GLOBAL ACADEMY OF TECHNOLOGY

Autonomous Institute affiliated to VTU, Belagavi,
Accredited by NAAC with 'A' grade
Ideal Homes Township, RR Nagar, Bengaluru – 560098

Department of Computer Science and Engineering (AI&ML)

COMPUTER NETWORKS
Laboratory Manual

V Semester
Course Code: CML23502

Academic Year 2025-26

Version 1.0
W.e.f, 4th August 2025

Prepared By
Mrs. NASRATH B K
Assistant Professor, Dept. of CSE(AI&ML)

Approved by
HOD,
Dept. of CSE(AI&ML

Vision of the Institute

Become a premier institution imparting quality education in engineering and management to meet the changing needs of society.

Mission of the Institute

M1: Create environment conductive for continuous learning through quality teaching and learning processes supported by modern infrastructure.

M2: Promote research and innovation through collaboration with industries.

M3: Inculcate ethical values and environmental conscious through holistic education programs.

Vision of the Department

To Become a leading hub of excellence in education, research in the field of Computer Science and Engineering providing AI-driven solutions with holistic development for the needs of the Society.

Mission of the Department

- To Provide aspiring engineers for industry and academia by offering excellent education in emerging AI techniques.
- To equip value-added technical and research-oriented education by satisfying societal needs
- To educate with professional integrity values and ethics for environment awareness.

PROGRAM EDUCATIONAL OBJECTIVES(PEOs)

- PEO1: Design and Develop innovative intelligent systems for the welfare of the Society.
- PEO2: Engage in lifelong learning process through higher education and to inculcate innovative ideas in research field.
- PEO3: Lead the IT Industry with management and Entrepreneurship skills.

PROGRAM SPECIFIC OUTCOMES(PSOs)

- PSO1: To Produce graduates with industry-ready skills with the knowledge of Computer Science & Machine Learning technology based to solve real world problems.
- PSO2: Ability to develop many successful applications and design efficient algorithms for intelligent systems within the realm of artificial intelligence incorporating in data analytics, Natural language processing and Internet of Things.

PROGRAM OUTCOMES (PO's)

PO1: Engineering Knowledge:

Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 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 sustainability 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 (WK2 and WK6).

PO6: The Engineer and The World:

Analyse 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 management principles and economic decision- making and apply these to one's own work, 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).

COURSE OUTCOMES

Upon successful completion of this course, students can:

CO502.1	Explain the fundamentals of computer networks.
CO502.2	Apply the concepts of computer networks to demonstrate the working of various layers and protocols in communication network.
CO502.3	Analyze the principles of protocol layering in modern communication systems.
CO502.4	Demonstrate various Routing protocols and their services
CO502.5	Understand principles of application layer protocols

Sl No	**PROGRAMS**
1.	Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth, and find the number of packets dropped.
2.	Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
3.	Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.
4.	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 client/server to display the messages on 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.

Mapping of CO-PO

CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO502.1	2	3	3		3	1			3					
CO502.2	2	3	3		3	1			3					
CO502.3	2	3	3		3	1			3					
CO502.4	2	3	3		3	1			3					3
CO502.5	3	3	3		3	1			3					3
Average	2	3	3		3	1			3					3

Low-1: Medium-2: High-3

Course name: Computer Networks

Course code: CML23502

LAB RUBRICS

Rubrics for Evaluation of Observation + Record

Attribute	Max. Marks	Good	Satisfactory	Poor
		2-3	1-2	0
Writeup	03	• Writes the program without errors.	• Writes the program with few mistakes.	• Unable to write the program.
	Max. Marks	4-5	2-4	0-2
Execution of program	05	• Debugs the program independently. • Executed the program for all possible inputs.	• Works with little help from faculty. • All possible input cases not covered.	• Unable to complete the execution of the program within the lab session.
	Max. Marks	2	1	0
Viva Voce	02	• Able to explain the logic of the program. • Answered all questions.	• Partially understood the logic of the program. • Answered few questions.	• Not understood the logic of the program. • Not answering any questions

Rubrics for Evaluation of Internal Test

Attribute	Max Marks	Good	Satisfactory	Poor
		10-15	5-9	0-4
Writeup	15	• Completes source code. • No syntax and logical errors. • All possible inputs listed with expected output.	• Completes source code. • Syntax and logical errors exist. • All possible inputs listed with expected output.	• Incomplete source code. • Change of program (0 Marks)
	Max Marks	60-70	20-59	0-19
Execution	70	• Able to debug the program and get the right set of outputs.	• Program, executed for some input cases or with very few mistakes	• Program executed only for one input-case/just to accept input, Not executed.
	Max Marks	10-15	5-9	0-4
Viva Voce	15	• Answering all questions.	• Answering few questions.	• Not Able to answer basic questions.

MARKS DISTRIBUTION

a) Continuous Internal Evaluation (CIE)	
1	**Observation + Record (Max. Marks 10) - (A)** **Observation:** Writeup + Execution + Viva Voce = 3+5+2 = **10 M Record: 10 M**
2	**Internal Assessment (Max. Marks 10) - (B)** Writeup + Execution + Viva Voce = 15+70+15 = 100 (scaled down to 10) **Total Marks = A + B = 20 Marks**

Course Coordinator **Module Coordinator** **HOD**

Introduction

- **NS2** is a discrete event network simulator, primarily used for research in network protocols (e.g., TCP, routing, multicast), traffic modelling, and network behaviour simulation.
- It supports various network elements like routers, switches, and network protocols (TCP, UDP, etc.).

Installation: NS2 is available for Linux and can be installed using the following steps:

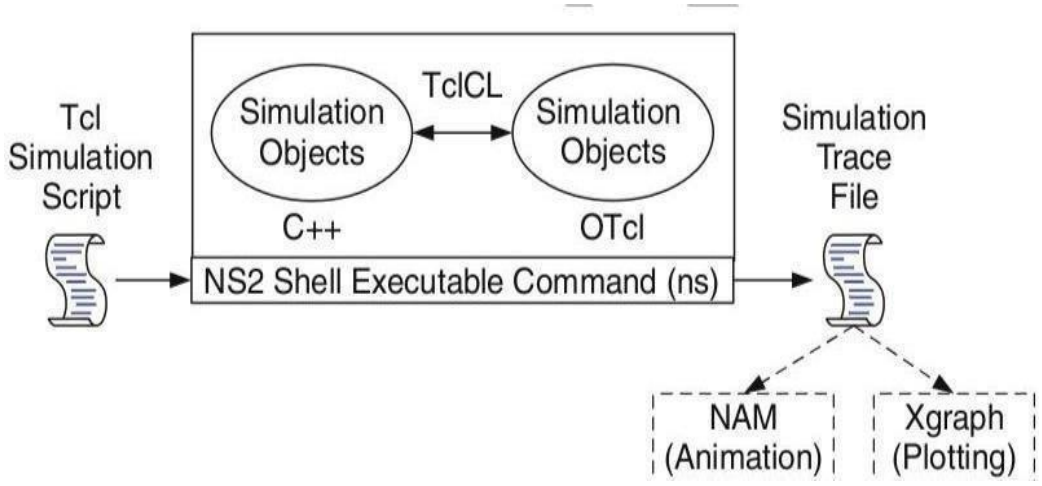
1. Download the NS2 package from the official website.
2. Extract and install it using the terminal:

```
tar -xvf ns-allinone-2.x.x.tar.gz
cd ns-allinone-2.x.x
./install
```

Basic Components of NS2:

- **Nodes:** Represents network devices.
- **Links:** Represents network connections between nodes.
- **Agents:** Define the protocol used (TCP, UDP).
- **Traffic Sources:** Generates data for simulation (e.g., FTP, CBR).
- **Schedulers:** Schedule events to simulate real-time behaviors.

Basic Architecture of NS2



Tcl Scripting:

- NS2 uses **Tcl (Tool Command Language)** for scripting the simulation scenarios.
- A simple NS2 script involves:
    1. **Defining the network:** Create nodes, links, and agents.
    2. **Creating traffic:** Attach traffic generators to simulate data transfer.
    3. **Simulation:** Run the simulation and visualize results using NAM (Network Animator).

NAM (Network Animator):

- **NAM** is a visualization tool used in NS2 to graphically represent the events in a network simulation.
- It reads trace files produced by NS2 during the simulation and provides a visual playback of packet movement, link statuses, node interactions, and more.
- It is useful for understanding how a network operates in real-time and for diagnosing issues like congestion, packet loss, or improper routing.

Key Features:

- Provides an animated view of nodes and packet flow.
- Can show packet drops, queue dynamics, and link usage.
- Offers controls to pause, replay, and navigate through the simulation events.

AWK:

- **AWK** is a powerful pattern-matching and text-processing language commonly used in NS2 for analyzing **trace files**.
- Trace files generated by NS2 contain detailed information about packet events, like when packets are sent, received, dropped, or forwarded.
- AWK helps extract specific metrics like throughput, delay, packet loss, etc., by scanning and processing these trace files based on patterns or conditions.

Trace Files:

- **Trace files** are log files that record all the significant events that occur during a simulation, such as packet transmission, reception, dropping, or routing.
- These files usually have an extension like .tr and are the primary source for post-simulation analysis.

Structure of Trace Files

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

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

1. The first field is the event type. It is given by one of four possible symbols r, +, -, d which correspond respectively to receive (at the output of the link), enqueued, dequeued and dropped.
2. The second field gives the time at which the event occurs.
3. Gives the input node of the link at which the event occurs.
4. Gives the output node of the link at which the event occurs.
5. Gives the packet type (eg CBR or TCP)
6. Gives the packet size
7. Some flags
8. This is the flow id (fid) of IPv6 that a user can set for each flow at the input OTcl script one can further use this field for analysis purposes; it is also used when specifying stream color for the NAM display.
9. This is the source address given in the form of "node.port".
10. This is the destination address, given in the same form.
11. This is the network layer protocol‟s packet sequence number. Even though UDP implementations in a real network do not use sequence number, ns keeps track of UDP packet

sequence number for analysis purposes

12. The last field shows the Unique id of the packet.

The process to structure a Wired NS2 TCL script with the mentioned components:

1. **Create the event scheduler**
   The event scheduler is created using the NS2 simulator object. This object manages the entire simulation process, including the timing and sequencing of events like packet transmissions, receptions, and drops.

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

2. **Open new files & turn on the tracing**
   Before the simulation begins, trace and NAM (Network Animator) files are opened. These files will record the details of the simulation, such as node interactions, packet events, and more. Tracing is turned on so that all events happening during the simulation are logged into these files, which can be analyzed after the simulation is complete.

```
set nf [open out.tr w]  ;# Open a trace file for writing
$ns trace-all $nf       ;# Turn on tracing for all events

set namfile [open out.nam w]  ;# Open NAM file for animation
$ns namtrace-all $namfile     ;# Turn on NAM tracing
```

3. **Create the nodes**
   Nodes represent devices or hosts in the network. Each node can act as a source or destination of traffic. In the script, nodes are created and assigned unique identifiers (like n0, n1, etc.), which allow you to reference them when setting up links and configuring traffic.

```
set n0 [$ns node]  ;# Create node 0
set n1 [$ns node]  ;# Create node 1
```

4. **Setup the links**
   A duplex link (bidirectional link) is set up between nodes. This link connects the nodes and defines the properties of the connection, such as the **bandwidth** (e.g., 1Mbps) and **delay** (e.g., 10ms). The queue type (like **DropTail**) specifies how packets will be handled when the buffer is full.

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail  ;# Link between n0 and n1 with 1Mbps bandwidth
```

5. Configure the traffic type (e.g., TCP, UDP, etc.)

The type of traffic (e.g., TCP or UDP) is configured by attaching traffic agents to the nodes. For TCP, you create a TCP agent at the source node and a TCP sink at the destination node. Similarly, for UDP, you attach a UDP agent to the source node and a Null agent to the destination node. This step defines how data will be sent and received between nodes.

TCP Example

```
set tcp [new Agent/TCP]          ;# Create a TCP agent
$ns attach-agent $n0 $tcp        ;# Attach TCP agent to node n0

set sink [new Agent/TCPSink]     ;# Create a TCP sink (receiver)
$ns attach-agent $n1 $sink       ;# Attach TCP sink to node n1

$ns connect $tcp $sink           ;# Connect TCP sender with the sink
```

UDP Example

```
set udp [new Agent/UDP]          ;# Create a UDP agent
$ns attach-agent $n0 $udp        ;# Attach UDP agent to node n0

set null [new Agent/Null]        ;# Create a Null agent (UDP sink)
$ns attach-agent $n1 $null       ;# Attach Null agent to node n1

$ns connect $udp $null           ;# Connect UDP sender with the sink
```

6. Set the time of traffic generation (e.g., CBR, FTP)

Traffic generation applications like **CBR (Constant Bit Rate)** or **FTP** are used to generate data that is transmitted across the network. You specify the time at which traffic starts by scheduling these applications to begin after a certain delay (e.g., 0.5 seconds). The traffic generator sends packets at regular intervals (for CBR) or in bulk (for FTP).

FTP for TCP

```
set ftp [new Application/FTP]        ;# Create an FTP application
$ftp attach-agent $tcp               ;# Attach FTP to the TCP agent

$ns at 0.5 "$ftp start"              ;# Start FTP at time 0.5 seconds
```

CBR For UDP

```tcl
set cbr [new Application/Traffic/CBR]  ;# Create a CBR traffic generator
$cbr set packetSize_ 1000              ;# Set packet size
$cbr set interval_ 0.01                ;# Set the sending interval (10ms)
$cbr attach-agent $udp                 ;# Attach CBR to the UDP agent

$ns at 0.5 "$cbr start"                ;# Start CBR at time 0.5 seconds
```

7. Terminate the simulation

To stop the simulation after a certain period, you define a procedure (like finish) that closes the trace and NAM files, flushes all remaining events to disk, and stops the simulator. This ensures all events are recorded properly and the simulation ends cleanly. The finish procedure is scheduled to execute at a specified time (e.g., 5 seconds), marking the end of the simulation. After that, the network animator (NAM) is executed, allowing you to visualize the simulation events.

