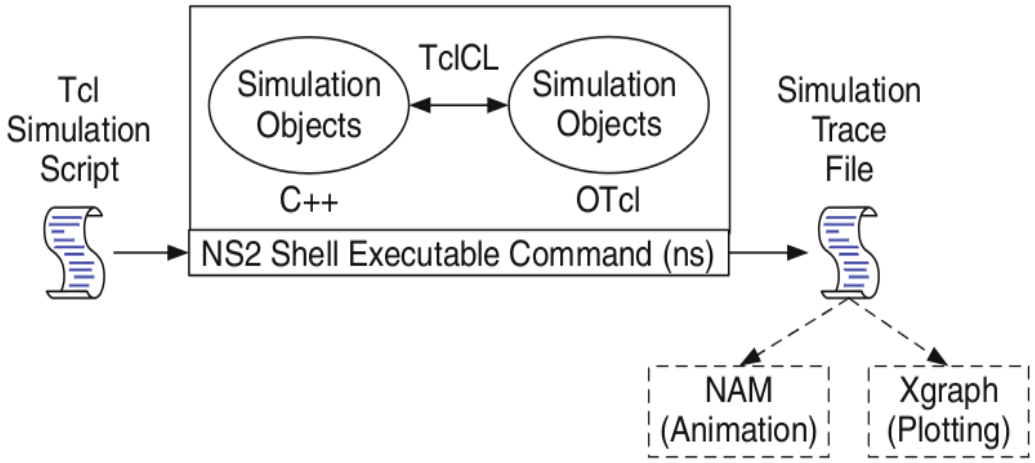
NS2 consists of two key languages: C++ and OTcl (Object-Oriented Tool Command Language). C++ language defines theinternal/ backend mechanism of the simulation by assembling and configuring the objects as well as scheduling the discrete/frontend events. Both C++ and OTcl are linked with each other using TclCL. The combination of both languages interpretthe scripts line by line the code written in gedit in Linux, ms-word or in the notepad in Windows etc. and saves the interpreted file with .tcl extension. NS2provides a large number of built-in C++ classes which can be used to set-up a simulation via a Tcl simulation script. One candevelop their own C++ classes and can use a OTcl configuration interface to put together the objects originated from these class.

After simulation, NS2 provides output as a Text-Based simulation results which can be interpreted graphically and interactivelyusing tools such as NAM (Network AniMator) and Xgraph. To analyze a particular behavior of the network, user can transportthat Text-Based data into a more conceivable presentation.

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**

```
procDiag {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 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 becauseit is an instance of the Simulator class, so an object the code[new Simulator] is indeed an instance 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 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 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]
```

The node is created which 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 includes 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, New Reno, Vegas. The type of agent appears in the first line:

```
settcp[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

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

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"

$ns at 124.5 "$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 (e.g. 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".

Design and simulation programs of Computer Networks lab

Tcl scripts for node, link creation, implementing NAM file, extracting trace file, plotting Xgraph, wireless LAN and wired LAN along with ping program. Coding in C includes CRC-16, routing algorithm, security implementation and TCP/IP socket programming. Congestion control algorithm to be implemented.

NAM

Nam is a Tcl/TK based animation tool for viewing network simulation traces and real world packet traces. It supports topology layout, packet level animation, and various data inspection tools.

The Network Animator (NAM) Tool

- The Network Animator (nam) is a completely separate program that is distributed with the NS simulator
- This program is named nam and it shows the progression of the packets through the network.
- The nam program reads an input file (containing the packet transmission events) and draw the network events graphically.

Running NAM

nam is a UNIX program and it is run as a command line.Example:

UNIX>> nam nam.input

nam.input is the file that contains network events

The key to making the animation input file is to tell NS to output network events into a file for nam to use.

Make NS output network event information for NAM

Do the following:

1. Create an output file

2. Activate the NAM trace feature in NS before running the simulation (this will tell NS to write NAM events outputs to the output file)

3. Close output file at the end of the simulation run

Then you can run nam with the output file (as input file for nam). Example:

Run the program using ns Reno1-nam.tcl; it will produce "out.nam" as output

When it finishes, run: nam out.nam to see the packets flow.

Making a better animation: If

1. The output does not look very good...
2. The nodes and links are placed very awkwardly
3. All packets (from all flows) are colored black.

We can change some parameters in the animation from inside NS !

Awk- An Advanced

awk is a programmable, pattern-matching, and processing tool available in UNIX. Itworks 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 there choice. You can now print a serial number, using the variable kount, and apply it those directors drawing a salary exceeding 6700:

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

$ catempawk.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.awkempn.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="|"}

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 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"}' empx.lst

1. NS2 INSTALLATION ON LINUX

1) Download **'ns-allinone-2.35'** from :
http://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.35/ns-allinone-2.35.tar.gz/download

2) Extract the downloaded zip file 'ns-allinone-2.35.tar.gz file' to desktop.

3) Now you need to download some essential packages for ns2,these packages can be downloaded by using the following command : **applications>accessories>terminal or dashhome>trminal**
then type the below line on the terminal window

"sudo apt-get install build-essential autoconf automake libxmu-dev libtool gcc"

or type this command

"sudo apt-get install autoconf automake gcc g++ build-essential libxmu-dev libtool libxt-dev"

4) Now change your directory(here i have already extracted the downloaded files to desktop,so my location is desktop) type the following codes in the command window to install NS2.

```
cd Desktop
cd ns-allinone-2.35
./install
```

The installation procedure will take a few minutes.....

5) After compleating the installation type the following command in the command window

```
gedit ~/.bashrc
```

6) Now an editor window appears,please copy and paste the follwing codes in the end of the text file (note that '/home/abhiram/Desktop/ns-allinone-2.35/octl-1.14' in each line in the below code should be replaced with your location where the 'ns-allinone-2.35.tar.gz'file is extracted)

```
# LD_LIBRARY_PATH
OTCL_LIB=/home/abhiram/Desktop/ns-allinone-2.35/otcl-1.14
NS2_LIB=/home/abhiram/Desktop/ns-allinone-2.35/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$USR_LOCAL_LIB

# TCL_LIBRARY
TCL_LIB=/home/abhiram/Desktop/ns-allinone-2.35/tcl8.5.10/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB
```

