DAYANANDA SAGAR ACADEMY OF TECHNOLOGY & MANAGEMENT

Affiliated to VTU
Approved by AICTE
Accredited by NAAC with A+ Grade
6 Programs Accredited by NBA
(CSE, ISE, ECE, EEE, MECH, CIVIL)

Department of CSE in Data Science

Computer Networks Laboratory Manual

Course Code :BCS502

Semester :Vth Sem

By,
Mahesh Basavaraj
Asst. Professor
DSATM

1. Vision and Mission of the Institution

Vision of the Institution:

To strive at creating the institution a center of highest caliber of learning, so as to create an overall intellectual atmosphere with each deriving strength from the other to be the best of engineers, scientists with management & design skills.

Mission of the Institution:

- To serve its region, state, the nation and globally by preparing students to make meaningful contributions in an increasing complex global society challenges.
- To encourage, reflection on and evaluation of emerging needs and priorities with state of art infrastructure at institution.
- To support research and services establishing enhancements in technical, economic, human and cultural development.
- To establish inter disciplinary center of excellence, supporting/ promoting student's implementation.
- To increase the number of Doctorate holders to promote research culture on campus.
- To establish IIPC, IPR, EDC, innovation cells with functional MOU's supporting student's quality growth.

QUALITY POLICY

Dayananda Sagar Academy of Technology and Management aims at achieving academic excellence through continuous improvement in all spheres of Technical and Management education. In pursuit of excellence cutting-edge and contemporary skills are imparted to the utmost satisfaction of the students and the concerned stakeholders.

2. Vision and Mission of the Department

Department Vision

To create an academic environment which trains the students as next generation data scientist solving grand challenges innovating through global research opportunities.

Department Mission

M1: To ensure the responsible use of data to benefit society.

M2: To ensure broader community in the translation of data into information to support and improve decision making.

M3: To develop skilled professionals in data science field.

M4: To establish industry conducive environment with State-of-Art data driven infrastructure.

M5: To facilitate high quality data science education, industry collaboration with research orientation.

M6: To maximize the power of data benefiting the social needs through science and engineering.

3. Programme Educational Objectives

PEO1: Graduates shall have robust knowledge of data handling, analytics platform.

PEO2: Graduates will be skilled professionals with global competence.

PEO3: Graduates shall have successful carrier as data science engineers with leadership and management skills.

4. Program Outcomes (POs)

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, research literature, and Analyze complex engineering problems

reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

5. Programme Specific Outcomes

PSO 1: Produce quality data science professionals with robust development knowledge

PSO 2: Develop global competency student quality to meet data science changes.

LIST OF EXPERIMENTS

SL No.	EXPERIMENT
1.	Implement three three-node point–to–point network with duplex links between them. Set the queue size, vary the bandwidth, and find the number of packets dropped.
2.	Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
3.	Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source/destination
4.	Develop a program for error-detecting code using CRC-CCITT (16-bits).
5.	Develop a program to implement a sliding window protocol in the data link layer.
6.	Develop a program to find the shortest path between vertices using the Bellman-Ford and path-vector routing algorithm.
7.	Using TCP/IP sockets, write a client-server program to make the client send the file name and to make the server send back the contents of the requested file if present.
8.	Develop a program on a datagram socket for the client/server to display the messages on the client side, typed at the server side.
9.	Develop a program for a simple RSA algorithm to encrypt and decrypt the data.
10.	Develop a program for congestion control using a leaky bucket algorithm.

CIE for the practical component of the IPCC

- **15 marks** for the conduction of the experiment and preparation of laboratory record, and **10 marks** for the test to be conducted after the completion of all the laboratory sessions.
- On completion of every experiment/program in the laboratory, the students shall be evaluated including viva-voce and marks shall be awarded on the same day.
- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to **15 marks**.
- The laboratory test **(duration 02/03 hours)** after completion of all the experiments shall be conducted for 50 marks and scaled down to **10 marks.**
- Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for **25 marks**.
- The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCC.

	Open Ended Questions
1.	**Socket Programming: Client-Server Communication** **Problem:** Write a program in Python (or any language) to implement a simple client-server communication using TCP sockets. The client should send a message to the server, and the server should send a response back to the client. **Expected Output:** The server and client should display the messages they send and receive, along with a confirmation that the communication was successful.
2.	**File Transfer using Sockets** **Problem:** Implement a file transfer protocol where the client can request a file from the server, and the server sends the file back to the client using TCP sockets. **Expected Output:** The cient should receive and save the file correctly, and both client and server should display logs of the transfer.
3.	**Ping Simulation** **Problem:** Write a program to simulate the **ping** command. The program should send ICMP echo request packets to a host and display the response time, along with packet loss percentage. **Expected Output:** Display the number of packets sent, received, lost, and the round-trip time for each packet.
4.	**Packet Sniffing using Wireshark** **Problem:** Use **Wireshark** to capture network traffic on your system. Identify the source and destination IP addresses, the protocol in use, and the packet length. Summarize the captured data. **Expected Output:** A report that includes screenshots and analysis of the captured traffic, such as identifying HTTP requests, DNS queries, etc.
5.	**Chat Application using Sockets** **Problem:** Design and implement a simple chat application using TCP sockets where multiple clients can send and receive messages to/from a server. **Expected Output:** Multiple clients should be able to connect to the server, and messages from one client should be relayed to all other connected clients.

INSTRUCTIONS FOR LAB :

1. **Algorithm for the program to be written on the left side (unruled side) of the record.**
2. **Programs to be written on the right side (Ruled side) of the record**
3. **Output and graph should be written on the left side (unruled side) of the record.**
4. **For simulation programs, please stick to the simulation screenshots of Wired scenario and network Animation output (nam output) as part of the output.**
5. **Open-ended questions are to be worked out by the groups created during the class/lab hour and solutions are to be explained. Marks will be given based on the solution and explanation.**

1. NSG2.1 & NS2

NS2 installation steps

- **Step 1:** open any Linux distribution such as Ubuntu, Mint, Fedora.
- **Step 2:** issue this command in the terminal "sudo apt-get install build-essential autoconf automake libxmu-dev" (without Quotes)
- **Step 3:** if you have Ubuntu issue this command first before u try step 2 : "sudo apt-get update"
- **Step 4:** Once installation succeeded, start the process of installing ns2
- **Step5:** copy the ns-allinone-2.3x.tar.gz in the home folder, in my case it is /home/DSATM
- **Step 6:** Go to home folder and execute the command **"tar zxvf ns-allinone-2.35.tar.gz"**
- **Step 7:** cd ns-allinone-2.35 and ./install (after these commands, wait for the successful installation), it may take 5 min to 15 minutes based on the speed of your computer.
- **Step 8:** there may be errors shown, in case if errors, the errors should be corrected and reissue the command ./install
- **Step 9:** There is an error on ns-2.35/linkstate/ls.h file, so the error has to be corrected as, In line number 137, change **erase to this->erase**
- **Step 10:** Once installation is succeeded, you will get some path information
- **Step 11:** set the paths in the .profile file which is under the home directory (**/home/hdl1(system name)/.profile**), copy the **PATH** url and **LD_LIBRARY_PATH** URL and paste in the .profile file.
- **Step 12:** Once the path information is set, run the command to test the path " source /home/DSATM/.profile) and see there should not be any errors
- **Step 13:** try ns and nam individually and see whether they are working when you type **ns** and type enter a **%** symbol indicates the installation is successful when you type **nam** and a network animation window pop up

NSG2.1

Installation steps:

- Step1 : sudo apt-get update"
- Step2 : sudo apt-get install default.jdk

What is NSG2?

NS2 Scenarios Generator 2(NSG2) is a JAVA based ns2 scenarios generator. Since NSG2 is written by JAVA language, you can run NSG on any platform. NSG2 is capable of generating both wired and wireless TCL scripts for ns2.

Some major functions of NSG2 are listed below:

1. Creating wired and wireless nodes
2. Creating connection between nodes
3. Creating links (Duplex-Link and Simplex-Link)
4. Creating agents (TCP and UDP)
5. Creating applications (CBR and FTP) 6. Node movement **Is NSG2 free?**

Yes, it is a free software.

Download NSG2 : NSG2.1 (2008/11/18)

Launch NSG2 :

To execute NSG2, you have to install JAVA6.0 first. You can download JAVA6.0 from http://java.sun.com/. The details of JAVA6.0 installation, please refer to Sun JAVA site.

NSG2 doesn't need to be installed in your computer. You just download it and launch it with following instruction under DOS command environment. In fact, on my computer system, Windows XP, I just double click the NSG2.jar, and NSG2 will automatically launch. If it doesn't launch, you can also launch NSG2 as following instructions.

General procedure to run the TCL script and verify the output

1. Right click in the folder where NSG is installed to open the terminal.
2. Type ‘ls‘ and check for list of copied files and type the command “ java –jar NSG2.1.jar”
3. Go the scenario and select “ New wired /wireless scenario”
4. Click on the nodes and place the nodes for the required scenario and connect them with link between nodes by choosing any one of them below mentioned according to the requirement.

