SHIV NADAR
UNIVERSITY
CHENNAI

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF ENGINEERING

LABORATORY RECORD

B.TECH
(YEAR : 20 - 20)

NAME : Rasuranth

REG. NO. : 2101110 2083

DEPT. : CSE **SEM.** : 4 **CLASS & SEC** : IOT-B

SHIV NADAR
UNIVERSITY
CHENNAI

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of the practical work done in the

.....Computer Networks..... **Laboratory by**

Name .....RASWANTH.....

Register Number .....2.10111.02083.....

Semester .....4.....Class & Sec. .....IOT-B.....

Branch .....CSE.....

SHIV NADAR UNIVERSITY Chennai

During the Academic year .....2022-23.....

K. Surya.

Faculty **Head of the Department**

Submitted for the.....Practical Examination held at SNU CHENNAI on.....

Internal Examiner **External Examiner**

INDEX

Name : Rasuanth Reg. No. 2101102083

Sem : 4 Class & Sec : IOT-B

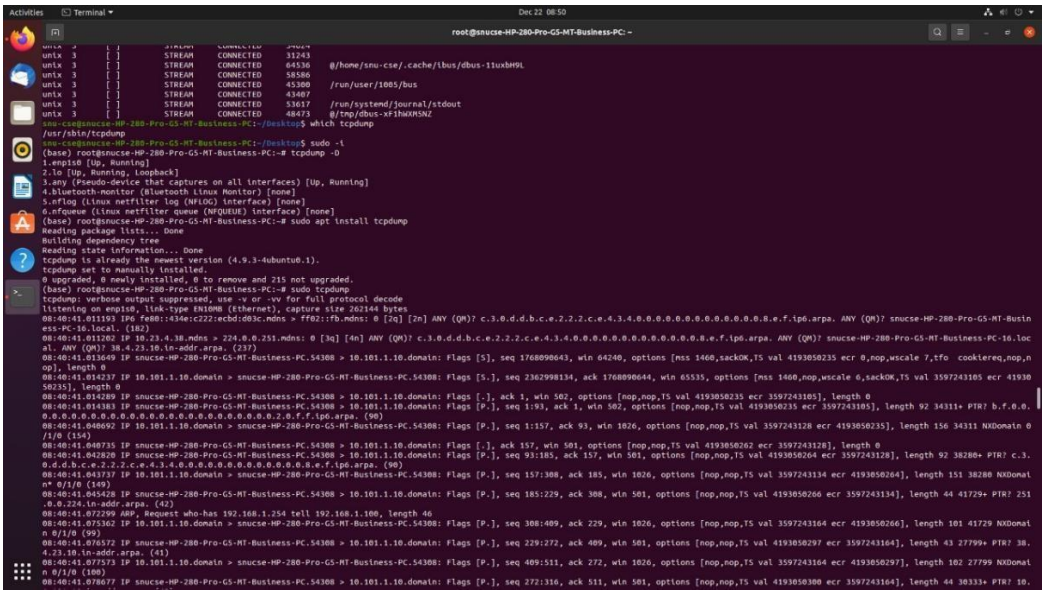
Ex. No.	Date of Expt.	Title of the Experiment	Page No.	Signature of the Faculty	Remarks
1.	20.12.22	Basic Commands	1		
2.	27.12.22	HTTP Web client Program	3		
3.	3.1.23	Applications using TCP Sockets	5		
5.	10.01.23	Simulation of ARP/RARP protocols	16		
4.	10.1.23	Simulation of DNS using UDP sockets	13		
6.	31.1.23	Study of NS & Simulation of Congestion Control algorithms using NS	21		
7.	21.2.23	Simulation of error correction code	28		
8.	28.2.23	Study of TCP/UDP performance using Simulation Tool	30		
9.	7.3.23	Simulation of distance/vector link state routing algorithm	33		
10.	14.3.23	Performance evaluation of Routing Protocols using Simulation Tool.	39		

Ex. No: 1	Basic Commands
20.12.2022	

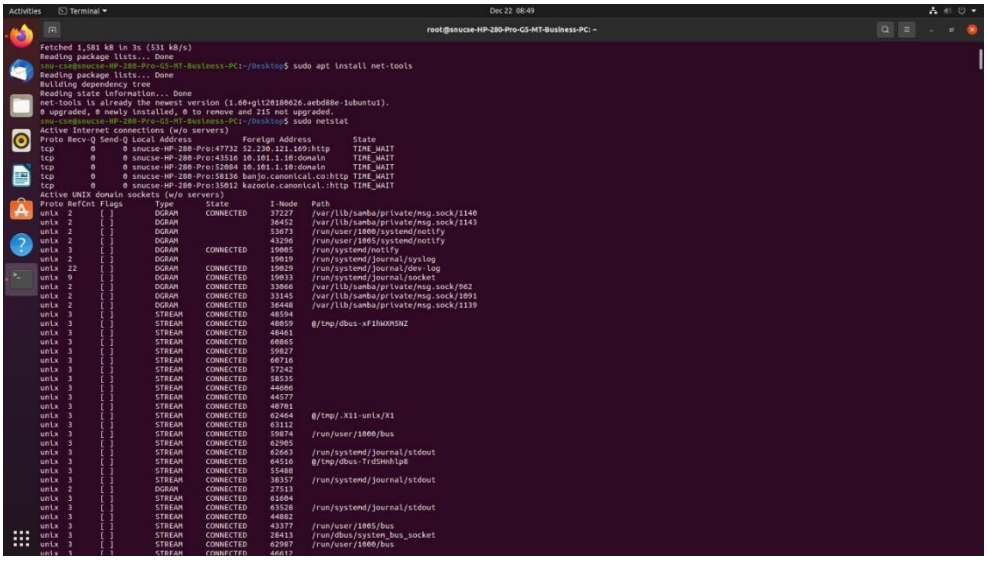
Aim: To use commands like tcpdump, netstat, ifconfig, nslookup and traceroute and to capture ping and trace route PDUs using a network protocol analyzer and examine.

Program and Sample Input/Output :

