(Approved by AICTE | Affiliated to VTU | Recognized by UGC with 2(f) & 12(B) status | Accredited by NBA and NAAC)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

V SEMESTER

COMPUTER NETWORKS LAB – MVJ22CS52

ACADEMIC YEAR 2025–2026 (ODD)

LABORATORY MANUAL

NAME OF THE STUDENT : ----------

BRANCH : ----------

UNIVERSITY SEAT NO. : ----------

SEMESTER & SECTION : ----------

BATCH : ----------

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION:

To create an enriching learning environment where talented and industry – ready AI and ML professionals are nurtured. This is our promise.

MISSION:

- To develop skilled and knowledgeable professionals in the field of Artificial Intelligence and Machine Learning.
- To impart high – quality education and career – oriented learning programs.
- To contribute towards advanced AI technologies that provide increased and better performance.
- To benefit the society through our contribution towards advancements in AI and Machine Learning.

PROGRAMEDUCATIONALOBJECTIVES (PEOs):

PEO1: Current Industry Practices: Graduates will analyze real world problems and give solution using current industry practices in computing technology.

PEO2: Research and Higher Studies: Graduates with strong foundation in mathematics and engineering fundamentals that will enable graduates to pursue higher learning, R&D activities and consultancy.

PEO3: Social Responsibility: Graduates will be professionals with ethics, who will provide industry growth and social transformation as responsible citizens.

PROGRAMOUTCOMES(POs):

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK 1 to WK 4 respectively to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

PO3: Design/ Development of Solutions: Design creative solutions for complex engineering problems and design/ develop systems/ components/ processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering

problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2andWK6)

PO6: **The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

PO8: Individual and Collaborative Teamwork: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10: Project Management and Finance: Apply knowledge and understanding of engineering managementprinciplesandeconomicdecision-makingandapplythesetoone'sownwork, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for

i) Independent and life-long learning

ii) Adaptability to new and emerging technologies and

iii) Critical thinking in the broadest context of technological change. (WK8)

PROGRAMSPECIFICOUTCOMES (PSOs):

1. **PSO1: Programming**: Ability to understand, analyze and develop computer programs In the areas related to algorithms, system software, multimedia, web design, DBMS, and networking for efficient design of computer-based systems of varying complexity.
2. **PSO2: Practical Solution**: Ability to practically provide solutions for real world problems with a broad range of programming language and opensource platforms in various computing domains.
3. **PSO3: Research**: Ability to use innovative ideas to do research in various domains to solve societal problems.

Course objective is to:

This course will enable students to

1. Introduction to fundamental concepts and types of computer networks
2. Demonstrate the TCP/IP and OSI models with merits and demerits.
3. Understand the difference between all communication protocols.

Prerequisites:

1. Data Communication
2. Computer Networks

Course Outcomes(CO's):

CO No	CO's
CO1	Interpret the basics of Computer Networks and Various Protocols.
CO2	Generalize functionalities and services of each layer of OSI model.
CO.3	Explains the concept of data framing and error control mechanisms
CO4	Compares Different routing protocols
CO5	Identify the concepts of network security, Mobile and adhoc networks

LabExperiments:

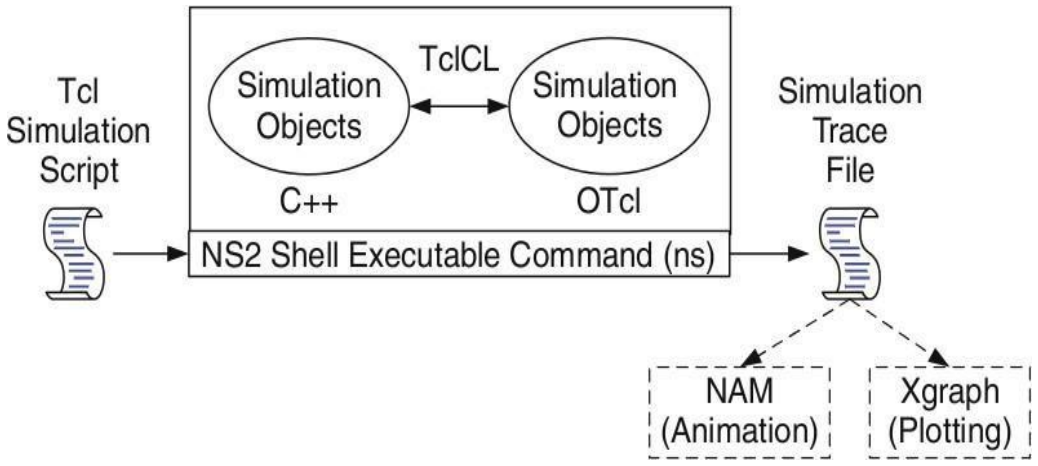
Sl. No.	Programs	RBTL
1	Learn to use commands like ping, tcpdump, netstat, ifconfig, nslookup and traceroute. Capture ping and traceroute PDUs using a network protocol Analyzer and examine.	**L3**
2	Write a program for error detecting code using CRC - CCITT(16-bits).	**L3**
3	Write a program to find the shortest pathbetweenverticesusing bellman-ford algorithm.	**L3**
4	ApplicationsusingTCPsockets like: a) Echoclientandechoserver b) Chat c) FileTransfer	**L3**
5	Simulation of DNS using UDP sockets.	**L3**
6	Write a code for simulating ARP/RARP protocols.	**L3**
7	Write a program for congestion control using leaky bucket algorithm.	**L3**
8	Implement three node point-to-point networks with duplex links between them. Set the queue size, vary the band width ,and find the Number of packets dropped.	**L3**
9	Simulate the transmission of ping messages/trace route over a network topologyconsistingof6nodesandfindthenumberofpacketsdropped due to congestion.	**L3**

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO	PSO
C207.1	3	3	3	2	1	-	-	-	-	-	-	-	1	3
C207.2	3	3	2	2	1	-	-	-	-	-	1	-	-	-
C207.3	3	3	2	2	1	-	-	-	-	-	1	-	3	-
C207.4	3	3	2	2	1	-	-	-	-	-	1	-	3	-
C207.5	3	2	2	2	1	-	-	-	-	-	-	-	1	3

IntroductiontoNS-2:

- WidelyknownasNS2,issimply anevent 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.
- Ingeneral,NS2providesuserswithawayofspecifyingsuchnetworkprotocolsandsimulating their corresponding behaviours.
-

BasicArchitectureofNS2



Tcl scripting

- Tcl is a general purpose scripting language.[Interpreter]
- Tcl run son 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 priort the usage.

Basics of TCL

Syntax: command arg1 arg2 arg3

- **Hello World!**

Putsstdout {Hello,World!}

Hello, World!

- **Variables** Command Substitution

seta5setlen[string length foo bar]

setb $ asetlen[expr[string length foo bar]+9]

- **Simple Arithmetic**

expr7.2 / 4

- **Procedures**

Proc Diag {ab}{

Set c [expr sqrt($a*$a+$b*$b)] return

$c }

puts“Diagonalofa3,4righttriangleis[Diag34]” Output:

Diagonal of a 3, 4 right triangle is 5.0

- **Loops**

while{$i<$n}{for{seti0}{$i<$n}{incri} {

. . .. . .

}}

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

NSS imulator Preliminaries.

1. Initialization and termination aspects of then simulator.
2. Definition of network nodes ,links ,queues and topology.
3. Definition of agents and of applications.
4. Then am visualization tool.
5. Tracing and random variables.

InitializationandTerminationofTCLScriptinNS-2

Anns simulation starts with the command

```
setns[new Simulator]
```

Whichisthusthefirstlineinthetclscript?Thislinedeclaresanewvariableasusingthesetcommand, you can call this variable as you wish, In general people declares it as ns because it is aninstance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new.

Inordertohaveoutputfileswithdataonthesimulation(tracefiles)orfiles used for visualization (nam files), we need to create the files using "open" command:

#Open the Trace file

```
settracefile1[openout.trw]
$nstrace-all$tracefile1
```

#OpentheNAMtracefile

```
setnamfile[openout.nam w]
$nsnamtrace-all$namfile
```

The above creates a dta trace file called "out.tr" and a nam visualization trace file called "out.nam".Withinthetclscript,thesefilesarenotcalledexplicitlybytheirnames,butinsteadbypointers thataredeclaredaboveandcalled"tracefile1"and"namfile"respectively.Remarkthattheybeginswith a#symbol.Thesecondlineopenthefile"out.tr"tobeusedforwriting,declaredwiththeletter"w".The third lineuses as imulator method called trace-all that have as parameter the name of the file where the traces will go.

ThelastlinetellsthesimulatortorecordallsimulationtracesinNAMinputformat.Italsogivesthefile 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.

#Definea‘finish’procedure

```
Procfinish{}{
globalnstracefile1namfile
$ns flush-trace
Close $tracefile1
Close$namfile
Execnamout.nam&
Exit0
```

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** willends the application andr eturnthenumber0 as status to the system. Zeroisthe default foracle anexit .Other values can be used to say that is a exit because something fails.

Attheendofnsprogramweshouldcalltheprocedure“finish”andspecifyatwhattimethetermination should occur. For example,

```
$nsat125.0“finish”
```

willbeusedtocall“**finish**”attime125sec.Indeed,the**at**methodofthesimulatorallowsustoschedule events explicitly.

The simulation can then begin using the command

```
$nsrun
```

Definition of a network of links and nodes

The way to define anodeis

```
setn0[$nsnode]
```

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

```
$nsduplex-link$n0$n210Mb10msDropTail
```

Which means that $n0 and $n2 are connected using a bi-directional link that has 10ms of propagation dela`y 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 (DeficitRoundRobin),thestochasticFairQueuing(SFQ)andtheCBQ(whichincludingapriorityand 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:

```
#setQueueSizeoflink(n0-n2)to20

$nsqueue-limit$n0$n220
```

Agents and Applications

We need tode finerouting (sources, destinations)the agents(protocols)the application that use them.

FTP over TCP

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

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

```
settcp[newAgent/TCP]
```

Thecommand**$nsattach-agent$n0$tcp**definesthesourcenodeofthetcpconnection. The command

```
setsink[newAgent/TCPSink]
```

Defines the behavior of the destination node of TCP and assigns toita pointer called sink.

#SetupaUDP connection

```
setudp[newAgent/UDP]
$nsattach-agent$n1$udp set
null [new Agent/Null]
$nsattach-agent$n5$null
$nsconnect$udp$null
$udpsetfid_2
```

#setup a CBR over UDP connection

The below shows the definition of a CBR application using a UDP agent Thecommand**$nsattach-agent$n4$sink**definesthedestinationnode.Thecommand**$nsconnect $tcp $sink** finallymakestheTCPconnectionbetweenthesourceanddestinationnodes.

```
set cbr [new
Application/Traffic/CBR]
$cbrattach-agent$udp
$cbrsetpacketsize_100
$cbrsetrate_0.01Mb
$cbrsetrandom_false
```

TCPhasmanyparameterswithinitialfixeddefaultsvaluesthatcanbechangedifmentionedexplicitly. For example,the default TCP packet size has a size of 1000bytes.Thiscanbechangedtoanothervalue, 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 aflowidentificationof"1".Weshalllatergivetheflow identification of "2"tothe UDP connection.