- Duplex link - > TCP
- Simplex link - > UDP

5. Connect the agent to the nodes suitably [TCP & UDP]
6. Fix the application for the agent as shown

- TCP - > application -> FTP
- UDP - > application -> CBR

7. Connect the agent between source node and destination node.
8. Click on parameters in the toolbar & rename trace file and nam file and save it.
9. Click on TCL and generate TCL script for the scenario
10. Save the TCL file with same name as trace and nam file with .tcl extension
11. Go the terminal and type command : "ns filename.tcl"
12. Run trace file and check the movement of packet from source to destination.
13. To measure the performance of the experiment we create AWK file.
14. To create AWK file goto menu->accessories-> gedit.
15. Type the awk code and save in the same location.
16. To execute awk file goto terminal and type command awk –f filename.awk filename.tr

Tool Command Language(TCL)

- 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.

Analysing Tcl scripting

1. Initialization of the TCL script in NS2.
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.

6.Termination of the TCL script in NS2.

- **Initialization of the TCL script in NS2.**

An ns simulation starts with the command

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

This line declares a new variable as using the **set** command, In general we declares it as **ns** because it is an instance of the Simulator class, so an object the code[**new Simulator**] is indeed the installation of the class Simulator using the reserved word new.

- **Definition of a network nodes**

The way to define a node is

```
set n0 [$ns node]
```

We created a node that is printed by the variable n0. When we shall refer to that node in the script we shall thus write $n0. Likewise any number of nodes can be created.

Definition of a network Link :

we can define the links to connect these nodes. 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 “simplexlink”.

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

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

An example would be:

```
#set Queue Size of link (n0-n2) to 20
$ns queue-limit $n0 $n2 20
```

- **Definition of agents and of 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.

The type of agent appears in the first line:

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

The command **$ns attach-agent $n0 $tcp**

defines the source node of the tcp connection.

The command

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

defines the behaviour of the destination node of TCP and assigns to it a pointer called sink.

FTP over UDP

#Setup a UDP connection

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

#setup a CBR over UDP connection

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

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

The command **$ns attach-agent $n4 $sink**

defines the destination node.

The command **$ns connect $tcp $sink**

makes the TCP connection between the source and destination nodes.

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

When we have several flows, we may wish to distinguish them so that we can identify them with different colors in the visualization part. This is done by the command **$tcp set fid_ 1** that assigns to the TCP connection a flow identification of "1".We shall later give the flow identification of "2" to the UDP connection.

CBR over UDP

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

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

```
$cbr set interval_ 0.005
```

The packet size can be set to some value using

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

Scheduling Events

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

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

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

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

```
$ns at 0.1 "$cbr start"
$ns at 1.0 " $ftp start"
$ns at 124.0 "$ftp stop"
```

- **Structure of Trace Files**

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

Analyzing the trace file

Trace file format for wired scenario

Event	Time	Where from node	To node	Pkt type	Pkt size	Flags	Flow ID	Src addr	Dest addr	Seq no	Pkt id

Event	Meaning
- r d	Added to queue Removed from queue Received at node Dropped

Trace file format for wireless scenario

Action	Time	Where from node	Layer	Flags	Seq no	Type	Size	flags

1. Action s – sent

   r – Rxd

   D - dropped

2. Time : when the action happened

3. From node : The node where action happened

4. Layer

   AGT -> Application layer (Application)

   RTR – Network Layer (Routing)

   LL – Link layer (ARP is done here)

   IFQ – Outgoing packet queue(between link & MAC layer)

   MAC –

   Phy – Physical

5. Flags :

6. Seq no : the sequence number of the packet

7. Type : Packet type Cbr -> CBR data stream pkt

   DSR -> DSR routing pkt

   RTS -> RTS pkt generated by MAC 802.11

   APR -> Link layer ARP Packet

8. Size : The size of pkt at current layer, when pkt goes dwon, size

   [a b c d] : a -> pkt duration in MAC layer header

   b -> The MAC address of destination

   c -> The MAC address of source

   d -> The MAC type of the pkt body

9. Flags :

Source node I/P	Port no of Dest node i/p (-1 Means broadcast)	Port no ip Header ttl	ip of next hop) means node 0 or broadc

The command to open the trace file

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

The above command creates a data trace file called “out.tr”. Within the tcl script, this file is not called explicitly by its names, but instead by pointers that are declared above and called “tracefile1”.

- **NAM** visualization

Command to open the **NAM** file

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

The above command creates a nam visualization file called “out.nam”. Within the tcl script, this files is not called explicitly by its name, but instead by pointers that are declared above and called “namfile”.

- **Termination of the program**

The termination of the program is done using 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 “**flushtrace”** 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
```

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
```

AWK

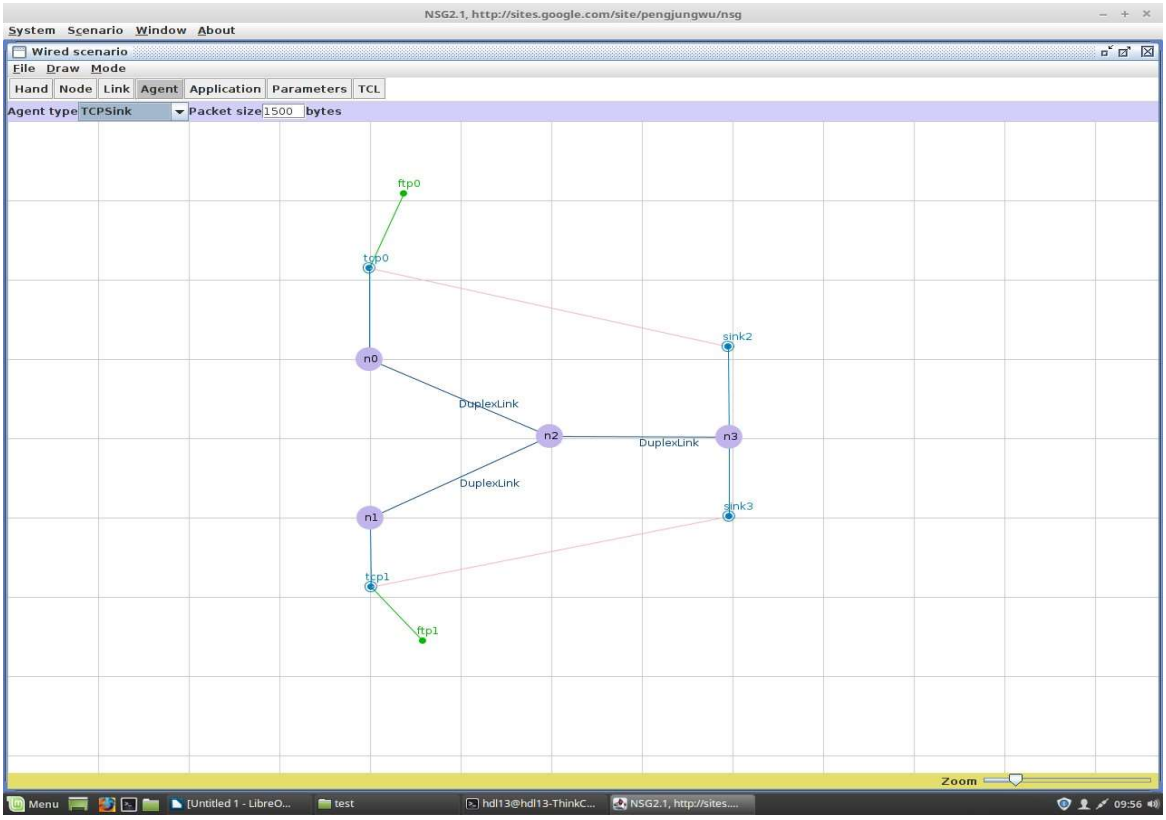
awk is a programmable, pattern-matching, and processing tool available in UNIX. It works equally well with text and numbers. awk is not just a command, but a programming language too. In other words, awk utility is a pattern scanning and processing language. It searches one or more files to see if they contain lines that match specified patterns and then perform associated actions, such as writing the line to the standard output or incrementing a counter each time it finds a match.