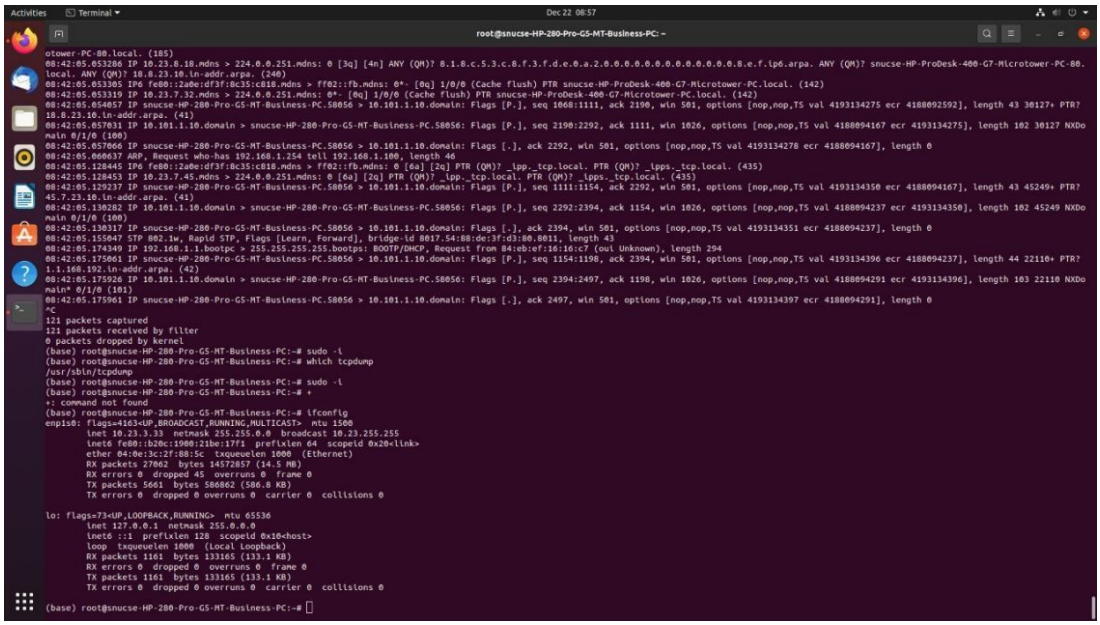
1) **Tcpdump :**



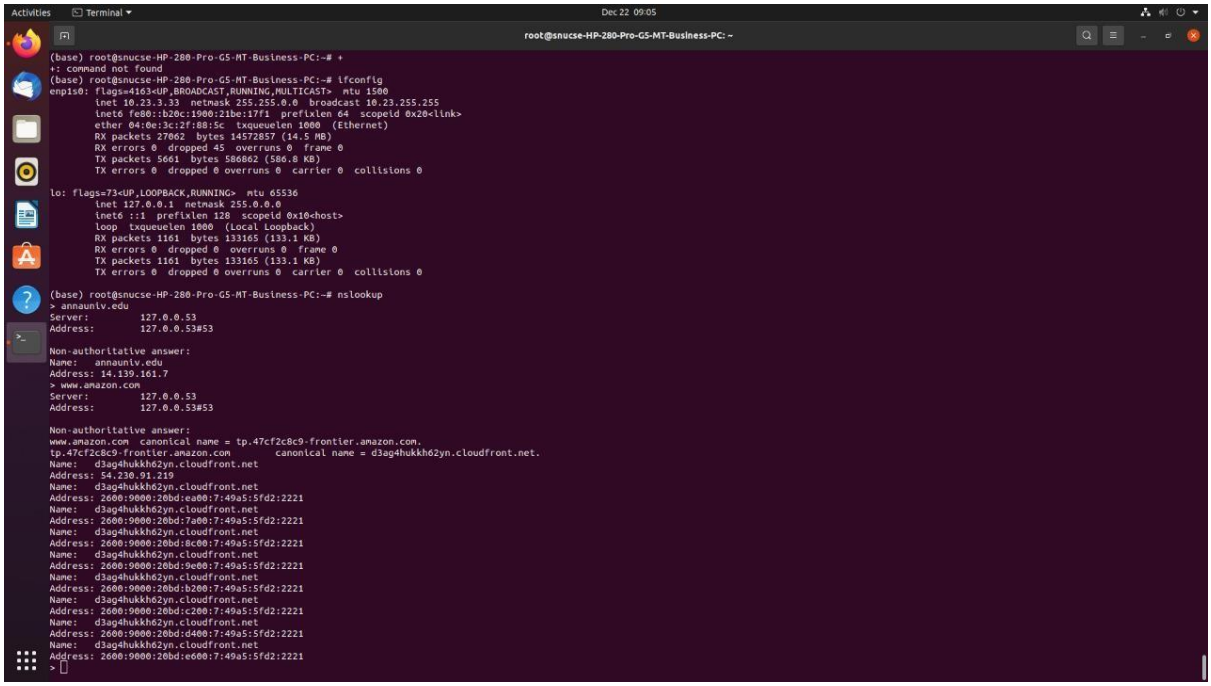
2) **netstat :**



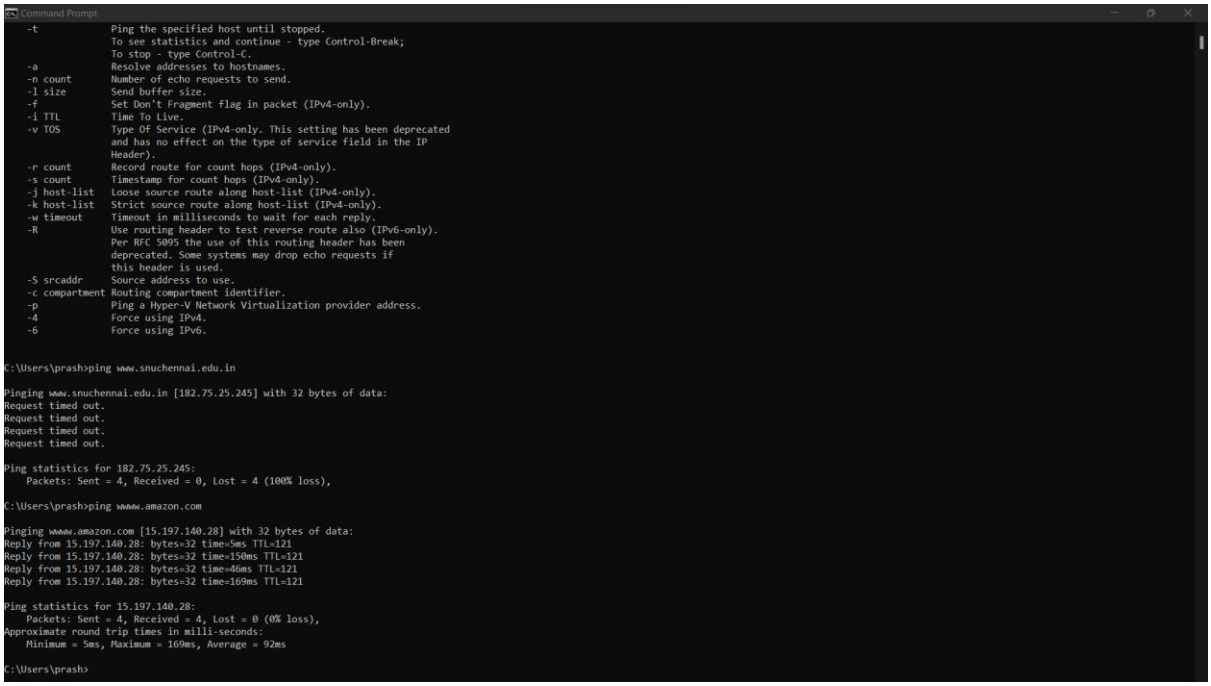
3) ifconfig :



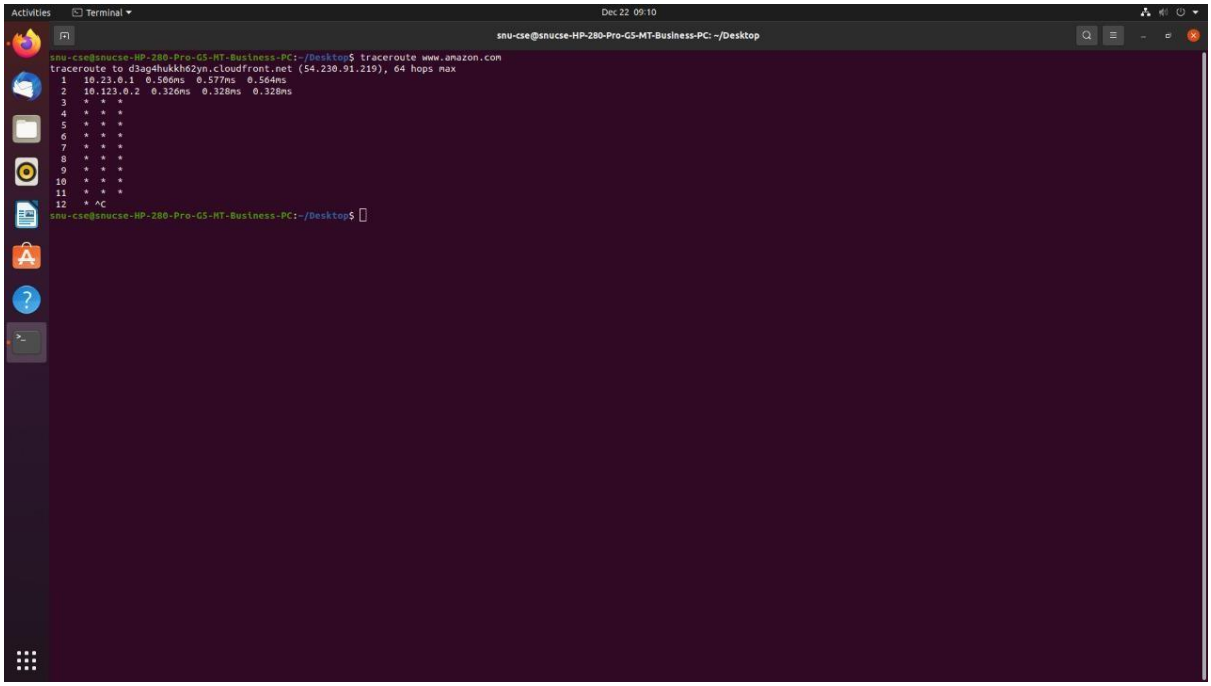
4) nslookup :



5) ping :



6) traceroute :



Result: The basic commands are executed; the ping is captured and trace route of PDUs are examined using network protocol analyzer successfully.

Ex. No: 2	HTTP Web Client Program
27.12.2022	

Aim: To write a HTTP web client program to download a web page using TCP sockets.

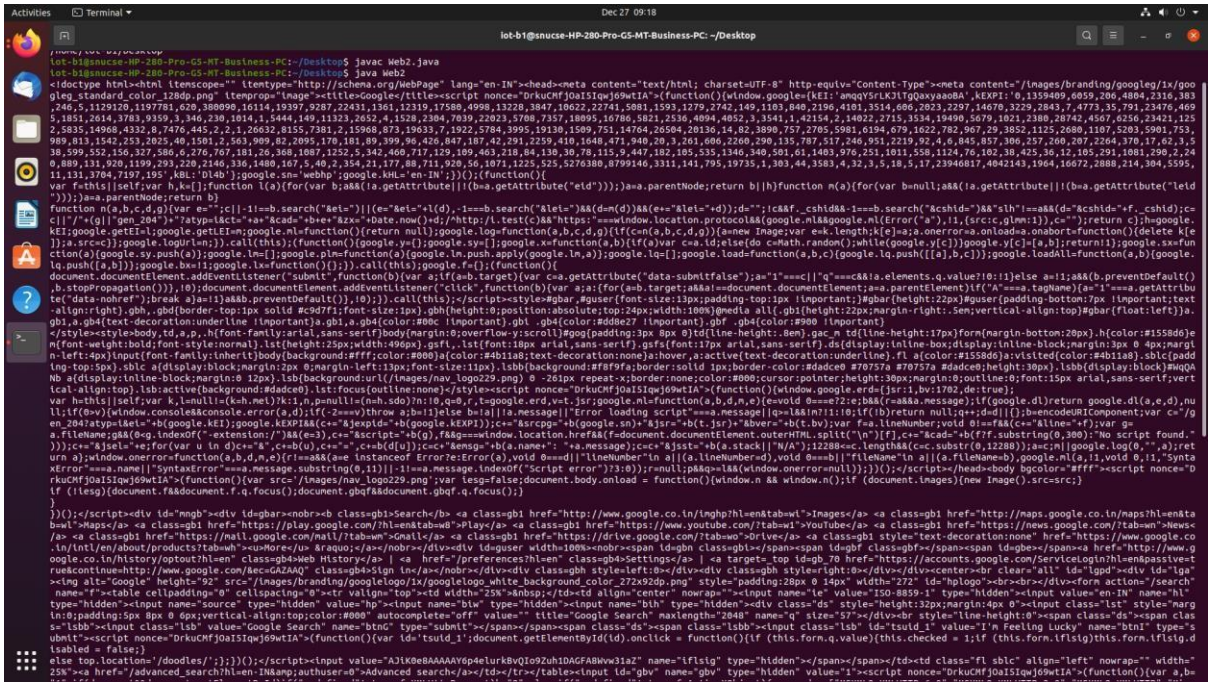
Algorithm:

1. Create a URL object and pass URL as string to download the webpage. URL example = new URL(pass URL of webpage you want to download)
2. Create Buffered Reader object and pass openStream(). Method of URL in Input Stream object.
3. Create a string object to read each line one by one from stream.
4. Write each line in html file where webpage will be downloaded.
5. Close all objects.
6. Catch exceptions if URL failed to download.