CBR over UDP

AUDP 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.

```
$cbrsetinterval_0.005
```

The packet size can beset to some value using

```
$cbrsetpacketSize_<packetsize>
```

Scheduling Events

NS is a discrete event based simulation. The tcp script defines when event should occur. The initializing command setns [new Simulator]creates an event scheduler, and events arethenscheduled using the format:

```
$nsat<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 an bed one through the following command

```
$nsat0.1"$cbr start"

$ns at1.0"$ftpstart"

$nsat124.0"$ftpstop"

$nsat124.5"$cbrstop"
```

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. These cond field gives the time at which the event occurs.
3. Gives the input node of the link at which the event occurs.
4. Gives the output node of the link at which the event occurs.
5. Gives the packet type (eg CBR or TCP)
6. Gives the packet size
7. Some flags
8. This is the flow id (fid) of IPv6 that a user can set for each flow at the input OTcl script one can further use this field for analys is purposes ;it is al soused 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 x graph 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 datasets 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 x graph window.

/-bg<color>(Background)

This specifies the background color of the x graph window.

/-fg<color>(Foreground)

This specifies the fore ground 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<unitname>(XunitText)

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

/-y<unitname>(YunitText)

This is the unit name for they-axis .Its default is "Y".

Awk- An Advanced

Awk is a programmable, pattern-matching ,and processing tool available in UNIX .It Works equally well with text and numbers.

Awk is not just a command, but a programming language too. In other words, awk utility is a pattern scanning and processing language. It searches one or more files tosee 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:

```
awkoption'selection_criteria{action}'file(s)
```

Here, selection criteria filter input and select lines for the action component to act upon. The selection criteria are 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

Awkallows the user touse variables of there choice .You can now print aserial 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 INA FILE

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

$cat empawk .awk

Observe that this time we haven't used quotes to enclose the awk program. You can now use awk with the –f *filename* option to obtain the same output:

```
Awk–F"|"–fempawk.awkempn.lst
```

THEBEGINANDENDSECTIONS

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-INVARIABLES

Awkhasseveralbuilt-invariables.Theyareallassignedautomatically,thoughitisalsopossible 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.FSredefinesthisfieldseparator,whichinthesampledatabasehappenstobethe|.Whenused atall,itmustoccurintheBEGINsectionsothatthebodyoftheprogramknowsitsvaluebeforeitstarts processing:

BEGIN {FS="|"}

This is anal ternative to the–F option which does the same thing.

TheOFSVariable*:*whenyouusedtheprintstatementwithcomma-separatedarguments,eachargument wasseparatedfromtheotherbyaspace.Thisisawk'sdefaultoutputfieldseparator,andcanreassigned 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"RecordNo",NR,"has", "fields"}'empx.lst

Experiment No:1 Learn to use commands like tcp dump, netstat, ifconfig, ns lookup and traceroute.

AIM: To Learn to use commands like tcp dump, netstat, ifconfig, ns lookup and traceroute ping.

PRE LAB DISCUSSION:
$]ping

```
C:\Users\ramakalyani>ping

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
            [-r count] [-s count] [[-j host-list] | [-k host-list]]
            [-w timeout] [-R] [-S srcaddr] [-c compartment] [-p]
            [-4] [-6] target_name

Options:
    -t             Ping the specified host until stopped.
                   To see statistics and continue - type Control-Break;
                   To stop - type Control-C.
    -a             Resolve addresses to hostnames.
    -n count       Number of echo requests to send.
    -l size        Send buffer size.
    -f             Set Don't Fragment flag in packet (IPv4-only).
    -i TTL         Time To Live.
    -v TOS         Type Of Service (IPv4-only. This setting has been deprecated
                   and has no effect on the type of service field in the IP
                   Header).
    -r count       Record route for count hops (IPv4-only).
    -s count       Timestamp for count hops (IPv4-only).
```

Tcp dump: The tcp dump utility allows you to capture packets that flow within your network to assist in network troubleshooting. The following are several examples of using tcp dump with different options. Traffic is captured based on a specified filter.

The tcp dump utility runs on the Linux command line. Tcp dump is a simple application that works well in Linux servers without Linux-based network devices, a GUI or various IoT nodes. These attributes give tcpdump an advantage over more powerful GUI-based analyzers, like Wireshark. Tcp dump is also scriptable, which means it can enable scheduled captures.

How to install tcp dump

Many Linux distributions include tcp dump, especially those used for enterprise networking. If your favorite distribution doesn't have it, use your preferred package manager to install it.

On Red Hat and similar distributions, type the following command:

```
# dnf install tcp dump
```

On Debian and similar distributions, type the following command:

```
# apt install tcp dump
```

MacOS users can manage tcp dump with the following brew command:

```
# brew install tcp dump
```

Because tcp dump is open source, you can also compile it yourself. Check the official website for more information and documentation.

Tcp dump is ready to use immediately after installation.

How to use tcp dump

Tcp dump contains many options and customizations that can help you find exactly what you want. Remember to check the main page for explanations and examples.

Tcp dump is a packet sniffing and packet analyzing tool for a System Administrator to troubleshoot connectivity issues in Linux. It is used to capture, filter, and analyze network traffic such as TCP/IP packets going through your system. It is many times used as a security tool as well. It saves the captured information in a p cap file, these p cap files can then be opened through Wireshark or through the command tool itself.

Installing tcp dump tool in Linux

Many Operating Systems have tcp dump command pre-installed but to install it, use the following commands. **For RedHat based linux OS**

```
yum install tcp dump
```

For Ubuntu/Debian OS

```
apt install tcp dump
```

Working with tcp dump command

1. To capture the packets of current network interface

```
sudo tcp dump
```

This will capture the packets from the current interface of the network through which the system is connected to the internet. **2.** To capture packets from a specific network interface

```
sudo tcp dump -i wlo1
```

This command will now capture the packets from wlo1 network interface. **3.** To capture specific number of packets

```
sudo tcp dump -c 4 -i wlo1
```

This command will capture only 4 packets from the wlo1 interface. **4.** To print captured packets in ASCII format

```
sudo tcp dump -A -i wlo1
```

This command will now print the captured packets from wlo1 to ASCII value. **5.** To display all available interfaces

```
sudo tcp dump -D
```

This command will display all the interfaces that are available in the system. **6.** To display packets in HEX and ASCII values

```
sudo tcp dump -XX -i wlo1
```

This command will now print the packets captured from the wlo1 interface in the HEX and ASCII values. **7.** To save captured packets into a file

```
sudo tcp dump -w captured_packets.pcap -i wlo1
```

This command will now output all the captures packets in a file named as captured_packets.pcap. **8.** To read captured packets from a file

```
sudo tcp dump -r captured_packets.pcap
```

This command will now read the captured packets from the captured_packets.pcap file. **9.** To capture packets with ip address

```
sudo tcpdump -n -i wlo1
```

---

his command will now capture the packets with IP addresses. **10.** To capture only TCP packets

```
sudo tcp dump -i wlo1 tcp
```

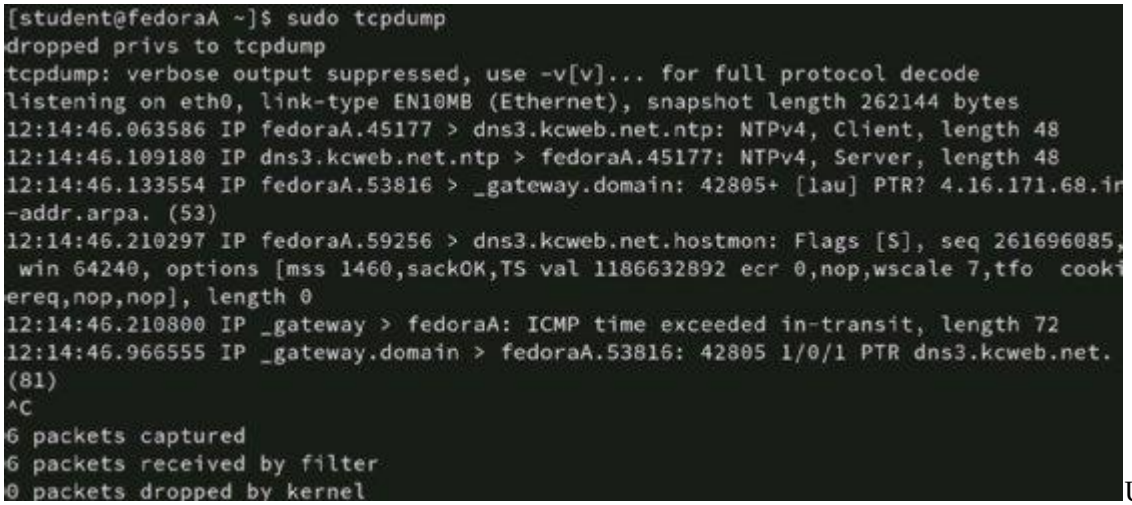
1. Start a capture

To get started with tcp dump, type the following command in the Linux terminal:

```
# tcp dump
```

Note that you may need sudo privileges.

Tcp dump displays captured packets in real time. This is useful if you know what to look for and if there's not a lot of traffic on the interface. However, it's much more likely that your screen quickly scrolls with nearly incomprehensible information.



Use real-time capture with tcp dump.

Interrupt the capture with Ctrl+C when this occurs.

You need a way to display only the information that is useful for your given task. The next sections explore options on how to focus tcp dump on what you need.

2. Select an interface

One way to narrow the capture is to specify the local network interface on which the analyzer captures. For example, captures are possible on a laptop's wired and wireless interfaces. Even more interfaces might be on network servers with multiple network interface cards or routers connected to several subnets.

Use the -i option to select the interface. To display the available interfaces, type tcpdump -D.

 Display available interfaces with the -D option.

Once you've identified the interface you want to use, type its name after the -i option:

```
# tcpdump -i eth0
```

This filter helps prevent data coming in overwhelming amounts from the capture results. However, you probably want to filter the results even more.

3. Select host information

You probably have a good idea of what to look for in troubleshooting or penetration testing scenarios. You also likely know where the packets you need come from or go. Specify the source or destination IP addresses you want tcpdump to watch for with the following flags.

Flag	Explanation
Host	Any packets with this host in the source or destination fields.
Src	Any packets with this host in the source field.
Dst	Any packets with this host in the destination field.
src and dst	Any packets with this host in both the source and destination fields.
src or dst	Any packets with this host either in the source field or destination field.

To capture packets from a specific host, type the following command:

```
# tcpdump -i eth0 host 10.1.1.42
```

If you want traffic that originates only from 10.1.1.42, type the following command:

```
# tcpdump -i eth0 src host 10.1.1.42
```

Develop more complex capture parameters with the and or or operators.

4. Filter by port number

You might be more interested in a type of traffic than the hosts involved. In that case, use a filter based on port numbers. If you need Simple Mail Transfer Protocol traffic, type the following command:

```
# tcpdump -i eth0 dst port 25
```

If you want to find insecure web traffic, type the following command:

```
# tcpdump -i eth0 dst port 80
```

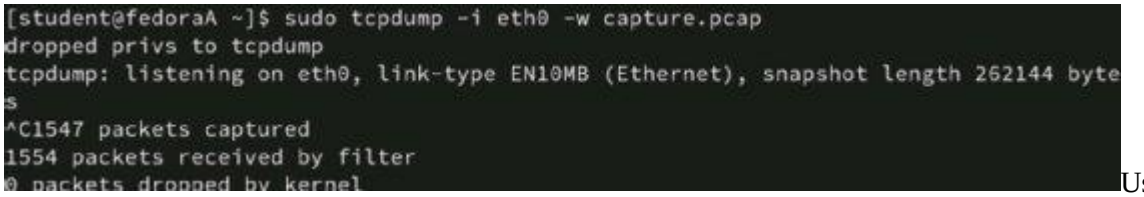
5. Write the capture to a file

One disadvantage of the examples above is tcp dump displays all results on the screen. They might roll by too quickly to analyze or detect patterns. It's usually better to write the capture results to a file instead.

Use the -w option with a file name to specify a destination.