```
proc finish {} {
    global ns nf namfile
    $ns flush-trace             ;# Flush all traces
    close $nf                   ;# Close the trace file
    close $namfile              ;# Close the NAM file
    exec nam out.nam &          ;# Open NAM file for animation
    exit 0                      ;# Terminate the simulation
}

$ns at 5.0 "finish"             ;# Schedule finish procedure at time 5.0
$ns run                         ;# Run the simulation
```

AWK in NS2 Context:

- **Trace Analysis**: AWK is most useful for analyzing NS2 trace files by filtering specific events (e.g., packet sent, received, or dropped), calculating performance metrics (e.g., throughput, delay, PDR), and processing large amounts of network data efficiently.
- **Automating Metrics**: AWK scripts can be written to automate the process of calculating key network metrics such as packet loss rate, throughput, and latency from trace files after running an NS2 simulation.

By leveraging these AWK commands, you can efficiently analyze and process the large volumes of trace

data generated in NS2 simulations.

1. **Print specific fields**: awk '{print $1, $2}' :
   This prints the **first** and **second** columns of each line in the trace file. In NS2 trace files, fields might represent event types, times, packet ids, etc.

2. **Count matching lines**: awk '$1 == "r" {count++} END {print count}'
   This command counts and prints the total number of packets that were **received** (denoted by "r" in NS2 trace files).

3. **Filter and print matching rows**: awk '$1 == "d"'
   This command prints all lines where the **first column** is "d" (representing a **dropped packet**) and the **third column** is "tcp" (for **TCP traffic**).

4. **Field separators:** awk -F, '{print $1, $2}'
   By default, AWK splits lines into fields using spaces or tabs, but you can change the **field separator** using the -F option.

XGRAPH

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

Syntax:

Xgraph [options] file-name

Options are listed here
/-bd <color> (Border)
This specifies the border color of the xgraph window.
/-bg <color> (Background)
This specifies the background color of the xgraph window.
/-fg<color> (Foreground)
This specifies the foreground color of the xgraph window.
/-lf <fontname> (LabelFont)
All axis labels and grid labels are drawn using this font.
/-t<string> (Title Text)
This string is centered at the top of the graph.
/-x <unit name> (XunitText)
This is the unit name for the x-axis. Its default is “X”.
/-y <unit name> (YunitText)
This is the unit name for the y-axis. Its default is “Y”.

Experiment 1:

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

Step1: Open text editor, type the below program and save with extention .tcl (prog1.tcl)

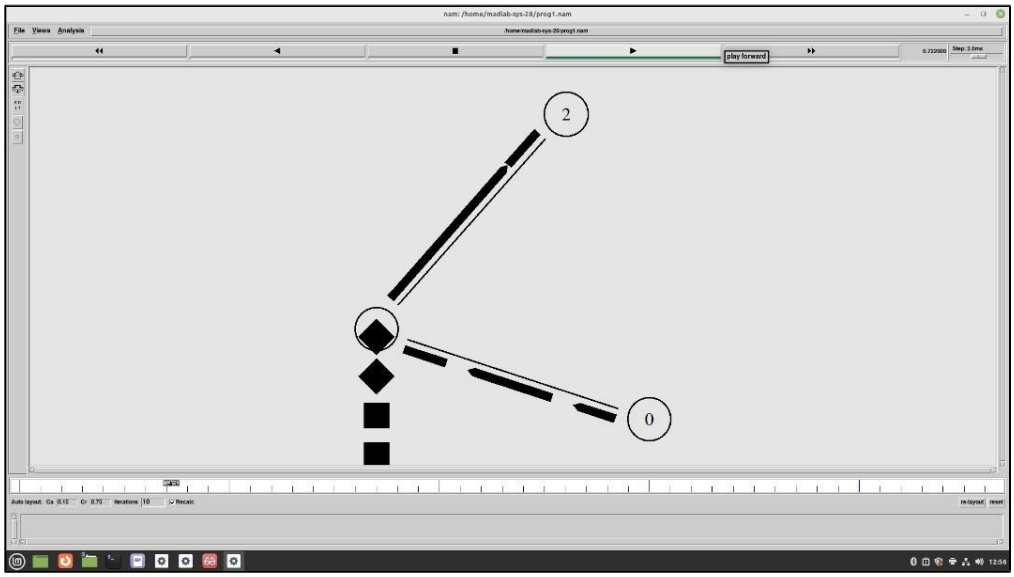
```
# Create a new Simulator instance
set ns [new Simulator]

# Open files to store NAM and trace output
set nf [open prog1.nam w]
$ns namtrace-all $nf
set nd [open prog1.tr w]
$ns trace-all $nd

# Define a procedure to end the simulation and close the trace files
proc finish { } {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog1.nam &
exit 0
}

# Create network nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

# Create duplex links between nodes with specific bandwidth, delay, and queue management
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 512kb 10ms DropTail

# Set a queue limit on the link between n1 and n2
$ns queue-limit $n1 $n2 10

# Attach a UDP agent to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

# Create a CBR (Constant Bit Rate) traffic generator
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
```

```
# Create a CBR (Constant Bit Rate) traffic generator
$cbr0 attach-agent $udp0

# Attach a NULL agent (sink) to node n2 to act as a receiver
set sink [new Agent/Null]
$ns attach-agent $n2 $sink

# Connect the UDP agent on node n0 to the sink on node n2
$ns connect $udp0 $sink

# Schedule simulation events (start/stop traffic and end the simulation)

$ns at 0.2 "$cbr0 start"      # Start CBR traffic at 0.2 seconds
$ns at 4.5 "$cbr0 stop"       # Stop CBR traffic at 4.5 seconds
$ns at 5.0 "finish"           # End the simulation at 5.0 seconds $ns run

# Run the simulation
$ns run
```

Step2: Open text editor, type the below program and save with extention .awk (prog1.awk)

```
BEGIN {
dcount = 0;
rcount = 0;
}
{
event = $1;
if(event == "d")
{
dcount++;
}

if(event == "r")
{
rcount++;
}
}
END {
printf("The no.of packets dropped : %d\n ",dcount);
printf("The no.of packets recieved : %d\n ",rcount);
}
```

Step3: Run the simulation program [root@localhost~] # ns prog1.tcl
(Here “ns” indicates network simulator. We get the topology shown in the snapshot.)

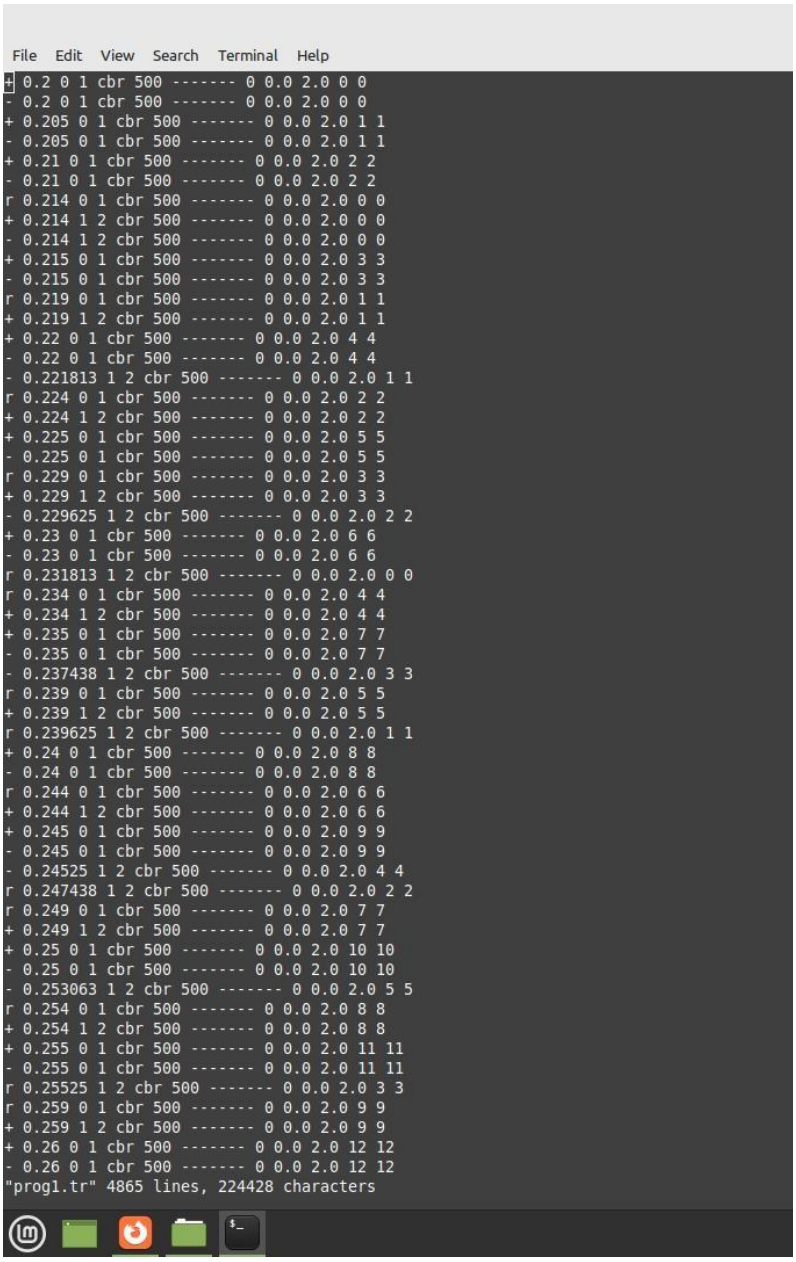


Step 4: Now press the play button in the simulation window and the simulation will begins.

Step 5: After simulation is completed run awk file to see the output ,
[root@localhost~]# awk –f prog1.awk prog1.tr

```
File  Edit  View  Search  Terminal  Help
madlab-sys-28@madlabsys28-MS-7D48:~$ gedit prog1.tcl
gedmadlab-sys-28@madlabsys28-MS-7D48:~$
madlab-sys-28@madlabsys28-MS-7D48:~$ gedit prog1.awk
madlab-sys-28@madlabsys28-MS-7D48:~$ ns prog1.tcl
madlab-sys-28@madlabsys28-MS-7D48:~$ Cannot connect to existing nam instance. Starting a new one...

madlab-sys-28@madlabsys28-MS-7D48:~$ awk -f prog1.awk prog1.tr
The no.of packets dropped : 301
 The no.of packets recieved : 1421
 madlab-sys-28@madlabsys28-MS-7D48:~$ ns prog1.tcl
madlab-sys-28@madlabsys28-MS-7D48:~$ awk -f prog1.awk prog1.tr
The no.of packets dropped : 301
 The no.of packets recieved : 1421
madlab-sys-28@madlabsys28-MS-7D48:~$
```

Step 6: To see the trace file contents open the file as , [root@localhost~]# vi prog1.tr



Experiment 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.

Step1: Open text editor, type the below program and save with extention .tcl (prog2.tcl)

```
# Create a new instance of the NS2 simulator
set ns [new Simulator]

# Open the files for writing the NAM (Network Animator) and trace outputs
set nf [open prog2.nam w]            # Open NAM trace file
$ns namtrace-all $nf                 # Enable NAM tracing for all events
set nd [open prog2.tr w]
$ns trace-all $nd

# Define the finish procedure to end the simulation
proc finish {} {
global ns nf nd
$ns flush-trace
close $nf
close $nd
exec nam prog2.nam &
 exit 0
}

# Create network nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]

# Establish duplex links (bidirectional links) between nodes and n0
$ns duplex-link $n1 $n0 1Mb 10ms DropTail
$ns duplex-link $n2 $n0 1Mb 10ms DropTail
$ns duplex-link $n3 $n0 1Mb 10ms DropTail
$ns duplex-link $n4 $n0 1Mb 10ms DropTail
$ns duplex-link $n5 $n0 1Mb 10ms DropTail
$ns duplex-link $n6 $n0 1Mb 10ms DropTail