Program:

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.net.URL;
public class Web2 {
  public static void main(String[] args) throws Exception {
    URL url = new URL("https://lms.snuchennai.edu.in/");
    BufferedReader reader = new BufferedReader(new
InputStreamReader(url.openStream()));
    BufferedWriter writer = new BufferedWriter(new
FileWriter("data.html"));
    String line;
    while ((line = reader.readLine()) != null){
      System.out.println(line);
      writer.write(line);
      writer.newLine();
    }
    reader.close();
    writer.close();
  }
}
```

Sample Input/Output:



Result: The web page has been downloaded using TCP Sockets successfully.

Ex. No: 3	Applications using TCP Sockets
03.01.2023	

Aim: To write client-server programs using TCP/IP.

a) Socket program for implementation of echo

Algorithm:

CLIENT

1. Create a socket which binds the Ip address of server and the port address to acquire service.
2. After establishing connection send a data to server.
3. Receive and print the same data from server.
4. Close the socket.

SERVER

1. Create a server socket to activate the port address.
2. Create a socket for the server socket which accepts the connection.
3. After establishing connection receive the data from client.
4. Print and send the same data to client.
5. Close the socket.

Program:

CLIENT

```
import java.io.*;
import java.net.*;
public class eclient{
  public static void main(String args[]){
    Socket c=null;
    String line;
    DataInputStream is,is1;
    PrintStream os;
    try{
      c=new Socket("localhost",8080);
    }
    catch(IOException e){
      System.out.println(e);
    }
    try{
      os=new PrintStream(c.getOutputStream());
      is=new DataInputStream(System.in);
```

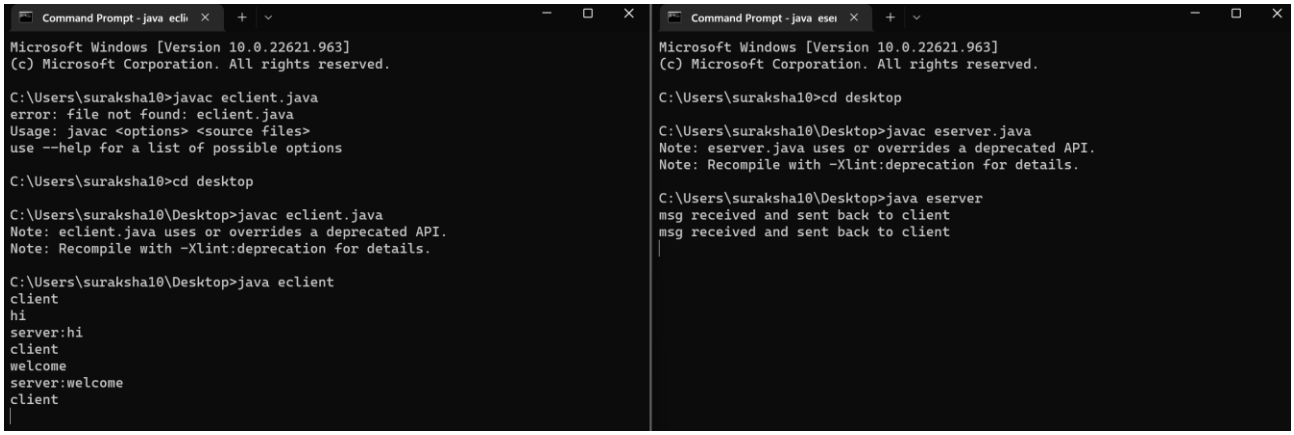
```
      is1=new DataInputStream(c.getInputStream());
      do{
        System.out.println("client");
        line=is.readLine();
        os.println(line);
        if(!line.equals("exit"))
        System.out.println("server:"+is1.readLine());
      }while(!line.equals("exit"));
    }
    catch(IOException e){
      System.out.println("socket closed");
    }
  }
}
```

SERVER

```
import java.io.*;
import java.net.*;
import java.lang.*;
public class eserver{
  public static void main(String args[])throws IOException{
    ServerSocket s=null;
    String line;
    DataInputStream is;
    PrintStream ps;
    Socket c=null;
    try{
      s=new ServerSocket(8080);
    }
    catch(IOException e){
      System.out.println(e);
    }
    try{
      c=s.accept();
      is=new DataInputStream(c.getInputStream());
      ps=new PrintStream(c.getOutputStream());
      while(true){
        line=is.readLine();
        System.out.println("msg received and sent back to client");
        ps.println(line);
      }
```

```
    }
    catch(IOException e){
      System.out.println(e);
    }
  }
}
```

Sample Input/Output:



b) To write a client-server application for chat using TCP

Algorithm:

CLIENT

1. Create a socket in client to server.
2. The client establishes a connection to the server.
3. The client accept the connection and to send the data from client to server.
4. The client communicates the server to send the end of the message.

SERVER

1. Create a socket in server to client
2. The server establishes a connection to the client.
3. The server accept the connection and to send the data from server to client and vice versa
4. The server communicate the client to send the end of the message.

Program:

CLIENT

```
import java.net.*;
import java.io.*;
public class TCPclient1{
 public static void main(String arg[]){
  Socket c=null;
  String line;
```

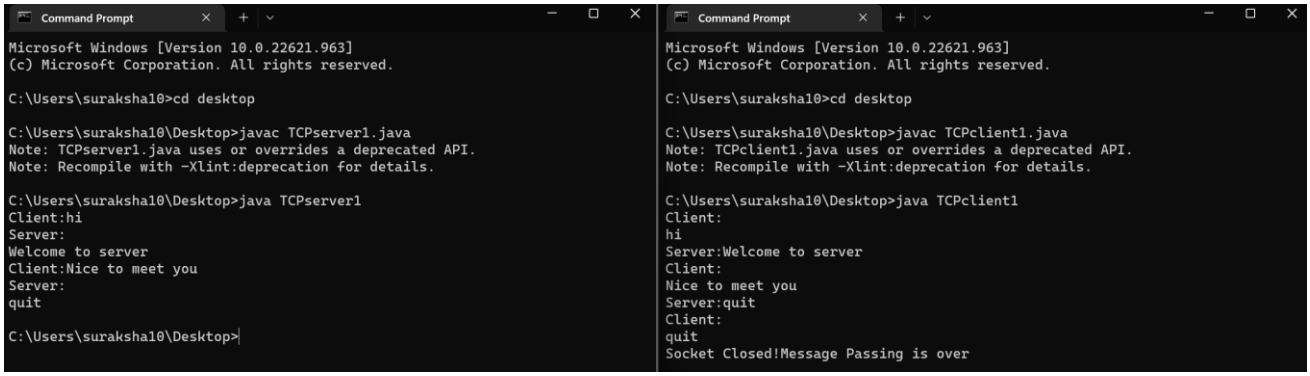
```
  DataInputStream is,is1;
  PrintStream os;
  try{
   c=new Socket("127.0.0.1",9999);
  }
  catch(IOException e){
   System.out.println(e);
  }
  try{
   os=new PrintStream(c.getOutputStream());
   is=new DataInputStream(System.in);
   is1=new DataInputStream(c.getInputStream());
   do{
    System.out.println("Client:");
    line=is.readLine();
    os.println(line);
    System.out.println("Server:" + is1.readLine());
   }
   while(line.equalsIgnoreCase("quit")==false);
   is1.close();
   os.close();
  }
  catch(IOException e){
   System.out.println("Socket Closed!Message Passing is over");
  }
 }
}
```

SERVER

```
import java.net.*;
import java.io.*;
public class TCPserver1{
 public static void main(String arg[]){
  ServerSocket s=null;
  String line;
  DataInputStream is=null,is1=null;
  PrintStream os=null;
  Socket c=null;
  try{
   s=new ServerSocket(9999);
  }
```

```
  catch(IOException e){
   System.out.println(e);
  }
  try{
   c=s.accept();
   is=new DataInputStream(c.getInputStream());
   is1=new DataInputStream(System.in);
   os=new PrintStream(c.getOutputStream());
   do{
    line=is.readLine();
    System.out.println("Client:"+line);
    System.out.println("Server:");
    line=is1.readLine();
    os.println(line);
   }
   while(line.equalsIgnoreCase("quit")==false);
   is.close();
   os.close();
  }
  catch(IOException e){
   System.out.println(e);
  }
 }
}
```

Sample Input/Output:



c) To Perform File Transfer in Client & Server Using TCP/IP.

Algorithm:

CLIENT

1. Establish a connection between the Client and Server.
2. Socket ss=new Socket(InetAddress.getLocalHost(),1100);

3. Implement a client that can send two requests:
   i. To get a file from the server.
   ii. To put or send a file to the server.
4. After getting approval from the server ,the client either get file from the server or send file to the server.

SERVER

1. Implement a server socket that listens to a particular port number.
2. Server reads the filename and sends the data stored in the file for the'get' request.
3. It reads the data from the input stream and writes it to a file in the server for the 'put' instruction.
4. Exit upon client's request.

Program:

CLIENT

