Lab Program 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.

Source Code:

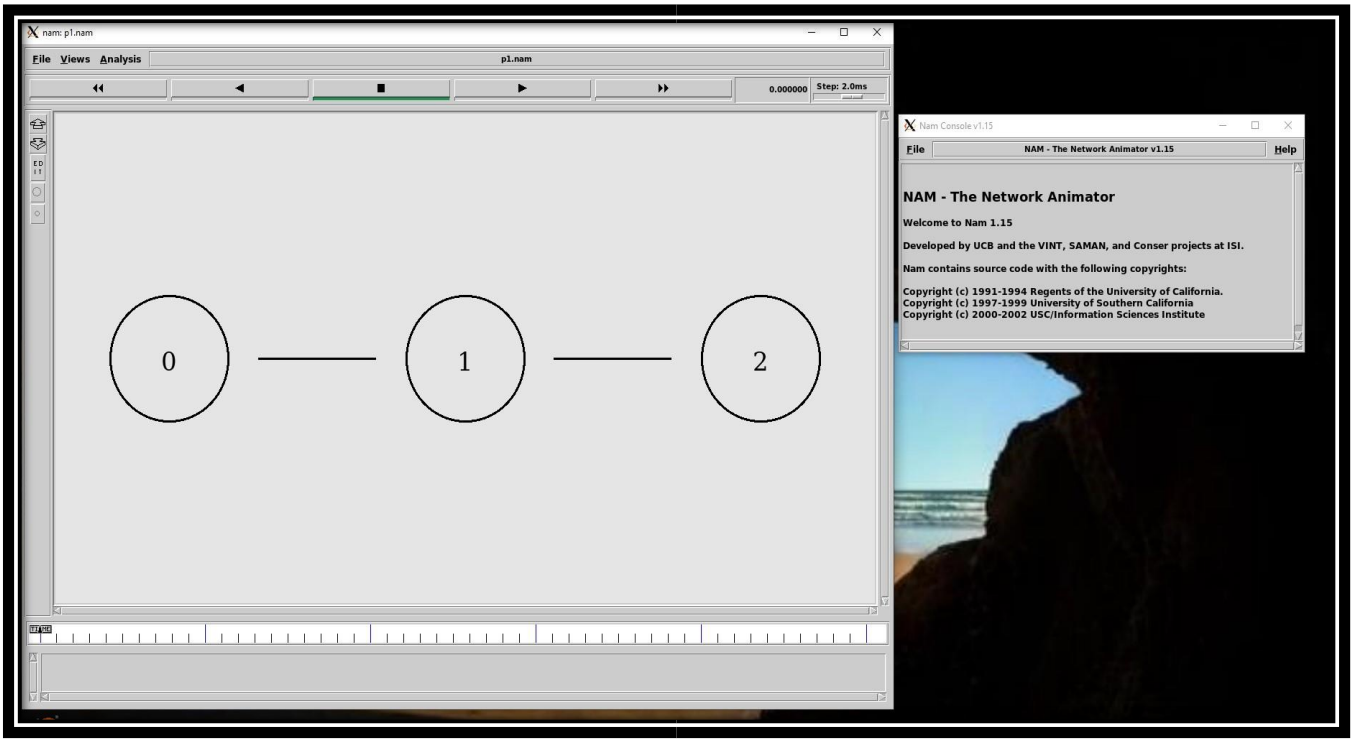
```
# Create Simulator set
ns [new Simulator]

# Open Trace file and NAM file
set ntrace [open p1.tr w] $ns
trace-all $ntrace set namfile
[open p1.nam w]
$ns namtrace-all $namfile

# Finish Procedure proc
Finish {} {     global ns
ntrace namfile
   # Dump all the trace data and close the files
   $ns flush-trace
close $ntrace     close
$namfile
   # Execute the nam animation file
exec nam p1.nam &
   # Show the number of packets dropped
exec echo "The number of packet drops is " &
exec grep -c "^d" p1.tr &     exit 0
}
```

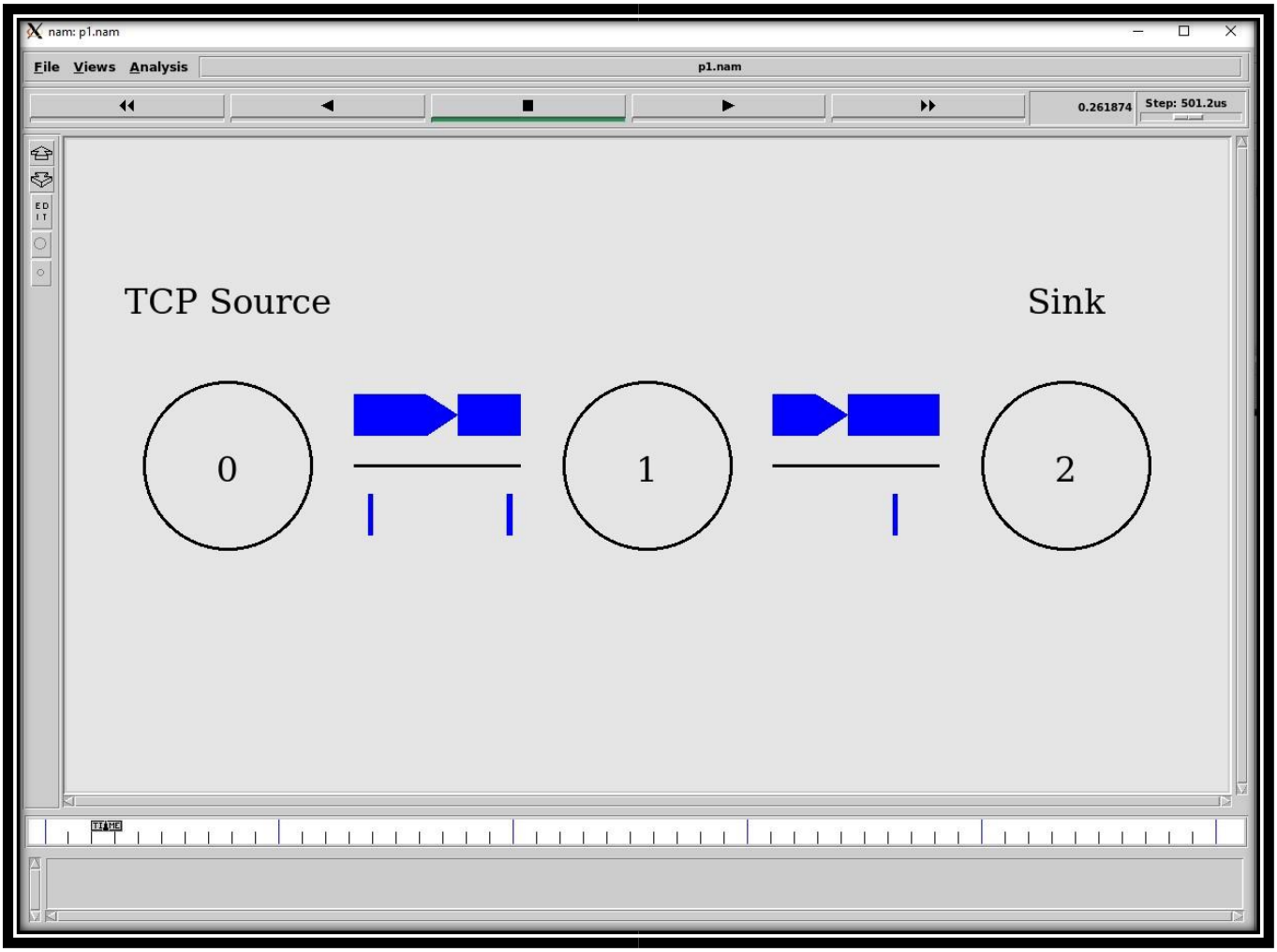
```
# Create 3 nodes set
n0 [$ns node] set
n1 [$ns node] set
n2 [$ns node]

# Label the nodes
$n0 label "TCP Source"
$n2 label "Sink"

# Set the color
$ns color 1 blue

# Create Links between nodes
# Modify the bandwidth to observe the variation in packet drop
$ns duplex-link $n0 $n1 1Mb 10ms DropTail $ns
duplex-link $n1 $n2 1Mb 10ms DropTail

# Make the Link Orientation
$ns duplex-link-op $n0 $n1 orient right $ns
duplex-link-op $n1 $n2 orient right

# Set Queue Size
# Modify the queue length to observe the variation in packet drop
$ns queue-limit $n0 $n1 10 $ns
queue-limit $n1 $n2 10

# Set up a Transport layer connection
set tcp0 [new Agent/TCP] $ns
attach-agent $n0 $tcp0 set sink0
[new Agent/TCPSink] $ns attach-
agent $n2 $sink0
```

```
# Corrected connect command
$ns connect $tcp0 $sink0

# Set up an Application layer Traffic set
cbr0 [new Application/Traffic/CBR]
$cbr0 set type_ CBR
$cbr0 set packetSize_ 100
$cbr0 set rate_ 1Mb
$cbr0 set random_ false
$cbr0 attach-agent $tcp0
$tcp0 set class_ 1

# Schedule Events
$ns at 0.0 "$cbr0 start"
$ns at 5.0 "Finish"

# Run the simulation
$ns run
```