# Create a custom procedure for the Ping agent to display the RTT (round-trip time)
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id] recieved ping answer from \
$from with round-trip-time $rtt ms." }
```

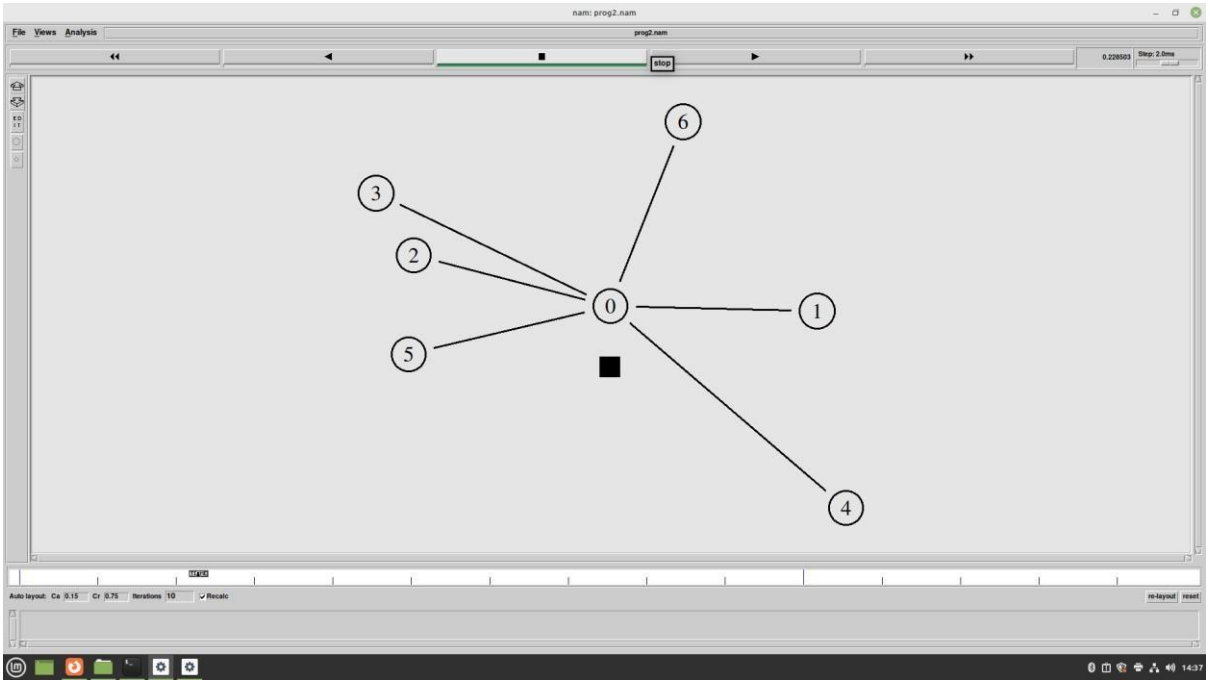
```
# Create Ping agents for each node

set p1 [new Agent/Ping]
set p2 [new Agent/Ping]
set p3 [new Agent/Ping]
set p4 [new Agent/Ping]
set p5 [new Agent/Ping]
set p6 [new Agent/Ping]

# Attach Ping agents to their respective nodes
$ns attach-agent $n1 $p1
$ns attach-agent $n2 $p2
$ns attach-agent $n3 $p3
$ns attach-agent $n4 $p4
$ns attach-agent $n5 $p5
$ns attach-agent $n6 $p6

# Set queue limits on links between node n0 and nodes n4, n5, and n6
$ns queue-limit $n0 $n4 3
$ns queue-limit $n0 $n5 2
$ns queue-limit $n0 $n6 2

# Connect Ping agents to simulate traffic
$ns connect $p1 $p4
$ns connect $p2 $p5
$ns connect $p3 $p6

# Schedule Ping packet transmissions at specific times
$ns at 0.2 "$p1 send"
$ns at 0.4 "$p2 send"
$ns at 0.6 "$p3 send"
$ns at 1.0 "$p4 send"
$ns at 1.2 "$p5 send"
$ns at 1.4 "$p6 send"

# Schedule the simulation to end at 2.0 seconds
$ns at 2.0 "finish"

# Schedule the simulation to end at 2.0 seconds
$ns run
```

Step2: Open text editor, type the below program and save with extention .awk (prog2.awk)

```
BEGIN {
count=0;
}
{
event=$1; if(event=="d")
{
```

```
count++;
}
}  END {
printf("No of packets dropped : %d\n",count);
}
```

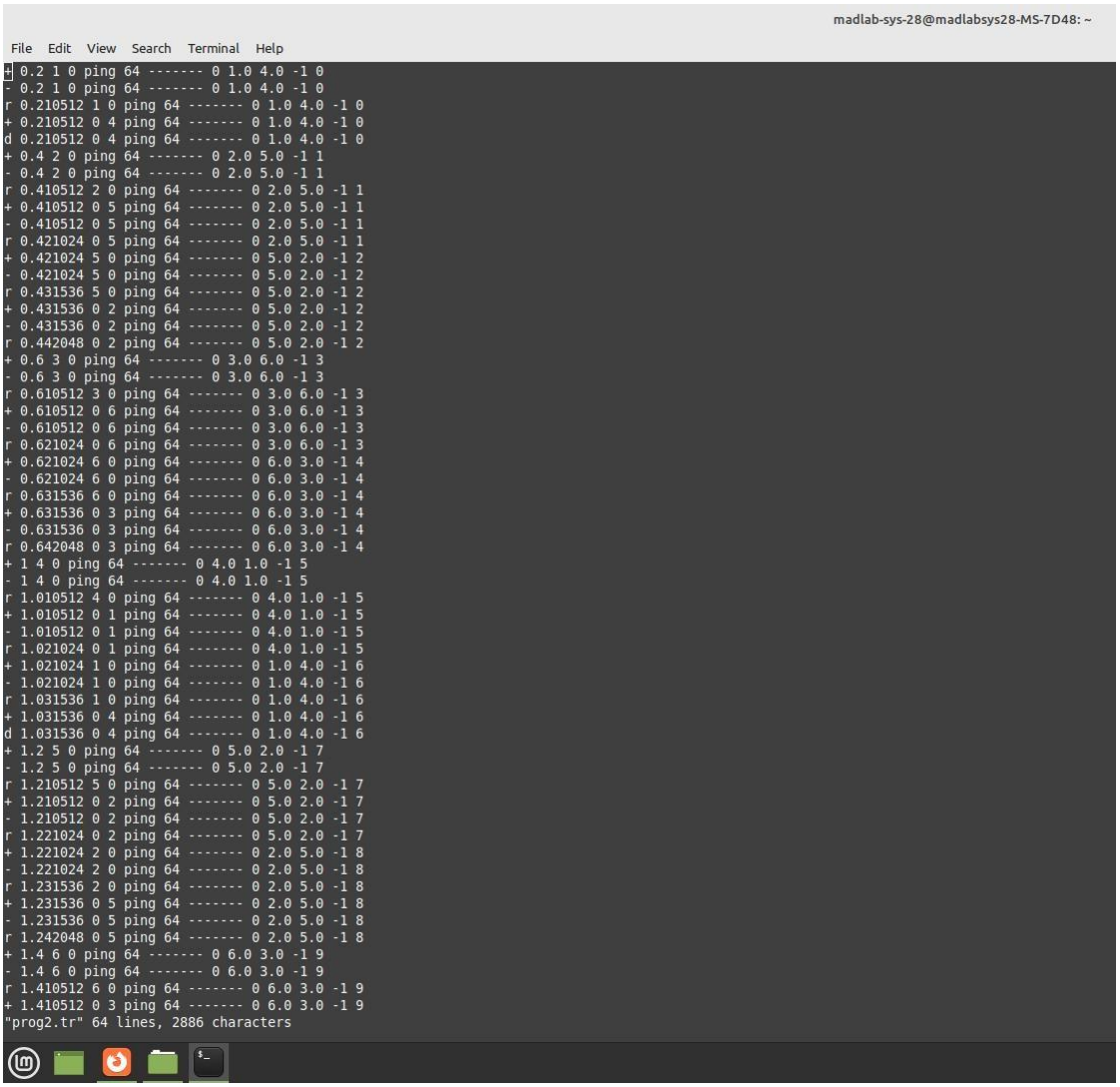
Step3: Run the simulation program [root@localhost~]# ns prog2.tcl
(Here "ns" indicates network simulator. We get the topology shown in the snapshot.)




```
File  Edit  View  Search  Terminal  Help
madlab-sys-28@madlabsys28-MS-7D48:~$ awk -f prg2.awk prog2.tr
No of packets dropped : 0
madlab-sys-28@madlabsys28-MS-7D48:~$ gedit prog2.tcl

madlab-sys-28@madlabsys28-MS-7D48:~$
madlab-sys-28@madlabsys28-MS-7D48:~$ ns prog2.tcl
node 2 recieved ping answer from  5 with round-trip-time 42.0 ms.
node 3 recieved ping answer from  6 with round-trip-time 42.0 ms.
node 5 recieved ping answer from  2 with round-trip-time 42.0 ms.
node 6 recieved ping answer from  3 with round-trip-time 42.0 ms.
madlab-sys-28@madlabsys28-MS-7D48:~$ awk -f prg2.awk prog2.tr
No of packets dropped : 2
madlab-sys-28@madlabsys28-MS-7D48:~$
```

Step 4: Now press the play button in the simulation window and the simulation will begins.
Step 5: After simulation is completed run awk file to see the output , [root@localhost~]# awk –f prog2.awk prog2.tr

Step 6: To see the trace file contents open the file as , [root@localhost~]# vi prog2.tr



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

Types of TCP Traffic:

1. **TCP Class 1 (TCP Vegas):**
   - TCP Vegas is a congestion control algorithm that emphasizes proactive congestion avoidance.
   - It estimates the network's bandwidth and adjusts the sending rate by comparing the expected and actual round-trip times (RTT).
   - Vegas increases its sending rate when it detects that the network is under-utilized and decreases the rate when it senses potential congestion.
   - This makes TCP Vegas more sensitive to early signs of congestion, avoiding packet loss more efficiently than TCP Reno.
   - In this simulation, class_ 1 refers to the TCP Vegas traffic on the n1 ↔ n7 link.

2. **TCP Class 2 (TCP Reno):**
   - TCP Reno is a reactive congestion control algorithm that uses Additive Increase Multiplicative Decrease (AIMD).
   - TCP Reno increases its congestion window gradually (additive increase) but reacts sharply by cutting the window in half upon detecting packet loss (multiplicative decrease).
   - Reno is simpler but less efficient compared to Vegas, as it waits for packet loss to occur before reducing its sending rate.
   - In this simulation, class_ 2 refers to the TCP Reno traffic on the n2 ↔ n8 link.

Key Differences between TCP Vegas and TCP Reno:

- Congestion Control Approach:
  - TCP Vegas tries to avoid congestion by adjusting the sending rate based on RTT estimates.
  - TCP Reno reacts to congestion after it has occurred, typically due to packet loss.
- Packet Loss Sensitivity:
  - TCP Vegas is more sensitive to early congestion and does not rely on packet loss for congestion control.
  - TCP Reno only reduces its rate after detecting packet loss, which can lead to more dropped packets during heavy traffic.

Congestion Window (CWND)

- The Congestion Window (CWND) is a dynamic TCP parameter that controls the amount of data a sender can transmit without receiving an acknowledgment (ACK).
- It grows or shrinks based on network conditions to manage congestion. CWND increases during normal operation (Additive Increase) but is reduced drastically when packet loss is detected (Multiplicative Decrease).
- The size of CWND determines the data flow rate, making it a key part of TCP's congestion control mechanism.

Step1: Open text editor, type the below program and save with extention .tcl (prog3.tcl)

```
# Create a new simulator object
set ns [new Simulator]

# Open a NAM trace file to visualize the network topology and simulation
set nf [open prog3.nam w]
$ns namtrace-all $nf
set nd [open prog3.tr w]
$ns trace-all $nd

# Set colors for nodes
$ns color 1 Blue
$ns color 2 Red

# Procedure to close the trace files and execute the NAM visualization tool
proc finish { } {
global ns nf nd
$ns flush-trace
close $nf close $nd
exec nam prog3.nam &
exit 0
}

# Define network nodes n0 to n8
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]

# Customize node shapes and colors for visualization
$n7 shape box
$n7 color Blue
$n8 shape hexagon
$n8 color Red

# Create duplex links between nodes with specified bandwidth, delay, and queue management type
$ns duplex-link $n1 $n0 2Mb 10ms DropTail
$ns duplex-link $n2 $n0 2Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 20ms DropTail
 # Create a LAN between nodes n3 to n8 with specific bandwidth, delay, and MAC protocol
$ns make-lan "$n3 $n4 $n5 $n6 $n7 $n8" 512Kb 40ms LL Queue/DropTail Mac/802_3