```
import java.io.*;
import java.net.Socket;
public class FClient {
 private static DataOutputStream dataOutputStream = null;
 private static DataInputStream dataInputStream = null;
 public static void main(String[] args) {
 // Create Client Socket connect to port 900
 try (Socket socket = new Socket("127.0.0.1", 900)) {
  dataInputStream = new DataInputStream(socket.getInputStream());
  dataOutputStream = new DataOutputStream(socket.getOutputStream());
  System.out.println("Sending the File to the Server");
  // Call SendFile Method
  sendFile("C:/Users/suraksha10/Desktop/Excs 3.pdf");
  dataInputStream.close();
  dataInputStream.close();
 }
 catch (Exception e) {
  e.printStackTrace();
 }
}
// sendFile function define here
private static void sendFile(String path) throws Exception {
 int bytes = 0;
 // Open the File where he located in your pc
 File file = new File(path);
 FileInputStream fileInputStream = new FileInputStream(file);
 // Here we send the File to Server
 dataOutputStream.writeLong(file.length());
```

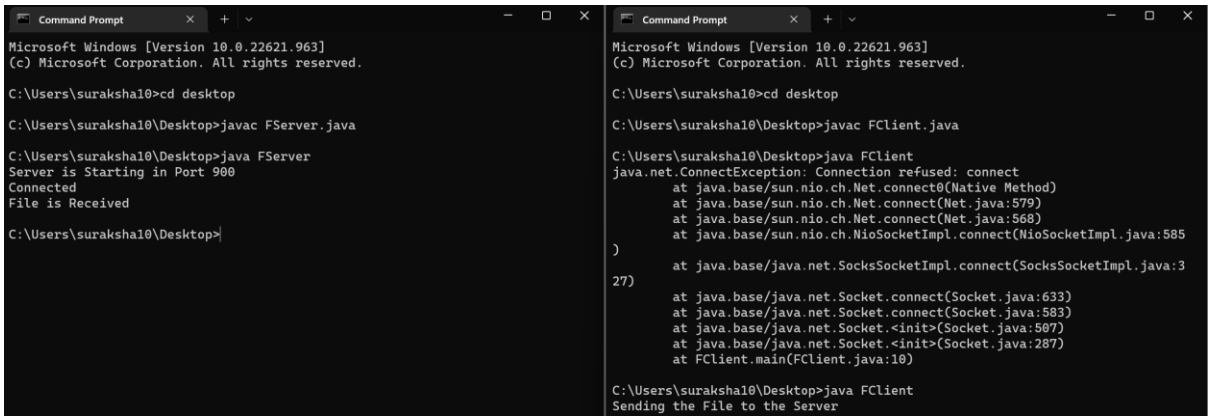
```
 // Here we break file into chunks
 byte[] buffer = new byte[4 * 1024];
 while ((bytes = fileInputStream.read(buffer)) != -1) {
  // Send the file to Server Socket
  dataOutputStream.write(buffer, 0, bytes);
  dataOutputStream.flush();
 }
 // close the file here
 fileInputStream.close();
 }
}
```

SERVER

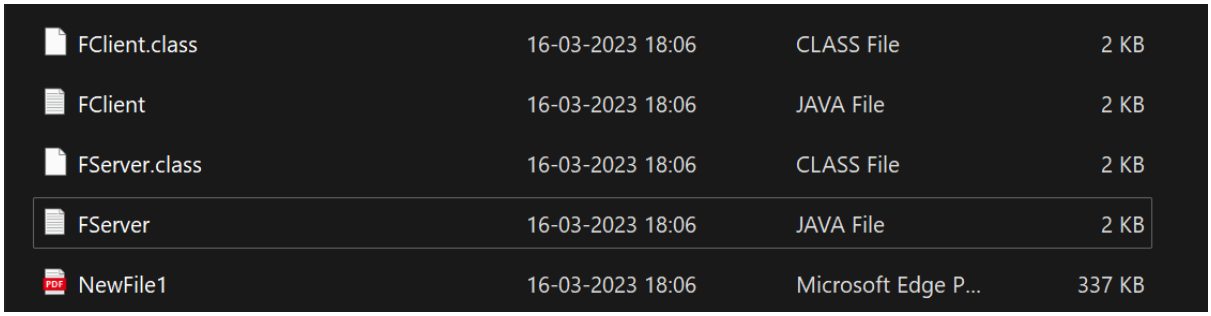
```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.FileOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
public class FServer {
 private static DataOutputStream dataOutputStream = null;
 private static DataInputStream dataInputStream = null;
 public static void main(String[] args){
 // Here we define Server Socket running on port 900
 try (ServerSocket serverSocket = new ServerSocket(900)) {
  System.out.println("Server is Starting in Port 900");
  // Accept the Client request using accept method
  Socket clientSocket = serverSocket.accept();
  System.out.println("Connected");
  dataInputStream = new DataInputStream(clientSocket.getInputStream());
  dataOutputStream = new DataOutputStream(clientSocket.getOutputStream());
  // Here we call receiveFile define new for that file
  receiveFile("NewFile1.pdf");
  dataInputStream.close();
  dataOutputStream.close();
  clientSocket.close();
 }
 catch (Exception e) {
  e.printStackTrace();
 }
}
// receive file function is start here
```

```
private static void receiveFile(String fileName) throws Exception{
 int bytes = 0;
 FileOutputStream fileOutputStream = new FileOutputStream(fileName);
 long size = dataInputStream.readLong(); // read file size
 byte[] buffer = new byte[4 * 1024];
 while (size > 0 && (bytes = dataInputStream.read(buffer, 0,
                              (int)Math.min(buffer.length, size)))!= -1) {
  // Here we write the file using write method
  fileOutputStream.write(buffer, 0, bytes);
  size -= bytes; // read upto file size
 }
 // Here we received file
 System.out.println("File is Received");
 fileOutputStream.close();
 }
}
```

Sample Input/Output:



A new file was created in the specified location.



Result: The Client-server programs using TCP/IP has been written and executed successfully in Java environment.

Ex. No: 4	**Simulation of DNS using UDP Sockets**
10.01.2023	

Aim: To write code for simulation of DNS using UDP sockets.

Algorithm:

CLIENT

1. Create a datagram socket
2. Get domain name from user
3. Create a datagram packet and send domain name to the server
4. Create a datagram packet to receive server message
5. Read server's response
6. If ip address is found then display it else display "Domain does not exist"
7. Close the client socket

SERVER

1. Create an array of hosts and its ip address in another array
2. Create a datagram socket and bind it to a port
3. Create a datagram packet to receive client request
4. Read the domain name from client to be resolved.
5. Lookup the host array for the domain name
6. If found then retrieve corresponding address
7. Create a datagram packet and send ip address to client
8. Repeat steps 3-7 to resolve further requests from clients
9. Close the server socket

Program:

CLIENT

```
import java.io.*;
import java.net.*;
public class udpdnsclient{
public static void main(String args[])throws IOException{
 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
 DatagramSocket clientsocket = new DatagramSocket();
 InetAddress ipaddress;
 if (args.length == 0)
  ipaddress = InetAddress.getLocalHost();
 else
  ipaddress = InetAddress.getByName(args[0]);
 byte[] senddata = new byte[1024];
 byte[] receivedata = new byte[1024];
```

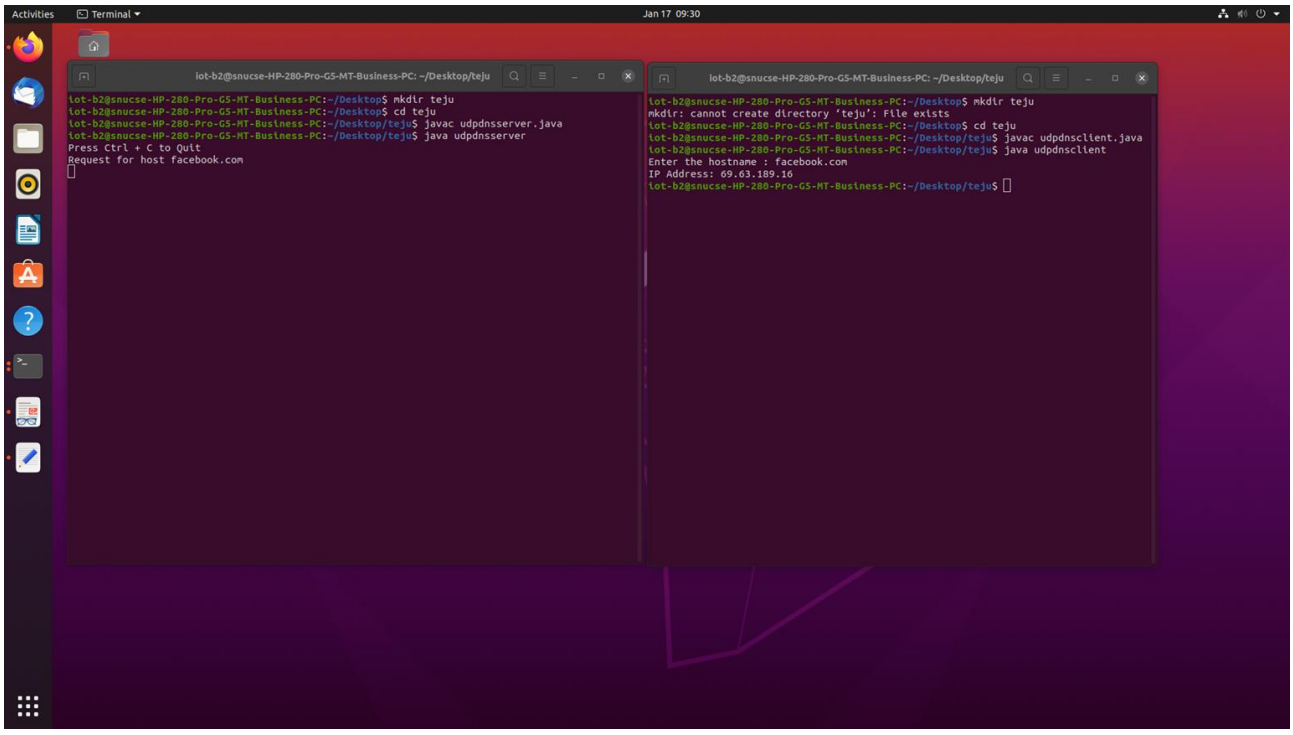
```
 int portaddr = 1362;
 System.out.print("Enter the hostname : ");
 String sentence = br.readLine();
 senddata = sentence.getBytes();
 DatagramPacket pack = new DatagramPacket(senddata,senddata.length,
ipaddress,portaddr);
 clientsocket.send(pack);
 DatagramPacket recvpack = new DatagramPacket(receivedata,
receivedata.length);
 clientsocket.receive(recvpack);
 String modified = new String(recvpack.getData());
 System.out.println("IP Address: " + modified);
 clientsocket.close();
 }
}
```

SERVER