Output: manohar@DESKTOP-6LK68TO:~ $ ns p1.tcl

The number of packet drops is 8





Lab Program 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

Source code:-

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

# Setup topography object  set
topo [new Topography]
$topo load_flatgrid 1500 1500

# Open the NS trace file  set
tracefile [open p4.tr w]
$ns trace-all $tracefile

# Open the NAM trace file  set
namfile [open p4.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile 1500 1500
#===================================
# Mobile node parameter setup
#===================================
$ns node-config -adhocRouting DSDV \
   -llType LL \
   -macType Mac/802_11 \
   -ifqType Queue/DropTail \
   -ifqLen 20 \
   -phyType Phy/WirelessPhy \
   -channelType Channel/WirelessChannel \
   -propType Propagation/TwoRayGround \
   -antType Antenna/OmniAntenna \
   -topoInstance $topo \
   -agentTrace ON \
```

```
   -routerTrace ON
#===================================
# Nodes Definition
#===================================
create-god 6  #
Create 6 nodes
set n0 [$ns node]
$n0 set X_ 630
$n0 set Y_ 501
$n0 set Z_ 0.0  $ns
initial_node_pos $n0 20  set
n1 [$ns node]  $n1 set X_
454
$n1 set Y_ 340
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20

set n2 [$ns node]  $n2
set X_ 785
$n2 set Y_ 326
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20

set n3 [$ns node]  $n3
set X_ 270
$n3 set Y_ 190
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20

set n4 [$ns node]
$n4 set X_ 539
```

```
$n4 set Y_ 131
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20

set n5 [$ns node]  $n5
set X_ 964
$n5 set Y_ 177
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
#===================================
# Agents Definition
#===================================
# Setup a UDP connection  set
udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

set null1 [new Agent/Null]
$ns attach-agent $n4 $null1
$ns connect $udp0 $null1
$udp0 set packetSize_ 1500

# Setup a TCP connection  set
tcp0 [new Agent/TCP]
$ns attach-agent $n3 $tcp0

set sink1 [new Agent/TCPSink]  $ns
attach-agent $n5 $sink1
$ns connect $tcp0 $sink1
#===================================
# Applications Definition
```

```
#=================================== # Setup a CBR Application over UDP connection  set
cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null