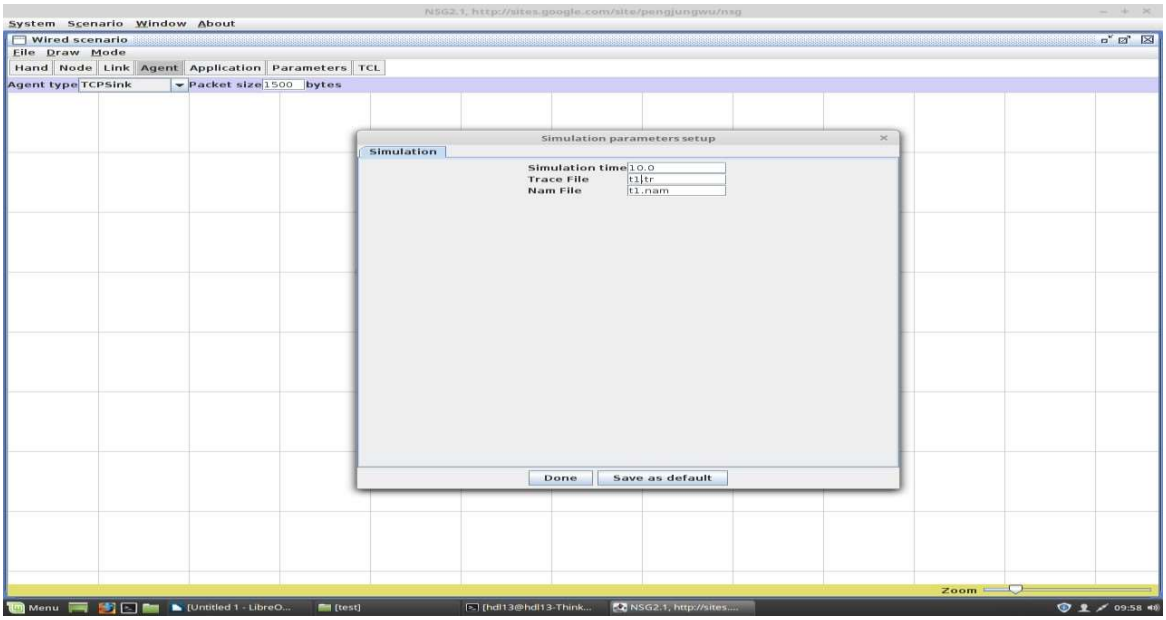
Program 1 : To implement a point to point network with four nodes and duplex links between them. Analyze the network performance by setting queue size and varying the bandwidth.

Procedure:

1) **Create a folder on desktop**
2) **Copy NSG2.1 jar file and paste it in the folder created in step 1.**
3) **Open terminal**
4) **Type ls (lists all files in the folder)**
5) **Type java –jar NSG2.1.jar (a window pops up)**
6) **Select Scenario(drop down list appears)**
7) **Select new wired scenario from the drop down list from step 6.**

Wired scenario for the above program:





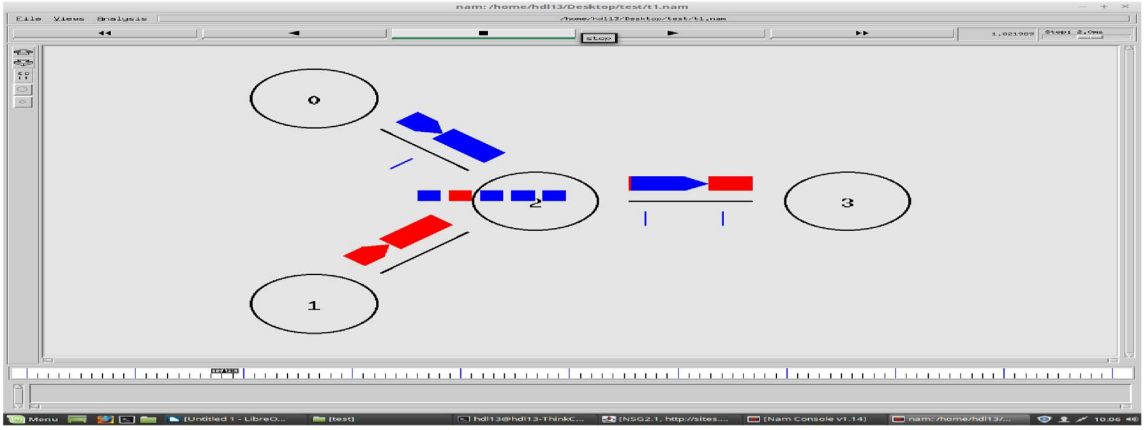

```
# This script is created by NSG2 beta1
#=====================================
#     Simulation parameters setup
#=====================================
set val(stop)   10.0                ;# time of simulation end
 #=====================================
#        Initialization
#=====================================
#Create a ns simulator set ns [new Simulator]
 $ns color 1 Blue
$ns color 2 Red

#Open the NS trace file
set tracefile [open t1.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
 set namfile [open t1.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 2.0Mb 10ms DropTail
$ns queue-limit $n0 $n2 10
$ns duplex-link $n1 $n2 2.0Mb 10ms DropTail
$ns queue-limit $n1 $n2 10
$ns duplex-link $n2 $n3 2.0Mb 10ms DropTail
$ns queue-limit $n2 $n3 10

#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 $n2 $n3 queuePos 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_ 1500
$tcp0 set fid_ 1

#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
$tcp1 set fid_ 2
 #=================================
#       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"

#Setup a FTP Application over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 1.0 "$ftp1 start"
$ns at 2.0 "$ftp1 stop"
 #=================================
#       Termination
#=================================
#Define a 'finish' procedure
proc finish {}
 {
   global ns tracefile namfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec nam t1.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
```

To run the TCL script use the below command

Network Animator for the executed TCL script



Sample of TRACE FILE for the executed TCL script

+ 1 0 2 tcp 40 ------- 1 0.0 3.0 0 0

- 1 0 2 tcp 40 ------- 1 0.0 3.0 0 0

+ 1 1 2 tcp 40 ------- 2 1.0 3.1 0 1

- 1 1 2 tcp 40 ------- 2 1.0 3.1 0 1

r 1.01016 0 2 tcp 40 ------- 1 0.0 3.0 0 0

+ 1.01016 2 3 tcp 40 ------- 1 0.0 3.0 0 0

- 1.01016 2 3 tcp 40 ------- 1 0.0 3.0 0 0

r 1.01016 1 2 tcp 40 ------- 2 1.0 3.1 0 1

+ 1.01016 2 3 tcp 40 ------- 2 1.0 3.1 0 1

- 1.01032 2 3 tcp 40 ------- 2 1.0 3.1 0 1

r 1.02032 2 3 tcp 40 ------- 1 0.0 3.0 0 0

+ 1.02032 3 2 ack 40 ------- 1 3.0 0.0 0 2

- 1.02032 3 2 ack 40 ------- 1 3.0 0.0 0 2

r 1.02048 2 3 tcp 40 ------- 2 1.0 3.1 0 1

+ 1.02048 3 2 ack 40 ------- 2 3.1 1.0 0 3

- 1.02048 3 2 ack 40 ------- 2 3.1 1.0 0 3

r 1.03048 3 2 ack 40 ------- 1 3.0 0.0 0 2

+ 1.03048 2 0 ack 40 ------- 1 3.0 0.0 0 2

- 1.03048 2 0 ack 40 ------- 1 3.0 0.0 0 2

r 1.03064 3 2 ack 40 ------- 2 3.1 1.0 0 3

+ 1.03064 2 1 ack 40 ------- 2 3.1 1.0 0 3

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

```
set ns [new Simulator]set tf
[open lab2.tr w]
$ns trace-all $tf

set nf [open lab2.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]

$n0 label "Ping0"
$n4 label "Ping4"
$n5 label "Ping5"
$n6 label "Ping6"
$n2 label "Router"

$ns color 1 "red"
```

```
$ns color 2 "green"

$ns duplex-link $n0 $n2 100Mb 300ms DropTail
$ns duplex-link $n1 $n2 1Mb 300ms DropTail
$ns duplex-link $n3 $n2 1Mb 300ms DropTail
$ns duplex-link $n5 $n2 100Mb 300ms DropTail
$ns duplex-link $n2 $n4 1Mb 300ms DropTail
$ns duplex-link $n2 $n6 1Mb 300ms DropTail

$ns queue-limit $n0 $n2 5
$ns queue-limit $n2 $n4 3
$ns queue-limit $n2 $n6 2
$ns queue-limit $n5 $n2 5

#The below code is used to connect between the ping agents to the node n0,#n4 ,
n5 and n6.
set ping0 [new Agent/Ping]
$ns attach-agent $n0 $ping0

set ping4 [new Agent/Ping]
$ns attach-agent $n4 $ping4

set ping5 [new Agent/Ping]
$ns attach-agent $n5 $ping5

set ping6 [new Agent/Ping]
$ns attach-agent $n6 $ping6

$ping0 set packetSize_ 50000
$ping0 set interval_ 0.0001
$ping5 set packetSize_ 60000
```

```
$ping5 set interval_ 0.00001
$ping0 set class_ 1
$ping5 set class_ 2

$ns connect $ping0 $ping4
$ns connect $ping5 $ping6

#Define a 'recv' function for the class 'Agent/Ping'
#The below function is executed when the ping agent receives a reply fromthe destination
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts " The node [$node_ id] received an reply from $from with round triptime of $rtt"
}

proc finish {} {
global ns nf tf
exec nam lab2.nam &
$ns flush-traceclose
$tf
close $nf
exit 0
}

#Schedule events
$ns at 0.1 "$ping0 send"
$ns at 0.2 "$ping0 send"
$ns at 0.3 "$ping0 send"
$ns at 0.4 "$ping0 send"
```