# Set orientations of the duplex links for better visualization
```

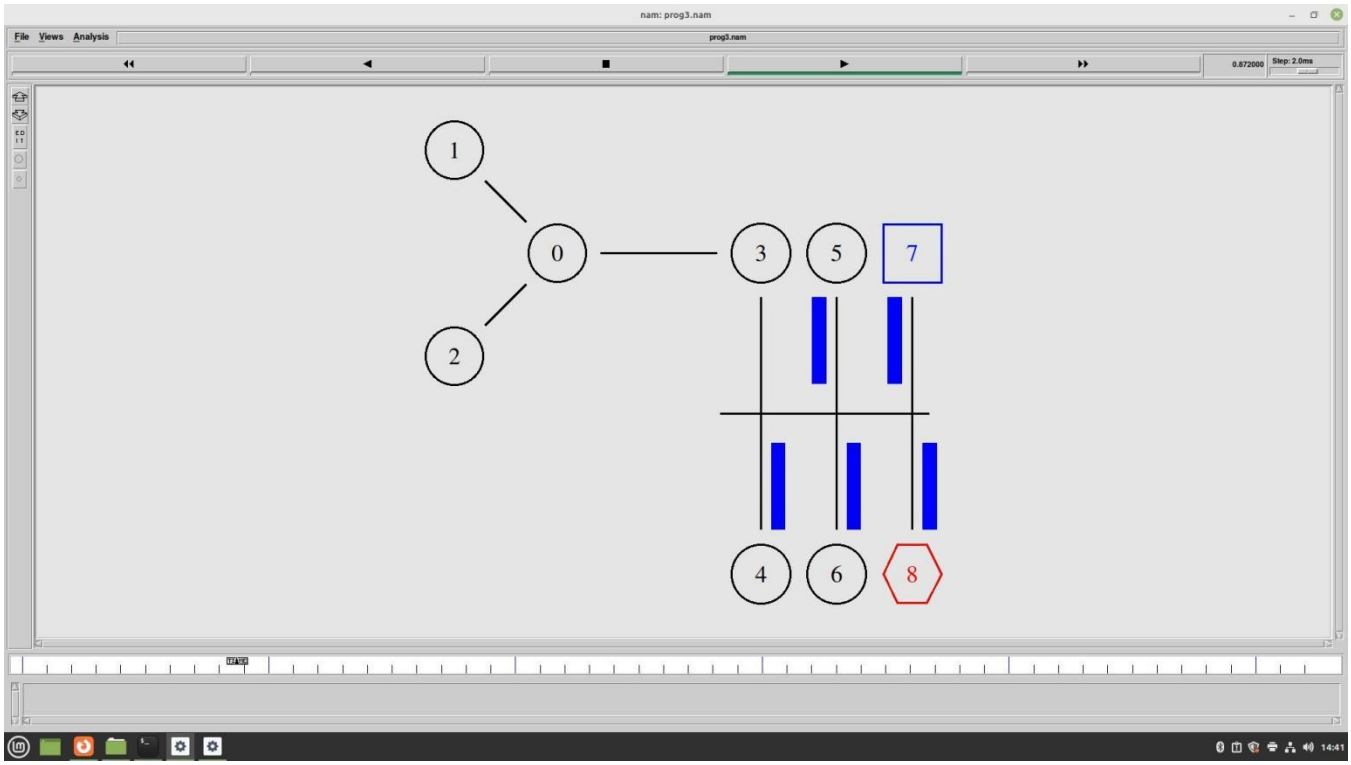
```
$ns duplex-link-op $n1 $n0 orient right-down
$ns duplex-link-op $n2 $n0 orient right-up
$ns duplex-link-op $n0 $n3 orient right

# Set queue limit for link between n0 and n3
$ns queue-limit $n0 $n3 20

# Setup TCP Vegas agent on node n1 and attach it to a TCPSink on node n7
set tcp1 [new Agent/TCP/Vegas]

$ns attach-agent $n1 $tcp1
set sink1 [new Agent/TCPSink]

$ns attach-agent $n7 $sink1
$ns connect $tcp1 $sink1
$tcp1 set class_ 1                                  # Set the TCP class to 1
$tcp1 set packetsize_ 55

# Attach FTP application to TCP Vegas agent on node n1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

# Open a trace file to log congestion window (cwnd) for TCP Vegas
set tfile [open cwnd.tr w]
$tcp1 attach $tfile
$tcp1 trace cwnd_

# Setup TCP Reno agent on node n2 and attach it to a TCPSink on node n8
set tcp2 [new Agent/TCP/Reno]
$ns attach-agent $n2 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $n8 $sink2
$ns connect $tcp2 $sink2
$tcp2 set class_ 2
$tcp2 set packetSize_ 55

# Attach FTP application to TCP Reno agent on node n2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

# Open a trace file to log congestion window (cwnd) for TCP Reno
set tfile2 [open cwnd2.tr w]
$tcp2 attach $tfile2
$tcp2 trace cwnd_

# Schedule FTP start/stop events for both connection
$ns at 0.5 "$ftp1 start"
$ns at 1.0 "$ftp2 start"
$ns at 5.0 "$ftp2 stop"
```

```
$ns at 5.0 "$ftp1 stop"

$ns at 5.5 "finish"
$ns run
```

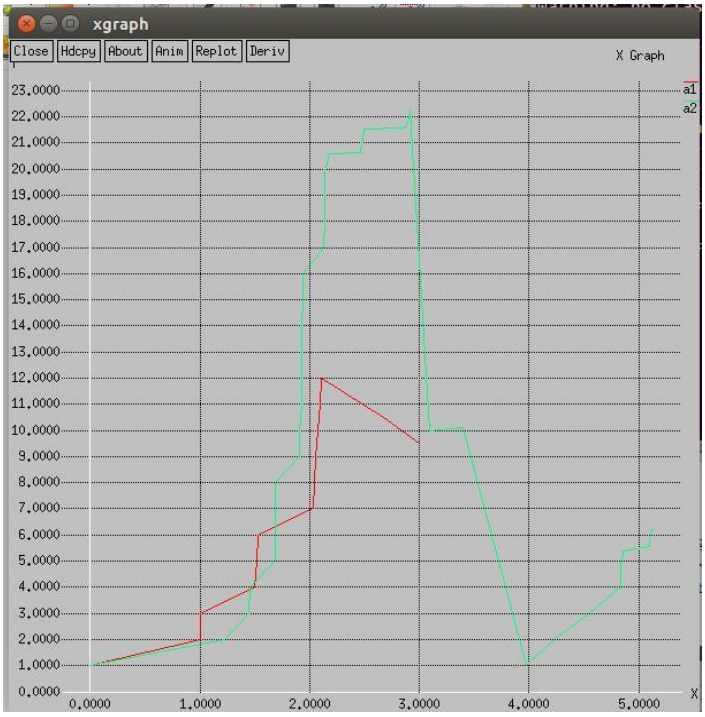
Step2: Open text editor, type the below program and save with extention .awk (prog3.awk)

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

Step3: Run the simulation program [root@localhost~]# ns prog3.tcl
(Here "ns" indicates network simulator. We get the topology shown in the snapshot.)



Step 4: Now press the play button in the simulation window and the simulation will begins.
Step 5: After simulation is completed run awk file and generate the graph , [root@localhost~]# awk –f prog3.awk cwnd.tr > a1 [root@localhost~]# awk –f prog3.awk cwnd2.tr > a2 [root@localhost~]#xgraph a1 a2



Step 6: To see the trace file contents open the file as ,

[root@localhost~]# vi prog3.tr

```
+ 0.5 1 0 tcp 1000 ------- 1 1.0 7.0 0 0
- 0.5 1 0 tcp 1000 ------- 1 1.0 7.0 0 0
r 0.514 1 0 tcp 1000 ------- 1 1.0 7.0 0 0
+ 0.514 0 3 tcp 1000 ------- 1 1.0 7.0 0 0
- 0.514 0 3 tcp 1000 ------- 1 1.0 7.0 0 0
r 0.542 0 3 tcp 1000 ------- 1 1.0 7.0 0 0
h 0.542 3 9 tcp 1000 ------- 1 1.0 7.0 0 0
+ 0.582 3 9 tcp 1000 ------- 1 1.0 7.0 0 0
- 0.582 3 9 tcp 1000 ------- 1 1.0 7.0 0 0
r 0.637848 9 7 tcp 1000 ------- 1 1.0 7.0 0 0
h 0.637848 7 9 ack 40 ------- 1 7.0 1.0 0 1
+ 0.677848 7 9 ack 40 ------- 1 7.0 1.0 0 1
- 0.677848 7 9 ack 40 ------- 1 7.0 1.0 0 1
r 0.718852 9 3 ack 40 ------- 1 7.0 1.0 0 1
+ 0.718852 3 0 ack 40 ------- 1 7.0 1.0 0 1
- 0.718852 3 0 ack 40 ------- 1 7.0 1.0 0 1
r 0.739172 3 0 ack 40 ------- 1 7.0 1.0 0 1
+ 0.739172 0 1 ack 40 ------- 1 7.0 1.0 0 1
- 0.739172 0 1 ack 40 ------- 1 7.0 1.0 0 1
r 0.749332 0 1 ack 40 ------- 1 7.0 1.0 0 1
+ 0.749332 1 0 tcp 1000 ------- 1 1.0 7.0 1 2
- 0.749332 1 0 tcp 1000 ------- 1 1.0 7.0 1 2
r 0.763332 1 0 tcp 1000 ------- 1 1.0 7.0 1 2
"prog3.tr" 4175 lines, 196769 characters
```

Experiment 4:

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

Theory

CRC(Cyclic Redundancy Check) is an error detecting technique used in digital networks and storage devices to detect the accidental changes to raw data. It cannot be used for correcting errors.

If an error is detected in the received message, a „Negative acknowledgement" is sent to the sender. The sender and the receiver agree upon a fixed polynomial called generator polynomial. The standard agreed generator polynomial is x16+x12+x5+x0 (any polynomial can be considered, of degree 16).
The CRC does error checking via polynomial division. The generated polynomial g(x) = x16+x12+x5+x0

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1

So the g(x) value is 10001000000100001

Algorithm:

1. Given a bit string (message to be sent), append 16 0^S to the end of it (the number of 0^s is the same as the degree of the generator polynomial) let this string + 0^S be called as modified string B
2. Divide B by agreed on polynomial g(x) and determine the remainder R(x). The 16-bit remainder received is called as checksum.
3. The message string is appended with checksum and sent to the receiver.
4. At the receiver side, the received message is divided by generator polynomial g(x).

If the remainder is 0, the receiver concludes that there is no error occurred otherwise, the receiver concludes an error occurred and requires a retransmission

PROGRAM:

```
#include <stdio.h>
#define MAX 200   // Maximum bits size
// Function to perform division
void divide(int divisor[], int rem[], int divisor_bits, int rem_length) {
    int cur = 0;
    while (1) {
        for (int i = 0; i < divisor_bits; i++) {
            rem[cur + i] = rem[cur + i] ^ divisor[i];  // XOR division
        }
        while (rem[cur] == 0 && cur != rem_length - 1) {
            cur++;
        }
        if ((rem_length - cur) < divisor_bits)
            break;
    }
}
int main() {
    int data[MAX], div[MAX], divisor[MAX], rem[MAX], crc[MAX];
    int data_bits, divisor_bits, tot_length;
    // Input number of data bits
    printf("Enter number of data bits: ");
    scanf("%d", &data_bits);
    printf("Enter data bits:\n");
    for (int i = 0; i < data_bits; i++)
        scanf("%d", &data[i]);
    // CRC-CCITT polynomial = 10001000000100001 (17 bits)
     divisor_bits = 17;
    int fixed_divisor[17] = {1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1};
    for (int i = 0; i < divisor_bits; i++)
        divisor[i] = fixed_divisor[i];
    // Total length = data + (divisor_bits - 1) zeros
    tot_length = data_bits + divisor_bits - 1;
```

```
// Append data into div[]
for (int i = 0; i < data_bits; i++)
    div[i] = data[i];
for (int i = data_bits; i < tot_length; i++)
    div[i] = 0;

// Copy dividend into remainder
for (int j = 0; j < tot_length; j++)
    rem[j] = div[j];

printf("Dividend (after appending 0's): ");
for (int i = 0; i < tot_length; i++)
    printf("%d", div[i]);
printf("\n");

// Perform division
divide(divisor, rem, divisor_bits, tot_length);

// Generate CRC codeword
for (int i = 0; i < tot_length; i++)
    crc[i] = div[i] ^ rem[i];

printf("CRC code: ");
for (int i = 0; i < tot_length; i++)
 printf("%d", crc[i]);
    printf("\n");

    // ---------------- ERROR DETECTION ----------------
    printf("Enter received codeword (%d bits):\n", tot_length);
    for (int i = 0; i < tot_length; i++)
        scanf("%d", &crc[i]);

    // Copy received codeword to remainder
    for (int j = 0; j < tot_length; j++)
```

```
        rem[j] = crc[j];


    divide(divisor, rem, divisor_bits, tot_length);
    int error = 0;
    for (int i = 0; i < tot_length; i++) {
        if (rem[i] != 0) {
            error = 1;
            break;
        }
    }
    if (error)
        printf("Error detected in received message.\n");
    else
        printf("No error detected.\n");


    printf("THANK YOU.\n");
    return 0;
}
```

OUTPUT:

```
"C:\Users\Kiran danush\Desktop\New folder\prg1\bin\Debug\prg1.exe"
Enter number of data bits: 3
Enter data bits (0 or 1):
1
1
0

Step 1: Appended Data (Dividend): 1100000000000000000
Step 2: CRC Checksum: 0110000011000110
Step 3: Transmitted Data (Data + CRC): 1100110000011000110

Step 4: Enter received data (19 bits):
1
1
0
0
1
1
0
0
0
0
0
1
1
0
0
0
1
1
0
Step 5: Result: Γ£à No error detected in received data.
Thank you.