# Setup a FTP Application over TCP connection  set
ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
#===================================
# Termination
#===================================
# Define a 'finish' procedure proc finish {} {
global ns tracefile namfile      $ns flush-trace
close $tracefile      close $namfile     exec nam
p4.nam &      exec echo "Number of packets
dropped is:" &      exec grep -c "^D" p4.tr &
exit 0
}
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$ftp0 start"
$ns at 180.0 "$ftp0 stop"
$ns at 200.0 "$cbr0 stop"
$ns at 200.0 "finish"
$ns at 70 "$n4 set dest 100 60 20"
$ns at 100 "$n4 set dest 700 300 20"
$ns at 150 "$n4 set dest 900 200 20"
$ns run
```

Output:-

manohar@DESKTOP-6LK68TO:~$ ns p2.tcl

warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl num_nodes

is set 6

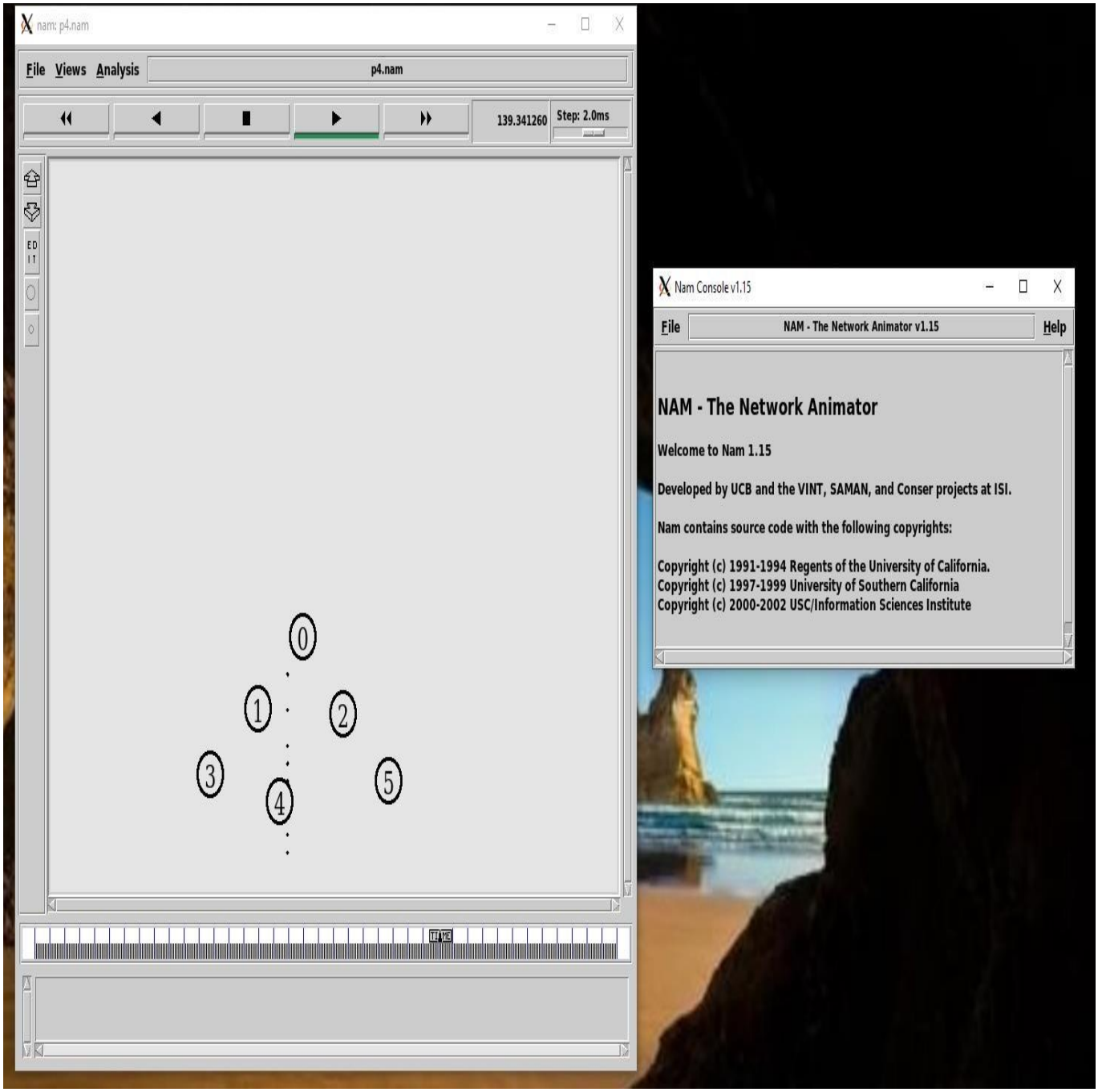
INITIALIZE THE LIST xListHead

channel.cc:sendUp - Calc highestAntennaZ_ and distCST_

highestAntennaZ_ = 1.5, distCST_ = 550.0 SORTING

LISTS ...DONE!

Number of packets dropped is: 15605



Lab Program 3:

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

Source code: # Create

Simulator set ns [new

Simulator]

Use colors to differentiate the traffic

$ns color 1 Blue

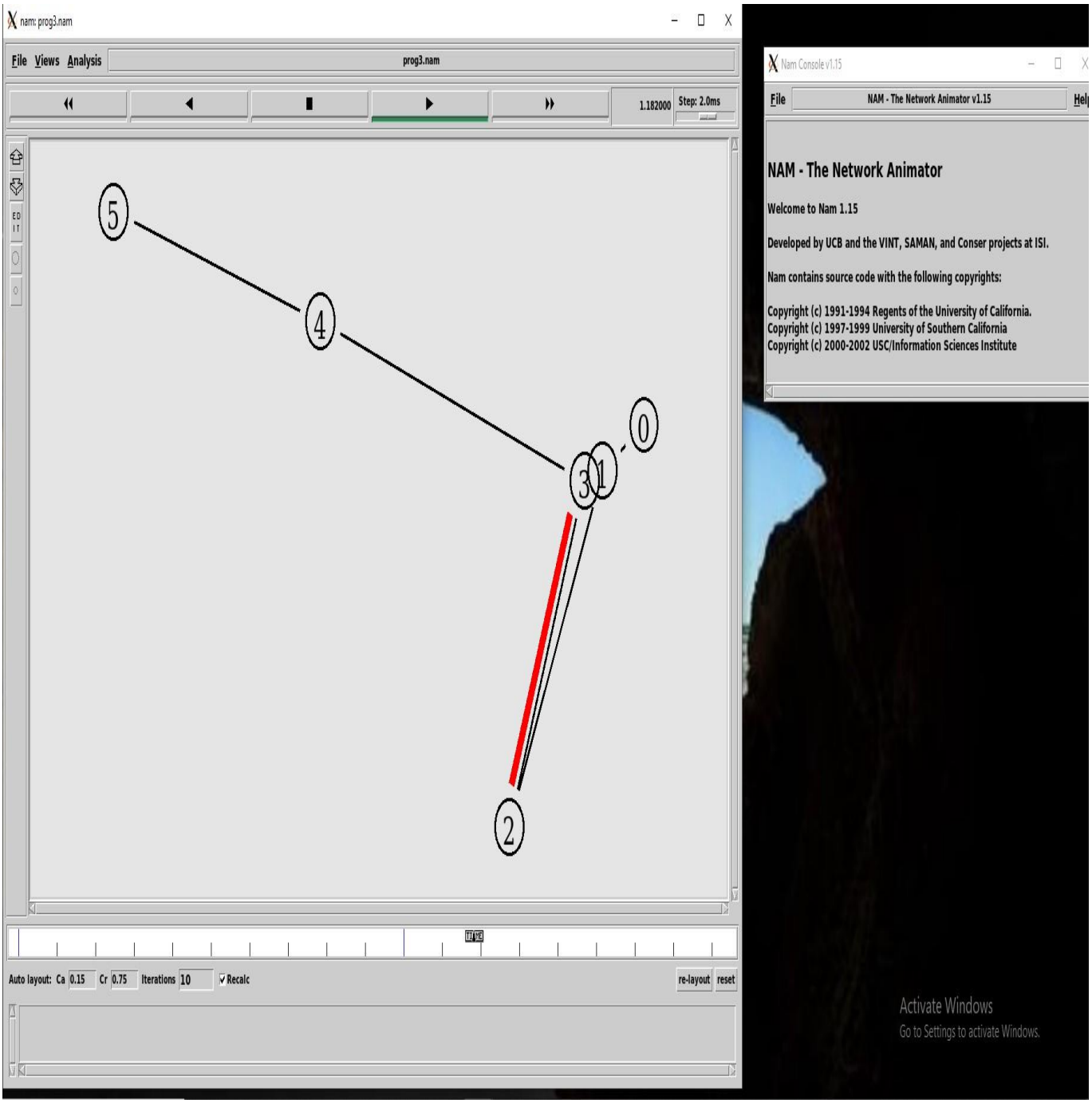
$ns color 2 Red

```
# Open trace and NAM trace file
set ntrace [open prog3.tr w]  $ns
trace-all $ntrace  set namfile
[open prog3.nam w]
$ns namtrace-all $namfile

# Finish Procedure  proc
Finish {} {     global ns
ntrace namfile
   # Dump all trace data and close the file
   $ns flush-trace
close $ntrace      close
$namfile
   # Execute the nam animation file
exec nam prog3.nam &
   # Find the number of ping packets dropped      puts "The
number of ping packets dropped are "      exec grep "^d"
prog3.tr | cut -d " " -f 5 | grep -c "ping" &     exit 0
}

# Create six nodes  for {set i
0} {$i < 6} {incr i} {      set
n($i) [$ns node]
}

# Connect the nodes for {set j
0} {$j < 5} {incr j} {
   $ns duplex-link $n($j) $n([expr ($j+1)]) 0.1Mb 10ms DropTail
}

# Define the recv function for the class 'Agent/Ping'
```

```
Agent/Ping instproc recv {from rtt} {
   $self instvar node_     if
{[info exists node_]} {
       puts "node [$node_ id] received ping answer from $from with round trip time $rtt ms"
   } else {
       puts "Error: node_ variable not set"
   }
}

# Create two ping agents and attach them to n(0) and n(5)
set p0 [new Agent/Ping]  $p0 set class_ 1
$ns attach-agent $n(0) $p0

set p1 [new Agent/Ping]  $p1
set class_ 1
$ns attach-agent $n(5) $p1

$ns connect $p0 $p1

# Set queue size and monitor the queue
# Queue size is set to 2 to observe the drop in ping packets
$ns queue-limit $n(2) $n(3) 2
$ns duplex-link-op $n(2) $n(3) queuePos 0.5