```
# tcpdump -i eth0 -w capture.pcap
```

Be sure to use the .pcap file extension. The capture results are not usable as a text file. In addition, Wireshark can open the tcpdump file if it has the .pcap extension.



Use the -w option and a file name to write the capture to a file.

View capture results

Now that you have a capture file to work with, you can display the results in two ways: with tcpdump or Wireshark. Tcpdump itself can read the file, but you might find it advantageous to use Wireshark.

View the capture file with tcpdump

To view the file with tcpdump, type the following command:

```
# tcpdump -r capture.pcap
```

```
[student@fedoraA ~]$ sudo tcpdump -r capture.pcap
reading from file capture.pcap, link-type EN10MB (Ethernet), snapshot length 262144
dropped privs to tcpdump
12:17:29.297048 IP fedoraA.60740 > mirror.math.princeton.edu.http: Flags [.], ack 41
65574085, win 21670, options [nop,nop,TS val 911386650 ecr 1882953590], length 0
12:17:29.307193 IP mirror.math.princeton.edu.http > fedoraA.60740: Flags [.], seq 1:
65161, ack 0, win 95, options [nop,nop,TS val 1882953605 ecr 911386551], length 6516
0: HTTP
```

Netstat : Netstat is a common command line TCP/IP networking available in most versions of Windows, Linux, UNIX and other operating systems. Netstat provides information and statistics about protocols in use and current TCP/IP network connection

/*Output of netstat*/

Command Prompt

```
C:\Users\ramakalyani>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    172.16.25.72:54300     sf-in-f188:5228        ESTABLISHED
  TCP    172.16.25.72:54303     4.213.25.240:https     ESTABLISHED
  TCP    172.16.25.72:54339     ec2-54-148-53-243:https  ESTABLISHED
  TCP    172.16.25.72:54581     156.227.14.43:https    CLOSE_WAIT
  TCP    172.16.25.72:54583     ec2-52-66-178-255:https  CLOSE_WAIT
  TCP    172.16.25.72:54585     ec2-52-38-134-115:https  CLOSE_WAIT
  TCP    172.16.25.72:54589     156.227.14.43:https    CLOSE_WAIT
  TCP    172.16.25.72:54624     156.227.14.43:https    CLOSE_WAIT
  TCP    172.16.25.72:54648     156.227.14.43:https    CLOSE_WAIT
  TCP    172.16.25.72:54684     whatsapp-chatd-edge-shv-01-bom1:https  ESTABLISHED
  TCP    172.16.25.72:55250     40.104.77.66:https     ESTABLISHED
  TCP    172.16.25.72:55277     52.111.232.51:https    ESTABLISHED
  TCP    172.16.25.72:55537     s3-r-w:https           CLOSE_WAIT
  TCP    172.16.25.72:55715     ec2-52-66-178-255:https  CLOSE_WAIT
  TCP    172.16.25.72:55744     ec2-3-108-149-0:https  CLOSE_WAIT
  TCP    172.16.25.72:55928     ec2-15-207-180-173:https  CLOSE_WAIT
  TCP    172.16.25.72:55929     ec2-15-207-180-173:https  CLOSE_WAIT
  TCP    172.16.25.72:55930     ec2-15-207-180-173:https  CLOSE_WAIT
  TCP    172.16.25.72:55983     4.213.25.241:https     ESTABLISHED
  TCP    172.16.25.72:55988     pnmaaa-ao-in-f14:https  ESTABLISHED
  TCP    172.16.25.72:55997     a23-193-165-89:https   TIME_WAIT
  TCP    172.16.25.72:55998     a23-193-165-89:https   TIME_WAIT
  TCP    172.16.25.72:55999     40.126.18.33:https     TIME_WAIT
  TCP    172.16.25.72:56000     a23-193-165-89:https   TIME_WAIT
```

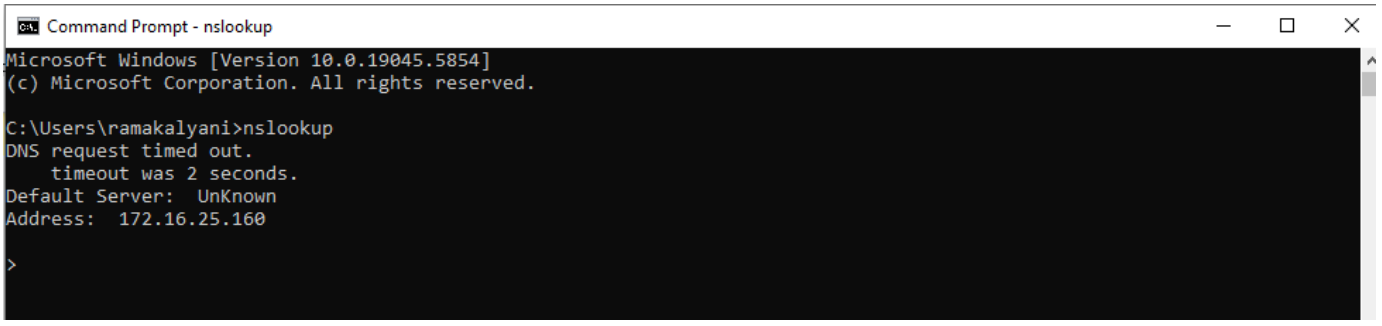
Ipconfig: ipconfig is a console application designed to run from the Windows command prompt. This utility allows you to get the IP address information of a Windows computer.

From the command prompt, type ipconfig to run the utility with default options. The output of the default command contains the IP address, network mask, and gateway for all physical and virtual network adapter.

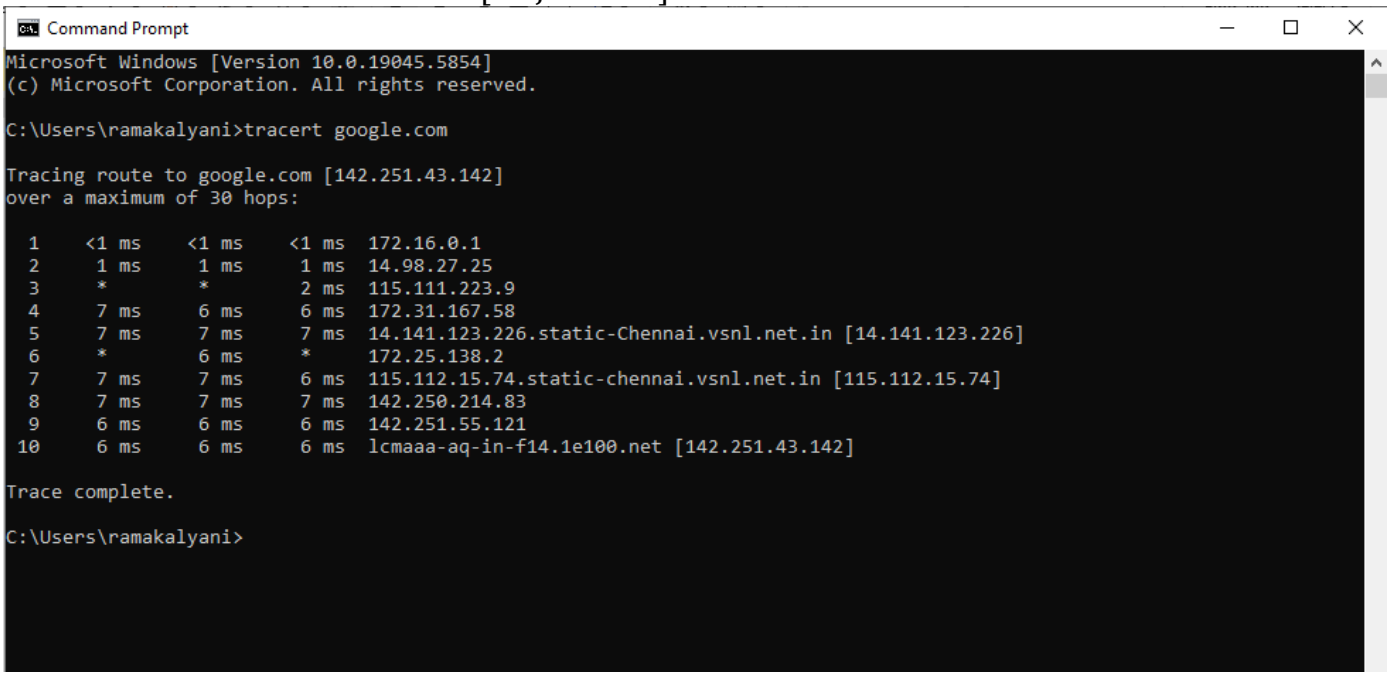
```
C:\Users\ramakalyani>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::fc2f:2d0c:5793:76ec%11
   IPv4 Address. . . . . . . . . . . : 172.16.25.72
   Subnet Mask . . . . . . . . . . . : 255.255.224.0
   Default Gateway . . . . . . . . . : 172.16.0.1

Ethernet adapter Local Area Connection* 9:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

Ns lookup :

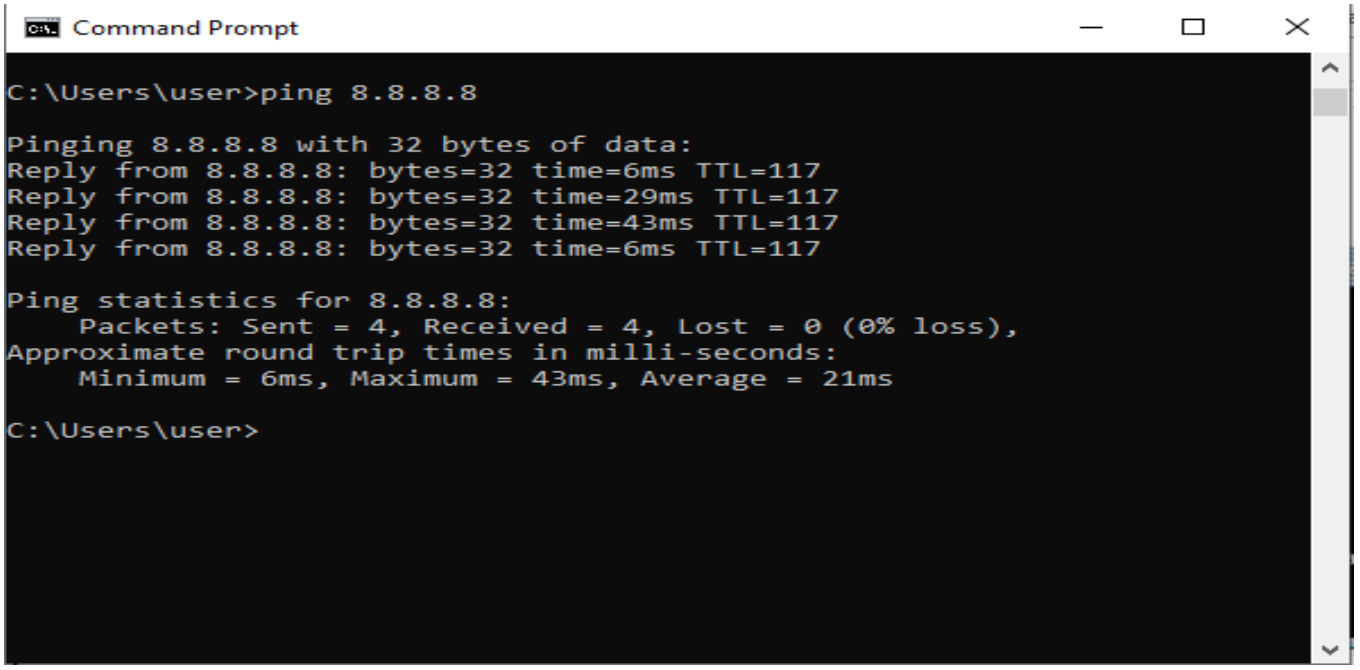
The ns lookup(which stands for name server lookup) command is a network utility program used to obtain information about internet servers. It finds name server information for domains by querying the Domain Name System.