```
# PATH
XGRAPH=/home/abhiram/Desktop/ns-allinone-2.35/bin:/home/abhiram/Desktop/ns-allinone-
2.35/tcl8.5.10/unix:/home/abhiram/Desktop/ns-allinone-2.35/tk8.5.10/unix
NS=/home/abhiram/Desktop/ns-allinone-2.35/ns-2.35/
NAM=/home/abhiram/Desktop/ns-allinone-2.35/nam-1.15/
PATH=$PATH:$XGRAPH:$NS:$NAM
```

7) Save and close the text editor and then type the following command on the terminal

```
source ~/.bashrc
```

8) Close the terminal window and start a new terminal window and now change the directory to ns-2.35 and validate ns-2.35 by exicuting the following command (it takes 30 to 45 minutes)

```
cd ns-2.35
./validate
```

9) If the installation is successful, then you will be able to see **%** at the command prompt while typing the following command

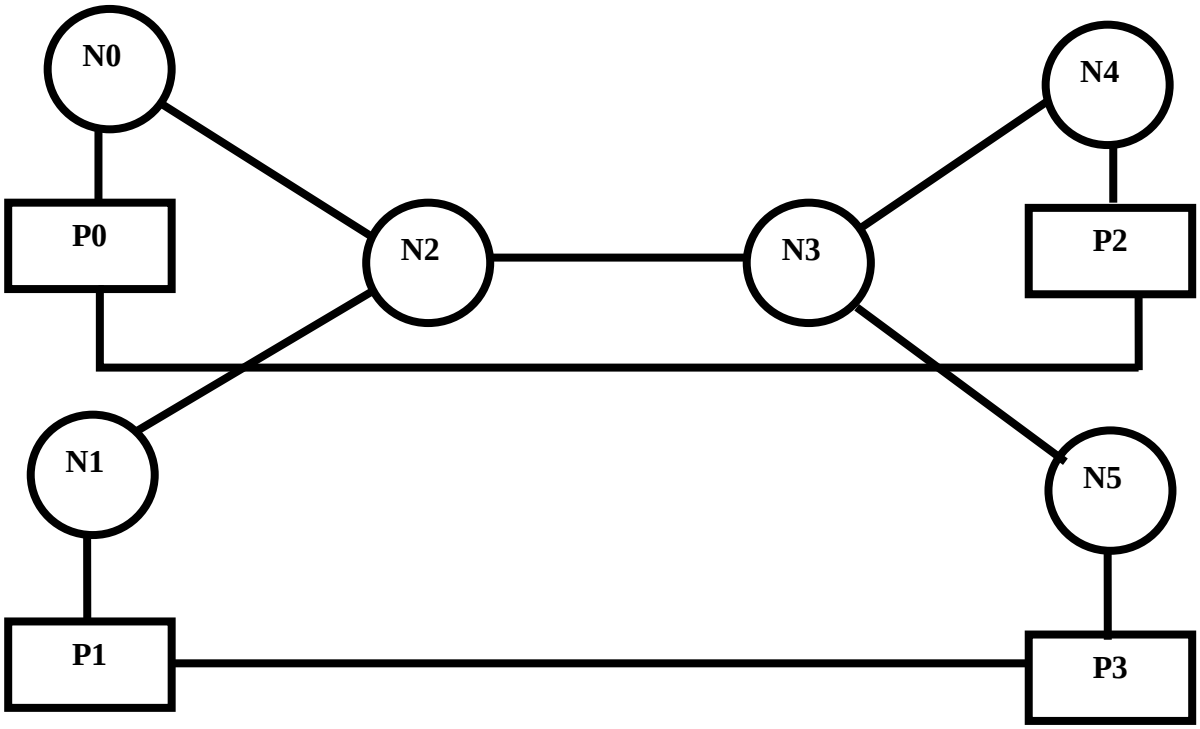
```
ns
```

10) Now type

```
exit
```

ADD-ON PROGRAMS

1.Simulate the transmission of Ping messages over a network topology consisting of six nodes and find the number of packets dropped due to congestion.



The above wired network consists of 6 nodes (N0 to N5) and Nodes N0, N1, N4, N5 are connected to ping agent, P0 will send message to P2 and P1 to P3, later P2 and P3 respond back to P0 and P1. This trace of communication can be seen in lab7.tr file and to check congestion effect, the bandwidth and queue size of link between N2 and N3 are reduced to 0.1Mb and 0/1.

```
#=================================
#       Initialization
#=================================
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
set tracefile [open lab7.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
```

```
set namfile [open lab7.nam w]
$ns namtrace-all $namfile

#=================================
#       Nodes Definition
#=================================
#Create 6 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
#=================================
#       Links Definition
#=================================
#Create links between nodes
$ns duplex-link $n0 $n2 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n2 50
$ns duplex-link $n1 $n2 100.0Mb 10ms DropTail
$ns queue-limit $n1 $n2 50
$ns duplex-link $n2 $n3 0.1Mb 10ms DropTail
$ns queue-limit $n2 $n3 0
$ns duplex-link $n3 $n4 100.0Mb 10ms DropTail
$ns queue-limit $n3 $n4 50
$ns duplex-link $n3 $n5 100.0Mb 10ms DropTail
$ns queue-limit $n3 $n5 50

#Give node position (for NAM)
$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
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down

#=================================
#       Agents Definition
#=================================
#Create two ping agents and attach them to the nodes n0 and n2
set p0 [new Agent/Ping]
$ns attach-agent $n0 $p0
```

```
$p0 set packetSize_ 50000

set p1 [new Agent/Ping]
$ns attach-agent $n1 $p1
$p1 set packetSize_ 50000
set p2 [new Agent/Ping]
$ns attach-agent $n4 $p2
$p2 set packetSize_ 50000

set p3 [new Agent/Ping]
$ns attach-agent $n5 $p3
$p3 set packetSize_ 50000

#Connect the two agents
$ns connect $p0 $p2
$ns connect $p1 $p3

#Schedule events
$ns at 0.2 "$p0 send"
$ns at 0.2 "$p1 send"

$ns at 0.6 "$p2 send"
$ns at 0.8 "$p3 send"

$ns at 1.0 "finish"