# Create Congestion
# Generate a Huge CBR traffic between n(2) and n(4)
set tcp0 [new Agent/TCP]  $tcp0 set class_ 2
$ns attach-agent $n(2) $tcp0

set sink0 [new Agent/TCPSink]  $ns
attach-agent $n(4) $sink0
$ns connect $tcp0 $sink0
```

```
# Apply CBR traffic over TCP  set cbr0
[new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set rate_ 1Mb
$cbr0 attach-agent $tcp0

# Schedule events
$ns at 0.2 "$p0 send"
$ns at 0.4 "$p1 send"
$ns at 0.4 "$cbr0 start"
$ns at 0.8 "$p0 send"
$ns at 1.0 "$p1 send"
$ns at 1.2 "$cbr0 stop"
$ns at 1.4 "$p0 send"
$ns at 1.6 "$p1 send"
$ns at 1.8 "Finish"

# Run the Simulation
$ns run
```

Output:-



manohar@DESKTOP-6LK68TO:~$ ns p3.tcl

node 0 received ping answer from 5 with round trip time 151.2 ms

node 0 received ping answer from 5 with round trip time 301.4 ms

node 5 received ping answer from 0 with round trip time 155.4 ms The

number of ping packets dropped are 3

Lab Program 4:

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

Source code: # Create

Simulator set ns [new

Simulator]

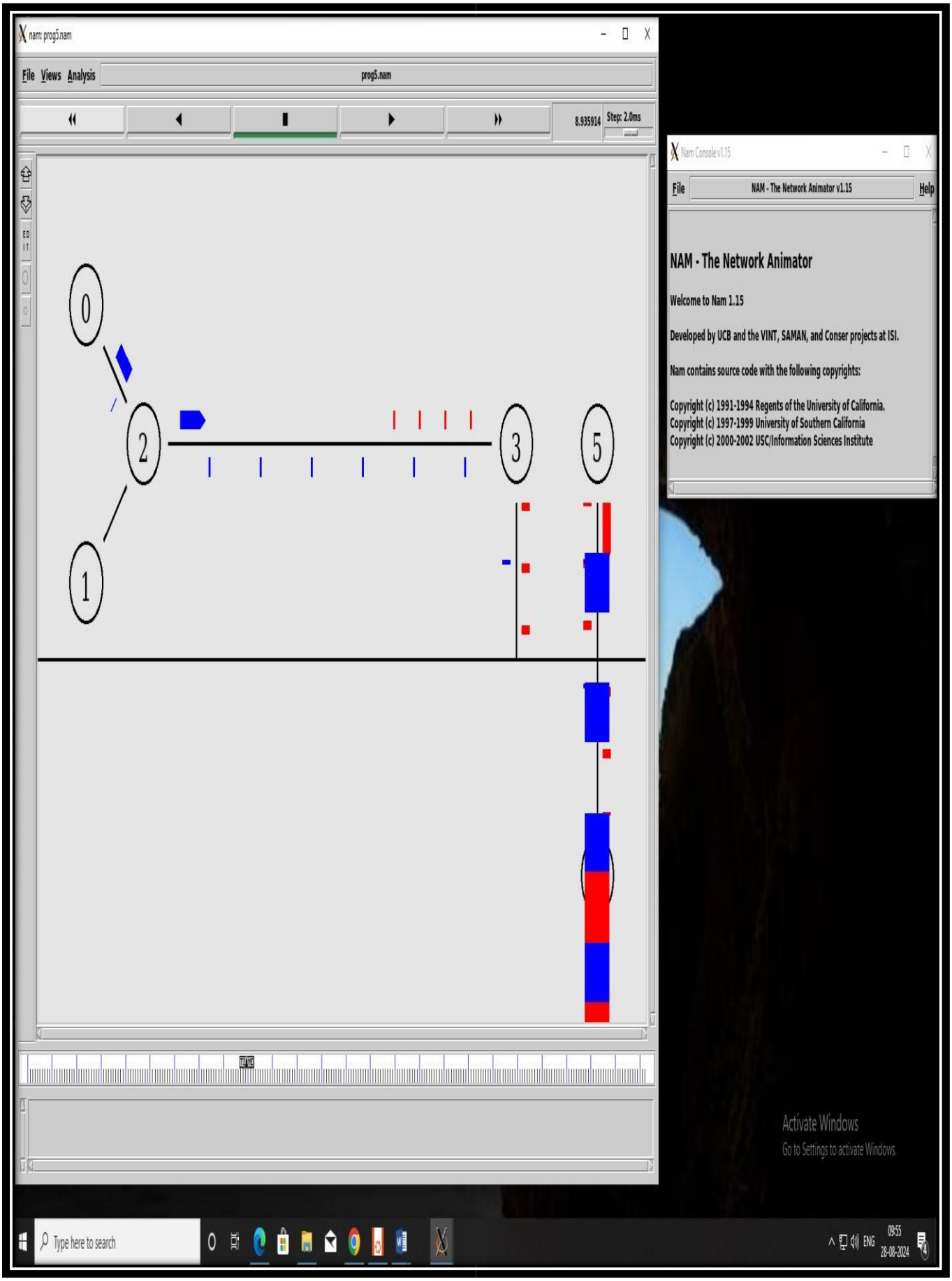
```
# Use colors to differentiate the traffics
$ns color 1 Blue
$ns color 2 Red

# Open trace and NAM trace file
set ntrace [open prog5.tr w] $ns
trace-all $ntrace set namfile
[open prog5.nam w]
$ns namtrace-all $namfile

# Use some flat file to create congestion graph windows
set winFile0 [open WinFile0 w] set winFile1 [open
WinFile1 w]

# Finish Procedure proc
Finish {} { global ns
ntrace namfile
  # Dump all trace data and Close the files
  $ns flush-trace
close $ntrace close
$namfile
  # Execute the NAM animation file
exec nam prog5.nam &
  # Plot the Congestion Window graph using xgraph
exec xgraph WinFile0 WinFile1 &
```

```
    exit 0
}

# Plot Window Procedure  proc
PlotWindow {tcpSource file} {
global ns      set time 0.1      set now
[$ns now]
    set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
    $ns at [expr $now+$time] "PlotWindow $tcpSource $file"  }

# Create 6 nodes
for {set i 0} {$i < 6} {incr i} {
set n($i) [$ns node]
}

# Create duplex links between the nodes
$ns duplex-link $n(0) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(1) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 0.6Mb 100ms DropTail

# Nodes n(3), n(4) and n(5) are considered in a LAN
set lan [$ns newLan "$n(3) $n(4) $n(5)" 0.5Mb 40ms LL Queue/DropTail MAC/802_3 Channel]

# Orientation to the nodes
$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
$ns duplex-link-op $n(2) $n(3) orient right

# Set queue between n(2) and n(3) and monitor the queue
$ns queue-limit $n(2) $n(3) 20
```

```
$ns duplex-link-op $n(2) $n(3) queuePos 0.5

# Set error model on link n(2) to n(3)  set
loss_module [new ErrorModel]
$loss_module ranvar [new RandomVariable/Uniform]
$loss_module drop-target [new Agent/Null]
$ns lossmodel $loss_module $n(2) $n(3)

# Set up the TCP connection between n(0) and n(4)  set
tcp0 [new Agent/TCP/Newreno]
$tcp0 set fid_ 1
$tcp0 set window_ 8000
$tcp0 set packetSize_ 552
$ns attach-agent $n(0) $tcp0

set sink0 [new Agent/TCPSink/DelAck]
$ns attach-agent $n(4) $sink0
$ns connect $tcp0 $sink0