Trace route: Traceroute is a network diagnostic tool used to track the pathway taken by a packet on an IP network from source to destination. Traceroute also records the time taken for each hop the packet makes during its route to the destination



Ping



Display traffic between 2 hosts: To display all traffic between two hosts (represented by variables host1 and host2):
tcpdump host host1 and host2

Display traffic from a source or destination host only:
To display traffic from only a source (src) or destination (dst) host:
tcpdumpsrc host
tcpdumpdst host

Display traffic for a specific protocol :
Provide the protocol as an argument to display only traffic for a specific protocol, for example tcp, udp, icmp, arp tcpdump protocol
For example to display traffic only for the tcp traffic :
tcpdumptcp

Filtering based on source or destination port :

To filter based on a source or destination port:
tcpdumpsrc port ftp
tcpdumpdst port http

2.Netstat:

Netstat is a common command line TCP/IP networking available in most versions of Windows, Linux, UNIX and other operating systems. Netstat provides information and statistics about protocols in use and current TCP/IP network connections. The Windows help screen (analogous to a Linux or UNIX for netstat reads as follows:
Displays protocol statistics and current TCP/IP network connections.
#netstat

3. ipconfig:

In Windows, ipconfig is a console application designed to run from the Windows command prompt. This utility allows you to get the IP address information of a Windows computer.
Using ipconfig : From the command prompt, type ipconfig to run the utility with default options. The output of the default command contains the IP address, network mask, and gateway for all physical and virtual network adapter.
#ipconfig

4.nslookup: The nslookup(which stands for name server lookup) command is a network utility program used to obtain information about internet servers. It finds name server information for domains by querying the Domain Name System. The nslookup command is a powerful tool for diagnosing DNS problems. You know you're experiencing a DNS problem when you can access a resource by specifying its IP address but not its DNS name.
#nslookup

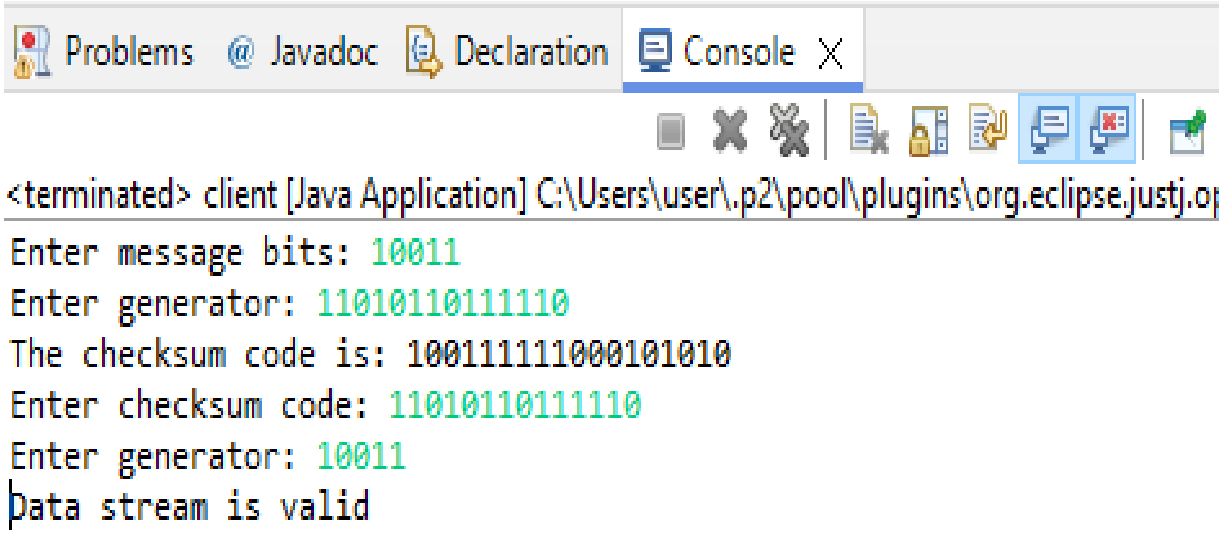
5.Trace route: Traceroute uses Internet Control Message Protocol (ICMP) echo packets with variable time to live (TTL) values. The response time of each hop is calculated. To guarantee accuracy, each hop is queried multiple times (usually three times) to better measure the response of that particular hop.
Traceroute is a network diagnostic tool used to track the pathway taken by a packet on an IP network from source to destination. Traceroute also records the time taken for each hop the packet makes during its route to the destination. Traceroute uses Internet Control Message Protocol (ICMP) echo packets with variable time to live (TTL) values. The response time of each hop is calculated. To guarantee accuracy, each hop is queried multiple times (usually three times) to better measure the response of that particular hop. Traceroute sends packets with TTL values that gradually increase from packet to packet, starting with TTL value of one. Routers decrement TTL values of packets by one when routing and discard packets whose TTL value has reached zero, returning the ICMP error message ICMP Time Exceeded.

Experiment 2: WRITE A PROGRAM FOR ERROR DETECTING CODE USING CRC-CCITT (16- BITS).

```
import java.util.Scanner;
import java.io.*;
public class Labman {
public static void main(String args[]) {
Scanner sc = new Scanner(System.in);
//Input Data Stream
System.out.print("Enter message bits: ");
String message = sc.nextLine();
System.out.print("Enter generator: ");
String generator = sc.nextLine();
int data[] = new int[message.length() + generator.length() - 1];
int divisor[] = new int[generator.length()];
for(int i=0;i<message.length();i++)
data[i] = Integer.parseInt(message.charAt(i)+"");
for(int i=0;i<generator.length();i++)
divisor[i] = Integer.parseInt(generator.charAt(i)+"");
//Calculation of CRC
for(int i=0;i<message.length();i++)
{
        if(data[i]==1)
                for(int j=0;j<divisor.length;j++)
                data[i+j] ^= divisor[j];
                }
                //Display CRC
                System.out.print("The checksum code is: ");
                for(int i=0;i<message.length();i++)
                data[i] = Integer.parseInt(message.charAt(i)+"");
                for(int i=0;i<data.length;i++)
                System.out.print(data[i]);
                System.out.println();
                //Check for input CRC code
                System.out.print("Enter checksum code: ");
                message = sc.nextLine();
                System.out.print("Enter generator: ");
                generator = sc.nextLine();
                data = new int[message.length() + generator.length() - 1];
                divisor = new int[generator.length()];
                for(int i=0;i<message.length();i++)
                data[i] = Integer.parseInt(message.charAt(i)+"");
                for(int i=0;i<generator.length();i++)
                divisor[i] = Integer.parseInt(generator.charAt(i)+"");
                //Calculation of remainder
                for(int i=0;i<message.length();i++) {
                if(data[i]==1)
                for(int j=0;j<divisor.length;j++)
                data[i+j] ^= divisor[j];
                }
                //Display validity of data
                boolean valid = true;
                for(int i=0;i<data.length;i++)
                if(data[i]==1){
                valid = false;
                break;
                }
```

```
if(valid==true)
                System.out.println("Data stream is valid");


else

System.out.println("Data stream is invalid. CRC error occurred.");
}
}
```

Output:



ExperimentNo:3

Aim: Write a program to find the shortest path between vertices using bellman-ford algorithm.

This is a java program to find shortest path from a single vertex. The Bellman–Ford algorithm is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph.[1] It iss lower than Dijkstra's algorithm for the same problem, but more versatile ,as it is capable of handling graphs in which some of the edge weights are negative numbers.

Here is the source code of the Java Program to Use the Bellman-Ford Algorithm to Find the Shortest Path Between Two Vertices Assuming that Negative Size Edges Exist in the Graph. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

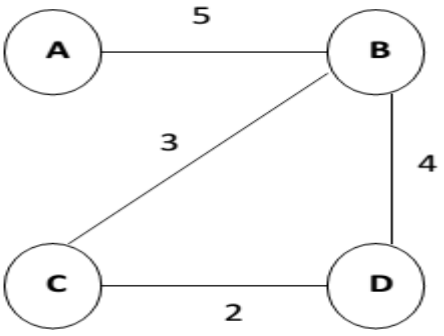
Program

```
importjava.util.Scanner;
 public class ford
 {
  private int D[];
  privateintnum_ver;
  publicstaticfinalintMAX_VALUE=999;
   public ford(int num_ver)
   {
this.num_ver=num_ver;
    D=newint[num_ver+1];
   }

  publicvoidBellmanFordEvaluation(intsource,int A[][])
  {
   for(int node =1; node<=num_ver; node++)
    {
       D[node]=MAX_VALUE;
    }

  D[source]=0;

  for(int node =1; node <=num_ver-1; node++)
    {
     for(int sn =1;sn<=num_ver; sn++)
     {
      for(int dn =1; dn<=num_ver; dn++)
        {
 if(A[sn][dn]!=MAX_VALUE)
 {
 if(D[dn] >D[sn]+A[sn][dn])
```

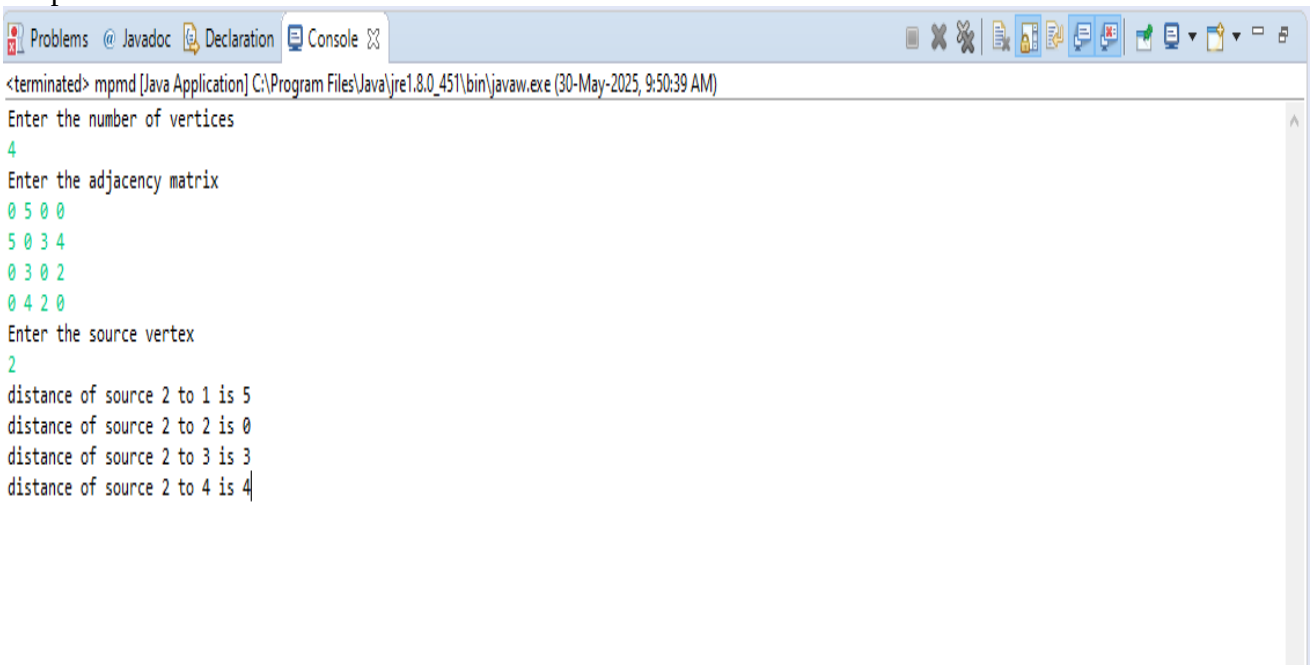
```
 D[dn]=D[sn]+ A[sn][dn];
 }
          }
        }
     }


  for(int sn = 1; sn<=num_ver; sn++)
  {
    for(int dn =1; dn<=num_ver; dn++)
     {
     if(A[sn][dn]!=MAX_VALUE)
     {
 if(D[dn] >D[sn]+A[sn][dn])
System.out.println("TheGraphcontainsnegative egdecycle");
 }
}
 }
  for(intvertex=1;vertex<=num_ver; vertex++)
  {
System.out.println("distanceofsource"+source+"to"+vertex+"is"+D[vertex]);
   }
}

 publicstatic void main(String[ ]args)
 {
    intnum_ver=0;
    int source;
    Scanner scanner = new Scanner(System.in);
System.out.println("Enterthenumberofvertices");
num_ver = scanner.nextInt();

     intA[][]=newint[num_ver+1][num_ver+1];
System.out.println("Enter the adjacency matrix");
   for(int sn = 1; sn<=num_ver; sn++)
   {
    for(int dn =1; dn<=num_ver; dn++)
   {
A[sn][dn]=scanner.nextInt(); if
(sn == dn)
{
          A[sn][dn]=0;
          continue;
     }
if(A[sn][dn]==0)
{
```

```
                A[sn][dn]=MAX_VALUE;
            }
        }
            }

        System.out.println("Enterthesourcevertex");
                source = scanner.nextInt();
               ford b = new ford (num_ver);
        b.BellmanFordEvaluation(source,A);
        scanner.close();
            }
         }
```

Input graph:



Output:



ExperimentNo4

Applications using TCP sockets like:
a) Echo client and echo server
b) Chat
c) File Transfer

Echo client and echo server

ALGORITHM

Client
1. Start
2. Create the TCP socket
3. Establish connection with the server
4. Get the message to be echoed from the user
5. Send the message to the server
6. Receive the message echoed by the server
7. Display the message received from the server
8. Terminate the connection
9. Stop

Server
1. Start
2. Create TCP socket, make it a listening socket
3. Accept the connection request sent by the client for connection establishment
4. Receive the message sent by the client
5. Display the received message
6. Send the received message to the client from which it receives
7. Close the connection when client initiates termination and server becomes a listening server, waiting for clients.
8. Stop.

```
import java.net.*;
import java.io.*;
publicclassEServer
{
publicstaticvoidmain(Stringargs[])
{
ServerSockets=null;
String line;
DataInputStream is;
PrintStreamps;
Socket c=null;
try
{
s=newServerSocket(9000);
```