Agent/Ping instproc recv {from rtt} {
$self instvar node_
}
#=================================
#       Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
   global ns tracefile namfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec nam lab7.nam &
   exit 0
}
```

$ns run

Steps for Execution

Step 1: Type the above program in text editor and save it with the filename and extension ".tcl"

For Example, lab7.tcl

- **tcl- Tool Command Language**

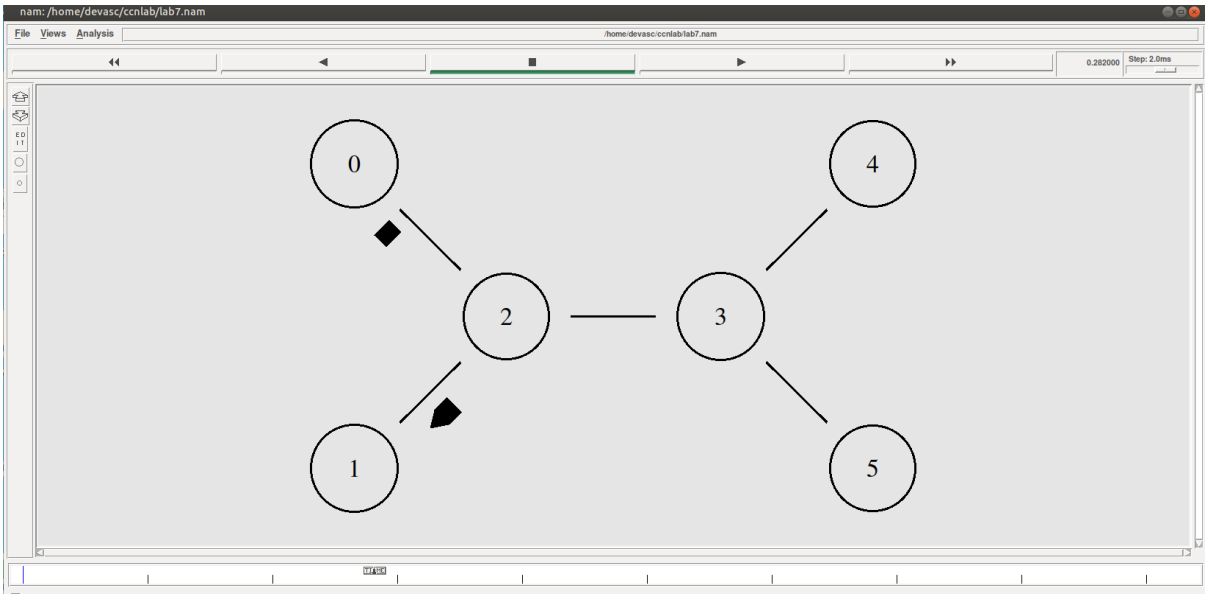
Step 2: Run the program in terminal window

[root@localhost ~]# ns lab7.tcl

- **ns-Network Simulator**

Step 3: Once the simulation is completed, run awk file to observe the output

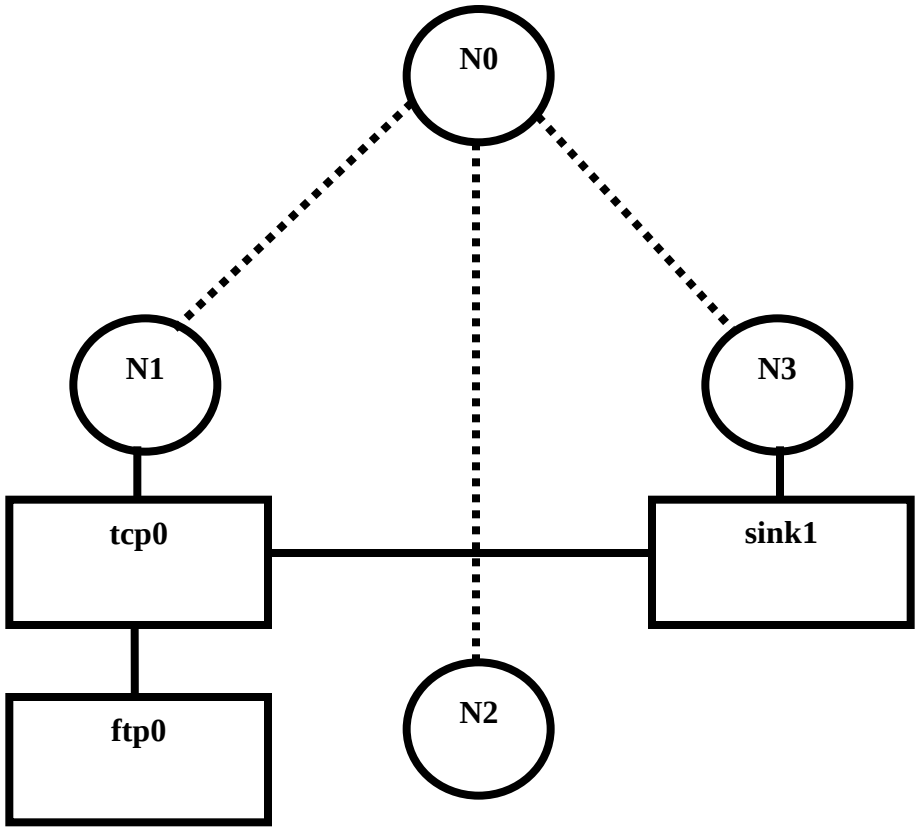
[root@localhost ~]# awk -f lab7.awk lab7.tr



lab7.tcl lab7.tr