Process returned 0 (0x0)   execution time : 206.928 s
Press any key to continue.
```

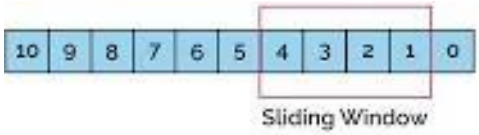
Experiment 5:

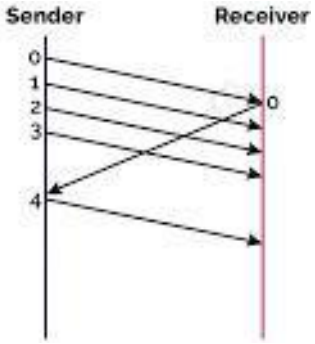
Develop a program to implement a sliding window protocol in the data link layer.

The sliding window is a technique for sending multiple frames at a time. It controls the data packets between the two devices where reliable and gradual delivery of data frames is needed. It is also used in TCP (Transmission Control Protocol).

In this technique, each frame has sent from the sequence number. The sequence numbers are used to find the missing data in the receiver end. The purpose of the sliding window technique is to avoid duplicate data, so it uses the sequence number.
Sliding Window Protocol allows a sender to send multiple packets before needing an acknowledgment (ACK) from the receiver.

- The window size defines how many packets the sender can send without receiving an ACK.
- As the sender receives ACKs, the window "slides" forward, allowing more packets to be sent.
- The protocol ensures that packets are sent in order and resent if necessary, making it useful for reliable data transfer (such as TCP in computer networks).
-





PROGRAM:

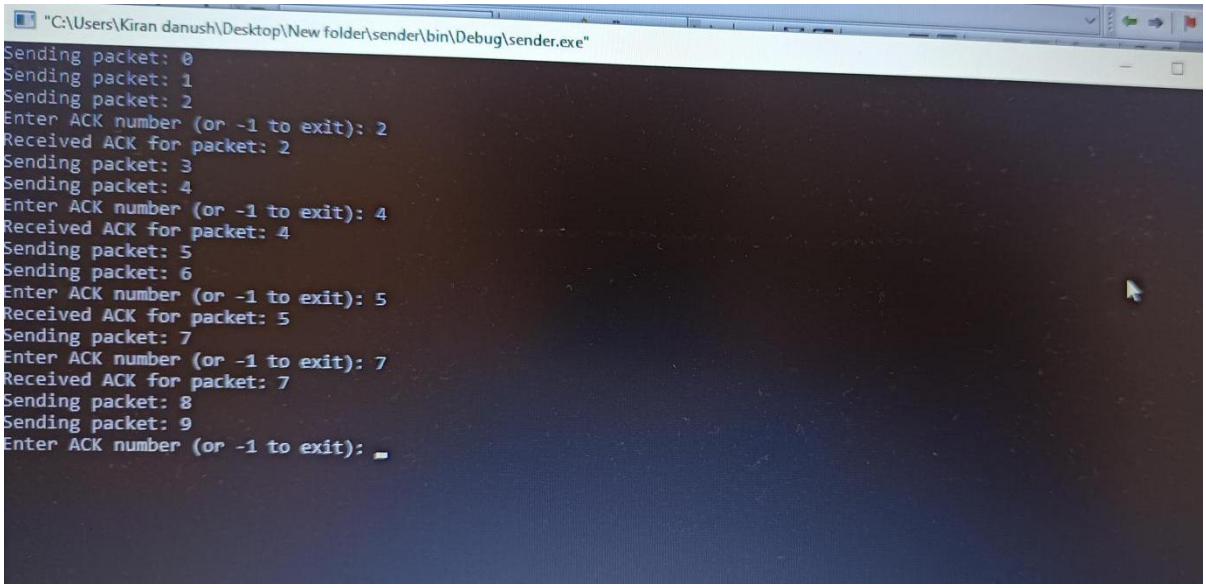
```
#include <stdio.h>
#include <unistd.h>   // for usleep()
#define MAX 100


// Sender structure
typedef struct {
    int windowSize;
    int window[MAX];
    int front, rear;
    int nextSeqNum;
    int ackNum;
} Sender;
```

```
// Initialize sender
void initSender(Sender *s, int windowSize) {
    s->windowSize = windowSize;
    s->front = 0;
    s->rear = -1;
    s->nextSeqNum = 0;
    s->ackNum = 0;
}
// Enqueue packet
void enqueue(Sender *s, int seq) {
    if (s->rear < MAX - 1) {
        s->rear++;
        s->window[s->rear] = seq;
    }
}
// Dequeue packet
void dequeue(Sender *s) {
    if (s->front <= s->rear) {
        s->front++;
    }
}
// Peek front of queue
int peek(Sender *s) {
    if (s->front <= s->rear)
        return s->window[s->front];
    return -1;
}
// Send packets within window
void sendPacket(Sender *s) {
    while (s->nextSeqNum < s->ackNum + s->windowSize) {
        printf("Sending packet: %d\n", s->nextSeqNum);
        enqueue(s, s->nextSeqNum);
        s->nextSeqNum++;
```

```
        usleep(500000); // simulate delay (0.5 sec)
    }
}

// Receive ACK
void receiveAck(Sender *s, int ack) {
    if (ack > s->ackNum && ack < s->nextSeqNum) {
        printf("Received ACK for packet: %d\n", ack);
        s->ackNum = ack;

        // Remove acknowledged packets
        while (s->front <= s->rear && peek(s) <= s->ackNum) {
            dequeue(s);
        }
         // Send next packets
                sendPacket(s);
            } else {
                printf("Invalid ACK: %d\n", ack);
            }
        }

        // Receiver simulation (manual ACK input)
        void simulateReceiver(Sender *s) {
            int ack;
            while (1) {
                printf("Enter ACK number (or -1 to exit): ");
                scanf("%d", &ack);

                if (ack == -1) {
                    printf("Exiting...\n");
                    break;
                }
                receiveAck(s, ack);
```

```
        }
    }
    int main() {
        int windowSize = 3;   // Change window size if needed
        Sender sender;
        initSender(&sender, windowSize);

        // Start sending packets
        sendPacket(&sender);

        // Receiver interaction
        simulateReceiver(&sender);

        return 0;
    }
```

OUTPUT



Experiment 6

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

Routing algorithm is a part of network layer software which is responsible for deciding which output line an incoming packet should be transmitted on. If the subnet uses datagram internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If the subnet uses virtual circuits (connection Oriented), routing decisions are made only when a new established route is being set up.

Routing algorithms can be grouped into two major classes: adaptive and nonadaptive. Nonadaptive algorithms do not base their routing decisions on measurement or estimates of current traffic and topology. Instead, the choice of route to use to get from *I* to *J* (for all *I* and *J*) is compute in advance, offline, and downloaded to the routers when the network ids booted. This procedure is sometime called static routing.

Adaptive algorithms, in contrast, change their routing decisions to reflect changes in the topology, and usually the traffic as well. Adaptive algorithms differ in where they get information (e.g., locally, from adjacent routers, or from all routers), when they change the routes (e.g., every ΔT sec, when the load changes, or when the topology changes), and what metric is used for optimization (e.g., distance, number of hops, or estimated transit time).

Two algorithms in particular, distance vector routing and link state routing are the most popular. Distance vector routing algorithms operate by having each router maintain a table (i.e., vector) giving the best known distance to each destination and which line to get there. These tables are updated by exchanging information with the neighbors.

The distance vector routing algorithm uses Bellman-Ford routing algorithm and Ford- Fulkerson algorithm. In distance vector routing, each router maintains a routing table that contains two parts: the preferred out going line to use for that destination, and an estimate of the time or distance to that destination. The metric used might be number of hops, time delay in milliseconds, total number of packets queued along the path, or something similar.

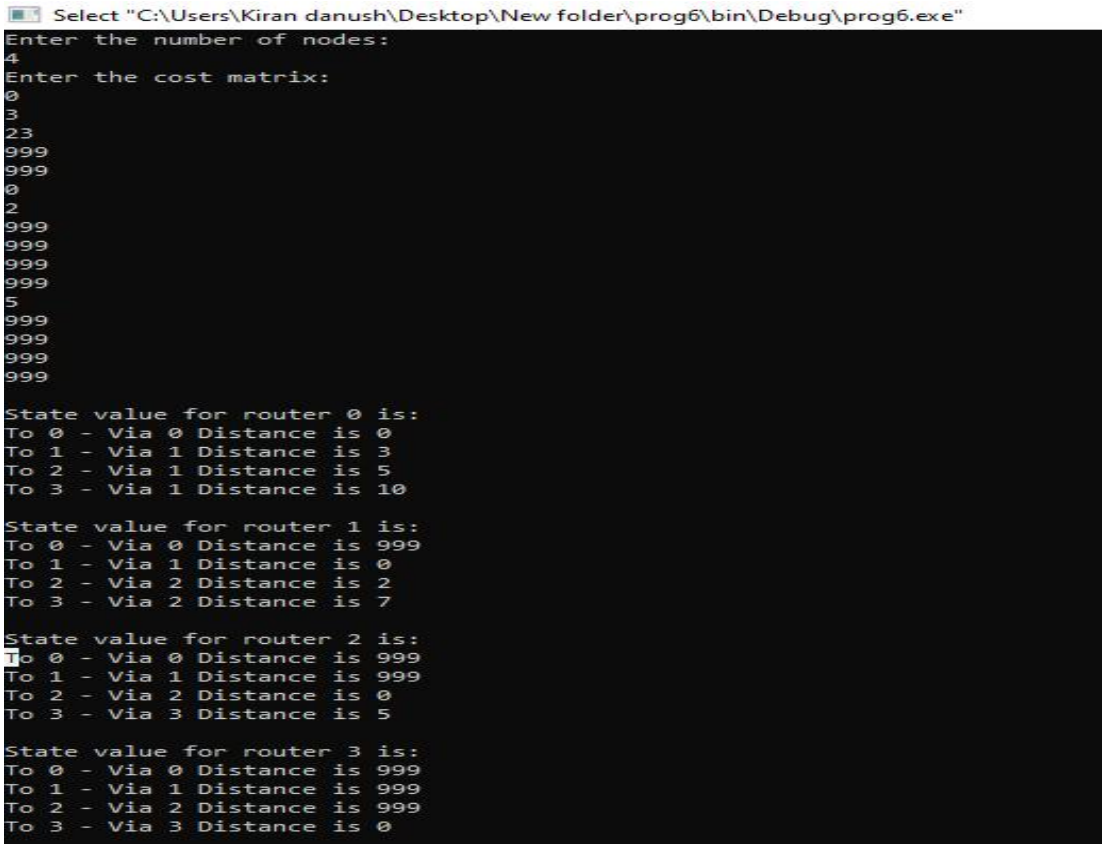
The Routing tables are shared among the neighbors, and the tables at the router are updated, such that the router will know the shortest path to the destination.

Program