```
import java.io.*;
import java.net.*;
public class udpdnsserver{
 private static int indexOf(String[] array, String str){
  str = str.trim();
  for (int i=0; i < array.length; i++){
   if (array[i].equals(str))
    return i;
  }
  return -1;
 }
 public static void main(String arg[])throws IOException{
  String[] hosts = {"yahoo.com", "gmail.com","cricinfo.com",
"facebook.com"};
  String[] ip = {"68.180.206.184", "209.85.148.19","80.168.92.140",
"69.63.189.16"};
  System.out.println("Press Ctrl + C to Quit");
  while (true){
   DatagramSocket serversocket=new DatagramSocket(1362);
   byte[] senddata = new byte[1021];
   byte[] receivedata = new byte[1021];
   DatagramPacket recvpack = new
   DatagramPacket(receivedata, receivedata.length);
   serversocket.receive(recvpack);
   String sen = new String(recvpack.getData());
```

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

Sample Input/Output:



Result: The code for simulation of DNS using UDP sockets have been written and executed successfully in Java Environment

Ex. No: 5	**Simulation of ARP / RARP Protocols**
17.01.2023	

Aim: To write codes simulating ARP /RARP protocols.

a) ARP

Algorithm:

CLIENT

1. Using socket connection is established between client and server.
2. Get the IP address to be converted into MAC address.
3. Send this IP address to server.
4. Server returns the MAC address to client.

SERVER

1. Accept the socket which is created by the client.
2. Server maintains the table in which IP and corresponding MAC addresses are stored.
3. Read the IP address which is send by the client.
4. Map the IP address with its MAC address and return the MAC address to client.

Program:

CLIENT

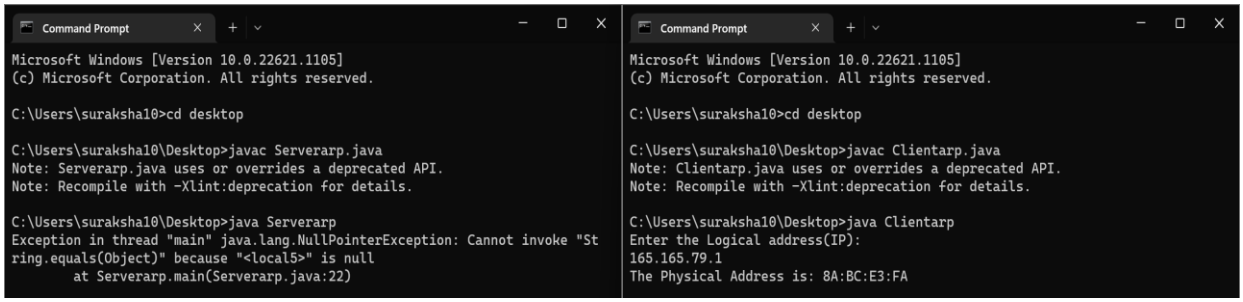
```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientarp{
 public static void main(String args[]){
  try{
   BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
   Socket clsct=new Socket("127.0.0.1",26732);
   DataInputStream din=new DataInputStream(clsct.getInputStream());
   DataOutputStream dout=new
   DataOutputStream(clsct.getOutputStream());
   System.out.println("Enter the Logical address(IP):");
   String str1=in.readLine();
   dout.writeBytes(str1+'\n');
   String str=din.readLine();
   System.out.println("The Physical Address is: "+str);
   clsct.close();
  }
  catch (Exception e){
```

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

SERVER

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverarp{
 public static void main(String args[])throws Exception{
  ServerSocket obj=new
  ServerSocket(5604);
  Socket obj1=obj.accept();
  while(true){
   DataInputStream din=new DataInputStream(obj1.getInputStream());
   DataOutputStream dout=new
   DataOutputStream(obj1.getOutputStream());
   String str=din.readLine();
   String ip[]={"165.165.80.80","165.165.79.1"};
   String mac[]={"6A:08:AA:C2","8A:BC:E3:FA"};
   for(int i=0;i<ip.length;i++){
    if(str.equals(ip[i])){
     dout.writeBytes(mac[i]+'\n');
     break;
    }
   }
   obj.close();
  }
 }
}
```

Sample Input/Output:



b) RARP

Algorithm:

CLIENT

1. Using datagram sockets, UDP function is established.
2. Get the MAC address to be converted into IP address.
3. Send this MAC address to server.
4. Server returns the IP address to client.

SERVER

1. Server maintains the table in which IP and corresponding MAC addresses are stored.
2. Read the MAC address which is send by the client.
3. Map the IP address with its MAC address and return the IP address to client.

Program:

CLIENT

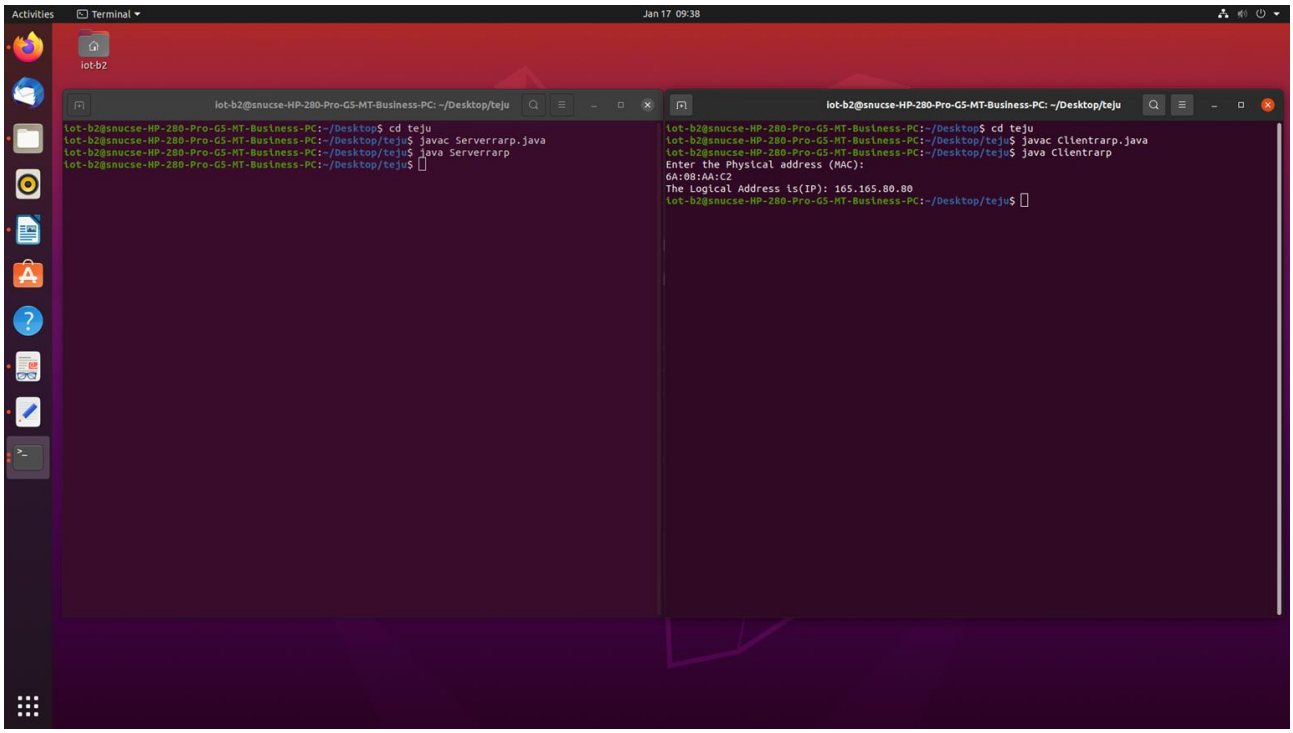
```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientrarp{
 public static void main(String args[]){
  try{
   DatagramSocket client=new DatagramSocket();
   InetAddress addr=InetAddress.getByName("127.0.0.1");
   byte[] sendbyte=new byte[1024];
   byte[] receivebyte=new byte[1024];
   BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
   System.out.println("Enter the Physical address (MAC):");
   String str=in.readLine(); sendbyte=str.getBytes();
   DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,
addr,1309);
   client.send(sender);
   DatagramPacket receiver=new DatagramPacket(receivebyte,
receivebyte.length);
   client.receive(receiver);
   String s=new String(receiver.getData());
   System.out.println("The Logical Address is(IP): "+s.trim());
   client.close();
  }
  catch(Exception e){
   System.out.println(e);
```

```
  }
 }
}
```