```
+ 0.2 0 2 ping 64 ------- 0 0.0 4.0 -1 0
- 0.2 0 2 ping 64 ------- 0 0.0 4.0 -1 0
+ 0.2 1 2 ping 64 ------- 0 1.0 5.0 -1 1
- 0.2 1 2 ping 64 ------- 0 1.0 5.0 -1 1
r 0.210512 0 2 ping 64 ------- 0 0.0 4.0 -1 0
+ 0.210512 2 3 ping 64 ------- 0 0.0 4.0 -1 0
d 0.210512 2 3 ping 64 ------- 0 0.0 4.0 -1 0
r 0.210512 1 2 ping 64 ------- 0 1.0 5.0 -1 1
+ 0.210512 2 3 ping 64 ------- 0 1.0 5.0 -1 1
d 0.210512 2 3 ping 64 ------- 0 1.0 5.0 -1 1
+ 0.6 4 3 ping 64 ------- 0 4.0 0.0 -1 2
- 0.6 4 3 ping 64 ------- 0 4.0 0.0 -1 2
r 0.610512 4 3 ping 64 ------- 0 4.0 0.0 -1 2
+ 0.610512 3 2 ping 64 ------- 0 4.0 0.0 -1 2
- 0.610512 3 2 ping 64 ------- 0 4.0 0.0 -1 2
r 0.621024 3 2 ping 64 ------- 0 4.0 0.0 -1 2
+ 0.621024 2 0 ping 64 ------- 0 4.0 0.0 -1 2
- 0.621024 2 0 ping 64 ------- 0 4.0 0.0 -1 2
r 0.631536 2 0 ping 64 ------- 0 4.0 0.0 -1 2
+ 0.631536 0 2 ping 64 ------- 0 0.0 4.0 -1 3
- 0.631536 0 2 ping 64 ------- 0 0.0 4.0 -1 3
r 0.642048 0 2 ping 64 ------- 0 0.0 4.0 -1 3
+ 0.642048 2 3 ping 64 ------- 0 0.0 4.0 -1 3
d 0.642048 2 3 ping 64 ------- 0 0.0 4.0 -1 3
+ 0.8 5 3 ping 64 ------- 0 5.0 1.0 -1 4
- 0.8 5 3 ping 64 ------- 0 5.0 1.0 -1 4
r 0.810512 5 3 ping 64 ------- 0 5.0 1.0 -1 4
+ 0.810512 3 2 ping 64 ------- 0 5.0 1.0 -1 4
- 0.810512 3 2 ping 64 ------- 0 5.0 1.0 -1 4
r 0.821024 3 2 ping 64 ------- 0 5.0 1.0 -1 4
+ 0.821024 2 1 ping 64 ------- 0 5.0 1.0 -1 4
- 0.821024 2 1 ping 64 ------- 0 5.0 1.0 -1 4
r 0.831536 2 1 ping 64 ------- 0 5.0 1.0 -1 4
+ 0.831536 1 2 ping 64 ------- 0 1.0 5.0 -1 5
- 0.831536 1 2 ping 64 ------- 0 1.0 5.0 -1 5
r 0.842048 1 2 ping 64 ------- 0 1.0 5.0 -1 5
+ 0.842048 2 3 ping 64 ------- 0 1.0 5.0 -1 5
d 0.842048 2 3 ping 64 ------- 0 1.0 5.0 -1 5
```

2.Simulate a simple BSS with transmitting nodes in Wireless LAN and determine the performance with respect to transmission of packets.



The group of two or more nodes connecting to one assess point (N0) is called Basic Service Set (BSS). In this wireless topology, Node N1 and N3 act as source and destination.

```
#=================================
#     Simulation parameters setup
#=================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     4                          ;# number of mobilenodes
set val(rp)     AODV                       ;# routing protocol
set val(x)      959                        ;# X dimension of topography
set val(y)      834                        ;# Y dimension of topography
set val(stop)   10.0                       ;# time of simulation end
#=================================
#       Initialization
#=================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open lab8.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open lab8.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel
#=================================
#     Mobile node parameter setup
#=================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
```

```
            -macType       $val(mac) \
            -ifqType       $val(ifq) \
            -ifqLen        $val(ifqlen) \
            -antType       $val(ant) \
            -propType      $val(prop) \
            -phyType       $val(netif) \
            -channel       $chan \
            -topoInstance  $topo \
            -agentTrace    ON \
            -routerTrace   ON \
            -macTrace      ON \
            -movementTrace ON
#=================================
#       Nodes Definition
#=================================
#Create 4 nodes
set n0 [$ns node]
$n0 set X_ 618
$n0 set Y_ 734
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 374
$n1 set Y_ 711
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 633
$n2 set Y_ 506
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 859
$n3 set Y_ 683
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20

#=================================
#       Agents Definition
#=================================
#Setup a TCP connection
```

```
set tcp0 [new Agent/TCP]
$ns attach-agent $n1 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500
#=================================
#        Applications Definition
#=================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 2.0 "$ftp0 stop"
#=================================
#        Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam lab8.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Awk Script (lab8.awk)

```
BEGIN{
a=0
b=0
c=0
```

```
}
{
if($1=="r"&&$3=="_3_"&&$4=="AGT"&&$7=="tcp"&&$8=="1540")
{
a++;
}
if($1=="s"&&$3=="_1_"&&$4=="AGT"&&$7=="tcp"&&$8=="1540")
{
b++;
}
if($1=="D"&&$3=="_0_"&&$4=="AGT"&&$7=="tcp"&&$8=="1540")
{
c++;
}
}
END{
printf("\n total number of packets drop: %d\n", c++);
printf("\n total number of packets sent: %d\n", b++);
printf("\n total number of packets received: %d\n", a++);
}
```

Steps for Execution

Step 1: Type the above program in text editor and save it with the filename and extension ".tcl"