```
#include <stdio.h>
int main() {
    int n, i, j, k, count;
    int dmat[10][10], dist[10][10], via[10][10];
    printf("Enter the number of nodes: ");
    scanf("%d", &n);
    printf("Enter the cost matrix:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &dmat[i][j]);
            if (i == j)
                dmat[i][j] = 0; // cost to itself is 0
            dist[i][j] = dmat[i][j];
            via[i][j] = j; // initially direct connection
        }
    }
    // Distance Vector Algorithm
    do {
        count = 0;
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                for (k = 0; k < n; k++) {
                    if (dist[i][j] > dmat[i][k] + dist[k][j]) {
                        dist[i][j] = dmat[i][k] + dist[k][j];
                        via[i][j] = k;
                        count++;
                    }
                }
            }
        }
    } while (count != 0);
    // Output routing table
```

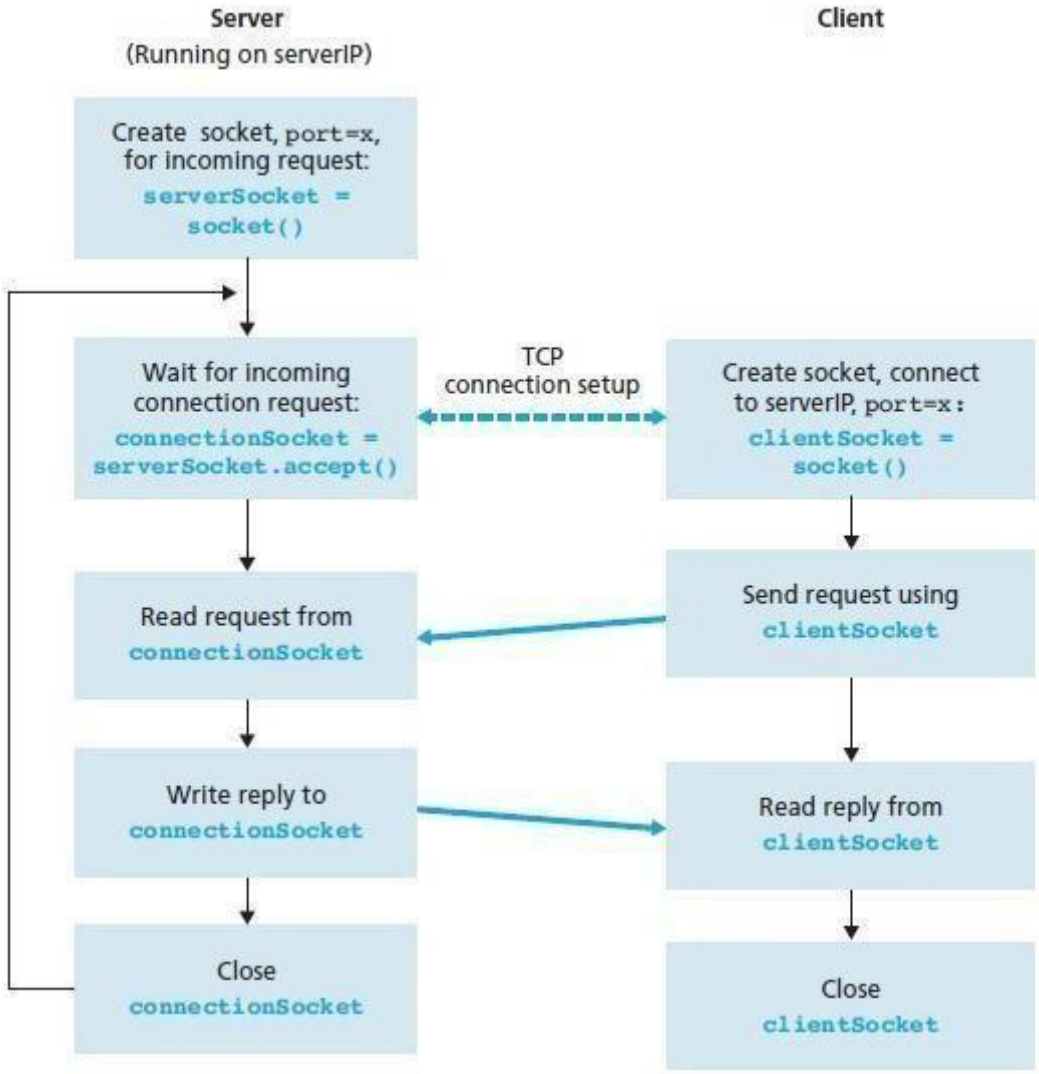
```
    for (i = 0; i < n; i++) {
        printf("\nState value for router %d:\n", i);
        for (j = 0; j < n; j++) {
      printf("To %d - Via %d Distance = %d\n", j, via[i][j], dist[i][j]);
        }
    }
    return 0;
}
```

Output:



Experiment 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.



Server Program

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 4000
#define BUF_SIZE 1024
```

```
int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char filename[BUF_SIZE], buffer[BUF_SIZE];
    FILE *fp;

    // Create socket
    server_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (server_fd == -1) {
        perror("Socket creation failed");
        exit(1);
    }

    // Bind socket
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("Bind failed");
        close(server_fd);
        exit(1);
    }

    // Listen for connections
    if (listen(server_fd, 3) < 0) {
        perror("Listen failed");
        close(server_fd);
        exit(1);
    }
    printf("Server ready. Waiting for connection...\n");
```

```
    // Accept connection
    new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t *)&addrlen);
    if (new_socket < 0) {
        perror("Accept failed");
        close(server_fd);
        exit(1);
    }
    printf("Connection established!\n");

    // Read filename from client
    read(new_socket, filename, BUF_SIZE);
    printf("Client requested file: %s\n", filename);

    // Open file
    fp = fopen(filename, "r");
    if (fp == NULL) {
        char *msg = "File not found\n";
        send(new_socket, msg, strlen(msg), 0);
    } else {
        // Send file content line by line
        while (fgets(buffer, BUF_SIZE, fp) != NULL) {
            send(new_socket, buffer, strlen(buffer), 0);
        }
        fclose(fp);
    }

    close(new_socket);
    close(server_fd);
    return 0;
}
```

Client Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 4000
#define BUF_SIZE 1024

int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;
    char filename[BUF_SIZE], buffer[BUF_SIZE];

    // Create socket
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock < 0) {
        perror("Socket creation error");
        exit(1);
    }
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    // Convert IPv4 address from text to binary
    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
        perror("Invalid address / Address not supported");
        exit(1);
    }
   // Connect to server
    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        perror("Connection Failed");
```

```
        exit(1);
    }

    // Get filename from user
    printf("Enter filename: ");
    scanf("%s", filename);

    // Send filename to server
    send(sock, filename, strlen(filename), 0);

    // Receive file contents
    printf("\n--- File Content ---\n");
    int n;
    while ((n = read(sock, buffer, BUF_SIZE - 1)) > 0) {
        buffer[n] = '\0';
        printf("%s", buffer);
    }
    printf("\n--------------------\n");

    close(sock);
    return 0;
}
```

OUTPUT:

How to Run

1. Save the above programs as server.c and client.c.
2. Compile them:

   gcc server.c -o server
   gcc client.c -o client

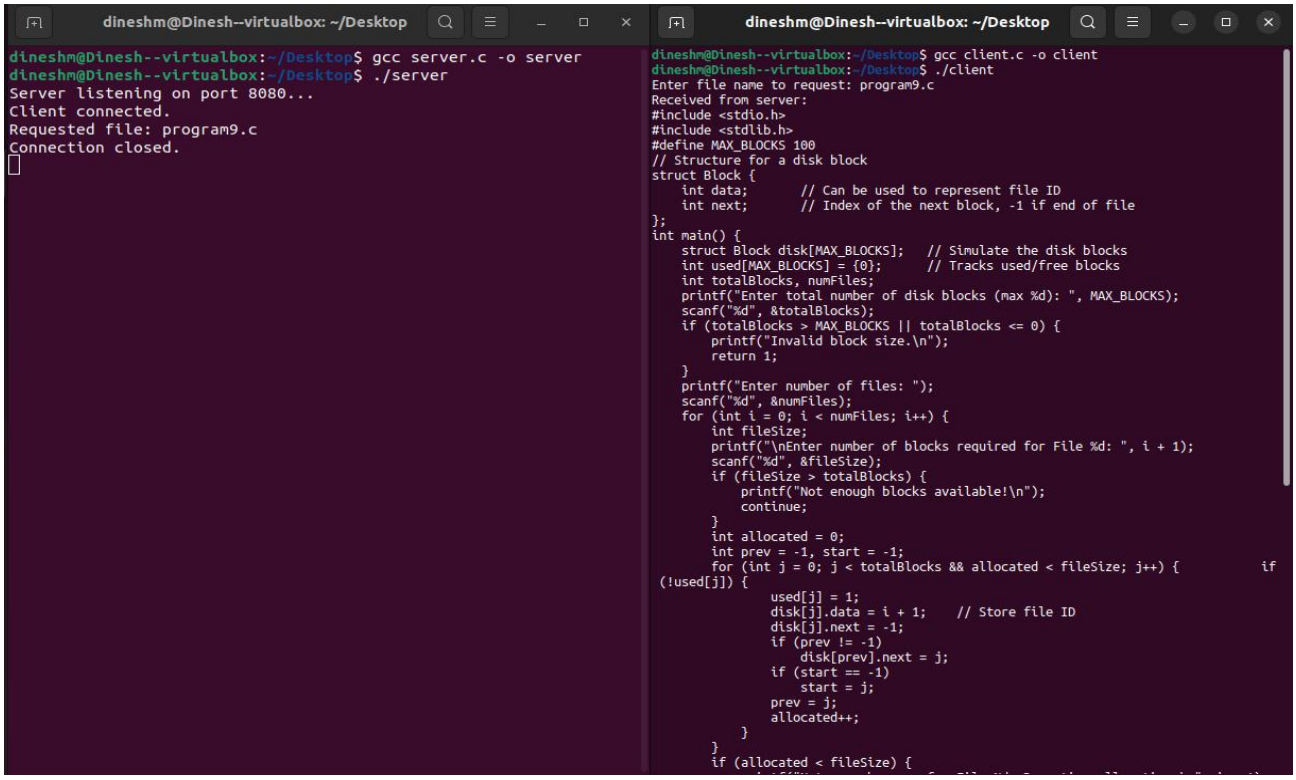
3. Create a text file (example: abc.txt) with some content.
4. Run server:

   CopyEdit
   ./server

5. In another terminal, run client:

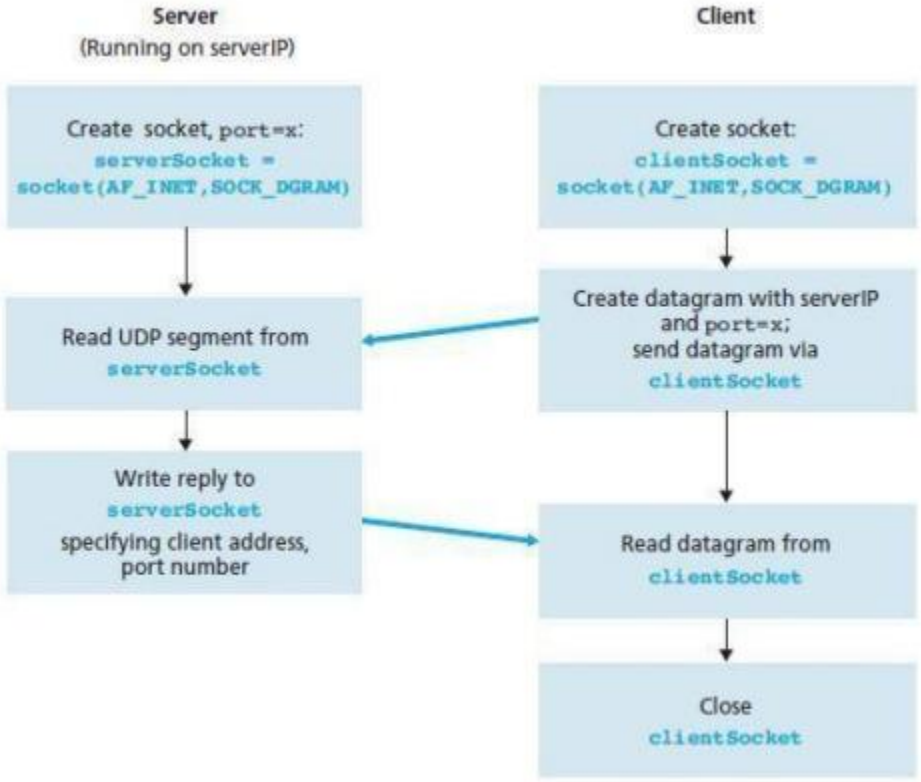
CopyEdit
./client

Sample Output



Experiment 8:

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



Program:

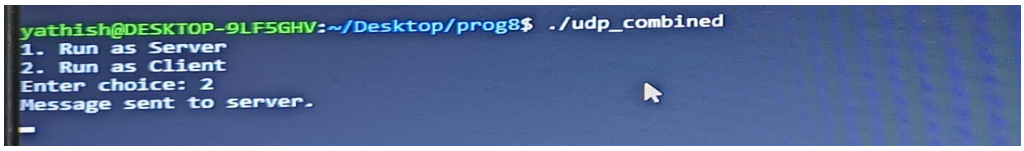
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define PORT 9876
#define BUF_SIZE 1024

void runServer();
void runClient();

int main() {
    int choice;
    printf("1. Run as Server\n");
```

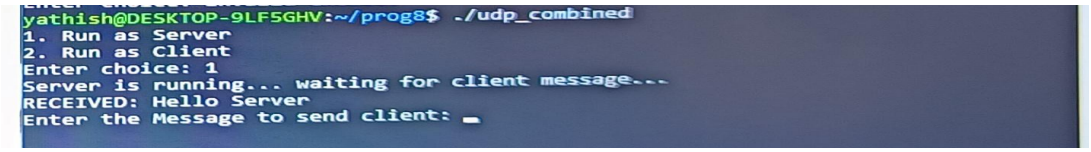
```
    printf("2. Run as Client\n");
    printf("Enter choice: ");
    scanf("%d", &choice);
    getchar();  // clear newline

    if (choice == 1) {
        runServer();
    } else if (choice == 2) {
        runClient();
    } else {
        printf("Invalid choice!\n");
    }
    return 0;
}
void runServer() {
    int sockfd;
    struct sockaddr_in serverAddr, clientAddr;
    socklen_t clientAddrLen = sizeof(clientAddr);
    char buffer[BUF_SIZE], sendBuffer[BUF_SIZE];

    // Create socket
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // Configure server address
    memset(&serverAddr, 0, sizeof(serverAddr));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = INADDR_ANY;
    serverAddr.sin_port = htons(PORT);

    // Bind socket
    if (bind(sockfd, (struct sockaddr *)&serverAddr, sizeof(serverAddr)) < 0) {
        perror("Bind failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }
```

```
    printf("Server is running... waiting for client message...\n");
    // Receive message
    int n = recvfrom(sockfd, buffer, BUF_SIZE, 0,
                     (struct sockaddr *)&clientAddr, &clientAddrLen);
    buffer[n] = '\0';
    printf("RECEIVED: %s\n", buffer);
// Send reply
    printf("Enter the Message to send client: ");
    fgets(sendBuffer, BUF_SIZE, stdin);

    sendto(sockfd, sendBuffer, strlen(sendBuffer), 0,
          (struct sockaddr *)&clientAddr, clientAddrLen);

    printf("Message sent to client.\n");

    close(sockfd);
}

void runClient() {
    int sockfd;
    struct sockaddr_in serverAddr;
    socklen_t serverAddrLen = sizeof(serverAddr);
    char buffer[BUF_SIZE];
    char *message = "Hello Server";

    // Create socket
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // Configure server address
    memset(&serverAddr, 0, sizeof(serverAddr));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); // localhost

    // Send message
```

```
    sendto(sockfd, message, strlen(message), 0,
           (struct sockaddr *)&serverAddr, serverAddrLen);
    printf("Message sent to server.\n");

    // Receive reply
    int n = recvfrom(sockfd, buffer, BUF_SIZE, 0,
                     (struct sockaddr *)&serverAddr, &serverAddrLen);
    buffer[n] = '\0';
    printf("FROM SERVER: %s\n", buffer);

    close(sockfd);
}
```

Output:





Experiment 9:

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

Cryptography is the study of creating ciphers(cipher text) and breaking them (cryptanalysis). The message to be encrypted, known as the plaintext, are transformed by a function that is parameterized by a key. The output of the encryption process, known as the ciphertext, is then transmitted. often by messenger or radio. The hacker, or intruder, hears and accurately copies down the complete ciphertext. However, unlike the intended recipient, he does not know the decryption key and so cannot decrypt the ciphertext easily.