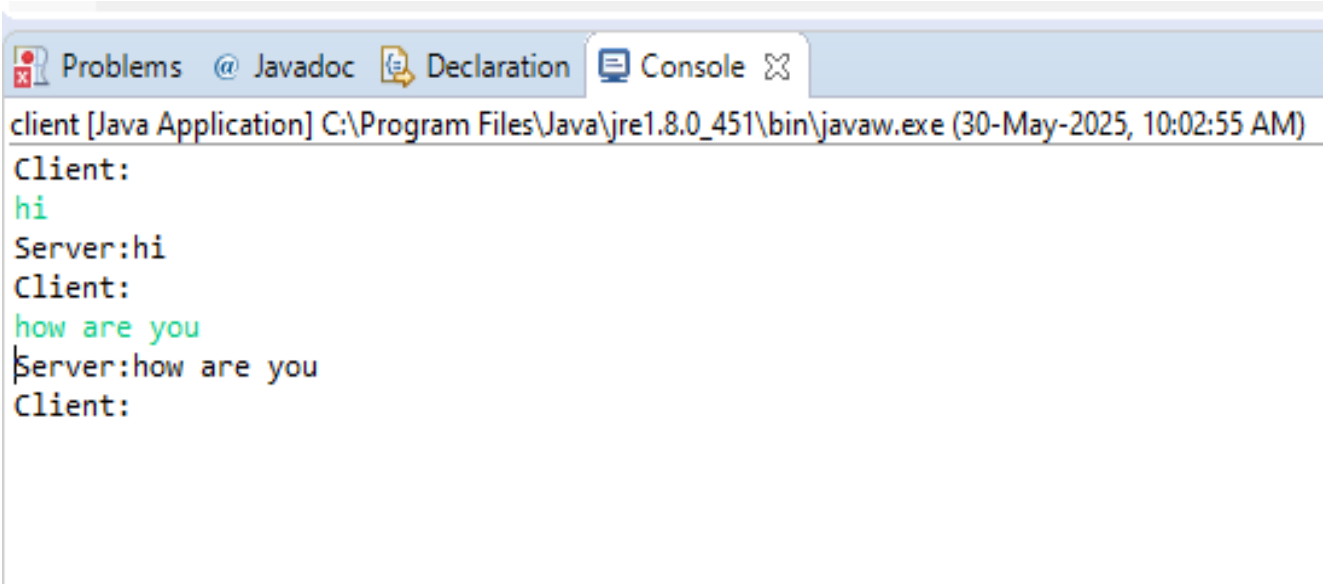
```
}
catch(IOExceptione)
{
System.out.println(e);
}
try
{
c=s.accept();
is=newDataInputStream(c.getInputStream());
ps=new PrintStream(c.getOutputStream());
while(true)
{
line=is.readLine();
ps.println(line);
}
}
catch(IOExceptione)
{
System.out.println(e);
}
}
}
```

EClient.java

```
import java.net.*;
import java.io.*;
publicclassEClient
{
publicstaticvoidmain(String arg[])
{
Socketc=null;
String line;
DataInputStreamis,is1;
PrintStreamos;
try
{
InetAddressia=InetAddress.getLocalHost();
c=new Socket(ia,9000);
}
catch(IOExceptione)
{
System.out.println(e);
}
try
{
os=newPrintStream(c.getOutputStream());
is=new DataInputStream(System.in);
is1=newDataInputStream(c.getInputStream());
```

```
while(true)
{
System.out.println("Client:");
line=is.readLine();
os.println(line);
System.out.println("Server:"+is1.readLine());
}}
catch(IOExceptione)
{
System.out.println("SocketClosed!");

}
}
```

OUTPUT



4b. Chat

ALGORITHM

Client

1. Start
2. Create the UDP datagram socket
3. Get the request message to be sent from the user
4. Send the request message to the server
5. If there quest message is “END” go to step 10
6. Wait for the reply message from the server
7. Receive the reply messages sent by the server
8. Display the reply message received from the server
9. Repeat the steps from3 to 8
10. Stop

Server

1. Start
2. Create UDP datagram socket, make it a listening socket
3. Receive there quest message sent by the client
4. If the received message is “END” go to step 10
5. Retrieve the client’s IP address from there quest message received
6. Display the received message
7. Get the reply message from the user
8. Send the reply message to the client
9. Repeat the steps from3 to 8.
10. Stop.

SERVER:

```
importjava.net.*;

import java.io.*;

importjava.util.*;


publicclassUDPServer

{

publicstaticvoidmain(Stringargs[])throwsSocketException,IOException

{

DatagramSocketserversocket=newDatagramSocket(9902);


Scannersc=newScanner(System.in);

while(true)

{

byte[]receivebuffer=newbyte[1024];

byte[] sendbuffer=new byte[1024];

DatagramPacketrecvdpkt=newDatagramPacket(receivebuffer,receivebuffer.length);

serversocket.receive(recvdpkt);

InetAddressip=recvdpkt.getAddress();

int portno=recvdpkt.getPort();

Stringclientdata=newString(recvdpkt.getData());
```

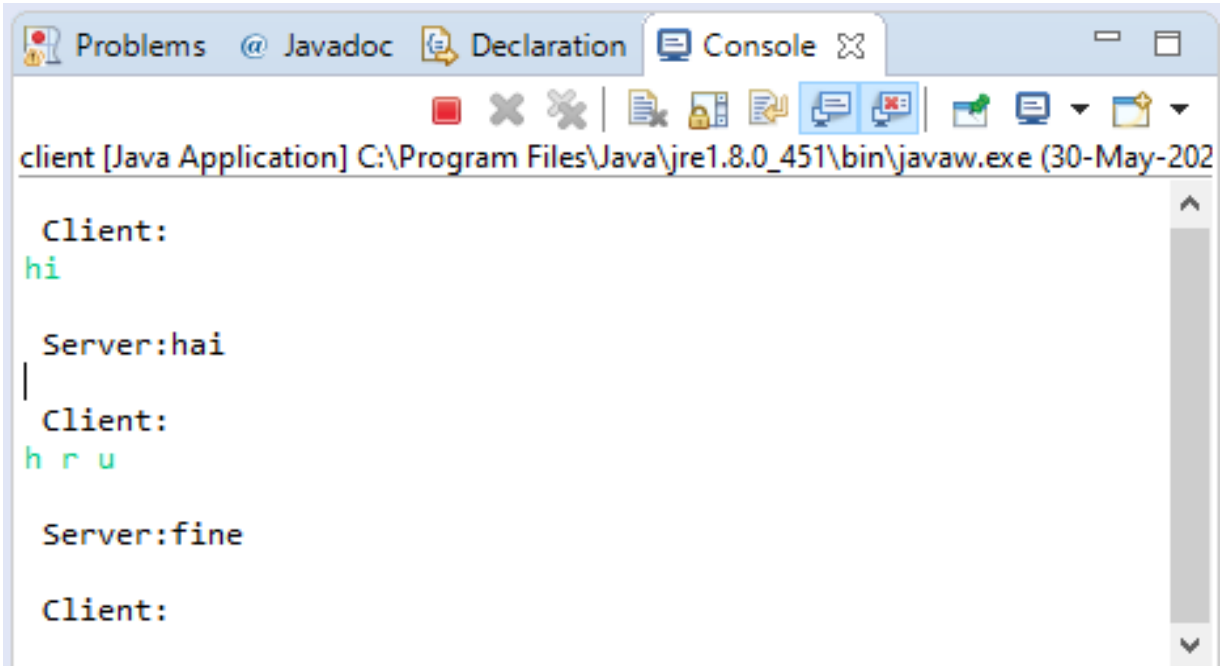
```
System.out.println("\nClient:"+clientdata);
System.out.println("\n Server:");
String serverdata=sc.nextLine();
sendbuffer=serverdata.getBytes();
DatagramPacketsendpacket=newDatagramPacket(sendbuffer,sendbuffer.length,ip,portno);
serversocket.send(sendpacket);
if(serverdata.equalsIgnoreCase("bye"))
{
System.out.println("Connected");
break;
}
}
serversocket.close();

}
}
```

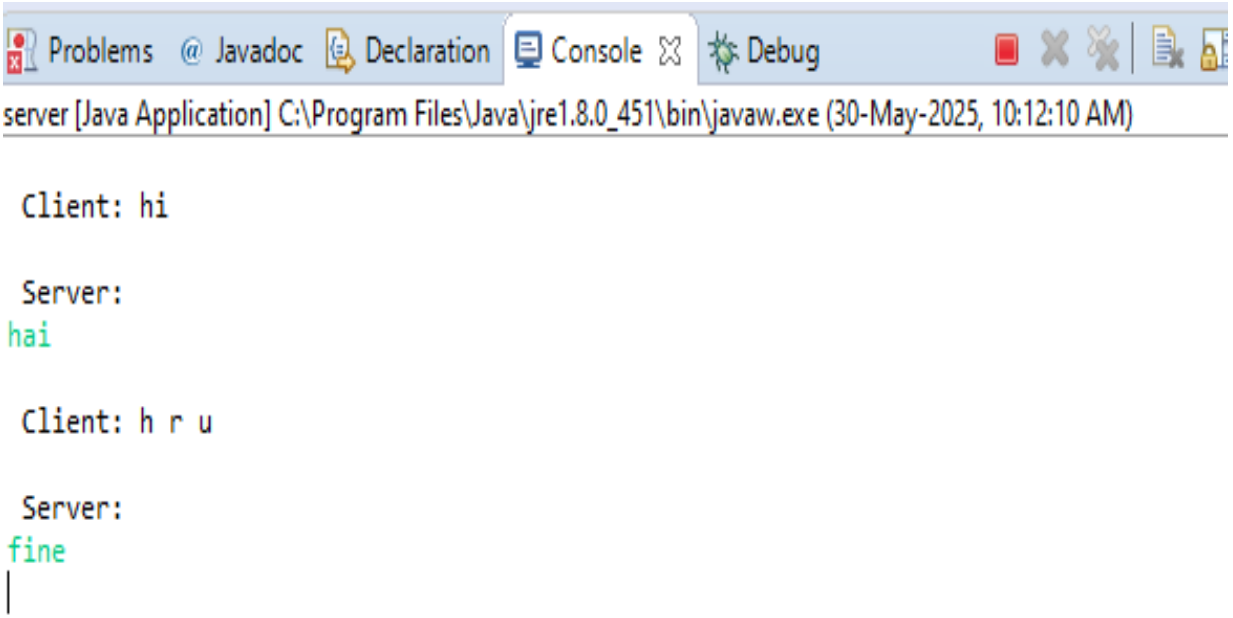
CLIENT :

```
importjava.io.*; import
java.util.*; import
java.net.*;
publicclassUDPClient
{
publicstaticvoidmain(Stringargs[])throwsSocketException,IOException
{
Scanner sc=new Scanner(System.in);
InetAddressip=InetAddress.getByName("localhost");
DatagramSocketclientsocket=newDatagramSocket();
```

```
while(true)
{
byte[] sendbuffer=new byte[1024];
byte[]receivebuffer=newbyte[1024];
System.out.println("\n Client:");
String clientdata=sc.nextLine();
sendbuffer=clientdata.getBytes();
DatagramPacketsendpacket=newDatagramPacket(sendbuffer,sendbuffer.length,ip,9902);
clientsocket.send(sendpacket);
if(clientdata.equalsIgnoreCase("bye"))
{
System.out.println("connectionendbyclient");
break;
}
DatagramPacketreceivepacket=newDatagramPacket(receivebuffer,receivebuffer.length);
clientsocket.receive(receivepacket);
Stringserverdata=newString(receivepacket.getData());
System.out.println("\n Server:"+serverdata);
}
clientsocket.close();
}
}
```

Output:


Problems
Javadoc
Declaration
Console
client [Java Application] C:\Program Files\Java\jre1.8.0_451\bin\javaw.exe (30-May-202
Client:
hi
Server:hai
Client:
h r u
Server:fine
Client:


Problems
Javadoc
Declaration
Console
Debug
server [Java Application] C:\Program Files\Java\jre1.8.0_451\bin\javaw.exe (30-May-2025, 10:12:10 AM)
Client: hi
Server:
hai
Client: h r u
Server:
fine

4c. File Transfer

AIM:
To write a java program for file transfer using TCP Sockets.