For Example, lab8.tcl

- **tcl- Tool Command Language**

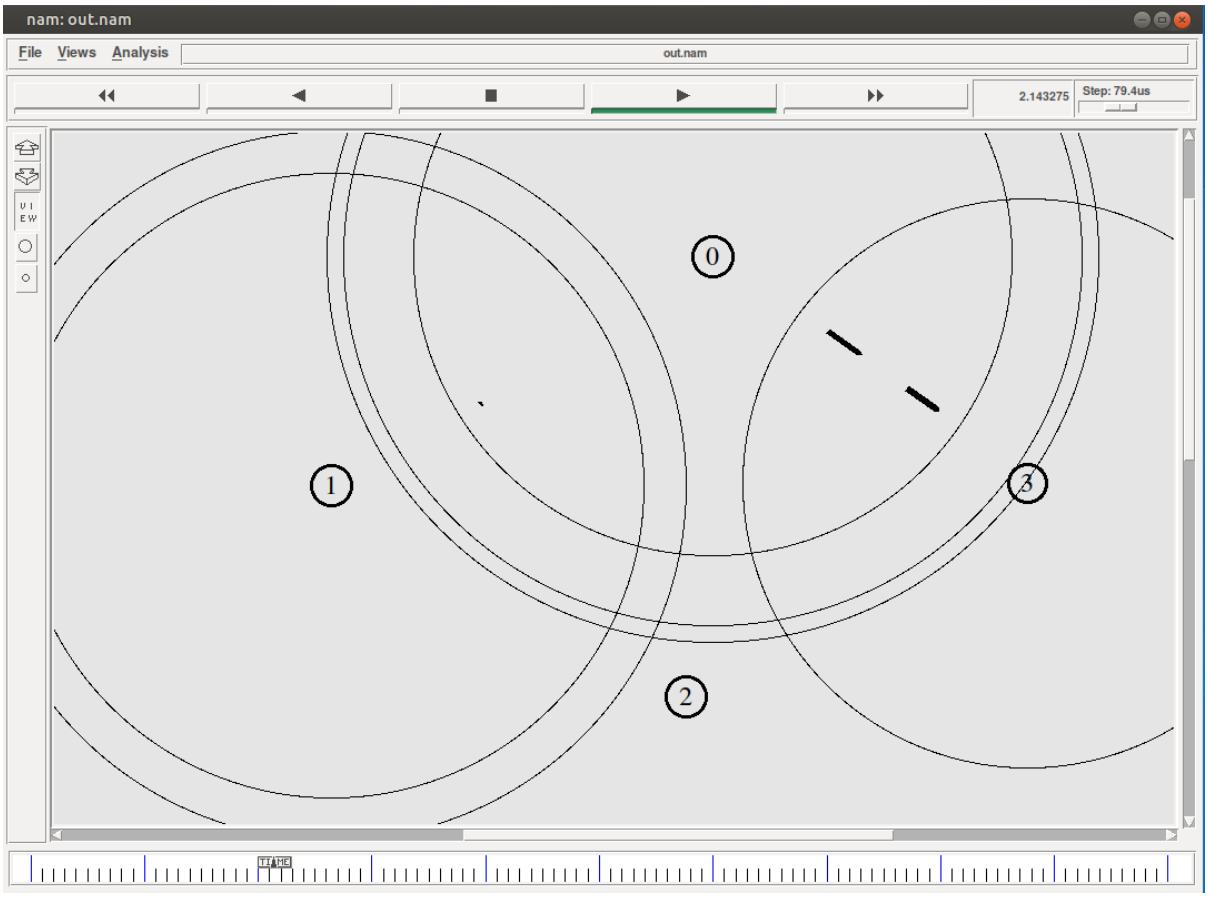
Step 2: Run the program in terminal window

[root@localhost ~]# ns lab8.tcl

- **ns-Network Simulator**

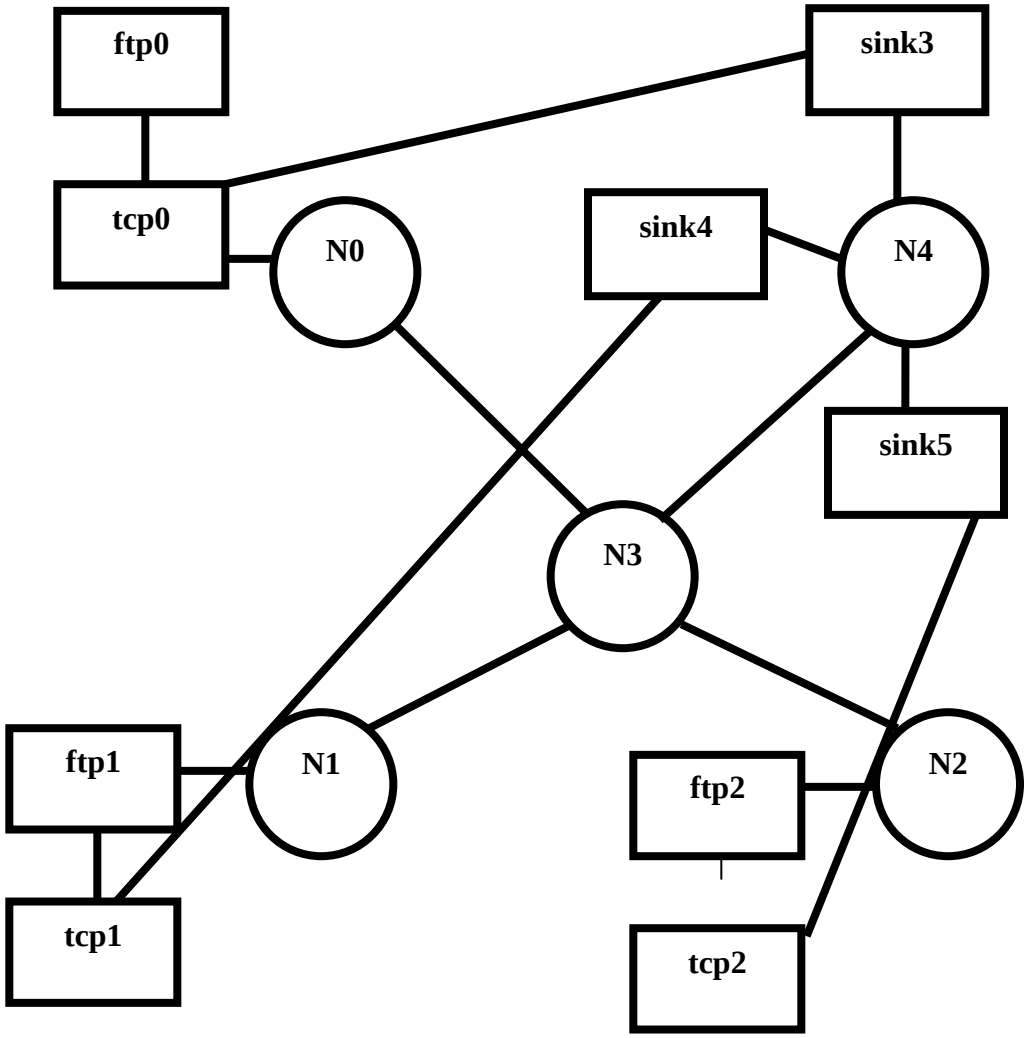
Step 3: Once the simulation is completed, run awk file to observe the output

[root@localhost ~]# awk -f lab8.awk lab8.tr




```
devasc@labvm: ~/ccnlab
File  Edit  View  Search  Terminal  Help
devasc@labvm:~$ cd ccnlab/
devasc@labvm:~/ccnlab$ awk -f lab8.awk lab8.tr

 total number of packets drop: 0

 total number of packets sent: 44

 total number of packets received: 44
devasc@labvm:~/ccnlab$
```