# Apply FTP Application over TCP  set
ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set type_ FTP

# Set up another TCP connection between n(5) and n(1)  set
tcp1 [new Agent/TCP/Newreno]
$tcp1 set fid_ 2
$tcp1 set window_ 8000
$tcp1 set packetSize_ 552
$ns attach-agent $n(5) $tcp1
```

```
set sink1 [new Agent/TCPSink/DelAck]
$ns attach-agent $n(1) $sink1
$ns connect $tcp1 $sink1

# Apply FTP application over TCP  set
ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP

# Schedule Events
$ns at 0.1 "$ftp0 start"
$ns at 0.1 "PlotWindow $tcp0 $winFile0"
$ns at 0.5 "$ftp1 start"
$ns at 0.5 "PlotWindow $tcp1 $winFile1"
$ns at 25.0 "$ftp0 stop"
$ns at 25.1 "$ftp1 stop"
$ns at 25.2 "Finish"

# Run the simulation
$ns run
```

Output:-



To develop and run a Java program that implements computer network protocol using the Ubuntu terminal, follow these steps:

1. Install Java on Ubuntu :-

Ensure that Java is installed on your Ubuntu system. You can check this by running:

java -version

If Java is not installed, install it using:

sudo apt-get update

sudo apt-get install default-jdk

2. Create a Directory for Your Project: Create a directory where you'll store your Java program:

mkdir **-p ~/sliding_window_protocol** cd

~/sliding_window_protocol

3. Write the Java Program :- Use a text editor like nano or vim to create your Java program file:

nano SlidingWindowProtocol.java

Then, paste the following Java code into the editor:

4. Save and Exit the Editor

- If using nano, press CTRL + O, then Enter to save. Press CTRL + X to exit.
- If using vim, press ESC, then type :wq and press Enter to save and exit.

5. Compile the Java Program: Compile the Java program using the javac command:

javac programname.java

This will create a `programname. Class` file if the compilation is successful.

6. Run the Java Program

Run the compiled Java program using the `java` command:

java programname

Lab Program 5:

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

Source Code:

```
import java.util.Random;

class SlidingWindowProtocol {
private int windowSize;     private
int numFrames;     private
boolean[] acknowledged;

   public SlidingWindowProtocol(int windowSize, int numFrames) {
this.windowSize = windowSize;         this.numFrames =
numFrames;
       this.acknowledged = new boolean[numFrames];
   }

   public void sendFrames() {         int
base = 0; // Start of the window
       Random random = new Random();

       while (base < numFrames) {
          // Sending frames within the window
          for (int i = 0; i < windowSize && (base + i) < numFrames; i++) {
if (!acknowledged[base + i]) {
                System.out.println("Sending frame " + (base + i));
                // Simulate a random frame loss
if (random.nextBoolean()) {
                   System.out.println("Frame " + (base + i) + " acknowledged.");
acknowledged[base + i] = true;
                } else {
                   System.out.println("Frame " + (base + i) + " lost.");
                }
             }
```

```
        }

        // Checking acknowledgments and sliding the window
while (base < numFrames && acknowledged[base]) {
base++;
        }

        // Simulate time delay for retransmission
try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted: " + e.getMessage());
        }
      }

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

   public static void main(String[] args) {
int windowSize = 4;        int numFrames
= 10;

      SlidingWindowProtocol swp = new SlidingWindowProtocol(windowSize, numFrames);
swp.sendFrames();
   }
}
```

Output:

manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$ javac SlidingWindowProtocol.java

manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$ java SlidingWindowProtocol

Sending frame 0 Frame 0 lost. Sending frame 1 Frame 1 lost.

Sending frame 2

Frame 2 acknowledged.

Sending frame 3 Frame

3 lost.

Sending frame 0

Frame 0 acknowledged.

Sending frame 1

Frame 1 acknowledged.

Sending frame 3

Frame 3 acknowledged.

Sending frame 4 Frame

4 lost.

Sending frame 5

Frame 5 acknowledged.

Sending frame 6 Frame

6 lost.

Sending frame 7

Frame 7 acknowledged.

Sending frame 4

Frame 4 lost. Sending

frame 6 Frame 6 lost.

Sending frame 4

Frame 4 acknowledged.

Sending frame 6 Frame

6 lost.

Sending frame 6

Frame 6 acknowledged.

Sending frame 8

Frame 8 acknowledged.

Sending frame 9 Frame

9 lost.

Sending frame 9 Frame

9 acknowledged.

All frames sent and acknowledged.

Lab Program 6:

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

Source code:

```
import java.util.ArrayList;
import java.util.Arrays; import
java.util.List;

class Edge {
   int source, destination, weight;

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

class Graph {
   private int V; // Number of vertices     private
List<Edge> edges; // List of all edges

   public Graph(int V) {
this.V = V;
      this.edges = new ArrayList<>();
   }

   // Function to add an edge to the graph
   public void addEdge(int source, int destination, int weight) {
edges.add(new Edge(source, destination, weight));
   }

   // Bellman-Ford algorithm to find the shortest path from a source vertex
public void bellmanFord(int src) {
      // Initialize distances from src to all other vertices as infinite
int[] dist = new int[V];
```

```
        Arrays.fill(dist, Integer.MAX_VALUE);
dist[src] = 0;

        // Relax all edges |V| - 1 times
for (int i = 1; i < V; ++i) {
for (Edge edge : edges) {
int u = edge.source;                int v
= edge.destination;                int
weight = edge.weight;
                if (dist[u] != Integer.MAX_VALUE && dist[u] + weight < dist[v]) {
dist[v] = dist[u] + weight;
                }
            }
        }

        // Check for negative-weight cycles
for (Edge edge : edges) {            int u =
edge.source;            int v =
edge.destination;            int weight =
edge.weight;
            if (dist[u] != Integer.MAX_VALUE && dist[u] + weight < dist[v]) {
System.out.println("Graph contains negative weight cycle");                return;
            }
        }

        printSolution(dist, src);
    }

    // Function to print the shortest distances from the source vertex
private void printSolution(int[] dist, int src) {
```

```
        System.out.println("Vertex Distance from Source " + src);
for (int i = 0; i < V; ++i)
            System.out.println("Vertex " + i + ": " + dist[i]);
    }

    // Path Vector Routing to show the path taken to each vertex
public void pathVectorRouting(int src) {
        // Initialize paths from src to all other vertices
List<Integer>[] paths = new ArrayList[V];
for (int i = 0; i < V; i++) {            paths[i] = new
ArrayList<>();
        }
        paths[src].add(src);

        int[] dist = new int[V];
        Arrays.fill(dist, Integer.MAX_VALUE);
dist[src] = 0;

        // Bellman-Ford logic with path tracking
for (int i = 1; i < V; ++i) {            for (Edge
edge : edges) {                int u = edge.source;
int v = edge.destination;                int weight
= edge.weight;
            if (dist[u] != Integer.MAX_VALUE && dist[u] + weight < dist[v]) {
dist[v] = dist[u] + weight;                    paths[v] = new ArrayList<>(paths[u]);
paths[v].add(v);
            }
          }
        }

        printPaths(paths, src);
```

```
    }

    // Function to print the paths from the source vertex
private void printPaths(List<Integer>[] paths, int src) {
System.out.println("Paths from Source " + src);         for
(int i = 0; i < paths.length; i++) {
System.out.print("Vertex " + i + ": ");              for (int
vertex : paths[i]) {
                System.out.print(vertex + " ");
            }
            System.out.println();
        }
    }
}

public class BellmanFordPathVector {
public static void main(String[] args) {
Graph graph = new Graph(5);
graph.addEdge(0, 1, -1);
graph.addEdge(0, 2, 4);
graph.addEdge(1, 2, 3);
graph.addEdge(1, 3, 2);
graph.addEdge(1, 4, 2);
graph.addEdge(3, 2, 5);
graph.addEdge(3, 1, 1);
graph.addEdge(4, 3, -3);


        int sourceVertex = 0;