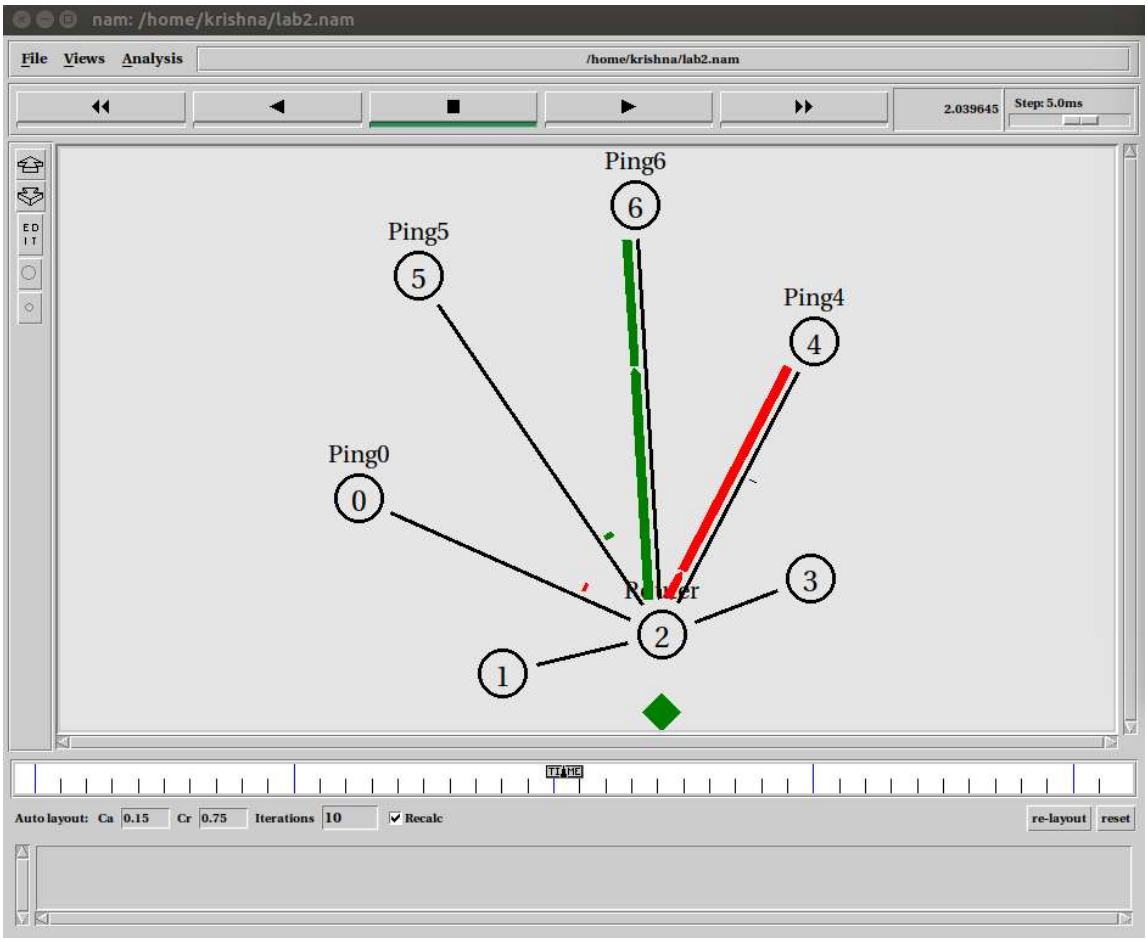
```
$ns at 0.5 "$ping0 send"
$ns at 0.6 "$ping0 send"
$ns at 0.7 "$ping0 send"
$ns at 0.8 "$ping0 send"
$ns at 0.9 "$ping0 send"

$ns at 1.0 "$ping0 send"
$ns at 1.1 "$ping0 send"
$ns at 1.2 "$ping0 send"
$ns at 1.3 "$ping0 send"
$ns at 1.4 "$ping0 send"
$ns at 1.5 "$ping0 send"
$ns at 1.6 "$ping0 send"
$ns at 1.7 "$ping0 send"
$ns at 1.8 "$ping0 send"

$ns at 0.1 "$ping5 send"
$ns at 0.2 "$ping5 send"
$ns at 0.3 "$ping5 send"
$ns at 0.4 "$ping5 send"
$ns at 0.5 "$ping5 send"
$ns at 0.6 "$ping5 send"
$ns at 0.7 "$ping5 send"
$ns at 0.8 "$ping5 send"
$ns at 0.9 "$ping5 send"

$ns at 1.0 "$ping5 send"
$ns at 1.1 "$ping5 send"
$ns at 1.2 "$ping5 send"
$ns at 1.3 "$ping5 send"
$ns at 1.4 "$ping5 send"
$ns at 1.5 "$ping5 send"
$ns at 1.6 "$ping5 send"
```

```
$ns at 1.7 "$ping5 send"
$ns at 1.8 "$ping5 send"

$ns at 5.0 "finish"
$ns run
```

AWK:

```
BEGIN{
count=0;
}
{
if($1=="d")
count++;
}
END
{
printf("The Total no of Packets Drop is :%d\n\n", count);
}
```

Topology:


nam: /home/krishna/lab2.nam
File Views Analysis
/home/krishna/lab2.nam
2.039645
Step: 5.0ms
Ping6
6
Ping5
5
Ping4
4
Ping0
0
3
2
1
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout
reset

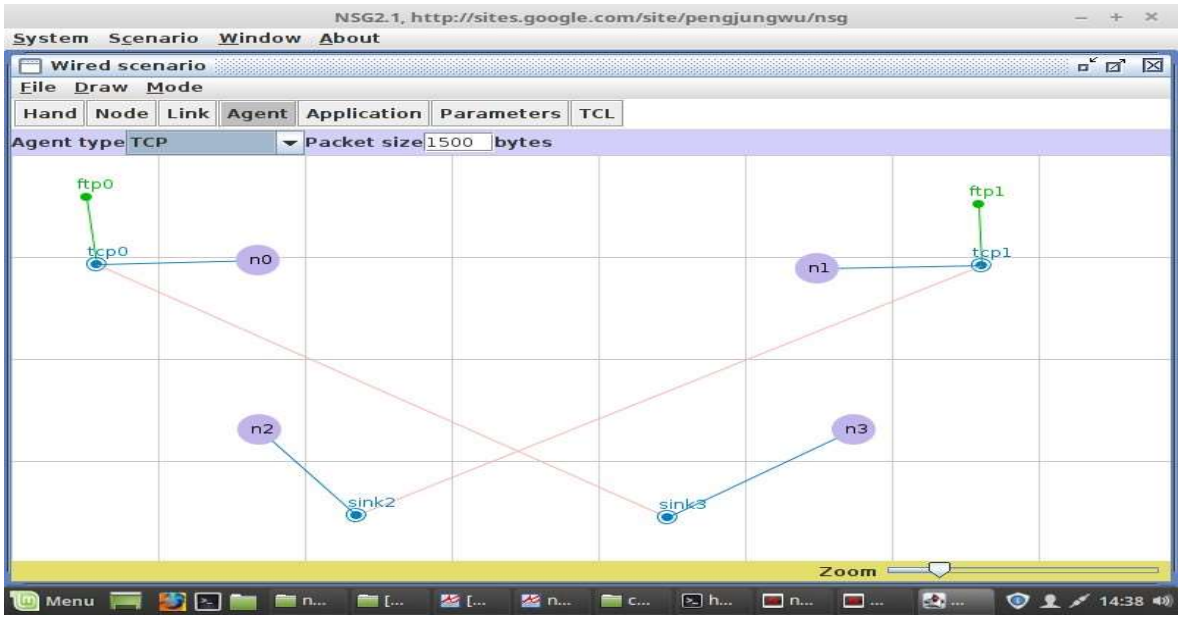
Output:


krishna@ubuntu: ~
krishna@ubuntu:~$ vi lab2.tcl
krishna@ubuntu:~$ vi lab2.awk
krishna@ubuntu:~$ awk -f lab2.awk lab2.tr
The Total no of Packets Drop is :24
krishna@ubuntu:~$

PROGRAM 3

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

Wired scenario for the above program:




```
# This script is created by NSG2 beta1
# <http://wushoupong.googlepages.com/nsg>
 #===================================
#     Simulation parameters setup
#===================================
set val(stop)   10.0                ;# time of simulation end
 #===================================
#        Initialization
#===================================
#Create a ns simulator
```