3.Simulate a network with star Topology (One router and several Hosts) Declare Applications (TCP or UDP) to send packets from hosts and to receiver (on One Host). Test the bandwidth and the Delay, When Buffers are of infinite capacities and buffers are limited Capacities.




```
#=================================
#     Simulation parameters setup
#=================================
set val(stop)   10.0                ;# time of simulation end
#=================================
#       Initialization
#=================================
#Create a ns simulator
set ns [new Simulator]
```

```
#Open the NS trace file
set tracefile [open lab9.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open lab9.nam w]
$ns namtrace-all $namfile

#=================================
#       Nodes Definition
#=================================
#Create 5 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
#=================================
#       Links Definition
#=================================
#Createlinks between nodes
$ns duplex-link $n0 $n3 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n3 50
$ns duplex-link $n1 $n3 100.0Mb 10ms DropTail
$ns queue-limit $n1 $n3 50
$ns duplex-link $n2 $n3 100.0Mb 10ms DropTail
$ns queue-limit $n2 $n3 50
$ns duplex-link $n3 $n4 100.0Mb 10ms DropTail
$ns queue-limit $n3 $n4 50
#Give node position (for NAM)
$ns duplex-link-op $n0 $n3 orient right-down
$ns duplex-link-op $n1 $n3 orient right-up
$ns duplex-link-op $n2 $n3 orient left-up
$ns duplex-link-op $n3 $n4 orient right-up
#=================================
#       Agents Definition
#=================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink3 [new Agent/TCPSink]
```

```
$ns attach-agent $n4 $sink3
$ns connect $tcp0 $sink3
$tcp0 set packetSize_ 1500

#Setup a TCP connection
set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set sink4 [new Agent/TCPSink]
$ns attach-agent $n4 $sink4
$ns connect $tcp1 $sink4
$tcp1 set packetSize_ 1500

#Setup a TCP connection
set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
set sink5 [new Agent/TCPSink]
$ns attach-agent $n4 $sink5
$ns connect $tcp2 $sink5
$tcp2 set packetSize_ 1500
#=================================
#       Applications Definition
#=================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 3.0 "$ftp0 stop"

#Setup a FTP Application over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 4.0 "$ftp1 start"
$ns at 7.0 "$ftp1 stop"

#Setup a FTP Application over TCP connection
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ns at 8.0 "$ftp2 start"
$ns at 9.0 "$ftp2 stop"
#=================================
#       Termination
```

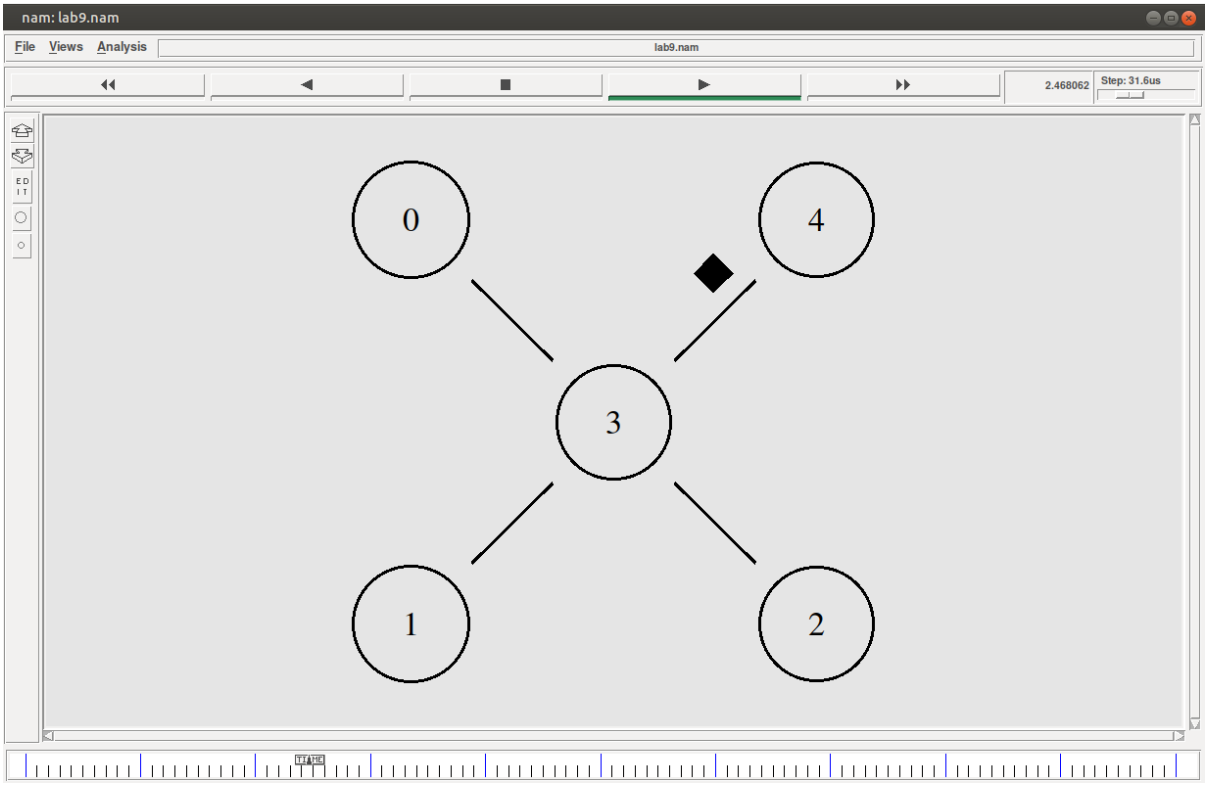
```
#=================================
#Define a 'finish' procedure
proc finish {} {
   global ns tracefile namfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec nam lab9.nam &
   exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Awk Script (lab9.awk)

```
BEGIN{
a=0
b=0
c=0
d=0
}
{
if($1=="r"&&$3=="0"&&$4=="3"&&$5=="tcp"&&$6=="1540")
{
a++;
}
if($1=="r"&&$3=="1"&&$4=="3"&&$5=="tcp"&&$6=="1540")
{
b++;
}
if($1=="r"&&$3=="2"&&$4=="3"&&$5=="tcp"&&$6=="1540")
{
c++;
}
if($1=="r"&&$3=="3"&&$4=="4"&&$5=="tcp"&&$6=="1540")
{
d++;
}
```