        System.out.println("Bellman-Ford Algorithm:");
graph.bellmanFord(sourceVertex);
```

```
        System.out.println("\nPath Vector Routing Algorithm:");
graph.pathVectorRouting(sourceVertex);
    }
}
```

Output:

manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$ nano BellmanFordPathVector.java

manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$ javac BellmanFordPathVector.java Note: BellmanFordPathVector.java uses unchecked or unsafe operations.

Note: Recompile with -Xlint:unchecked for details.

manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$ java BellmanFordPathVector Bellman-Ford Algorithm:

Vertex Distance from Source 0

Vertex 0: 0

Vertex 1: -1

Vertex 2: 2

Vertex 3: -2

Vertex 4: 1

Path Vector Routing Algorithm:

Paths from Source 0

Vertex 0: 0

Vertex 1: 0 1

Vertex 2: 0 1 2

Vertex 3: 0 1 4 3

Vertex 4: 0 1 4

Lab Program 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.

Source code:

FileServer.java

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

public class FileServer {
   public static void main(String[] args) {
try {
        ServerSocket serverSocket = new ServerSocket(8080);
        System.out.println("Server is listening on port 8080...");

        while (true) {
           Socket clientSocket = serverSocket.accept();
           System.out.println("Client connected!");

      BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
           PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

           String fileName = in.readLine();
           System.out.println("Client requested file: " + fileName);

           File file = new File(fileName);
if (file.exists() && !file.isDirectory()) {
               System.out.println("File found, sending contents...");
               BufferedReader fileReader = new BufferedReader(new FileReader(file));
               String line;
               while ((line = fileReader.readLine()) != null) {
out.println(line);
                   System.out.println("Sent: " + line);
               }
               out.flush();
fileReader.close();
```

```
                System.out.println("File content sent successfully.");
} else {
                System.out.println("File not found.");
out.println("File not found");
            }

            clientSocket.close();
            System.out.println("Client connection closed.");
          }
      } catch (IOException e) {
e.printStackTrace();
      }
   }
}
```

FileClient.Java

```
import java.io.*; import
java.net.*; public class
FileClient {
   public static void main(String[] args) {
try {
          Socket socket = new Socket("localhost", 8080);
          System.out.println("Connected to the server!");

          BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));
          PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
          BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

          System.out.print("Enter the filename: ");
          String fileName = userInput.readLine();
out.println(fileName);
```

```
        String serverResponse;
        System.out.println("Contents of the file:");
while ((serverResponse = in.readLine()) != null) {
if (serverResponse.equals("File not found")) {
System.out.println(serverResponse);                    break;
            }
            System.out.println(serverResponse);
        }
socket.close();
        System.out.println("Connection closed.");
} catch (IOException e) {
e.printStackTrace();
    }   }  }
```

Output:- first

terminal

manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$ java FileServer

Server is listening on port 8080...

Second terminal

manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$ java FileClient Connected

to the server!

Enter the filename: one Contents

of the file:

Connection closed.

Note:-

Above is partial output :

a)Verify Port Availability b) Check for Firewall Rules c) Verify Network Configuration d) **Check for Listening Ports**

Lab Program 8:

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

Source code:-

UDPServer.java

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

public class UDPServer {
   public static void main(String[] args) {
DatagramSocket socket = null;        try {
        // Create a DatagramSocket on port 9876
socket = new DatagramSocket(9876);
        System.out.println("Server is listening on port 9876...");

        byte[] receiveData = new byte[1024];

        while (true) {
           // Receive the incoming datagram packet
           DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
socket.receive(receivePacket);

           String message = new String(receivePacket.getData(), 0, receivePacket.getLength());
           System.out.println("Received message: " + message);

           // Send the message back to the client
           InetAddress clientAddress = receivePacket.getAddress();
int clientPort = receivePacket.getPort();
           DatagramPacket sendPacket = new DatagramPacket(message.getBytes(), message.length(),
clientAddress, clientPort);              socket.send(sendPacket);
        }
     } catch (IOException e) {
e.printStackTrace();        }
finally {
```

```
            if (socket != null && !socket.isClosed()) {
socket.close();
            }
        }
    }
}
```

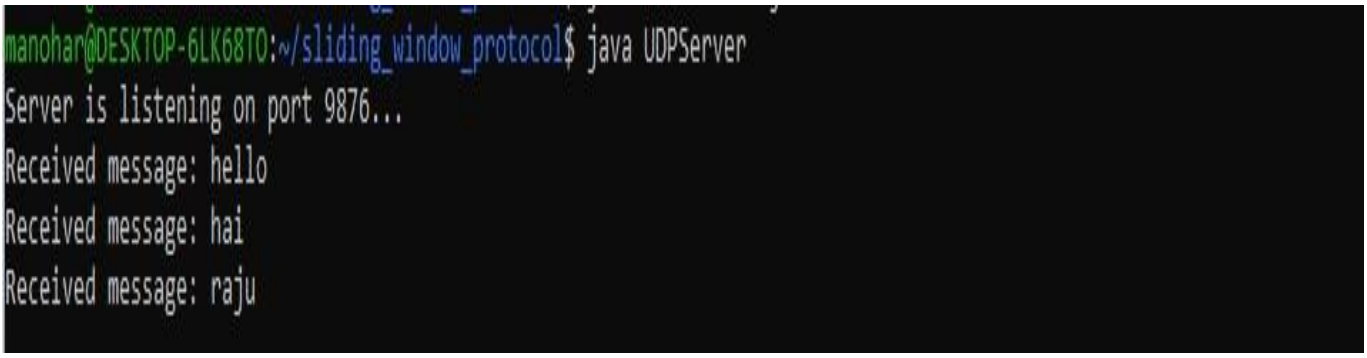
UDPClient.java

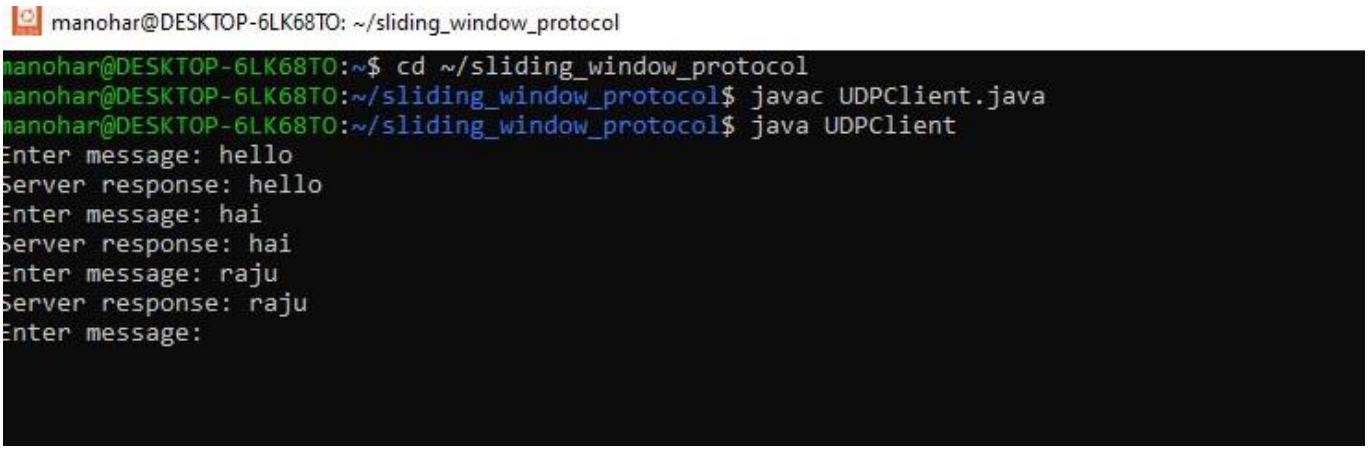
```
import java.io.*; import
java.net.*;
```