Algorithm

Server

1. Import java packages and create class file server.
2. Create a new server socket and bind it to the port.
3. Accept the client connection
4. Get the file name and stored into the Buffered Reader.
5. Create a new object class file and realine.
6. Iffileis existsthenFileReaderreadthecontentuntil EOFis reached.
7. Stop the program.

Client

1. Import java packages and create class file server.
2. Create a new server socket and bind it to the port.
3. Now connection is established.
4. The object of a Buffer Reader classis used for storing data content which has been retrieved from socket object.
5. The connection is closed.
6. Stop the program.

File Server:

```
Impor tjava.io.BufferedInputStream;
import java.io.File;
importjava.io.FileInputStream;
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket
publicclassFileServer
{
publicstatic void main(String[]args)throws Exception


]
```

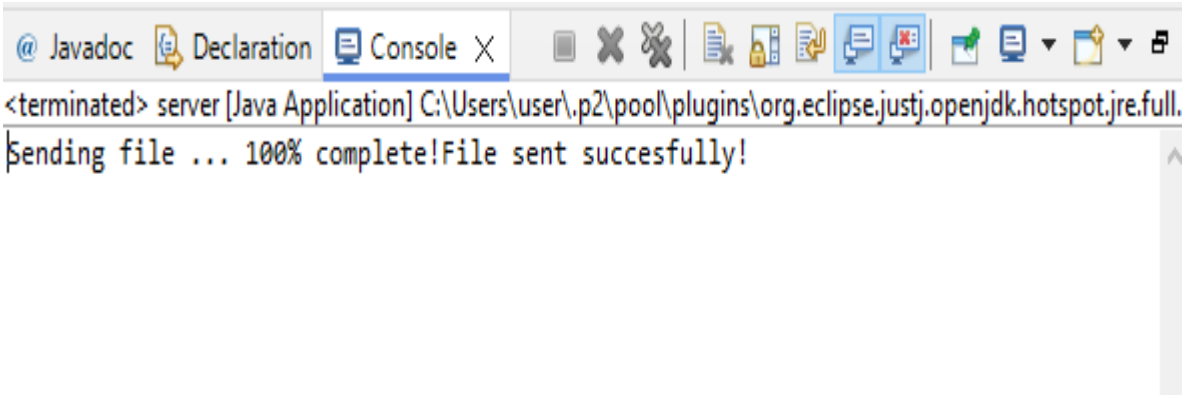
```
{
//InitializeSockets
ServerSocketssock=newServerSocket(5000);Socket socket
= ssock.accept();
//TheInetAddressspecification
InetAddressIA= InetAddress.getByName("localhost");
//Specifythefile
File file = new File("e:\\Bookmarks.html");
FileInputStreamfis = new FileInputStream(file);
BufferedInputStreambis=newBufferedInputStream(fis);

OutputStreamos=socket.getOutputStream();//Read
File Contents into contents array
byte[]contents;
longfileLength=file.length();
long current = 0;
longstart=System.nanoTime();
while(current!=fileLength){
int size=10000;
if(fileLength-current>=size)
current += size;
else{
size=(int)(fileLength-current); current
= fileLength;
}
contents=newbyte[size];

bis.read(contents,0,size);
os.write(contents);
System.out.print("Sendingfile..."+(current*100)/fileLength+"%complete!");
}
os.flush();
//Filetransferdone.Closethesocketconnection!
socket.close();
ssock.close();
System.out.println("Filesentsuccesfully!");
} }
```

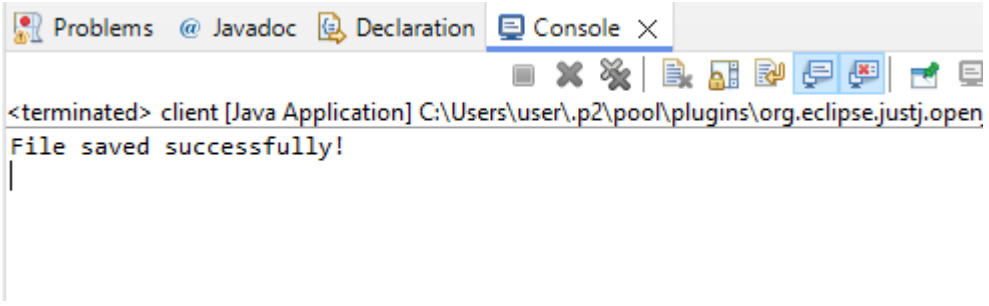
FileClient:

```
importjava.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
importjava.net.InetAddress;
import java.net.Socket;
public class FileClient {
publicstatic void main(String[]args)throws Exception{
//Initializesocket
Socketsocket=newSocket(InetAddress.getByName("localhost"),5000);byte[]
contents = new byte[10000];
//InitializetheFileOutputStream totheoutputfile'sfullpath.FileOutputStreamfos= new
```

```
FileOutputStream("e:\\Bookmarks1.html");
BufferedOutputStreambos=newBufferedOutputStream(fos);
InputStream is = socket.getInputStream();
//Noofbytesreadinoneread()call int
bytesRead = 0;
while((bytesRead=is.read(contents))!=-1)
bos.write(contents, 0, bytesRead);.

bos.flush();
socket.close();
System.out.println("Filesavedsuccessfully!");
}

}
```

Output





Experiment 5:

AIM : To write a java program for DNS application

PRELAB DISCUSSION:

The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities.

The domain name space refers a hierarchy in the internet naming structure. This hierarchy has multiple levels (from 0 to 127), with a root at the top. The following diagram shows the domain name space hierarchy.

Name server contains the DNS database. This database comprises of various names and their correspondingIPaddresses.SinceitisnotpossibleforasingleservertomaintainentireDNSdatabase, therefore ,the information is distributed among many DNS servers. Types of Name Servers Root Server is the top level server which consists of the entire DNS tree. It does not contain the information about domains but delegates the authority to the other server Primary Server stores a file about its zone. It has authority to create, maintain, and update The zone file. Secondary Servertransferscompleteinformationaboutazonefromanotherserverwhich may be primary or secondary server. The secondary server does not have authority to create or update a zone file.

DNSisaTCP/IPprotocolusedondifferentplatforms.Thedomainnamespaceisdivided into three different sections: generic domains, country domains, and inverse domain.

The main function of DNS is to translate domain names into IP Addresses, which computers can understand. It also provides a list of mail servers which accept Emails for each domain name. Each domain name in DNS will nominate a set of name servers to be authoritative for its DNS records

ALGORITHM

Server

1. Start
2. Create UDP datagram socket
3. Create a table that maps host name and IP address
4. Receive the host name from the client
5. Retrieve the client's IP address from the received datagram
6. Get the IP address mapped for the hostname from the table.
7. Display the host name and corresponding IP address
8. Send the IP address for the requested host name to the client
9. Stop.

Client

1. Start
2. Create UDP datagram socket.
3. Get the host name from the client
4. Send the host name to the server
5. Wait for the reply from the server
6. Receive the reply datagram and read the IP address for the requested hostname
7. Display the IP address.
8. Stop

Simulation of DNS using UDP Sockets

DNS Server

```
javaimportjava.io.*;
import java.net.*;
publicclassudpdnsserver
{
privatestaticintindexOf(String[]array,Stringstr)
{
str =str.trim();
for(inti=0;i<array.length;i++)
{
if(array[i].equals(str))
return i;
}
return-1;
}
publicstaticvoidmain(Stringarg[])throwsIOException
{
String[] hosts = {"yahoo.com", "gmail.com","cricinfo.com", "facebook.com"};
String[]ip={"68.180.206.184","209.85.148.19","80.168.92.140","69.63.189.16"};
System.out.println("PressCtrl+CtoQuit"); while
(true)
{
DatagramSocketserversocket=newDatagramSocket(1362);
byte[] senddata = new byte[1021]; byte[] receivedata = new byte[1021];
DatagramPacketrecvpack=newDatagramPacket(receivedata,receivedata.length);
serversocket.receive(recvpack);
Stringsen=newString(recvpack.getData());

InetAddressipaddress=recvpack.getAddress(); int
port = recvpack.getPort();
String capsent;
System.out.println("Requestforhost"+sen); if(indexOf
(hosts, sen) != -1)
capsent=ip[indexOf(hosts,sen)]; else
capsent = "Host Not Found";
senddata=capsent.getBytes();
DatagramPacketpack=newDatagramPacket(senddata,senddata.length,ipaddress,port);
serversocket.send(pack);
serversocket.close();
}}}
```

UDP DNS Client

```
javaimportjava.io.*;
import java.net.*;
publicclassudpdnsclient
{
publicstaticvoidmain(Stringargs[])throwsIOException
{
BufferedReaderbr=newBufferedReader(newInputStreamReader(System.in));
DatagramSocketclientsocket = new DatagramSocket();
```

```
InetAddressipaddress;
if (args.length == 0)
ipaddress=InetAddress.getLocalHost(); else
ipaddress=InetAddress.getByName(args[0]);
byte[] senddata = new byte[1024];
byte[]receivedata=newbyte[1024]; int
portaddr = 1362;
System.out.print("Enterthehostname:");
String sentence = br.readLine();
Senddata=sentence.getBytes();
DatagramPacketpack=newDatagramPacket(senddata,senddata.length,ipaddress,portaddr);
clientsocket.send(pack);
DatagramPacketrecvpack=newDatagramPacket(receivedata,receivedata.length);
clientsocket.receive(recvpack);
Stringmodified=newString(recvpack.getData());
System.out.println("IP Address: " + modified);
clientsocket.close();
}}
```

OUTPUT

```
Enterthehostname:yahoo.com IP
Address: 68.180.206.184

Enterthehostname:cricinfo.com IP
Address: 80.168.92.140

Enterthehostname: youtube.com
IPAddress:HostNotFound
```

ExperimentNo6

Write a code for simulating ARP/RARP protocols.

A**IM:**

To write a java program for simulating ARP and RARP protocols using TCP.

PRELAB DISCUSSION:

Address Resolution Protocol(ARP)is a low-level network protocol for translating network layer addresses into link layer addresses.ARP lies between layers 2 and 3 of the OSI model, although ARP was not included in the OSI framework and allows computers to introduce each other a cross a network prior to communication.Because protocols are basic network communication units, address resolution is dependent on protocols such as ARP, which is the only reliable method of handling required tasks. The Address Resolution Protocol(ARP)is a communication protocol used for discovering the link layer address, such as a MAC address, associated with a given internet layer address,

When configuring a new network computer, each system is assigned an Internet Protocol (IP) address for primary identification and communication. A computer also has a unique media access control (MAC)address identity .Manufacturers embed the MAC address in the local area network(LAN)card. The MAC address is also known as the computer’s physical address.

Address Resolution Protocol (ARP) is used to resolve an IPv4 address (32 bit Logical Address) to the physical address (48 bit MAC Address). Network Applications at the Application Layer use IPv4 Address to communicate with another device.

Reverse Address Resolution Protocol (RARP) is a network protocol used to resolve a data link layer address to the corresponding net work layer address. For example,RARP isused toresolvea Ethernet MAC address to an IP address.

The client broadcasts a RARP packet with an ethernet broadcast address, and it's own physical address in the dataportion.The server responds byte lling the clientit's IPaddress. Notethereisno name sent. Also note there is no security.