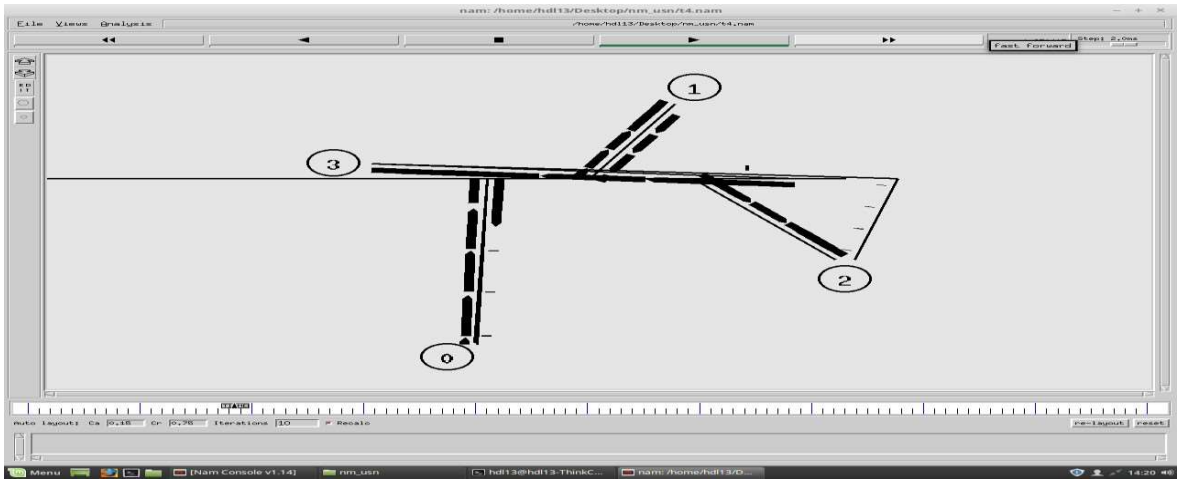
```
set ns [new Simulator]


#Open the NS trace file

set tracefile [open t4.tr w]

$ns trace-all $tracefile


#Open the NAM trace file

set namfile [open t4.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


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

 #Give node position (for NAM)

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

#        Agents Definition
```

```
#=================================

#Setup a TCP connection

 set tcp1 [new Agent/TCP]

$ns attach-agent $n1

$tcp1 set sink4 [new Agent/TCPSink]

$ns attach-agent $n3 $sink4

$ns connect $tcp1 $sink4

$tcp1 set packetSize_ 1500


#Setup a TCP connection

set tcp2 [new Agent/TCP]

$ns attach-agent $n0 $tcp2

set sink3 [new Agent/TCPSink]

 $ns attach-agent $n2 $sink3

$ns connect $tcp2 $sink3

$tcp2 set packetSize_ 1500

  #=================================
#       Applications Definition

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

#Setup a FTP Application over TCP connection

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp1

$ns at 1.0 "$ftp0 start"

$ns at 2.0 "$ftp0 stop"

 #Setup a FTP Application over TCP connection
```

```
set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp2

$ns at 1.0 "$ftp1 start"

$ns at 2.0 "$ftp1 stop"


set file1 [open file1.tr w]
$tcp1 attach $file1

$tcp1 trace cwnd_

$tcp1 set maxcwnd_ 10

set file2 [open file2.tr w]

$tcp2 attach $file2

$tcp2 trace cwnd_

 #=================================
#        Termination

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

#Define a 'finish' procedure

proc finish {} {

    global ns tracefile namfile

    $ns flush-trace

     close $tracefile

     close $namfile

    exec nam t4.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
```

Network Animator for the executed TCL script



4. Develop a program for error detecting code using CRC-CCITT (16- bits).

```
import java.util.Scanner;
import java.io.*;
public class CRC1
{
 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:

Output sample 1:

Enter message bits: 11010011101100

Enter generator: 10011

The checksum code is: 11010011101100101

Enter checksum code: 11010011101100101

Enter generator: 10011

Data stream is valid

Output sample 2:

Enter message bits: 1010101010

Enter generator: 1101

The checksum code is: 1010101010110

Enter checksum code: 1010101010111

Enter generator: 1101

Data stream is invalid. CRC error occurred.

5. Develop a program to implement a sliding window protocol in the data link layer.

```
import java.util.Arrays;
import java.util.Random;

class GoBackNProtocol {
   private final int WINDOW_SIZE;  // Window size
   private final int TOTAL_FRAMES;  // Total number of frames
   private int nextFrameToSend = 0;  // Index of the next frame to send
   private int frameExpectedByReceiver = 0;  // The frame that the receiver expects

   public GoBackNProtocol(int totalFrames, int windowSize) {
      this.TOTAL_FRAMES = totalFrames;
      this.WINDOW_SIZE = windowSize;
   }

   public void sendFrames() {
      Random random = new Random();
      while (nextFrameToSend < TOTAL_FRAMES) {
         // Simulate sending frames in the current window
         for (int i = 0; i < WINDOW_SIZE && nextFrameToSend < TOTAL_FRAMES; i++) {
            System.out.println("Sender: Sending frame " + nextFrameToSend);
            nextFrameToSend++;
         }

         // Simulate receiving an ACK with some random loss
         if (random.nextBoolean()) {
            frameExpectedByReceiver += WINDOW_SIZE;
            System.out.println("Receiver: ACK received for frames up to " + frameExpectedByReceiver);
         } else {
            System.out.println("Receiver: ACK lost, resending frames from " +
(frameExpectedByReceiver));
            nextFrameToSend = frameExpectedByReceiver;  // Go back to the unacknowledged frame
         }
      }

      System.out.println("All frames sent and acknowledged successfully.");
   }
```

```
    public static void main(String[] args) {
        int totalFrames = 10;  // Total number of frames to send
        int windowSize = 4;  // Window size

        GoBackNProtocol protocol = new GoBackNProtocol(totalFrames, windowSize);
        protocol.sendFrames();
    }
}
```

OUTPUT:

Output sample 1:

```
Sender: Sending frame 0
Sender: Sending frame 1
Sender: Sending frame 2
Sender: Sending frame 3
Receiver: ACK received for frames up to 4
Sender: Sending frame 4
Sender: Sending frame 5
Sender: Sending frame 6
Sender: Sending frame 7
Receiver: ACK received for frames up to 8
Sender: Sending frame 8
Sender: Sending frame 9
Receiver: ACK received for frames up to 12
All frames sent and acknowledged successfully.
```

Output sample 2:

```
Sender: Sending frame 0
Sender: Sending frame 1
Sender: Sending frame 2
Sender: Sending frame 3
Receiver: ACK received for frames up to 4
Sender: Sending frame 4
```

Sender: Sending frame 5

Sender: Sending frame 6

Sender: Sending frame 7

Receiver: ACK lost, resending frames from 4

Sender: Sending frame 4

Sender: Sending frame 5

Sender: Sending frame 6

Sender: Sending frame 7

Receiver: ACK received for frames up to 8

Sender: Sending frame 8

Sender: Sending frame 9

Receiver: ACK received for frames up to 12

All frames sent and acknowledged successfully.

6. Develop a program to find the shortest path between vertices using the Bellman-Ford and path vector routing algorithm.

```
import java.util.Arrays;
class Graph {
    class Edge {
        int source, destination, weight;

        Edge(int source, int destination, int weight) {
            this.source = source;
            this.destination = destination;
            this.weight = weight;
        }
    }

    int vertices, edges;
    Edge[] edgeList;

    Graph(int vertices, int edges) {
        this.vertices = vertices;
        this.edges = edges;
        edgeList = new Edge[edges];
    }

    void addEdge(int edgeIndex, int source, int destination, int weight) {
        edgeList[edgeIndex] = new Edge(source, destination, weight);
    }

    void bellmanFord(int startVertex) {
        int[] distances = new int[vertices];
        Arrays.fill(distances, Integer.MAX_VALUE);
        distances[startVertex] = 0;

        for (int i = 1; i < vertices; i++) {
            for (int j = 0; j < edges; j++) {
                int u = edgeList[j].source;
                int v = edgeList[j].destination;
```

```
            int weight = edgeList[j].weight;

            if (distances[u] != Integer.MAX_VALUE && distances[u] + weight < distances[v]) {
                distances[v] = distances[u] + weight;
            }
        }
    }

    // Check for negative-weight cycles
    for (int j = 0; j < edges; j++) {
        int u = edgeList[j].source;
        int v = edgeList[j].destination;
        int weight = edgeList[j].weight;

        if (distances[u] != Integer.MAX_VALUE && distances[u] + weight < distances[v]) {
            System.out.println("Graph contains a negative-weight cycle");
            return;
        }
    }

    printSolution(distances, startVertex);
}

void printSolution(int[] distances, int startVertex) {
    System.out.println("Vertex distances from source vertex " + startVertex + ":");
    for (int i = 0; i < vertices; i++) {
        System.out.println("To Vertex " + i + " is " + distances[i]);
    }
}

void distanceVectorRouting(int[][] graph, int startVertex) {
    int[] distances = new int[vertices];
    Arrays.fill(distances, Integer.MAX_VALUE);
    distances[startVertex] = 0;

    boolean updated;

    do {
        updated = false;
```

```
        for (int u = 0; u < vertices; u++) {
            for (int v = 0; v < vertices; v++) {
                if (graph[u][v] != Integer.MAX_VALUE && distances[u] != Integer.MAX_VALUE &&
                    distances[u] + graph[u][v] < distances[v]) {
                    distances[v] = distances[u] + graph[u][v];
                    updated = true;
                }
            }
        }
    } while (updated);
    printSolution(distances, startVertex);
}

public static void main(String[] args) {
    int vertices = 5;
    int edges = 8;
    Graph graph = new Graph(vertices, edges);
    // Adding edges to the graph
    graph.addEdge(0, 0, 1, -1);
    graph.addEdge(1, 0, 2, 4);
    graph.addEdge(2, 1, 2, 3);
    graph.addEdge(3, 1, 3, 2);
    graph.addEdge(4, 1, 4, 2);
    graph.addEdge(5, 3, 2, 5);
    graph.addEdge(6, 3, 1, 1);
    graph.addEdge(7, 4, 3, -3);