There are several ways of classifying cryptographic algorithms. They are generally categorized based on the number of keys that are employed for encryption and decryption, and further defined by their application and use. The three types of algorithms are as follows:

1. Secret Key Cryptography (SKC): Uses a single key for both encryption and decryption. It is also known as symmetric cryptography.
2. Public Key Cryptography (PKC): Uses one key for encryption and another for decryption. It is also known as asymmetric cryptography.
3. Hash Functions: Uses a mathematical transformation to irreversibly "encrypt" information

Public-key cryptography has been said to be the most significant new development in cryptography. Modern PKC was first described publicly by Stanford University professor Martin Hellman and graduate student Whitfield Diffie in 1976. In public key cryptography, one key is used to encrypt the plaintext and the other key is used to decrypt the ciphertext.

In PKC, one of the keys is designated the public key and may be advertised as widely as the owner wants. The other key is designated the private key and is never revealed to another party. It is straight forward to send messages under this scheme. Public key of the receiver is used for encryption, so that only the receiver can decrypt the message (using his private key).

The RSA algorithm is named after Ron Rivest, Adi Shamir and Len Adleman, who invented it in 1977. The RSA algorithm can be used for both public key encryption and digital signatures.

Algorithm

1. Generate two large random primes, P and Q, of approximately equal size.
2. Compute $N = P \times Q$
3. Compute $Z = (P-1) \times (Q-1)$.
4. Choose an integer E, $1 < E < Z$, such that GCD $(E, Z) = 1$
5. Compute the secret exponent D, $1 < D < Z$, such that $E \times D \equiv 1 \pmod{Z}$
6. The public key is (N, E) and the private key is (N, D).

Note: The values of P, Q, and Z should also be kept secret.

The message is encrypted using public key and decrypted using private key.

An example of RSA encryption

1. Select primes P=11, Q=3.
2. $N = P \times Q = 11 \times 3 = 33$

   $Z = (P-1) \times (Q-1) = 10 \times 2 = 20$

3. Lets choose E=3

   Check GCD(E, P-1) = GCD(3, 10) = 1 (i.e. 3 and 10 have no common factors except 1), and check GCD(E, Q-1) = GCD(3, 2) = 1
   therefore GCD(E, Z) = GCD(3, 20) = 1
4. Compute D such that E x D ≡ 1 (mod Z) compute $D = E^{-1} \bmod Z$ = $3^{-1} \bmod 20$

   find a value for D such that Z divides ((E x D)-1) find D such that 20 divides 3D-1. Simple testing (D = 1, 2, ...) gives D = 7
   Check: (E x D)-1 = 3.7 - 1 = 20, which is divisible by Z.
5. Public key = (N, E) = (33,

   3) Private key = (N, D) = (33, 7).

Now say we want to encrypt the message m = 7, Cipher code = $M^E \bmod N$
= $7^3 \bmod 33$
= 343 mod 33
= 13.
Hence the ciphertext c = 13.

To check decryption we compute Message‟ = $C^D \bmod N$
= $13^7 \bmod 33$
= 7.

Note that we don't have to calculate the full value of 13 to the power 7 here. We can make use of the fact that a = bc mod n = (b mod n).(c mod n) mod n so we can break down a potentially large number into its components and combine the results of easier, smaller calculations to calculate the final value.

Now say we want to encrypt the message m = 7, Cipher code = $M^E \bmod N$
= $7^3 \bmod 33$
= 343 mod 33
= 13.
Hence the ciphertext c = 13.

To check decryption we compute Message‟ = $C^D \bmod N$
= $13^7 \bmod 33$
= 7.

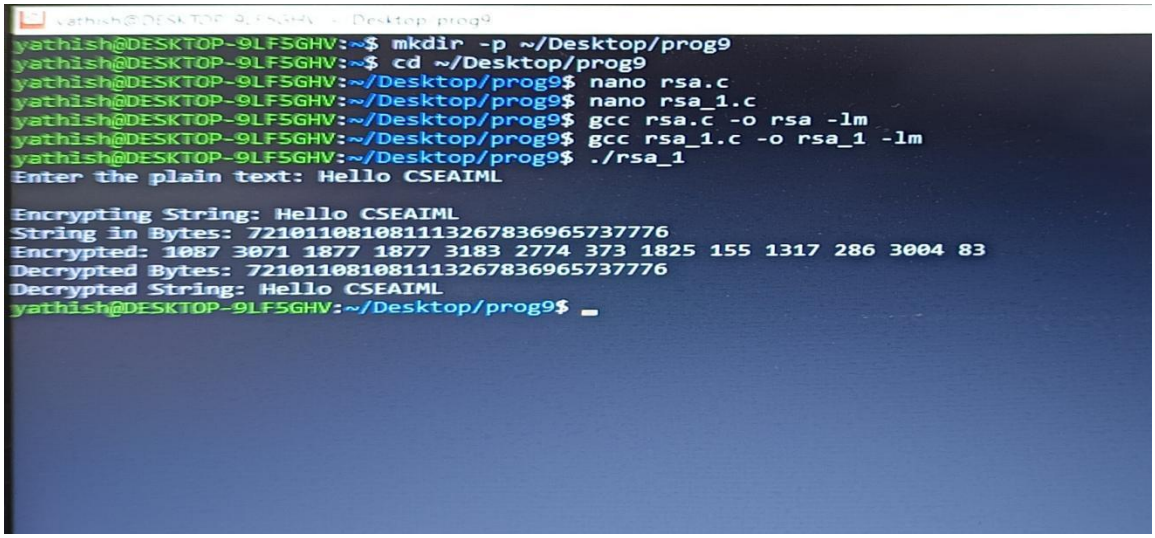
Note that we don't have to calculate the full value of 13 to the power 7 here. We can make use of the fact that a = bc mod n = (b mod n).(c mod n) mod n so we can break down a potentially large number into its components and combine the results of easier, smaller calculations to calculate the final value.

Program:

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

// Function to find gcd
int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}

// Function to perform modular exponentiation
long long int mod_pow(long long int base, long long int exp, long long int mod) {
    long long int result = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1)
            result = (result * base) % mod;
        exp = exp >> 1;
        base = (base * base) % mod;
    }
    return result;
}

int main() {
    // Two prime numbers (small, for demo)
    int p = 61;
    int q = 53;
    int n = p * q;          // modulus
    int phi = (p - 1) * (q - 1);

    // Choose e
    int e = 2;
    while (e < phi) {
        if (gcd(e, phi) == 1)
            break;
        else
            e++;
    }

    // Choose d (multiplicative inverse of e mod phi)
    int d = 0;
    int k = 1;
    while (1) {
        k = k + phi;
        if (k % e == 0) {
            d = k / e;
```

```
            break;
        }
    }
    char msg[100];
    printf("Enter the plain text: ");
    fgets(msg, sizeof(msg), stdin);
    msg[strcspn(msg, "\n")] = 0; // remove newline
    printf("\nEncrypting String: %s\n", msg);
    printf("String in Bytes: ");
    for (int i = 0; i < strlen(msg); i++) {
        printf("%d", msg[i]);
    }
    printf("\n");
     // Encryption
    long long int encrypted[100];
    printf("Encrypted: ");
    for (int i = 0; i < strlen(msg); i++) {
        encrypted[i] = mod_pow(msg[i], e, n);
        printf("%lld ", encrypted[i]);
    }
    printf("\n");
    // Decryption
    char decrypted[100];
    printf("Decrypted Bytes: ");
    for (int i = 0; i < strlen(msg); i++) {
        decrypted[i] = mod_pow(encrypted[i], d, n);
        printf("%d", decrypted[i]);
    }
    decrypted[strlen(msg)] = '\0';
    printf("\nDecrypted String: %s\n", decrypted);
    return 0;
}
```

Output:



Experiment 10:

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

The congesting control algorithms are basically divided into two groups: open loop and closed loop. Open loop solutions attempt to solve the problem by good design, in essence, to make sure it does not occur in the first place. Once the system is up and running, midcourse corrections are not made. Open loop algorithms are further divided into ones that act at source versus ones that act at the destination.

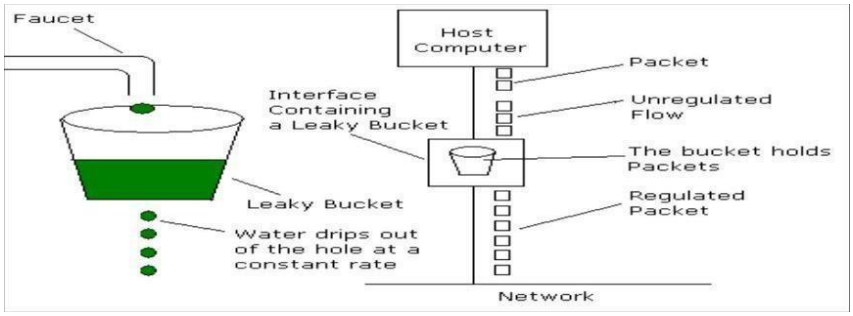
In contrast, closed loop solutions are based on the concept of a feedback loop if there is any congestion. Closed loop algorithms are also divided into two sub categories: explicit feedback and implicit feedback. In explicit feedback algorithms, packets are sent back from the point of congestion to warn the source. In implicit algorithm, the source deduces the existence of congestion by making local observation, such as the time needed for acknowledgment to come back.

The presence of congestion means that the load is (temporarily) greater than the resources (in part of the system) can handle. For subnets that use virtual circuits internally, these methods can be used at the network layer.

Another open loop method to help manage congestion is forcing the packet to be transmitted at a more predictable rate. This approach to congestion management is widely used in ATM networks and is called traffic shaping.

The other method is the leaky bucket algorithm. Each host is connected to the network by an interface containing a leaky bucket, that is, a finite internal queue. If a packet arrives at the queue when it is full, the packet is discarded. In other words, if one or more process are already queued, the new packet is unceremoniously discarded. This arrangement can be built into the hardware interface or simulate d by the host operating system. In fact it is nothing other than a single server queuing system with constant service time.

The host is allowed to put one packet per clock tick onto the network. This mechanism turns an uneven flow of packet from the user process inside the host into an even flow of packet onto the network, smoothing out bursts and greatly reducing the chances of congestion.

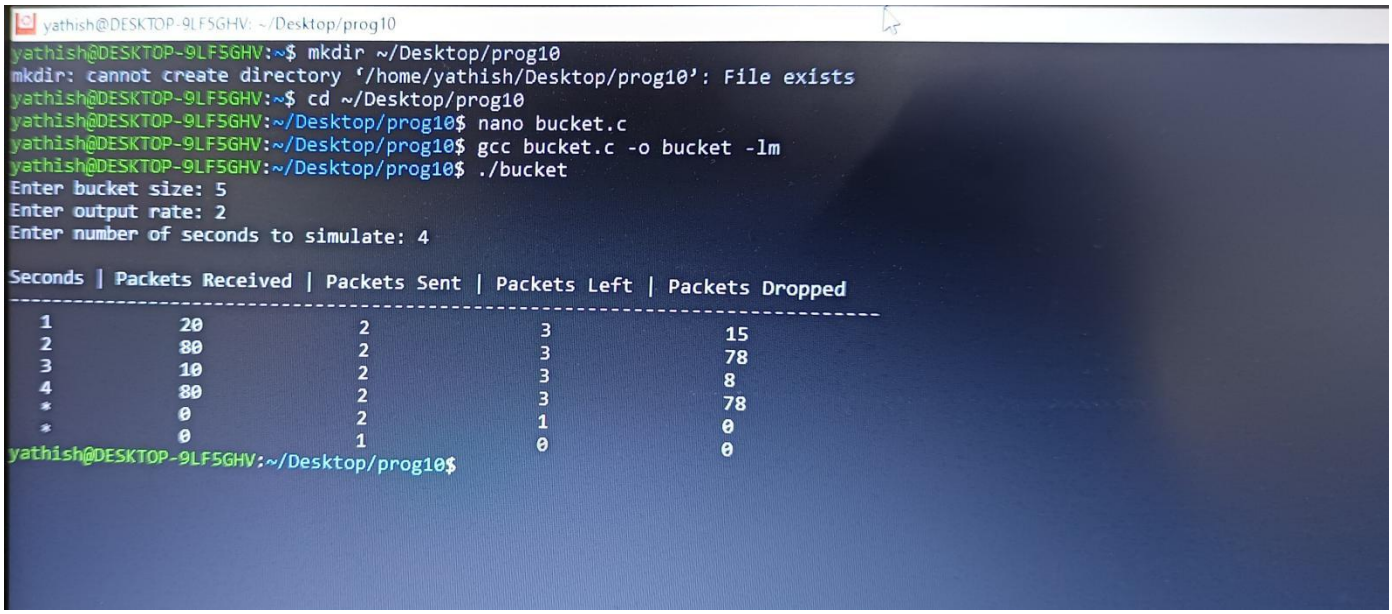


Program :

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

int main() {
   int bucket_size, output_rate, nsec;
   int packets[100], i, p_remain = 0, drop = 0, mini;

   printf("Enter bucket size: ");
   scanf("%d", &bucket_size);

   printf("Enter output rate: ");
   scanf("%d", &output_rate);

   printf("Enter number of seconds to simulate: ");
   scanf("%d", &nsec);

   // Random packets for each second
   for (i = 0; i < nsec; i++) {
      packets[i] = (rand() % 9 + 1) * 10;  // random 10–90
   }

   printf("\nSeconds | Packets Received | Packets Sent | Packets Left | Packets Dropped\n");
   printf("-------------------------------------------------------------------------\n");

   for (i = 0; i < nsec; i++) {
      p_remain += packets[i];

      if (p_remain > bucket_size) {
         drop = p_remain - bucket_size;
         p_remain = bucket_size;
      }

   mini = (p_remain < output_rate) ? p_remain : output_rate;

   printf("   %d\t\t%d\t\t%d\t\t%d\t\t%d\n",
         i + 1, packets[i], mini, p_remain - mini, drop);

   p_remain -= mini;
   drop = 0;
}
// Drain remaining packets
while (p_remain > 0) {
   mini = (p_remain < output_rate) ? p_remain : output_rate;
   printf("   *\t\t0\t\t%d\t\t%d\t\t%d\n",
         mini, p_remain - mini, drop);
   p_remain -= mini;
}
```

```
    return 0;
}
```