Media Access Control (MAC) addresses need to be individually configured on the servers by an administrator. RARP is limited to serving only IP addresses.Reverse ARP differs from the Inverse Address Resolution Protocol which is designed to obtain the IP address associated with a local Frame Relay data link connection identifier. InARP is not used in Ethernet

ALGORITHM:

Client

1. Start the program
2. Create socket and establish connection with the server.
3. Get the IP address to be converted into MAC address from the user.
4. Send this IP address to server.
5. Receive the MAC address for the IP address from the server.
6. Display the received MAC address
7. Terminate the connection

Server

1. Start the program
2. Create the socket ,bind the socket created with IP address and port number and make it a listening socket.
3. Accept the connection request when it is requested by the client.
4. Server maintains the table in which IP and corresponding MAC addresses are stored.
5. Receive the IP address sent by the client.
6. Retrieve the corresponding MAC address for the IP address and send it to the client.
7. Close the connection with the client and now the server be comes a listening server waiting for the connection request from other clients
8. Stop

.

```
import java.io.*;

import  java.net.*;
importjava.util.*;
class Clientarp
{
publicstaticvoidmain(Stringargs[])
{
try
{
BufferedReaderin=newBufferedReader(newInputStreamReader(System.in)); Socket
clsct=new Socket("127.0.0.1",5604)
DataInputStream din=new DataInputStream(clsct.getInputStream());
DataOutputStreamdout=newDataOutputStream(clsct.getOutputStream());
System.out.println("Enter the Logical address(IP):");
Stringstr1=in.readLine();
dout.writeBytes(str1+'\n';
Stringstr=din.readLine();

System.out.println("ThePhysicalAddressis:"+str);

clsct.close();
}
catch(Exceptione)
{
System.out.println(e);
}
}
}
```

Server:

```
import java.io.*;
importjava.net.*;
importjava.util.*;


classServerarp
{
publicstaticvoidmain(Stringargs[])
{
try{ServerSocketobj=new
ServerSocket(139);
```

```
Socketobj1=obj.accept();

while(true)

{

DataInputStream din=new DataInputStream(obj1.getInputStream());

DataOutputStreamdout=newDataOutputStream(obj1.getOutputStream());

String str=din.readLine();


String ip[]={"165.165.80.80","165.165.79.1"};
Stringmac[]={"6A:08:AA:C2","8A:BC:E3:FA"};
for(inti=0;i<ip.length;i++)
{
if(str.equals(ip[i]))
{
dout.writeBytes(mac[i]+'\n');
break;
}
}
obj.close();
}
}
catch(Exceptione)


{


System.out.println();
}
}
}
```

Output:

```
EntertheLogicaladdress(IP):
165.165.80.80
ThePhysicalAddress is:6A:08:AA:C2
```

(b)Program for Reverse Address Resolution Protocol(RARP)usingUDP ALGORITHM:
Client
1. Start the program
2. Create datagram socket
3. Get the MAC address to be converted into IP address from the user.
4. Send this MAC address to server using UDP datagram.
5. Receive the datagram from the server and display he corresponding IPaddress.
6. Stop

Server
1. Start the program.
2. Server maintains the table in which IP and corresponding MAC addresses are stored.
3. Create the datagram socket
4. Receive the datagram sent by the client and read the MAC address sent.
5. Retrieve the IP address for the received MAC address from the table.
6. Display the corresponding IPaddress.
7. Stop

```
import java.io.*;
importjava.net.*;
importjava.util.*;
class Clientrarp
{
publicstaticvoidmain(Stringargs[])

{
try
{
DatagramSocket client=new DatagramSocket();
InetAddressaddr=InetAddress.getByName("127.0.0.1");
byte[] sendbyte=new byte[1024];
byte[]receivebyte=newbyte[1024];
BufferedReaderin=newBufferedReader(newInputStreamReader(System.in));
System.out.println("Enter the Physical address (MAC):");
Stringstr=in.readLine();sendbyte=str.getBytes();
DatagramPacketsender=newDatagramPacket(sendbyte,sendbyte.length,addr,1309);
client.send(sender);
DatagramPacketreceiver=newDatagramPacket(receivebyte,receivebyte.length);
client.receive(receiver);
Strings=newString(receiver.getData());

System.out.println("TheLogicalAddressis(IP):"+s.trim());
client.close();
}
catch(Exceptione)
{
System.out.println(e);
}
}
}
```

Server:

```
import java.io.*;
importjava.net.*;
importjava.util.*;
class Serverrarp
{
publicstaticvoid main(String args[])
{
try
{
DatagramSocketserver=newDatagramSocket(1309);
while(true)
{
byte[] sendbyte=new byte[1024];
byte[]receivebyte=newbyte[1024];
DatagramPacketreceiver=newDatagramPacket(receivebyte,receivebyte.length);
server.receive(receiver);
Stringstr=newString(receiver.getData());
String s=str.trim();
InetAddressaddr=receiver.getAddress();
int port=receiver.getPort();
String ip[]={"165.165.80.80","165.165.79.1"};
Stringmac[]={"6A:08:AA:C2","8A:BC:E3:FA"};
for(inti=0;i<ip.length;i++)
{
if(s.equals(mac[i]))
{
sendbyte=ip[i].getBytes();
DatagramPacketsender=newDatagramPacket(sendbyte,sendbyte.length,addr,port);
server.send(sender);
break;
}
}
break;
}
}
catch(Exceptione)
{
System.out.println(e);

} }}
```

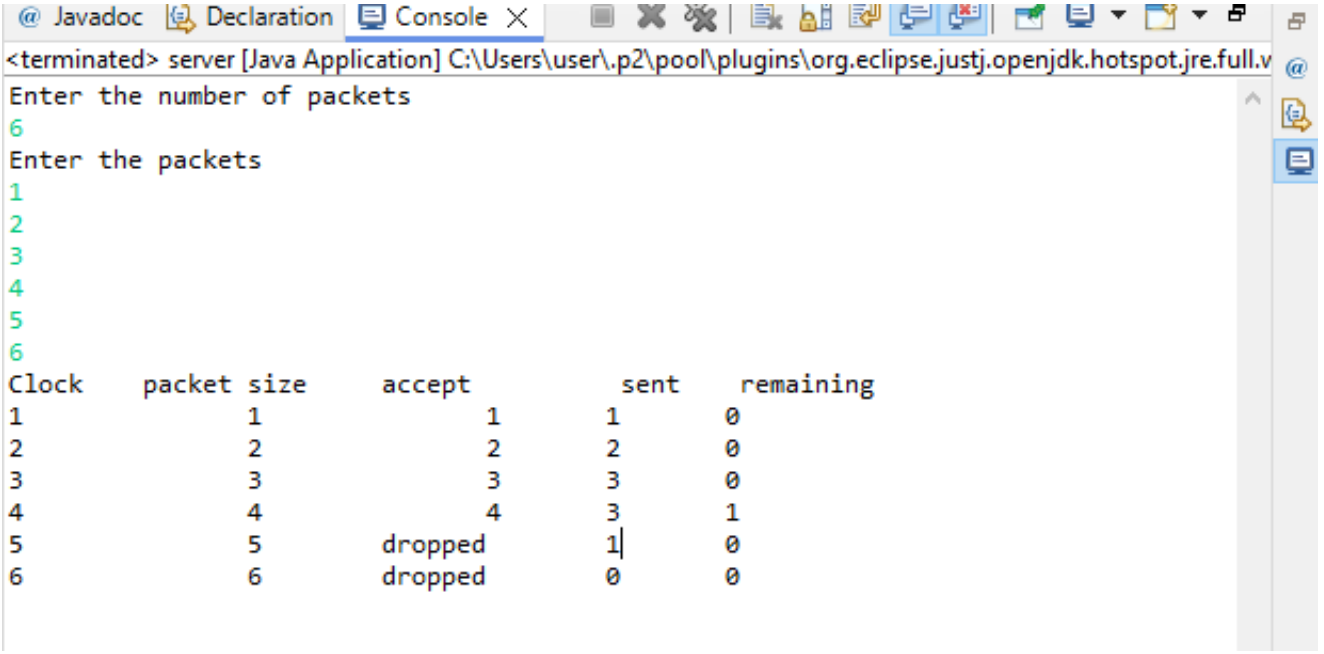
Output:

```
EnterthePhysicaladdress(MAC):
6A:08:AA:C2
TheLogical Address is(IP): 165.165.80.80
```

7. Writeaprogramforcongestioncontrol usingleakybucket algorithm.

Aim:Towriteaprogramforcongestioncontrolusingleakybucketalgorithm.

```
import java.util.Scanner;

import java.lang.*;
publicclassLEAKY_BUCKET{
publicstaticvoidmain(String[]args)
{
Int i;
int a[]=newint[20];
intbuck_rem=0,buck_cap=4,rate=3,sent,recv;
Scanner in = new Scanner(System.in);
System.out.println("Enterthenumberofpackets"); int
n = in.nextInt();
System.out.println("Enterthepackets");
for(i=1;i<=n;i++)
a[i]=in.nextInt();
System.out.println("Clock\tpacketsize\taccept \tsent\tremaining");
for(i=1;i<=n;i++)
{
if(a[i]!=0)
{
if(buck_rem+a[i]>buck_cap)
recv=-1;
else
{
recv=a[i];
buck_rem+=a[i];
}
}
else
recv=0;
if(buck_rem!=0)
{
if(buck_rem<rate)
{sent=buck_rem;
buck_rem=0;
}
else
{
sent=rate;
buck_rem=buck_rem-rate;
}
}
else
sent=0;
if(recv==-1)
System.out.println(+i+"\t\t"+a[i]+"\tdropped\t"+sent+"\t"+buck_rem);
else
System.out.println(+i+"\t\t"+a[i]+"\t\t"+recv+"\t"+sent+"\t"+buck_rem);
}
}    }
```


Javadoc
Declaration
Console
<terminated> server [Java Application] C:\Users\user\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.v
Enter the number of packets
6
Enter the packets
1
2
3
4
5
6
Clock packet size accept sent remaining
1 1 1 1 0
2 2 2 2 0
3 3 3 3 0
4 4 4 3 1
5 5 dropped 1 0
6 6 dropped 0 0

ExperimentNo08.

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

Program:

```
set ns [new Simulator]
set nf [open PA1.nam w]
$ns namtrace-all $nf
set tf [open PA1.tr w]
$ns trace-all $tf
proc finish { } {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam PA1.nam &
exit 0
}
set n0 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 80Mb 10ms DropTail
$ns duplex-link $n2 $n3 79.5Mb 10ms DropTail
$ns queue-limit $n0 $n2 100
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 5000
$cbr0 set interval_ 0.0005
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
$ns connect $udp0 $null0
$ns at 0.1 "$cbr0 start"
$ns at 1.0 "finish"
$ns run
```

AWK file:(Open a new editor using “vi command” and write awk file and save with “.awk” extension)

```
#immediately after BEGIN should open braces ‘{‘

BEGIN{ c=0;}
{
  if($1=="d")
  {c++;
        printf("%s\t%s\n",$5,$11);
```

```
  }
 }
END{ printf("The number of packets dropped =%d\n",c); }
```

Steps for execution:

- Open vi editor and type program. Program name should have the extension " **.tcl** "
  [root@localhost ~]# vi lab1.tcl
- Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.
- Open vi editor and type **awk** program. Program name should have the extension "**.awk** "
  [root@localhost ~]# vi lab1.awk
- Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.
- Run the simulation program
  [root@localhost~]# ns lab1.tcl
- Here **"ns"** indicates network simulator. We get the topology shown in the snapshot.
- Now press the play button in the simulation window and the simulation will begins.
- After simulation is completed run **awk file** to see the output ,
  [root@localhost~]# awk –f lab1.awk lab1.tr
- To see the trace file contents open the file as ,
  [root@localhost~]# vi lab1.tr

Trace file contains 12 columns:

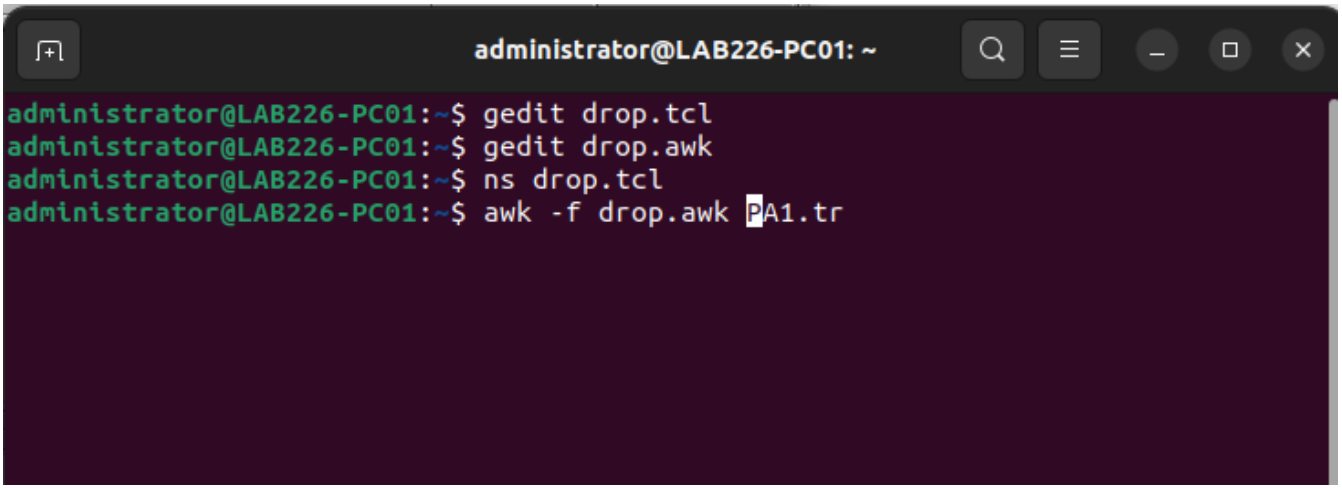
Event type, Event time, From Node, To Node, Packet Type, Packet Size, Flags (indicated by), Flow ID, Source address, Destination address, Sequence ID, Packet ID