    // Running Bellman-Ford Algorithm
    System.out.println("Bellman-Ford Algorithm:");
    graph.bellmanFord(0);
    // Distance Vector Routing
    System.out.println("\nDistance Vector Routing Algorithm:");
    int[][] routingGraph = {
        {0, -1, 4, Integer.MAX_VALUE, Integer.MAX_VALUE},
        {Integer.MAX_VALUE, 0, 3, 2, 2},
        {Integer.MAX_VALUE, Integer.MAX_VALUE, 0, Integer.MAX_VALUE,
Integer.MAX_VALUE},
        {Integer.MAX_VALUE, 1, 5, 0, Integer.MAX_VALUE},
```

```
            {Integer.MAX_VALUE, Integer.MAX_VALUE, Integer.MAX_VALUE, -3, 0},
        };
        graph.distanceVectorRouting(routingGraph, 0);
    }}
```

OUTPUT :

Output sample 1:

Bellman-Ford Algorithm:
Vertex distances from source vertex 0:
To Vertex 0 is 0
To Vertex 1 is -1
To Vertex 2 is 2
To Vertex 3 is -2
To Vertex 4 is 1
Distance Vector Routing Algorithm:
Vertex distances from source vertex 0:
To Vertex 0 is 0
To Vertex 1 is -1
To Vertex 2 is 2
To Vertex 3 is -2
To Vertex 4 is 1

Output sample 2:
Bellman-Ford Algorithm:
Vertex distances from source vertex 0:
To Vertex 0 is 0
To Vertex 1 is -1
To Vertex 2 is 2
To Vertex 3 is -2
To Vertex 4 is 1

Distance Vector Routing Algorithm:
Vertex distances from source vertex 0:
To Vertex 0 is 0
To Vertex 1 is -1
To Vertex 2 is 2
To Vertex 3 is -2
To Vertex 4 is 1

7.Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.

// TCP Server :

```
import java.net.*;
import java.io.*;
public class TCPS
 {
 public static void main(String[] args)  throws Exception
 {
 ServerSocket sersock=new ServerSocket(4000);
 System.out.println("Server ready for connection");
 Socket sock=sersock.accept();
System.out.println("Connection Is successful and waiting for chatting");
InputStream istream=sock.getInputStream();
BufferedReader fileRead=new BufferedReader(new InputStreamReader(istream));
String fname=fileRead.readLine();
BufferedReader ContentRead=new BufferedReader(new FileReader(fname));
 OutputStream ostream=sock.getOutputStream();
PrintWriter pwrite=new PrintWriter(ostream,true);
String str;
while((str=ContentRead.readLine())!=null)
{
pwrite.println(str);
 }
 sock.close();
sersock.close();
pwrite.close();
fileRead.close();
 ContentRead.close();
}
 }
```

TCP Client:

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

```
public class TCPC
{
 public static void main(String[] args) throws Exception
{
Socket sock=new Socket("127.0.01",4000);
System.out.println("Enter the filename");
 BufferedReader keyRead=new BufferedReader(new InputStreamReader(System.in));
String fname=keyRead.readLine();
OutputStream ostream=sock.getOutputStream();
PrintWriter pwrite=new PrintWriter(ostream,true);
 pwrite.println(fname);
 InputStream istream=sock.getInputStream();
BufferedReader socketRead=new BufferedReader(new InputStreamReader(istream));
String str;
while((str=socketRead.readLine())!=null)
 {
 System.out.println(str);
 }
pwrite.close();
 socketRead.close();
keyRead.close();
}
 }
```

OUTPUT:

Server :

Server ready for connection

Connection Is successful and waiting for chatting

Client :

Enter the filename

sample.txt

Hello, this is a test file.

It contains multiple lines of text.

This is the third line.

8. Develop a program on datagram socket for client/server to display the messages on client side, typed at the server side.

//UDP Sever:

```
 import java.net.*;
import java.net.InetAddress;
class UDPServer
 {
 public static void main(String args[])throws Exception
 {
DatagramSocket serverSocket = new DatagramSocket(9876);
byte[] receiveData=new byte[1024];
byte[] sendData=new byte[1024];
while(true)
 {
System.out.println("Server is Up");
DatagramPacket receivePacket=new DatagramPacket(receiveData,receiveData.length);
serverSocket.receive(receivePacket);
 String sentence=new String(receivePacket.getData());
System.out.println("RECEIVED:"+sentence);
InetAddress IPAddress=receivePacket.getAddress();
int port=receivePacket.getPort();
 String capitalizedSentence=sentence.toUpperCase();
 sendData=capitalizedSentence.getBytes();
 DatagramPacket sendPacket=new DatagramPacket(sendData,sendData.length,IPAddress,port);
serverSocket.send(sendPacket);
}
 }
}
```

//UDP Client :

```
import java.io.*;
import java.net.*;
import java.net.InetAddress;
class UDPClient
{
 public static void main(String[] args)throws Exception
 {
```

```
BufferedReader inFromUser=new BufferedReader(new InputStreamReader(System.in));
DatagramSocket clientSocket=new DatagramSocket();
InetAddress IPAddress=InetAddress.getByName("localhost");
byte[] sendData=new byte[1024];
byte[] receiveData=new byte[1024];
System.out.println("Enter the sting to be converted in to Upper case");
String sentence=inFromUser.readLine(); sendData=sentence.getBytes();
DatagramPacket sendPacket=new DatagramPacket(sendData,sendData.length,IPAddress,9876);
clientSocket.send(sendPacket);
DatagramPacket receivePacket=new DatagramPacket(receiveData,receiveData.length);
clientSocket.receive(receivePacket);
String modifiedSentence=new String(receivePacket.getData());
System.out.println("FROM SERVER:"+modifiedSentence);
clientSocket.close();
}
}
```

OUTPUT :

Output 1:

Server output :

Server is Up
RECEIVED: Hello, Server!
Server is Up
RECEIVED: How are you?

Client output :

Enter the string to be converted into uppercase:
Hello, Server!
FROM SERVER: HELLO, SERVER!
Enter the string to be converted into uppercase:
How are you?
FROM SERVER: HOW ARE YOU?

9. Develop a program for simple RSA algorithm to encrypt and decrypt the data.

```
import java.io.DataInputStream;
 import java.io.IOException;
import java.math.BigInteger;
 import java.util.Random;
public class RSA {
 private BigInteger p,q,N,phi,e,d;
private int bitlength=1024;
private Random r;
public RSA()
 {
r=new Random();
 p=BigInteger.probablePrime(bitlength,r);
 q=BigInteger.probablePrime(bitlength,r);
System.out.println("Prime number p is"+p);
System.out.println("prime number q is"+q);
N=p.multiply(q);
 phi=p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
e=BigInteger.probablePrime(bitlength/2,r);
while(phi.gcd(e).compareTo(BigInteger.ONE)>0&&e.compareTo(phi)<0)
 {
 e.add(BigInteger.ONE);
 }
 System.out.println("Public key is"+e); d=e.modInverse(phi);
System.out.println("Private key is"+d);
 }
public RSA(BigInteger e,BigInteger d,BigInteger N)
 {
this.e=e;
this.d=d;
 this.N=N;
}
public static void main(String[] args)throws IOException
 {
 RSA rsa=new RSA();
 DataInputStream in=new DataInputStream(System.in);
String testString; System.out.println("Enter the plain text:");
testString=in.readLine();
```

```
System.out.println("Encrypting string:"+testString);
System.out.println("string in bytes:"+bytesToString(testString.getBytes()));
byte[] encrypted=rsa.encrypt(testString.getBytes());
byte[] decrypted=rsa.decrypt(encrypted);
System.out.println("Dcrypting Bytes:"+bytesToString(decrypted));
System.out.println("Dcrypted string:"+new String(decrypted));
}
private static String bytesToString(byte[] encrypted)
{
String test=" "; for(byte b:encrypted)
{
test+=Byte.toString(b);
}
return test;
}
public byte[]encrypt(byte[]message)
{
return(new BigInteger(message)).modPow(e,N).toByteArray();
}
public byte[]decrypt(byte[]message)
{
return(new BigInteger(message)).modPow(d,N).toByteArray();
}
}
```

OUTPUT :

Certainly! Here are two sample outputs with single-digit prime numbers for the RSA implementation:

Example 1:

****Input:****