SERVER

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverrarp{
 public static void main(String args[]){
  try{
   DatagramSocket server=new DatagramSocket(1309);
   while(true){
    byte[] sendbyte=new byte[1024];
    byte[] receivebyte=new byte[1024];
    DatagramPacket receiver=new DatagramPacket(receivebyte,
receivebyte.length);
    server.receive(receiver);
    String str=new String(receiver.getData());
    String s=str.trim();
    InetAddress addr=receiver.getAddress();
    int port=receiver.getPort();
    String ip[]={"165.165.80.80","165.165.79.1"};
    String mac[]={"6A:08:AA:C2","8A:BC:E3:FA"};
    for(int i=0;i<ip.length;i++){
     if(s.equals(mac[i])){
      sendbyte=ip[i].getBytes();
      DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,
addr,port);
      server.send(sender);
      break;
     }
    }
    break;
   }
  }
  catch(Exception e){
   System.out.println(e);
  }
 }
}
```

Sample Input/Output:



Result: The codes for simulating ARP/RARP protocols has been written and executed successfully in Java environment.

Ex. No: 6	**Study of Network Stimulator (NS) and Simulation of Congestion Control Algorithms using NS**
31.01.2023	

Aim: To write codes for creating a simple topology and simulation of congestion control algorithms using Network Simulator

a) Simple topology using NS2

Algorithm:

1. Start network simulator OTCL editor.
2. Create new simulator using **set ns [new Simulator]**
3. Create Trace route to Network Animator **set nf [open out.nam w] $ns namtrace-all $nf**
4. Create procedure to trace all path.

   proc create_testnet {} {

   global s1 s2 r1 k1

   set s1 [$ns node]

   set s2 [$ns node]

   set r1 [$ns node]

   set k1 [$ns node]

   }
5. Create full duplex connection using:
   $ns duplex-link $s1 $r1 8Mb 5ms drop-tail
   $ns duplex-link $s2 $r1 8Mb 5ms drop-tail
   set L [ns_duplex $r1 $k1 800Kb 100ms drop-tail]
6. Connect with TCP and SINK command. **$ns connect $tcp $sink**
7. Run and execute the program. **$ns run**

Program:

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

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf
set tf [open out.tr w]
$ns trace-all $tf
```

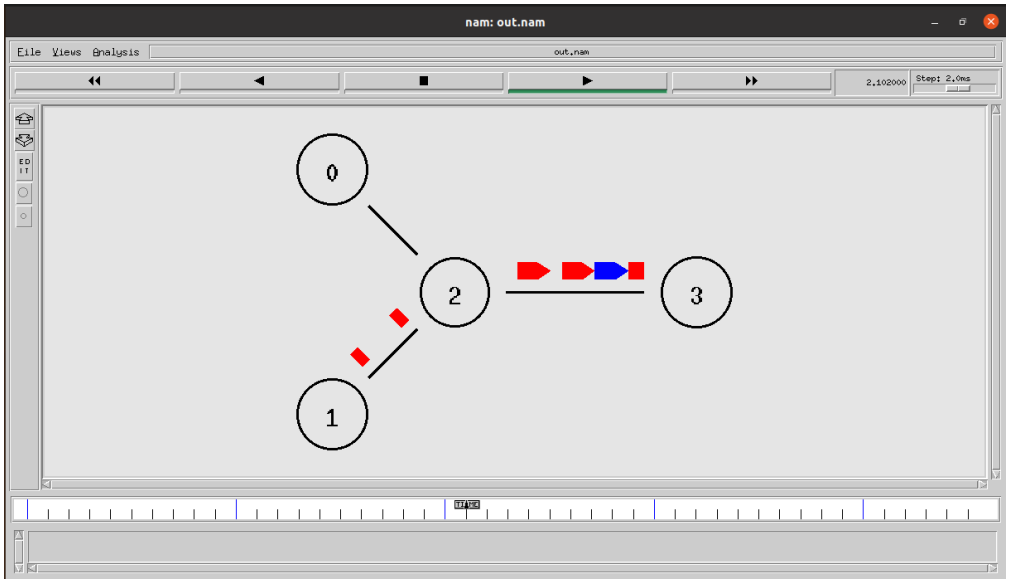
```
#Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Execute NAM on the trace file
        exec nam out.nam &
        exit 0
}

#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 5Mb 10ms DropTail
$ns duplex-link $n1 $n2 5Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 2

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

#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5

#Setup a TCP connection
#using these we are pointing to tcp agent which is object in ns
set tcp [new Agent/TCP]

#class_2 means traffic will be of vbr(variable bit rate) type
$tcp set class_ 2

#$tcp will be source node
```

```
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]

#$sink that will be destination node
$ns attach-agent $n3 $sink
$ns connect $tcp $sink

#fid means color id
$tcp set fid_ 1

#Setup a FTP over TCP connection
#by default ftp will be using tcp connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
#set type using type_ftp
$ftp set type_ FTP

#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp

#ending node wll be always null for udp as there is no destination for udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR

#1000 bits/second
$cbr set packet_size_ 1000
#rate of transfer or bandwidth
$cbr set rate_ 1mb
#to send packets in random order
$cbr set random_ false

#Schedule events for the CBR and FTP agents
#at 0.1 second start cbr appllication
```

```
#$ns at <time> <event>
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#scheduler is started when we will run ns i.e. through $ns run
#Detach tcp and sink agents after 4.5 second detach agent from tcp
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Print CBR packet size and interval
#puts is the command for showing something on console view
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

#Run the simulation
$ns run
```

Sample Input/Output:

```
snu-cse@linux:~$ gedit simple.tcl
snu-cse@linux:~$ ns simple.tcl
 When configured, ns found the right version of tclsh in /usr/bin/tclsh8.6
but it doesn't seem to be there anymore, so ns will fall back on running the first tclsh in your
 path. The wrong version of tclsh may break the test suites. Reconfigure and rebuild ns if this
is a problem.
CBR packet size = 1000
CBR interval = 0.0080000000000000002
```



b) Simulation of Congestion Control Algorithm (CAA) using NS

Algorithm:

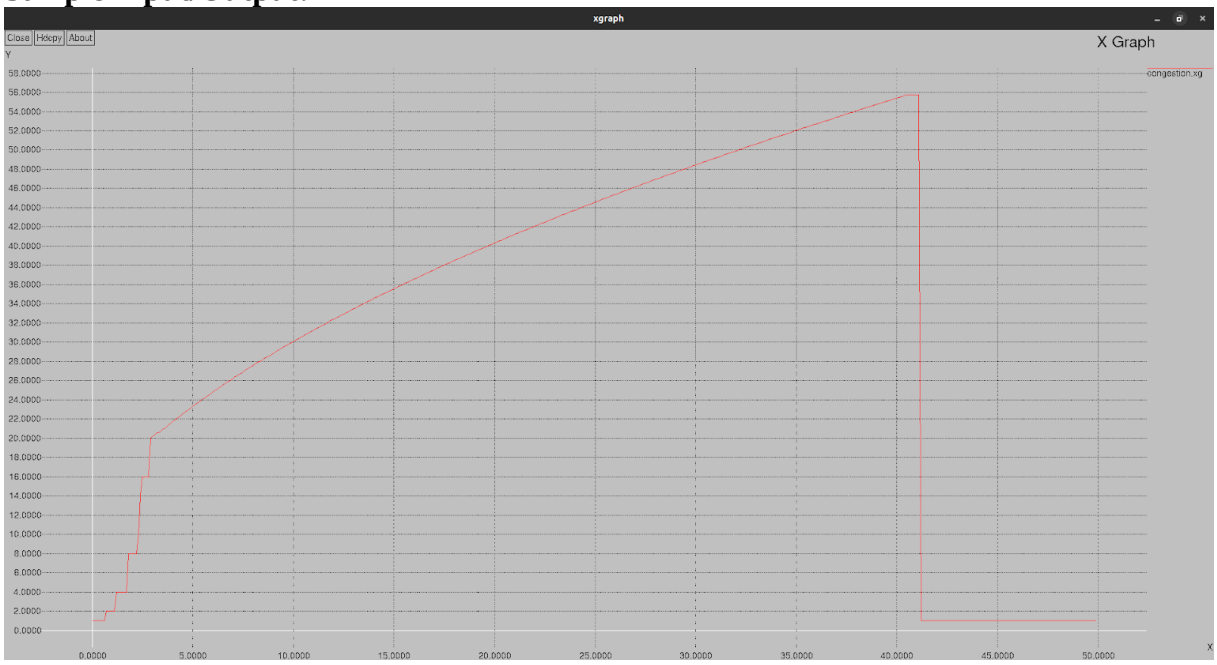
1. Create a simulation object.
2. Set routing protocol to routing.
3. Trace packets and all links onto NAM trace and to trace file.
4. Create right nodes.
5. Describe their layout topology as octagon.
6. Add a sink agent to node.
7. Connect source and sink.
8. Run the simulation.