```
}
END{
printf("\n total number of packets received at Node 3 due to Node 0: %d\n", a++);
printf("\n total number of packets received at Node 3 due to Node 1: %d\n", b++);
printf("\n total number of packets received at Node 3 due to Node 2: %d\n", c++);
printf("\n total number of packets received at Node 4 due to host Nodes: %d\n", d++);
}
```

```
devasc@labvm: ~/ccnlab
File Edit View Search Terminal Help
devasc@labvm:~/ccnlab$ awk -f lab9.awk lab9.tr

 total number of packets received at Node 3 due to Node 0: 930

 total number of packets received at Node 3 due to Node 1: 1430

 total number of packets received at Node 3 due to Node 2: 430

 total number of packets received at Node 4 due to host Nodes: 2790
```



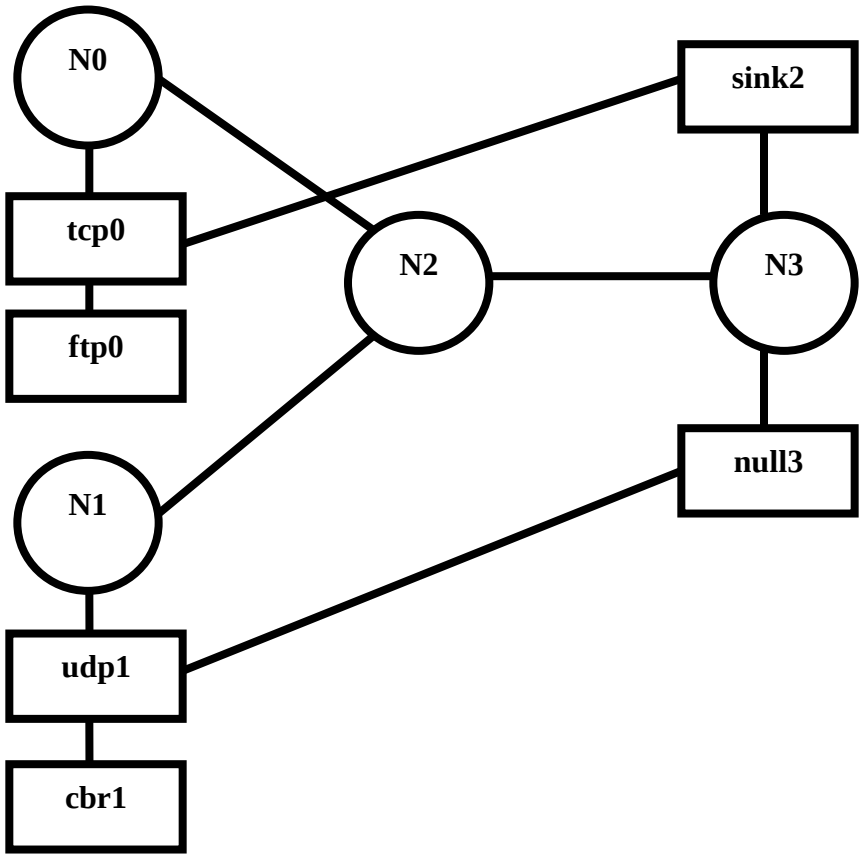
Simulation Result

Serial No.	Bandwidth(Mb)	Queue Size	Delay(ms)	Packet Received at Node 4
1	100	50	10	2790
2	500	30	30	810
3	700	10	50	390

4.Build a Four Node Point to Point Network with links N0-N2, N1-N2 and N2-N3.Connect a TCP link between N0-N3 and UDP link between N1-N3.

(i) Define BERs for Links. Compare TCP and UDP Protocols when errors occur.

(ii) Modify to Simulate a Link Failure between the Host and the Target Node. Compare TCP and UDP Protocols, when the Target Node is not accessible.



In this network topology there are two source nodes N0 and N1, they will be transmitting TCP and UDP packets to destination node N3.

```
#=================================
#     Simulation parameters setup
#=================================
set val(stop)   50.0                ;# time of simulation end

#=================================
#       Initialization
#=================================
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
set tracefile [open lab10.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open lab10.nam w]
$ns namtrace-all $namfile

#=================================
#       Nodes Definition
#=================================
#Create 4 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#=================================
#       Links Definition
#=================================
#Createlinks between nodes
$ns duplex-link $n0 $n2 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n2 50
$ns duplex-link $n1 $n2 100.0Mb 10ms DropTail
$ns queue-limit $n1 $n2 50
$ns duplex-link $n2 $n3 100.0Mb 10ms DropTail
$ns queue-limit $n2 $n3 50
```

```
#Give node position (for NAM)
$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

set err [new ErrorModel]
$ns lossmodel $err $n2 $n3
$err set rate_ 0.1

#==================================
#        Agents Definition
#==================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink2 [new Agent/TCPSink]
$ns attach-agent $n3 $sink2
$ns connect $tcp0 $sink2
$tcp0 set packetSize_ 50000

#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set null3 [new Agent/Null]
$ns attach-agent $n3 $null3
$ns connect $udp1 $null3
$udp1 set packetSize_ 50000

#==================================
#        Applications Definition
#==================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 20.0 "$ftp0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
```

```
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$ns at 25.0 "$cbr1 start"
$ns at 48.0 "$cbr1 stop"

#=================================
#       Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
   global ns tracefile namfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec nam lab10.nam &
   exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