```
**Enter the plain text:**
**Hello, RSA!**
```

****Output:****

```
```

Prime number p is 7

Prime number q is 11

Public key (e) is 3

Modulus (N) is 77

Private key (d) is 47

Encrypting string: Hello, RSA!

String in bytes:

721011081081113441321181101071211083210819210111021081111431171071011161151101081181
1610110108100

Encrypted bytes: 64 11 64 3 5 3 64 3 27 27 23 27 23 5 3 23 2 13 3 2 2 5 2 1 11 5 8 9 5 10 11 11 8 4 3 3 11 4 27 27 18 5 18 18 11 27 5 6 3 23 9 13 5 8 11 3 13 2 7 5 18 9 5 5 18 11 5 2 27 5 5 3 5 5 11 11 5 27 2 23 5 5 6 4 23 27 5 5 11 5 11 27 5 9 2 2 27 9 9 5 6 27 11 8 5 6 27 5 11 2 5 2 5 11 2 5 6 9 27 6 5 11 27 11 27 6 27 6 2 27 5 9 5 9 9 18 27 9 18 5 9 5 9 5 11 6 11 11 5 27 27 2 11 11 27 2 11 27 5 11 27 18 9 9 5 18 27 18 18 2 5 27 18 2 9 27 27 27 2 5 5 11 5 5 11 11 27 2 9 27 11 11 2 5 2 5 6 9 27 9 9 5 6 27 5 27 9 11 2 11 27 27 5 2 11 11 27 6 27 5 27 9 11 9 11 11 5 11 27 9 5 5 2 27 5 11 5 5 11 2 27 27 2 5 27 5 2 11 27 2 5 27 5 5 27 27 27 5 6 27 5 5 9 5 27 5 2 27 11 27 27 6 27 9 9 11 6 5 27 9 9 5 6 2 27 9 5 6 27 27 11 6 27 27 6 5 27 27 9 5 11 27 5 5 11 2 11 27 27 2 5 5 27 9 27 9 27 11 27 2 5 11 27 2 5 27 2 5 5 27 9 27 27 2 27 27 6 27 27 27 27 5 6 27 5 5 9 5 27 11 27 2 9 9 2 27 11 2 9 27 27 2 27 2 9 27 27 6 27 5 2 5 11 27 5 27 5 27 5 27 9 5 2 27 27 5 11 11 27 27 27 27 27 27 5 9 27 5 5 11 27 27 9 11 27 5 27 11 27 27 27 2 11 27 2 5 27 5 27 27 5 11 27 11 27 9 5 27 27 5 27 5 11 2 27 5 5 27 5 2 27 27 27 27 11 27 2 5 27 9 9 2 27 27 5 5 5 5 9 27 9 11 27 11 27 2 11 5 27 27 27 2 27 2 27 2 27 9 27 5 27 5 11 2 5 11 27 2 27 11 27 2 27 5 11 11 2 5 5 27 2 5 2 27 5 9 27 5 27 11 11 5 5 5 5 27 11 27 2 11 27 27 5 2 27 11 27 27 27 11 5 2 27 5 2 27 27 5 27 11 27 27 27 27 2 27 27 11 11 27 2 5 27 27 27 2 27 27 27 9 5 11 27 27 11 27 27 11 27 11 5 27 11 5 27 2 5 11 27 27 27 27 5 27 27 27 5 5 5 27 27 11 2 11 27 27 5 27 5 11 27 2 27 5 27 11 27 27 11 27 27 11 5 27 2 5 27 11 5 27 27 27 5 2 27 5 5 27 27 27 27 27 27 27 5 5 5 27 11 27 5 27 2 27 27 27 27 27 5 27 11 11 27 5 2 27 5 5 27 27 27 27 5 2 11 27 5 27 27 11 27 27 5 27 5 2 27 27 5 5 27 27 27 27 27 5 2 5 2 5 2 11 5 27 11 5 2 27 27 5 11 27 5 5 11 27 27 11 2 27 27 27 5 11 2 27 27 5 27 27 27 27 27 11 27 11 5 27 27 27 27 27 27 5 27 5 5 11 27 27 2 5 5 5 27 11 27 27 27 27 27 2 5 11 27 27 27 5 2 27 27 27 11 27 27 27 27 27 27 5 11 27 27 27 2 27 27 27 27 5 2 27 27 27 27 27 27 27 27 27 27 27 5 27 27 5 5 5 2 27 27 5 27 27 27 2 27 27 5 5 27 27 27 27 27 5 27 2 27 27 27 5 27 27 27 2 27 5 5 27 27 27 27 27 5 27 2 27 27 27 5 27 27 27 2 27 5 5 27 27 27 27 27 27 27 27 27 27 27 5 27 27 5 27 27 5 2 5 27 27 27 5 27 27 5 2 27 5 27 27 5 5 27 27 27 27 27 27 27 27 27 27 27 5 27 27 27 2 27 2 27 27 5 27 27 27 27 5 2 27 27 27 2 5 27 27 5 27 27 5 27 27 27 5 27 5 27 27 27 27 27 27 27 5 5 27 5 27 27 27 27 27 5 27 27 27 27 27 27 27 27 27 27 5 27 5 2 27 27 27 27 27 27 27 5 27 27

10. Develop a program for congestion control using leaky bucket algorithm.