Program:

```
#create simulator
set ns [new Simulator]
#to create 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]

# to create the link between the nodes with bandwidth, delay and queue
$ns duplex-link $n0 $n2   2Mb  10ms DropTail
$ns duplex-link $n1 $n2   2Mb  10ms DropTail
$ns duplex-link $n2 $n3 0.3Mb 200ms DropTail
$ns duplex-link $n3 $n4 0.5Mb  40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb  30ms DropTail

# Sending node is 0 with agent as Reno Agent
set tcp1 [new Agent/TCP/Reno]
$ns attach-agent $n0 $tcp1
# receiving (sink) node is n4
set sink1 [new Agent/TCPSink]
$ns attach-agent $n4 $sink1
# establish the traffic between the source and sink
$ns connect $tcp1 $sink1
# Setup a FTP traffic generator on "tcp1"
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP
```

```
# start/stop the traffic
$ns at 0.1   "$ftp1 start"
$ns at 40.0 "$ftp1 stop"

# Set simulation end time
$ns at 50.0 "finish"
# procedure to plot the congestion window
proc plotWindow {tcpSource outfile} {
   global ns
   set now [$ns now]
   set cwnd [$tcpSource set cwnd_]
# the data is recorded in a file called congestion.xg (this can be plotted
# using xgraph or gnuplot. this example uses xgraph to plot the cwnd_
   puts  $outfile  "$now $cwnd"
   $ns at [expr $now+0.1] "plotWindow $tcpSource  $outfile"
}
set outfile [open  "congestion.xg"  w]
$ns  at  0.0  "plotWindow $tcp1  $outfile"
proc finish {} {
      exec xgraph congestion.xg -geometry 300x300 &
   exit 0
}
# Run simulation
$ns run
```

Sample Input/Output:



```
0 1
0.10000000000000001 1
0.20000000000000001 1
0.30000000000000004 1
0.40000000000000002 1
0.5 1
0.59999999999999998 1
0.69999999999999996 2
0.79999999999999993 2
0.89999999999999991 2
0.99999999999999989 2
1.0999999999999999 2
1.2 4
1.3 4
1.4000000000000001 4
1.5000000000000002 4
1.6000000000000003 4
1.7000000000000004 4
1.8000000000000005 8
1.9000000000000006 8
2.0000000000000004 8
2.1000000000000005 8
2.2000000000000006 8
2.3000000000000007 10
2.4000000000000008 14
2.5000000000000009 16
2.600000000000001 16
2.7000000000000011 16
2.8000000000000012 16
2.9000000000000012 20
3.0000000000000013 20.1993
3.1000000000000014 20.3474
3.2000000000000015 20.5433
3.3000000000000016 20.592
3.4000000000000017 20.689
3.5000000000000018 20.8816
3.6000000000000019 21.025
3.700000000000002 21.2146
3.800000000000002 21.4025
3.9000000000000021 21.5424
4.0000000000000018 21.7275
4.1000000000000014 21.8653
4.2000000000000011 22.0476
4.3000000000000007 22.2285
4.4000000000000004 22.3632
4.5 22.5415
4.5999999999999996 22.6743
4.6999999999999993 22.8502
4.7999999999999989 23.0248
4.8999999999999986 23.1548
4.9999999999999982 23.3271
5.0999999999999979 23.4555
5.1999999999999975 23.6256
5.2999999999999972 23.7944
"congestion.xg" 500L, 12633C
```

Result: The codes for creating a simple topology and simulation of congestion control algorithms have been written and executed successfully using Network Simulator.

Ex. No: 7	**Simulation of Error Correction Code**
21.02.2023	

Aim:

To write a java code for simulation of any error correction code.

Algorithm:

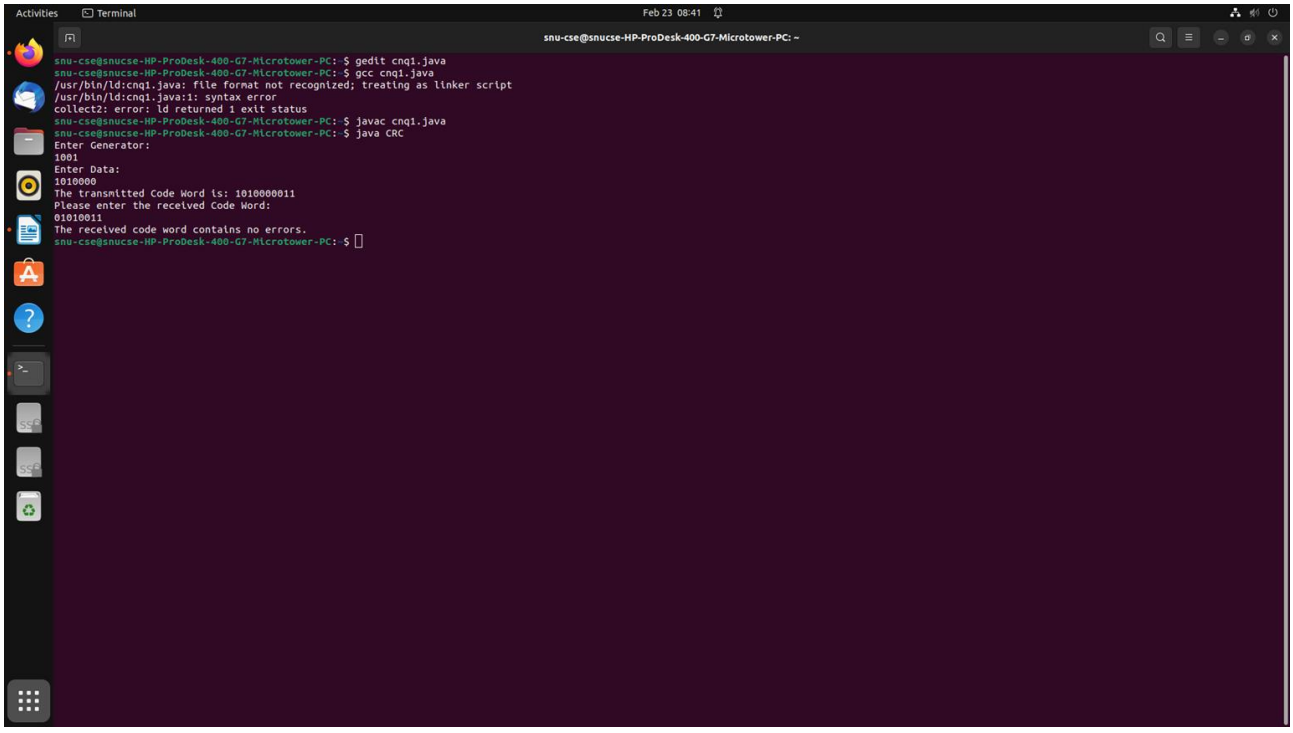
1. The original message (dataword) is considered as M(x) consisting of ‘k’ bits and the divisor as C(x) consists of ‘n+1’ bits.
2. The original message M(x) is appended by ‘n’ bits of zeros. Let us call this zero-extended message as T(x).
3. Divide T(x) by C(x) and find the remainder.
4. The division operation is performed using XOR operation.
5. The resultant remainder is appended to the original message M(x) as CRC and sent by the sender(codeword).

Program:

```
import java.io.*;
class CRC{
public static void main(String args[]) throws IOException{
 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
 System.out.println("Enter Generator:");
 String gen = br.readLine();
 System.out.println("Enter Data:");
 String data = br.readLine();
 String code = data;
 while(code.length() < (data.length() + gen.length() - 1))
 code = code + "0";
 code = data + div(code,gen);
 System.out.println("The transmitted Code Word is: " + code);
 System.out.println("Please enter the received Code Word: ");
 String rec = br.readLine();
 if(Integer.parseInt(div(rec,gen)) == 0)
  System.out.println("The received code word contains no errors.");
 else
  System.out.println("The received code word contains errors.");
}

static String div(String num1,String num2){
 int pointer = num2.length();
 String result = num1.substring(0, pointer);
 String remainder = "";
 for(int i = 0; i < num2.length(); i++){
  if(result.charAt(i) == num2.charAt(i))
   remainder += "0";
```

```
  else
   remainder += "1";
 }
 while(pointer < num1.length()){
  if(remainder.charAt(0) == '0'){
   remainder = remainder.substring(1, remainder.length());
   remainder = remainder + String.valueOf(num1.charAt(pointer));
   pointer++;
  }
  result = remainder;
  remainder = "";
  for(int i = 0; i < num2.length(); i++){
   if(result.charAt(i) == num2.charAt(i))
    remainder += "0";
   else
    remainder += "1";
  }
 }
 return remainder.substring(1,remainder.length());
}
}
```

Sample Input/Output:



Result: The simulation of an error correction code has been done successfully in the Java environment.

Ex. No: 8	**Study of TCP/UDP Performance using Simulation tool**
28.02.2023	

Aim: To study the TCP/UDP performance using Simulation tool.

Algorithm:

1. Create a simulation object.
2. Set routing protocol to routing.
3. Trace packets and all links onto NAM trace and to trace file.
4. Create right nodes.
5. Describe their layout topology.
6. Add a sink agent to node.
7. Connect source and sink.
8. Observe the traffic route when link is up and down.
9. View the simulated events and trace file and analyze it.
10. Start the schedule..