```
public class UDPClient {
   public static void main(String[] args) {
DatagramSocket socket = null;
BufferedReader userInput = null; try
{
        // Create a DatagramSocket
socket = new DatagramSocket();
        InetAddress serverAddress = InetAddress.getByName("localhost");
int serverPort = 9876;

        userInput = new BufferedReader(new InputStreamReader(System.in));
String message;

        while (true) {
            // Read user input
            System.out.print("Enter message: ");
message = userInput.readLine();

            if (message.equalsIgnoreCase("exit")) {
break;
            }

            // Send the message to the server
            DatagramPacket sendPacket = new DatagramPacket(message.getBytes(), message.length(),
serverAddress, serverPort);              socket.send(sendPacket);

            // Receive the response from the server
            byte[] receiveData = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
socket.receive(receivePacket);
```

```
            String response = new String(receivePacket.getData(), 0, receivePacket.getLength());
System.out.println("Server response: " + response);
        }
    } catch (IOException e) {
e.printStackTrace();        }
finally {
        if (socket != null && !socket.isClosed()) {
socket.close();
        }
try {
            if (userInput != null) userInput.close();
} catch (IOException e) {
e.printStackTrace();
        }
    }
  }
}
```

Output:-





Lab Program 9:

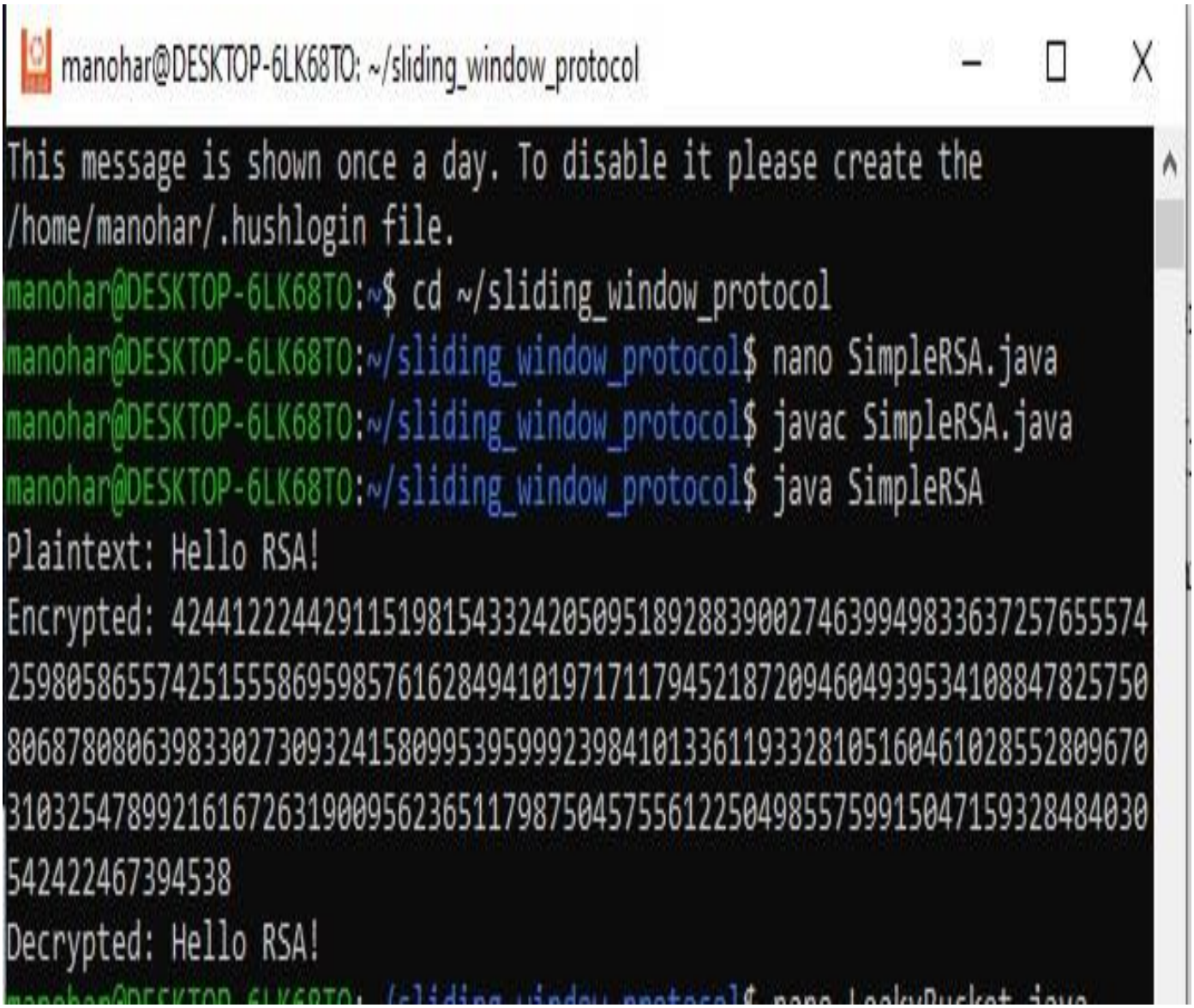
Develop a program for a simple RSA algorithm to encrypt and decrypt the data.

Source code:-

```
import java.math.BigInteger; import
java.security.SecureRandom;

public class SimpleRSA {
private BigInteger n, d, e;
private int bitLength = 1024;
   private SecureRandom random = new SecureRandom();

   public SimpleRSA() {
generateKeys();
   }
   // Method to generate public and private keys
private void generateKeys() {
      BigInteger p = BigInteger.probablePrime(bitLength / 2, random);
BigInteger q = BigInteger.probablePrime(bitLength / 2, random);        n
= p.multiply(q);

      BigInteger phi = (p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));

      e = BigInteger.probablePrime(bitLength / 2, random);
while (phi.gcd(e).intValue() > 1) {
         e = BigInteger.probablePrime(bitLength / 2, random);
      }

      d = e.modInverse(phi);
   }

   // Method to encrypt a plaintext message
   public BigInteger encrypt(String message) {        return new
BigInteger(message.getBytes()).modPow(e, n);
```

```
    }

    // Method to decrypt a ciphertext message     public String
decrypt(BigInteger encrypted) {         return new
String(encrypted.modPow(d, n).toByteArray());
    }

    // Getters for the public key components
public BigInteger getN() {         return n;
    }

    public BigInteger getE() {
return e;
    }

    public static void main(String[] args) {
        SimpleRSA rsa = new SimpleRSA();

        String plaintext = "Hello RSA!";
        System.out.println("Plaintext: " + plaintext);

        // Encrypt the plaintext
        BigInteger encrypted = rsa.encrypt(plaintext);
        System.out.println("Encrypted: " + encrypted);