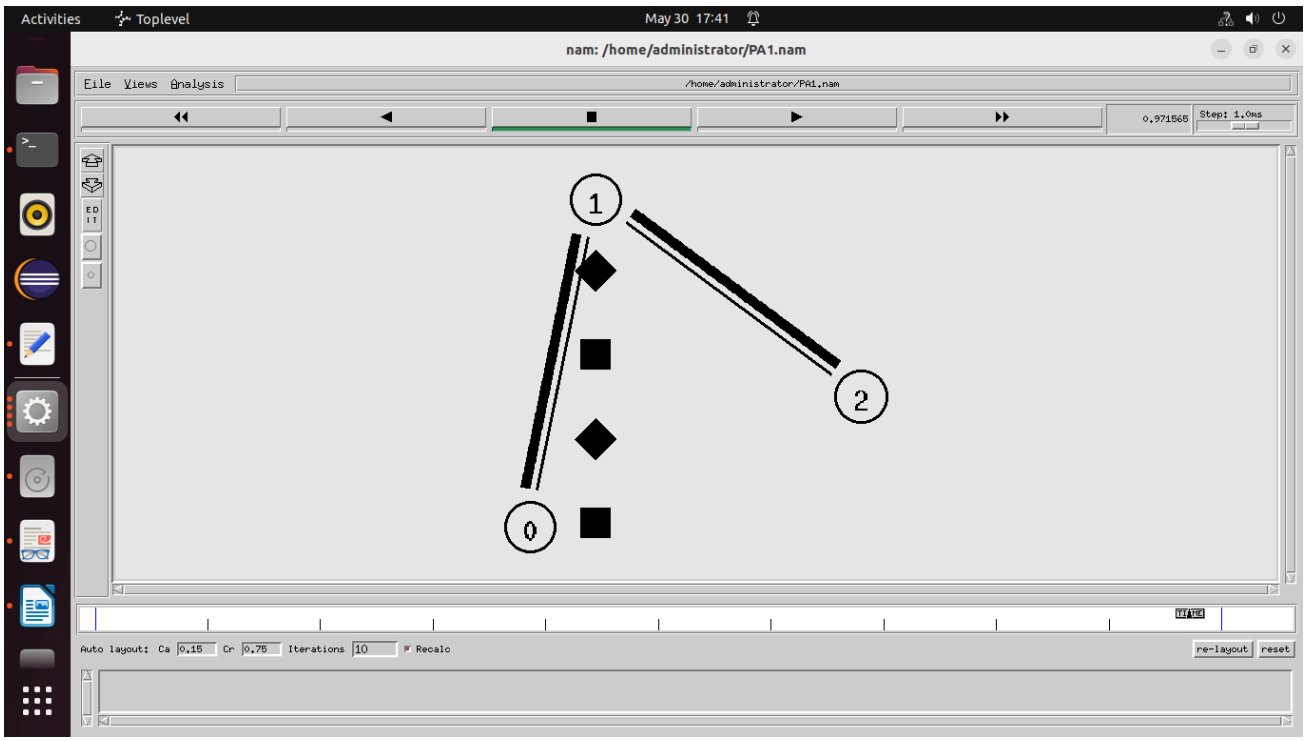
Output

Contents of Trace File Topology

SimulatedNetwork:

PacketDropped:






```
administrator@LAB226-PC01: ~
administrator@LAB226-PC01:~$ ns drop.tcl
administrator@LAB226-PC01:~$ awk -f drop.awk PA1.tr
cbr      7840
cbr      8000
cbr      8160
cbr      8320
cbr      8480
cbr      8640
cbr      8800
The number of packets dropped =7
administrator@LAB226-PC01:~$
```

Experiment 9

Simulate the transmission of ping messages/trace route overa network topology consisting of 6 nodes and find the number of packets dropped due to congestion

Aim: **Implement transmission of ping messages/traceroute over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion**.

```
set ns [ new Simulator ]
set nf [ open lab4.nam w ]
$ns namtrace-all $nf
set tf [ open lab4.tr w ]
$ns trace-all $tf
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$ns duplex-link $n0 $n4 1005Mb 1ms DropTail
$ns duplex-link $n1 $n4 50Mb 1ms DropTail
$ns duplex-link $n2 $n4 2000Mb 1ms DropTail
$ns duplex-link $n3 $n4 200Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 1ms DropTail
set p1 [new Agent/Ping]
$ns attach-agent $n0 $p1
$p1 set packetSize_ 50000
$p1 set interval_ 0.0001
set p2 [new Agent/Ping]
$ns attach-agent $n1 $p2
set p3 [new Agent/Ping]
$ns attach-agent $n2 $p3
$p3 set packetSize_ 30000
$p3 set interval_ 0.00001
set p4 [new Agent/Ping]
$ns attach-agent $n3 $p4
set p5 [new Agent/Ping]
$ns attach-agent $n5 $p5
$ns queue-limit $n0 $n4 5
$ns queue-limit $n2 $n4 3
$ns queue-limit $n4 $n5 2
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id]received answer from $from with round trip time $rtt msec"
}
$ns connect $p1 $p5
$ns connect $p3 $p4
proc finish { } {
global ns nf tf
```

```
$ns flush-trace
close $nf
close $tf
exec nam lab4.nam &
exit 0
}
$ns at 0.1 "$p1 send"
$ns at 0.2 "$p1 send"
$ns at 0.3 "$p1 send"
$ns at 0.4 "$p1 send"
$ns at 0.5 "$p1 send"
$ns at 0.6 "$p1 send"
$ns at 0.7 "$p1 send"
$ns at 0.8 "$p1 send"
$ns at 0.9 "$p1 send"
$ns at 1.0 "$p1 send"
$ns at 1.1 "$p1 send"
$ns at 1.2 "$p1 send"
$ns at 1.3 "$p1 send"
$ns at 1.4 "$p1 send"
$ns at 1.5 "$p1 send"
$ns at 1.6 "$p1 send"
$ns at 1.7 "$p1 send"
$ns at 1.8 "$p1 send"
$ns at 1.9 "$p1 send"
$ns at 2.0 "$p1 send"
$ns at 2.1 "$p1 send"
$ns at 2.2 "$p1 send"
$ns at 2.3 "$p1 send"
$ns at 2.4 "$p1 send"
$ns at 2.5 "$p1 send"
$ns at 2.6 "$p1 send"
$ns at 2.7 "$p1 send"
$ns at 2.8 "$p1 send"
$ns at 2.9 "$p1 send"
$ns at 0.1 "$p3 send"
$ns at 0.2 "$p3 send"
$ns at 0.3 "$p3 send"
$ns at 0.4 "$p3 send"
$ns at 0.5 "$p3 send"
$ns at 0.6 "$p3 send"
$ns at 0.7 "$p3 send"
$ns at 0.8 "$p3 send"
$ns at 0.9 "$p3 send"
$ns at 1.0 "$p3 send"
$ns at 1.1 "$p3 send"
$ns at 1.2 "$p3 send"
$ns at 1.3 "$p3 send"
$ns at 1.4 "$p3 send"
$ns at 1.5 "$p3 send"
$ns at 1.6 "$p3 send"
```

```
$ns at 1.7 "$p3 send"
$ns at 1.8 "$p3 send"
$ns at 1.9 "$p3 send"
$ns at 2.0 "$p3 send"
$ns at 2.1 "$p3 send"
$ns at 2.2 "$p3 send"
$ns at 2.3 "$p3 send"
$ns at 2.4 "$p3 send"
$ns at 2.5 "$p3 send"
$ns at 2.6 "$p3 send"
$ns at 2.7 "$p3 send"
$ns at 2.8 "$p3 send"
$ns at 2.9 "$p3 send"
$ns at 3.0 "finish"
$ns run
```

AWK File:

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

```
administrator@LAB226-PC01:~$ gedit ping.tcl
administrator@LAB226-PC01:~$ ns ping.tcl
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 404.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 704.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 804.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 804.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 804.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 804.9 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 2received answer from 3 with round trip time 5.3 msec
node 0received answer from 5 with round trip time 804.9 msec
node 2received answer from 3 with round trip time 5.3 msec
administrator@LAB226-PC01:~$ awk -f ping.awk lab4.tr
Total number of ping packets dropped due to congestion =20
administrator@LAB226-PC01:~$
```

VIVA QUESTIONS

1. What is socket? How is it implemented?
2. What is the different system call son clients ide and servers ide implementation ?
3. How to convert single client into multiple clients handling in server?
4. What are the differences between SOCK_STREAM, SOCK_DGRAM, and SOCK_RAM? What is their level of implementation?
5. Explain need of sock adder _instructure and IN ADDR_ANY.
6. What is FIFO? How to implement Multiple FIFOs. Disadvantages of FIFO?
7. Which are the different IPCP rimitives ?Explain Why IPC Primitives are required?
8. Explain the purpose of RS algorithm? What is Public and Private Key? How to find public and private key in RSA algorithm?
9. Listsome Cryptographic algorithms?
10. What is congestion? Explain leaky bucket and token bucket algorithm? What are its disadvantages of each?
11. List other congestion control algorithm.
12. What are the disadvantages of RSAAl gorithm?
13. What is the need for routing protocols? List some routing protocols. Which are common routing algorithms used by routing protocols
14. Disadvantages of Distance vector routing. Explain the method to overcome such disadvantage.
15. What is CRC ?What is the need for CRC ?Which Layer it is implemented?
16. Can CRC be employed for error correction if so how.
17. Explain steps of error correction and error detection .What is encoding and decoding, why it is required?
18. What is Hamming code ?Can it be used for error detection and correction how?
19. What are linear block codes ?Explain their advantages.
20. What is protocol? List some protocols .How to modify existing protocol?
21. Difference between TCP/IP and OSI Layer Model?
22. Difference between Router ,Bridge ,Gateway channel .In which layer it is used ?
23. Difference between hub and switch. Which is more compatible and reliable?
24. Explain various Ethernet standards.
25. DifferencebetweenIPandMACaddress.
26. In which layer ATM protocol isused ? Explain the purpose.
27. In Which layer or protocol which converts IP to MAC Address.
28. Which are the different collision detection protocols?

29. Which is the different channelization protocol and explain the ir needs?
30. Explain term error rate, bandwidth, throughput transmission delay, BER, Interarrival time. What is unit of bandwidth?

Do's

Do wear ID card and follow dress code.

- Do log off the computers when you finish.
- Do ask for assistance if youneed help.
- Do keep your voice low when speaking to others in theLAB.
- Do ask for assistancein downloading any software.
- Do make suggestions as to how we can improve the LAB.
- In case of any hardware related problem, ask LAB incharge for solution.
- If you are the last one leaving the LAB, make sure that the staff in charge of the LAB is informed to close the LAB.
- Beon time to LAB sessions.
- Do keep the LAB as clean as possible.

Don'ts

- Do not use mobile phone inside the lab.
- Don't do anything that can make the LAB dirty(like eating ,throwing waste papers etc).
- Do not carry any external devices without permission.
- Don't move the chairs of the LAB.
- Don't inter change any part of one computer with another.
- Don't leave the computers of the LAB turned on while leaving the LAB.
- Do not install or download any software or modify or delete any system files on any lab computers.
- Do not damage ,remove ,ordis connectany labels, parts, cables, or equipment.
- Don't attempt to by pass the computer security system.
- Dono tread or modify ther user's file.
- If you leave the lab, donot leave your personal belonging sun attended .We are not responsible for any theft.
- Do not install or down load any software or modify or delete any system files on any lab computers.
- Do not damage, remove, ordisconnectany labels, parts, cables, or equipment.
- Don't attempt to by pass the computer security system.
- Do not read or modify ther user's file.
- If you leave the lab, donot leave your personal belonging sun attended. We are not responsible for any theft.