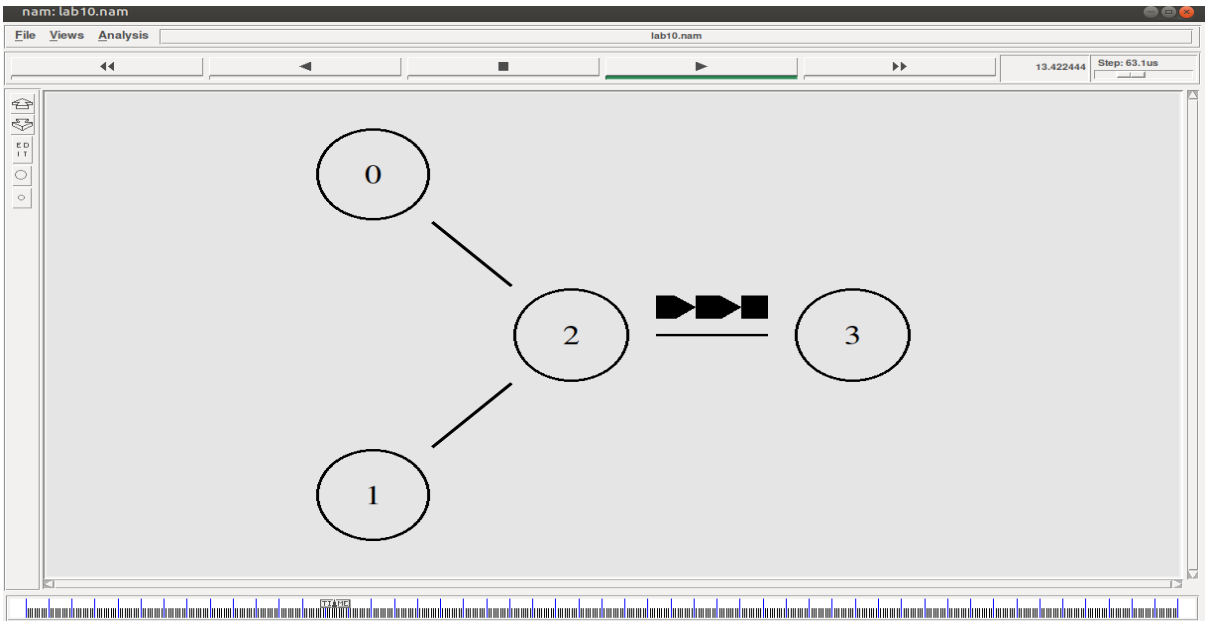
Awk Script (lab10.awk)

```
BEGIN{
a=0
b=0
}
{
if($1=="r"&&$3=="2"&&$4=="3"&&$5=="tcp"&&$6=="50040")
{
a++;
}
if($1=="r"&&$3=="2"&&$4=="3"&&$5=="cbr"&&$6=="1000")
{
b++;
}
}
END{
printf("\n total number of packets received at tcp: %d\n", a++);
printf("\n total number of packets received at udp: %d\n", b++);
```

}

Simulation Result

Serial No.	Error Rate	UDP Packet Received at Node 3	TCP Packet Received at Node 3
1	0.1	2566	423
2	0.01	2838	3077
3	0.001	2868	4479



```
File  Edit  View  Search  Terminal  Help
devasc@labvm:~/ccnlab$ awk -f lab10.awk lab10.tr

 total number of packets received at tcp: 423

 total number of packets received at udp: 2566
devasc@labvm:~/ccnlab$ ns lab10.tcl
devasc@labvm:~/ccnlab$ awk -f lab10.awk lab10.tr

 total number of packets received at tcp: 423

 total number of packets received at udp: 2566
devasc@labvm:~/ccnlab$ ns lab10.tcl
devasc@labvm:~/ccnlab$ awk -f lab10.awk lab10.tr

 total number of packets received at tcp: 3077

 total number of packets received at udp: 2838
devasc@labvm:~/ccnlab$ ns lab10.tcl
devasc@labvm:~/ccnlab$ awk -f lab10.awk lab10.tr

 total number of packets received at tcp: 4479

 total number of packets received at udp: 2868
devasc@labvm:~/ccnlab$
```

Socket-Server.c

```
#`include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>

int main(void)
{
  int listenfd = 0,connfd = 0;

  struct sockaddr_in serv_addr;

  char sendBuff[1025];
  int numrv;

  listenfd = socket(AF_INET, SOCK_STREAM, 0);
  printf("socket retrieve success\n");

  memset(&serv_addr, '0', sizeof(serv_addr));
  memset(sendBuff, '0', sizeof(sendBuff));
```

```
  serv_addr.sin_family = AF_INET;
  serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
  serv_addr.sin_port = htons(5000);

  bind(listenfd, (struct sockaddr*)&serv_addr,sizeof(serv_addr));

  if(listen(listenfd, 10) == -1){
     printf("Failed to listen\n");
     return -1;
  }

  while(1)
    {
     connfd = accept(listenfd, (struct sockaddr*)NULL ,NULL); // accept awaiting request

     strcpy(sendBuff, "Message from server");
     write(connfd, sendBuff, strlen(sendBuff));

     close(connfd);
     sleep(1);
    }

  return 0;
}
```

Socket-client.c

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>

int main(void)
{
  int sockfd = 0,n = 0;
  char recvBuff[1024];
  struct sockaddr_in serv_addr;

  memset(recvBuff, '0' ,sizeof(recvBuff));
  if((sockfd = socket(AF_INET, SOCK_STREAM, 0))< 0)
```

```
    {
      printf("\n Error : Could not create socket \n");
      return 1;
    }

  serv_addr.sin_family = AF_INET;
  serv_addr.sin_port = htons(5000);
  serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

  if(connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr))<0)
    {
      printf("\n Error : Connect Failed \n");
      return 1;
    }

  while((n = read(sockfd, recvBuff, sizeof(recvBuff)-1)) > 0)
    {
      recvBuff[n] = 0;
      if(fputs(recvBuff, stdout) == EOF)
    {
      printf("\n Error : Fputs error");
    }
      printf("\n");
    }

  if( n < 0)
    {
      printf("\n Read Error \n");
    }

  return 0;
}
```

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, are 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 RSA algorithm?
RSA is short for Rivest-Shamir-Adleman algorithm. It is the most commonly used public key encryption algorithm in use today.

99) 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.

REFERENCE

1. **Communication Networks: Fundamental Concepts and Key Architectures -** Alberto Leon, Garcia and IndraWidjaja, 3rd Edition, Tata McGraw- Hill, 2004.
2. **Data and Computer Communication,** William Stallings, 8thtEdition, Pearson Education, 2007.
3. **Computer Networks: A Systems Approach -** Larry L. Peterson and Bruce S. David, 4th Edition, Elsevier, 2007.
4. **Introduction to Data Communications and Networking** – Wayne Tomasi, Pearson Education, 2005.
5. **Communication Networks – Fundamental Concepts and Key architectures –** Alberto Leon-Garcia and IndraWidjaja:, 2rd Edition, Tata McGraw-Hill, 2004
6. **Computer and Communication Networks –** Nader F. Mir:, Pearson Education, 2007.