        // Decrypt the ciphertext
        String decrypted = rsa.decrypt(encrypted);
        System.out.println("Decrypted: " + decrypted);     } }
```

Output:-


manohar@DESKTOP-6LK68TO: ~/sliding_window_protocol
This message is shown once a day. To disable it please create the
/home/manohar/.hushlogin file.
manohar@DESKTOP-6LK68TO:~$ cd ~/sliding_window_protocol
manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$ nano SimpleRSA.java
manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$ javac SimpleRSA.java
manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$ java SimpleRSA
Plaintext: Hello RSA!
Encrypted: 424412224429115198154332420509518928839002746399498336372576555742598058655742515558695985761628494101971711794521872094604939534108847825750806878080639833027309324158099539599923984101336119332810516046102855280967031032547899216167263190095623651179875045755612250498557599150471593284840305424224673945 38
Decrypted: Hello RSA!

Lab Program 10:

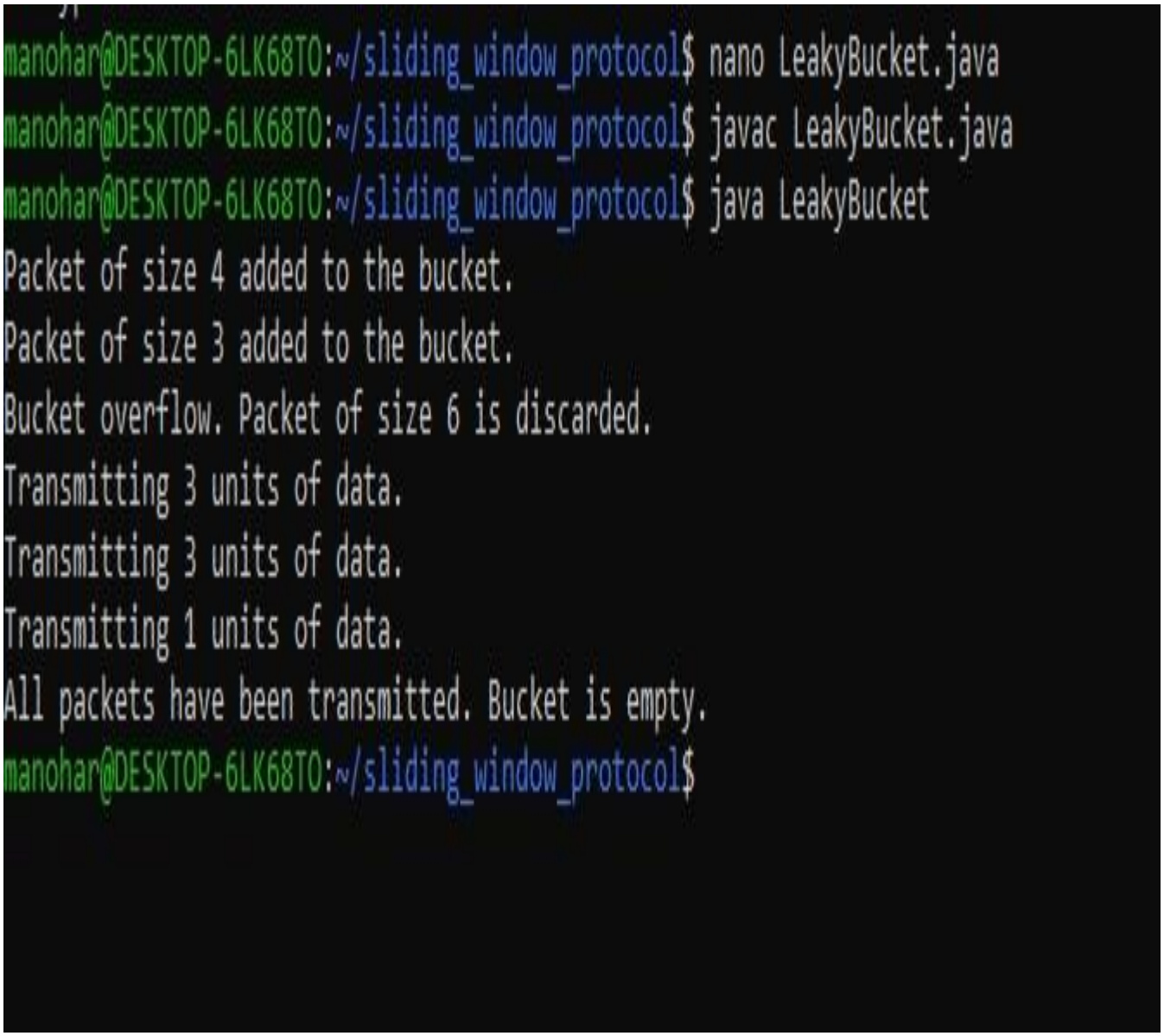
Develop a program for congestion control using a leaky bucket algorithm.

Source code:- import java.util.LinkedList; import java.util.Queue;

```
public class LeakyBucket {
   private final int bucketSize;  // Maximum capacity of the bucket
   private final int outputRate;  // Rate at which water (packets) leaks (transmits)
private int currentWater;      // Current amount of water (packets) in the bucket

   private Queue<Integer> packetQueue;

   public LeakyBucket(int bucketSize, int outputRate) {
this.bucketSize = bucketSize;         this.outputRate =
outputRate;         this.currentWater = 0;
       this.packetQueue = new LinkedList<>();
   }
   // Method to add packets to the bucket     public
void addPacket(int packetSize) {         if
(packetSize + currentWater > bucketSize) {
           System.out.println("Bucket overflow. Packet of size " + packetSize + " is discarded.");
       } else {
           currentWater += packetSize;
packetQueue.add(packetSize);
           System.out.println("Packet of size " + packetSize + " added to the bucket.");
       }
   }

   // Method to simulate packet transmission (leakage)
public void transmitPackets() {         while
(!packetQueue.isEmpty()) {
           int transmitted = Math.min(currentWater, outputRate);
```

```
            System.out.println("Transmitting " + transmitted + " units of data.");
currentWater -= transmitted;
            while (!packetQueue.isEmpty() && packetQueue.peek() <= transmitted) {
transmitted -= packetQueue.poll();
            }
            if (transmitted > 0 && !packetQueue.isEmpty()) {
packetQueue.add(packetQueue.poll() - transmitted);
            }

            // Simulate a time delay between packet transmissions
try {
                Thread.sleep(1000); // 1 second delay
} catch (InterruptedException e) {
e.printStackTrace();
            }
        }
        System.out.println("All packets have been transmitted. Bucket is empty.");
    }
    public static void main(String[] args) {
        LeakyBucket leakyBucket = new LeakyBucket(10, 3); // Bucket size 10 units, output rate 3 units per
second

        leakyBucket.addPacket(4);
leakyBucket.addPacket(3);
        leakyBucket.addPacket(6);  // This packet will be discarded due to overflow

        leakyBucket.transmitPackets();        }
```

Output:-


manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$ nano LeakyBucket.java
manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$ javac LeakyBucket.java
manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$ java LeakyBucket
Packet of size 4 added to the bucket.
Packet of size 3 added to the bucket.
Bucket overflow. Packet of size 6 is discarded.
Transmitting 3 units of data.
Transmitting 3 units of data.
Transmitting 1 units of data.
All packets have been transmitted. Bucket is empty.
manohar@DESKTOP-6LK68TO:~/sliding_window_protocol$