Output:



Viva Questions

1) What is a Link?

A link refers to the connectivity between two devices. It includes the type of cables and protocols used in order for one device to be able to communicate with the other.

2) What are the layers of the OSI reference model?

There are 7 OSI layers: Physical Layer, Data Link Layer, Network Layer, Transport Layer, Session Layer, Presentation Layer and Application Layer.

3) What is backbone network?

A backbone network is a centralized infrastructure that is designed to distribute different routes and data to various networks. It also handles management of bandwidth and various channels.

4) What is a LAN?

LAN is short for Local Area Network. It refers to the connection between computers and other network devices that are located within a small physical location.

5) What is a node?

A node refers to a point or joint where a connection takes place. It can be computer or device that is part of a network. Two or more nodes are needed in order to form a network connection.

6) What are routers?

Routers can connect two or more network segments. These are intelligent network devices that store information in its routing table such as paths, hops and bottlenecks. With this info, they are able to determine the best path for data transfer. Routers operate at the OSI Network Layer.

7) What is point to point link?

It refers to a direct connection between two computers on a network. A point to point connection does not need any other network devices other than connecting a cable to the NIC cards of both computers.

8) What is anonymous FTP?

Anonymous FTP is a way of allowed access to data in these an anonymous guest.granting user access to files in public servers. Users that are servers do not need to identify themselves, but instead log in as

9) What is subnet mask?

A subnet mask is combined with an IP address in order to identify two parts: the extended network address and the host address. Like an IP address, a subnet mask is made up of 32 bits.

10) What is the maximum length allowed for a UTP cable?

A single segment of UTP cable has an allowable length of 90 to 100 meters. This limitation can be overcome by using repeaters and switches.

11) What is data encapsulation?

Data encapsulation is the process of breaking down information into smaller manageable chunks before it is transmitted across the network. It is also in this process that the source and destination addresses are attached into the headers, along with parity checks.

12) Describe Network Topology
Network Topology refers to the layout of a computer network. It shows how devices and cables are physically laid out, as well as how they connect to one another.

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

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

15) What is the job of the Network Layer under the OSI reference model?
The Network layer is responsible for data routing, packet switching and control of network congestion. Routers operate under this layer.

16) How does a network topology affect your decision in setting up a network?
Network topology dictates what media you must use to interconnect devices. It also serves as basis on what materials, connector and terminations that is applicable for the setup.

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

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

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

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

21) What is the importance of the OSI Physical Layer?
The physical layer does the conversion from data bits to electrical signal, and vice versa. This is where network devices and cable types are considered and setup.

22) How many layers are there under TCP/IP?
There are four layers: the Network Layer, Internet Layer, Transport Layer and Application Layer.

23) What are proxy servers and how do they protect computer networks?
Proxy servers primarily prevent external users who identifying the IP addresses of an internal network.

Without knowledge of the correct IP address, even the physical location of the network cannot be identified. Proxy servers can make a network virtually invisible to external users.

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

25) What is the importance of implementing a Fault Tolerance System? Are there limitations?
A fault tolerance system ensures continuous data availability. This is done by eliminating a single point of failure. However, this type of system would not be able to protect data in some cases, such as in accidental deletions.

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

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

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

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

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

31) What is the purpose of cables being shielded and having twisted pairs?
The main purpose of this is to prevent crosstalk. Crosstalks are electromagnetic interferences or noise that can affect data being transmitted across cables.

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

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

34) What is the equivalent layer or layers of the TCP/IP Application layer in terms of OSI reference model?
The TCP/IP Application layer actually has three counterparts on the OSI model: the Session layer, Presentation Layer and Application Layer.

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

36) What is the main purpose of OSPF?
OSPF, or Open Shortest Path First, is a link-state routing protocol that uses routing tables to determine the best possible path for data exchange.

37) What are firewalls?
Firewalls serve to protect an internal network from external attacks. These external threats can be hackers who want to steal data or computer viruses that can wipe out data in an instant. It also prevents other users from external networks from gaining access to the private network.

38) Describe star topology
Star topology consists of a central hub that connects to nodes. This is one of the easiest to setup and maintain.

39) What are gateways?
Gateways provide connectivity between two or more network segments. It is usually a computer that runs the gateway software and provides translation services. This translation is a key in allowing different systems to communicate on the network.

40) What is the disadvantage of a star topology?
One major disadvantage of star topology is that once the central hub or switch get damaged, the entire network becomes unusable.

41) What is SLIP?
SLIP, or Serial Line Interface Protocol, is actually an old protocol developed during the early UNIX days. This is one of the protocols that are used for remote access.

42) Give some examples of private network addresses.
10.0.0.0 with a subnet mask of 255.0.0.0
172.16.0.0 with subnet mask of 255.240.0.0 192.168.0.0 with subnet mask of 255.255.0.0

43) What is tracert?
Tracert is a Windows utility program that can used to trace the route taken by data from the router to the destination network. It also shows the number of hops taken during the entire transmission route.

44) What are the functions of a network administrator?
A network administrator has many responsibilities that can be summarize into 3 key functions: installation of a network, configuration of network settings, and maintenance/troubleshooting of networks.

45) Describe at one disadvantage of a peer to peer network.
When you are accessing the resources that are shared by one of the workstations on the network, that workstation takes a performance hit.

46) What is Hybrid Network?
A hybrid network is a network setup that makes use of both client-server and peer-to-peer architecture.

47) What is DHCP?
DHCP is short for Dynamic Host Configuration Protocol. Its main task is to automatically assign an IP address to devices across the network. It first checks for the next available address not yet taken by any device, then assigns this to a network device.

48) What is the main job of the ARP?
The main task of ARP or Address Resolution Protocol is to map a known IP address to a MAC layer address.

49) What is TCP/IP?
TCP/IP is short for Transmission Control Protocol / Internet Protocol. This is a set of protocol layers that is designed to make data exchange possible on different types of computer networks, also known as heterogeneous network.

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

51) What protocol can be applied when you want to transfer files between different platforms, such between UNIX systems and Windows servers?
Use FTP (File Transfer Protocol) for file transfers between such different servers. This is possible because FTP is platform independent.

52) What is the use of a default gateway?
Default gateways provide means for the local networks to connect to the external network. The default gateway for connecting to the external network is usually the address of the external router port.

53) One way of securing a network is through the use of passwords. What can be considered as good passwords?
Good passwords are made up of not just letters, but by combining letters and numbers. A password that combines uppercase and lowercase letters is favorable than one that uses all upper case or all lower case letters. Passwords must be not words that can easily be guessed by hackers, such as dates, names, favorites, etc. Longer passwords are also better than short ones.

54) What is the proper termination rate for UTP cables?
The proper termination for unshielded twisted pair network cable is 100 ohms.

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

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

57) What happens when you use cables longer than the prescribed length?
Cables that are too long would result in signal loss. This means that data transmission and reception would be affected, because the signal degrades over length.

58) What common software problems can lead to network defects? Software related problems can be any or a combination of the following: - client server problems
-application conflicts
-error in configuration - protocol mismatch
-security issues
-user policy and rights issues

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

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

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

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

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

64) What is the difference between a hub and a switch?
A hub acts as a multiport repeater. However, as more and more devices connect to it, it would not be able to efficiently manage the volume of traffic that passes through it. A switch provides a better alternative

that can improve the performance especially when high traffic volume is expected across all ports.

65) What are the different network protocols that are supported by Windows RRAS services? There are three main network protocols supported: NetBEUI, TCP/IP, and IPX.

66) What are the maximum networks and hosts in a class A, B and C network? For Class A, there are 126 possible networks and 16,777,214 hosts
For Class B, there are 16,384 possible networks and 65,534 hosts For Class C, there are 2,097,152 possible networks and 254 hosts

67) What is the standard color sequence of a straight-through cable? orange/white, orange, green/white, blue, blue/white, green, brown/white, brown.

68) What protocols fall under the Application layer of the TCP/IP stack?
The following are the protocols under TCP/IP Application layer: FTP, TFTP, Telnet and SMTP.

69) You need to connect two computers for file sharing. Is it possible to do this without using a hub or router?
Yes, you can connect two computers together using only one cable. A crossover type cable can be use in this scenario. In this setup, the data transmit pin of one cable is connected to the data receive pin of the other cable, and vice versa.

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

71) What is the difference between a straight-through and crossover cable?
A straight-through cable is used to connect computers to a switch, hub or router. A crossover cable is used to connect two similar devices together, such as a PC to PC or Hub to hub.

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

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

74) When you move the NIC cards from one PC to another PC, does the MAC address gets transferred as well?
Yes, that's because MAC addresses are hard-wired into the NIC circuitry, not the PC. This also means that a PC can have a different MAC address when the NIC card was replace by another one.

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

76) In a network that contains two servers and twenty workstations, where is the best place to install an Anti-virus program?
An anti-virus program must be installed on all servers and workstations to ensure protection. That's because individual users can access any workstation and introduce a computer virus when plugging in their removable hard drives or flash drives.

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

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

79) What is the difference between CSMA/CD and CSMA/CA?
CSMA/CD, or Collision Detect, retransmits data frames whenever a collision occurred. CSMA/CA, or Collision Avoidance, will first broadcast intent to send prior to data transmission.

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

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

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

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

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

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

86) What are the different technologies involved in establishing WAN links?
Analog connections - using conventional telephone lines; Digital connections - using digitalgrade telephone lines; switched connections - using multiple sets of links between sender and receiver to move data.

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

88) When troubleshooting computer network problems, what common hardware-related problems can occur?
A large percentage of a network is made up of hardware. Problems in these areas can range from malfunctioning hard drives, broken NICs and even hardware startups. Incorrectly hardware configuration is also one of those culprits to look into.

89) What can be done to fix signal attenuation problems?
A common way of dealing with such a problem is to use repeaters and hub, because it will help regenerate the signal and therefore prevent signal loss. Checking if cables are properly terminated is also a must.

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

91) Explain profile in terms of networking concept?
Profiles are the configuration settings made for each user. A profile may be created that puts a user in a group, for example.

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

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

94) What protocols fall under the TCP/IP Internet Layer?
There are 4 protocols that are being managed by this layer. These are ICMP, IGMP, IP and ARP.

95) When it comes to networking, what are rights?
Rights refer to the authorized permission to perform specific actions on the network. Each user on the network can be assigned individual rights, depending on what must be allowed for that user.

96) What is one basic requirement for establishing VLANs?
A VLAN requires dedicated equipment on each end of the connection that allows messages entering the Internet to be encrypted, as well as for authenticating users.

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

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

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