```
import java.util.Scanner;
import java.lang.*;
public class lab7
 {
public static void main(String[] args)
 {
 int i;
 int a[]=new int[20];
int buck_rem=0,buck_cap=4,rate=3,sent,recv;
 Scanner in = new Scanner(System.in);
System.out.println("Enter the number of packets");
 int n = in.nextInt();
System.out.println("Enter the packets");
for(i=1;i<=n;i++)
a[i]= in.nextInt();
 System.out.println("Clock \t packet size \t accept \t sent \t remaining");
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]+ "\t dropped \t" + sent +"\t" +buck_rem);
else
 System.out.println(+i+ "\t\t" +a[i] +"\t\t" +recv +"\t" +sent + "\t" +buck_rem);
 }
 }
 }
```

OUTPUT:

Enter the number of packets:

5

Enter the packets:

3

4

2

1

3

Clock	packet size	accept	sent	remaining
1	3	3	3	1
2	4	dropped	0	4
3	2	2	3	0
4	1	1	1	3
5	3	3	3	1

OUTPUT 2:

Enter the number of packets:

4

Enter the packets:

5

6

2

3

Clock	packet size	accept	sent	remaining
1	5	dropped	0	5
2	6	dropped	0	6
3	2	2	3	0
4	3	3	3	0

Computer Networks VIVA Questions

1. Explain What is Network?
A network is a set of devices connected by physical media links. A network is recursively is a connection of two or more nodes by a physical link or two or more networks connected by one or more nodes.

2. What is a Link?
At the lowest level, a network can consist of two or more computers directly connected by some physical medium such as coaxial cable or optical fiber. Such a physical medium is called as Link.

3. What is a node?
A network can consist of two or more computers directly connected by some physical medium such as coaxial cable or optical fiber. Such a physical medium is called as Links and the computer it connects is called as Nodes.

4. What is a gateway or Router?
A node that is connected to two or more networks is commonly called as router or Gateway. It generally forwards message from one network to another.

5. What is point-point link?
If the physical links are limited to a pair of nodes it is said to be point-point link.

6. What is Multiple Access?
If the physical links are shared by more than two nodes, it is said to be Multiple Access.

7. What are the advantages of Distributed Processing?
a. Security/Encapsulation
b. Distributed database
c. Faster Problem solving
d. Security through redundancy

e. Collaborative Processing

8. What are the criteria necessary for an effective and efficient network?
a. Performance
It can be measured in many ways, including transmit time and response time. b. Reliability
It is measured by frequency of failure, the time it takes a link to recover from a failure, and the networks robustness.
c. Security
Security issues includes protecting data from unauthorized access and virues.

9. Name the factors that affect the performance of the network?
a. Number of Users
b. Type of transmission medium
c. Hardware
d. Software

10. Name the factors that affect the reliability of the network?
a. Frequency of failure
b. Recovery time of a network after a failure

11. Name the factors that affect the security of the network?
a. Unauthorized Access
b. Viruses

12. What is Protocol?
A protocol is a set of rules that govern all aspects of information communication.

13. What are the key elements of protocols?
The key elements of protocols are
a. Syntax
It refers to the structure or format of the data, that is the order in which they are presented.
b. Semantics
It refers to the meaning of each section of bits.
c. Timing
Timing refers to two characteristics: When data should be sent and how fast they can be sent.

14. What are the key design issues of a computer Network?
a. Connectivity
b. Cost-effective Resource Sharing
c. Support for common Services
d. Performance

15. Define Bandwidth and Latency?
Network performance is measured in Bandwidth (throughput) and Latency (Delay). Bandwidth of a network is given by the number of bits that can be transmitted over the network in a certain period of time. Latency corresponds to how long it t5akes a message to

travel from one end off a network to the other. It is strictly measured in terms of time.

16. Define Routing?
The process of determining systematically hoe to forward messages toward the destination nodes based on its address is called routing.

17. What is a peer-peer process?
The processes on each machine that communicate at a given layer are called peer-peer process.

18. When a switch is said to be congested?
It is possible that a switch receives packets faster than the shared link can accommodate and stores in its memory, for an extended period of time, then the switch will eventually run out of buffer space, and some packets will have to be dropped and in this state is said to congested state.

19. What is semantic gap?
Defining a useful channel involves both understanding the applications requirements and recognizing the limitations of the underlying technology. The gap between what applications expects and what the underlying technology can provide is called semantic gap.

20. What is Round Trip Time?
The duration of time it takes to send a message from one end of a network to the other and back, is called RTT.

21. Define the terms Unicasting, Multicasting and Broadcasting.

If the message is sent from a source to a single destination node, it is called Unicasting.
If the message is sent to some subset of other nodes, it is called Multicasting.
If the message is sent to all the m nodes in the network it is called Broadcasting.
22. What is Multiplexing?
Multiplexing is the set of techniques that allows the simultaneous transmission of multiple signals across a single data link.

23. Name the categories of Multiplexing?
a. Frequency Division Multiplexing (FDM)
b. Time Division Multiplexing (TDM)
i. Synchronous TDM
ii. ASynchronous TDM Or Statistical TDM.
c. Wave Division Multiplexing (WDM)

24. What is FDM?
FDM is an analog technique that can be applied when the bandwidth of a link is greater than the combined bandwidths of the signals to be transmitted.

25. What is WDM?

WDM is conceptually the same as FDM, except that the multiplexing and demultiplexing involve light signals transmitted through fiber optics channel.

26. What is TDM?
TDM is a digital process that can be applied when the data rate capacity of the transmission medium is greater than the data rate required by the sending and receiving devices.

27. What is Synchronous TDM?
In STDM, the multiplexer allocates exactly the same time slot to each device at all times, whether or not a device has anything to transmit.

28. List the layers of OSI
a. Physical Layer
b. Data Link Layer
c. Network Layer
d. Transport Layer
e. Session Layer
f. Presentation Layer
g. Application Layer

29. Which layers are network support layers?
a. Physical Layer
b. Data link Layer and
c. Network Layers

30. Which layers are user support layers?
a. Session Layer
b. Presentation Layer and
c. Application Layer

31. Which layer links the network support layers and user support layers?
The Transport layer links the network support layers and user support layers.

32. What are the concerns of the Physical Layer?
Physical layer coordinates the functions required to transmit a bit stream over a physical medium.
a. Physical characteristics of interfaces and media
b. Representation of bits
c. Data rate
d. Synchronization of bits
e. Line configuration
f. Physical topology
g. Transmission mode

33. What are the responsibilities of Data Link Layer?
The Data Link Layer transforms the physical layer, a raw transmission facility, to a reliable

link and is responsible for node-node delivery.
a. Framing
b. Physical Addressing
c. Flow Control
d. Error Control
e. Access Control

34. What are the responsibilities of Network Layer?
The Network Layer is responsible for the source-to-destination delivery of packet possibly across multiple networks (links).
a. Logical Addressing
b. Routing

35. What are the responsibilities of Transport Layer?
The Transport Layer is responsible for source-to-destination delivery of the entire message.
a. Service-point Addressing
b. Segmentation and reassembly
c. Connection Control
d. Flow Control
e. Error Control

36. What are the responsibilities of Session Layer?
The Session layer is the network dialog Controller. It establishes, maintains and synchronizes the interaction between the communicating systems.
a. Dialog control
b. Synchronization

37. What are the responsibilities of Presentation Layer?
The Presentation layer is concerned with the syntax and semantics of the information exchanged between two systems.
a. Translation
b. Encryption
c. Compression

38. What are the responsibilities of Application Layer?
The Application Layer enables the user, whether human or software, to access the network. It provides user interfaces and support for services such as e-mail, shared database management and other types of distributed information services.
a. Network virtual Terminal
b. File transfer, access and Management (FTAM)
c. Mail services
d. Directory Services

39. What are the two classes of hardware building blocks?
Nodes and Links.

40. What are the different link types used to build a computer network?
a. Cables
b. Leased Lines
c. Last-Mile Links
d. Wireless Links

41. What are the categories of Transmission media?
a. Guided Media
i. Twisted Pair cable
1. Shielded TP
2. Unshielded TP
ii. Coaxial Cable
iii. Fiber-optic cable
b. Unguided Media
i. Terrestrial microwave
ii. Satellite Communication

42. What are the types of errors?
a. Single-Bit error
In a single-bit error, only one bit in the data unit has changed
b. Burst Error
A Burst error means that two or more bits in the data have changed.

43. What is Error Detection? What are its methods?
Data can be corrupted during transmission. For reliable communication errors must be deducted and Corrected. Error Detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination. The common Error Detection methods are
a. Vertical Redundancy Check (VRC)
b. Longitudinal Redundancy Check (VRC)
c. Cyclic Redundancy Check (VRC)
d. Checksum

44. What is Redundancy?
The concept of including extra information in the transmission solely for the purpose of comparison. This technique is called redundancy.

45. What is VRC?
It is the most common and least expensive mechanism for Error Detection. In VRC, a parity bit is added to every data unit so that the total number of 1s becomes even for even parity. It can detect all single-bit errors. It can detect burst errors only if the total number of errors in each data unit is odd.

46. What is LRC?
In LRC, a block of bits is divided into rows and a redundant row of bits is added to the whole block. It can detect burst errors. If two bits in one data unit are damaged and bits in exactly

the same positions in another data unit are also damaged, the LRC checker will not detect an error. In LRC a redundant data unit follows n data units.

47. What is CRC?
CRC, is the most powerful of the redundancy checking techniques, is based on binary division.

48. What is Checksum?
Checksum is used by the higher layer protocols (TCP/IP) for error detection

49. List the steps involved in creating the checksum.
a. Divide the data into sections
b. Add the sections together using 1s complement arithmetic
c. Take the complement of the final sum, this is the checksum.

50. What are the Data link protocols?
Data link protocols are sets of specifications used to implement the data link layer. The categories of Data Link protocols are

Asynchronous Protocols
Synchronous Protocols
a. Character Oriented Protocols
b. Bit Oriented protocols
51. Compare Error Detection and Error Correction:
The correction of errors is more difficult than the detection. In error detection, checks only any error has occurred. In error correction, the exact number of bits that are corrupted and location in the message are known. The number of the errors and the size of the message are important factors.

52. What is Forward Error Correction?
Forward error correction is the process in which the receiver tries to guess the message by using redundant bits.

53. Define Retransmission?
Re transmission is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Re sending is repeated until a message arrives that the receiver believes is error-freed.

54. What are Data Words?
In block coding, we divide our message into blocks, each of k bits, called datawords. The block coding process is one-to-one. The same dataword is always encoded as the same codeword.

55. What are Code Words?
r redundant bits are added to each block to make the length n = k + r. The resulting n-bit blocks are called codewords. 2n 2k codewords that are not used. These codewords are invalid

or illegal.

56. What is a Linear Block Code?
A linear block code is a code in which the exclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.

57. What are Cyclic Codes?
Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.

58. Define Encoder?
A device or program that uses predefined algorithms to encode, or compress audio or video data for storage or transmission use. A circuit that is used to convert between digital video and analog video.

59. Define Decoder?
A device or program that translates encoded data into its original format (e.g. it decodes the data). The term is often used in reference to MPEG-2 video and sound data, which must be decoded before it is output.

60. What is Framing?
Framing in the data link layer separates a message from one source to a destination, or from other messages to other destinations, by adding a sender address and a destination address. The destination address defines where the packet has to go and the sender address helps the recipient acknowledge the receipt.
61. What is Fixed Size Framing?
In fixed-size framing, there is no need for defining the boundaries of the frames. The size itself can be used as a delimiter.

62. Define Character Stuffing?
In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC), which has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag.

63. What is Bit Stuffing?
Bit stuffing is the process of adding one extra 0 whenever five consecutive Is follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

64. What is Flow Control?
Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.

65. What is Error Control ?
Error control is both error detection and error correction. It allows the receiver to inform the

sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term error control refers primarily to methods of error detection and re transmission.

66. What Automatic Repeat Request (ARQ)?
Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term error control refers primarily to methods of error detection and retransmission. Error control in the data link layer is often implemented simply: Any time an error is detected in an exchange, specified frames are retransmitted. This process is called automatic repeat request (ARQ).

67. What is Stop-and-Wait Protocol?
In Stop and wait protocol, sender sends one frame, waits until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame.

68. What is Stop-and-Wait Automatic Repeat Request?
Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.

69. What is usage of Sequence Number in Relaible Transmission?
The protocol specifies that frames need to be numbered. This is done by using sequence numbers. A field is added to the data frame to hold the sequence number of that frame. Since we want to minimize the frame size, the smallest range that provides unambiguous communication. The sequence numbers can wrap around.

70. What is Pipelining ?
In networking and in other areas, a task is often begun before the previous task has ended. This is known as pipelining.

71. What is Sliding Window?
The sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver. In other words, he sender and receiver need to deal with only part of the possible sequence numbers.

72. What is Piggy Backing?
A technique called piggybacking is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.

73. What are the two types of transmission technology available?
(i) Broadcast and (ii) point-to-point