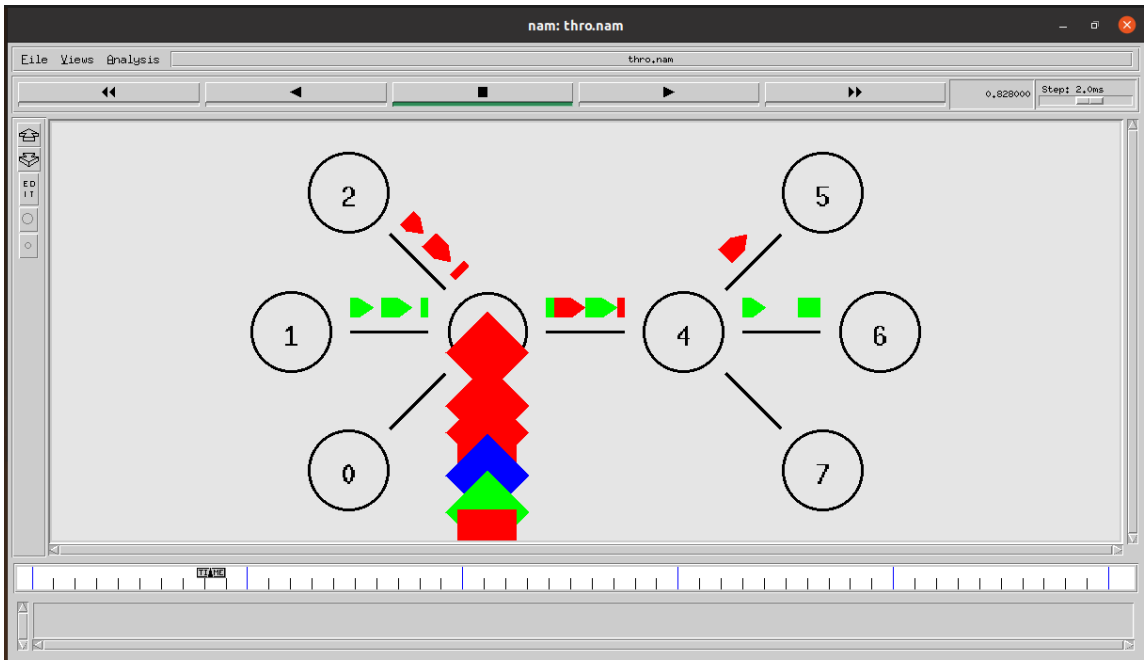
Program:

```
set ns [new Simulator]
set nr [open thro_dt.tr w]
$ns trace-all $nr
set nf [open thro.nam w]
$ns namtrace-all $nf
        proc finish { } {
        global ns nr nf
        $ns flush-trace
        close $nf
        close $nr
        exec nam thro.nam &
        exit 0            }
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
```

```
set n6 [$ns node]
set n7 [$ns node]

$ns duplex-link $n0 $n3 1Mb 10ms DropTail
$ns duplex-link $n1 $n3 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 1Mb 10ms DropTail
$ns duplex-link $n4 $n6 1Mb 10ms DropTail
$ns duplex-link $n4 $n7 1Mb 10ms DropTail
$ns duplex-link-op $n0 $n3 orient right-up
$ns duplex-link-op $n1 $n3 orient right
$ns duplex-link-op $n2 $n3 orient right-down
$ns duplex-link-op $n3 $n4 orient middle
$ns duplex-link-op $n4 $n5 orient right-up
$ns duplex-link-op $n4 $n7 orient right-down
$ns duplex-link-op $n6 $n4 orient left
set udp0 [new Agent/UDP]
$ns attach-agent $n2 $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set  interval_ 0.005
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n5 $null0
$ns connect $udp0 $null0
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set  interval_ 0.005
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n6 $null0
$ns connect $udp1 $null0
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
```

```
set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 500
$cbr2 set  interval_ 0.005
$cbr2 attach-agent $tcp0
set tcpsink0 [new Agent/TCPSink]
$ns attach-agent $n7 $tcpsink0
$ns connect $tcp0 $tcpsink0

$udp0 set fid_ 1
$udp1 set fid_ 2
$tcp0 set fid_ 3
$ns color 1 Red
$ns color 2 Green
$ns color 3 blue

$ns at 0.2 "$cbr0 start"
$ns at 3.5 "$cbr0 stop"
$ns at 0.3 "$cbr1 start"
$ns at 4.5 "$cbr1 stop"
$ns at 0.4 "$cbr2 start"
$ns at 4.5 "$cbr2 stop"
$ns at 5.0 "finish"
$ns run
```

Sample Input/Output:



Result: The TCP/UDP performance has been studied using Simulation tool.

Ex. No: 9	**Simulation of Distance Vector/Link State Routing algorithm**
07.03.2023	

Aim:

To write a code in java for simulation of Distance Vector and Link State Routing algorithm.

a) Distance Vector Routing

Algorithm:

1. Each router prepares its routing table.
2. By their local knowledge. each router knows about-
   i. All the routers present in the network
   ii. Distance to its neighbouring routers
3. Each router exchanges its distance vector with its neighbouring routers.
4. Each router prepares a new routing table using the distance vectors it has obtained from its neighbours. This step is repeated for (n-2) times if there are n routers in the network.
5. After this, routing tables converge / become stable.

Program:

```
import java.io.*;
public class DVR {
 static int graph[][];
```

```
 static int via[][];
 static int rt[][];
 static int v;
 static int e;
 public static void main(String args[]) throws IOException {
  BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
  System.out.println("Please enter the number of Vertices: ");
  v = Integer.parseInt(br.readLine());
  System.out.println("Please enter the number of Edges: ");
  e = Integer.parseInt(br.readLine());
  graph = new int[v][v];
  via = new int[v][v];
  rt = new int[v][v];
  for(int i = 0; i < v; i++)
   for(int j = 0; j < v; j++)  {
    if(i == j)
     graph[i][j] = 0;
    else
     graph[i][j] = 9999;
   }
  for(int i = 0; i < e; i++) {
   System.out.println("Please enter data for Edge " + (i + 1) + ":");
   System.out.print("Source: ");
   int s = Integer.parseInt(br.readLine());
   s--;
   System.out.print("Destination: ");
   int d = Integer.parseInt(br.readLine());
   d--;
   System.out.print("Cost: ");
   int c = Integer.parseInt(br.readLine());
   graph[s][d] = c;
   graph[d][s] = c;
  }
  dvr_calc_disp("The initial Routing Tables are: ");
  System.out.print("Please enter the Source Node for the edge whose cost
has changed: ");
  int s = Integer.parseInt(br.readLine());
  s--;
  System.out.print("Please enter the Destination Node for the edge whose
cost has changed: ");
  int d = Integer.parseInt(br.readLine());
```

```
 d--;
 System.out.print("Please enter the new cost: ");
 int c = Integer.parseInt(br.readLine());
 graph[s][d] = c;
 graph[d][s] = c;
 dvr_calc_disp("The new Routing Tables are: ");
}

static void dvr_calc_disp(String message) {
 System.out.println();
 init_tables();
 update_tables();
 System.out.println(message);
 print_tables();
 System.out.println();
}

static void update_table(int source) {
 for(int i = 0; i < v; i++) {
  if(graph[source][i] != 9999) {
   int dist = graph[source][i];
   for(int j = 0; j < v; j++) {
    int inter_dist = rt[i][j];
    if(via[i][j] == source)
     inter_dist = 9999;
    if(dist + inter_dist < rt[source][j])  {
     rt[source][j] = dist + inter_dist;
     via[source][j] = i;
    }
   }
  }
 }
}

static void update_tables() {
 int k = 0;
 for(int i = 0; i < 4*v; i++) {
  update_table(k);
  k++;
  if(k == v)
   k = 0;
```

```
  }
 }

 static void init_tables() {
  for(int i = 0; i < v; i++) {
   for(int j = 0; j < v; j++) {
    if(i == j)  {
     rt[i][j] = 0;
     via[i][j] = i;
    }
    else {
     rt[i][j] = 9999;
     via[i][j] = 100;
    }
   }
  }
 }

 static void print_tables(){
  for(int i = 0; i < v; i++) {
   for(int j = 0; j < v; j++)
    System.out.print("Dist: " + rt[i][j] + "    ");
   System.out.println();
  }
 }
}
```

Sample Input/Output:

```
snu-cse@snucse-HP-280-Pro-G5-MT-Business-PC:~$ gedit dvr.java
snu-cse@snucse-HP-280-Pro-G5-MT-Business-PC:~$ javac dvr.java
snu-cse@snucse-HP-280-Pro-G5-MT-Business-PC:~$ java dvr
Please enter the number of Vertices:
3
Please enter the number of Edges:
4
Please enter data for Edge 1:
Source: 1
Destination: 3
Cost: 2
Please enter data for Edge 2:
Source: 3
Destination: 2
Cost: 4
Please enter data for Edge 3:
Source: 3
Destination: 1
Cost: 4
Please enter data for Edge 4:
Source: 3
Destination: 2
Cost: 5

The initial Routing Tables are:
Dist: 0    Dist: 9    Dist: 4
Dist: 9    Dist: 0    Dist: 5
Dist: 4    Dist: 5    Dist: 0

Please enter the Source Node for the edge whose cost has changed: 2
Please enter the Destination Node for the edge whose cost has changed: 1
Please enter the new cost: 5

The new Routing Tables are:
Dist: 0    Dist: 5    Dist: 4
Dist: 5    Dist: 0    Dist: 5
Dist: 4    Dist: 5    Dist: 0

snu-cse@snucse-HP-280-Pro-G5-MT-Business-PC:~$
```

b) Link State Routing

Algorithm:

1. Discover its neighbours and build its neighbour table.
2. Measure the cost (delay, bandwidth, etc) to each of its neighbours.
3. Construct and send a routing update telling all it has learned to all routers in the network.
4. Apply the Dijkstra algorithm to construct the shortest path to all possible destinations.

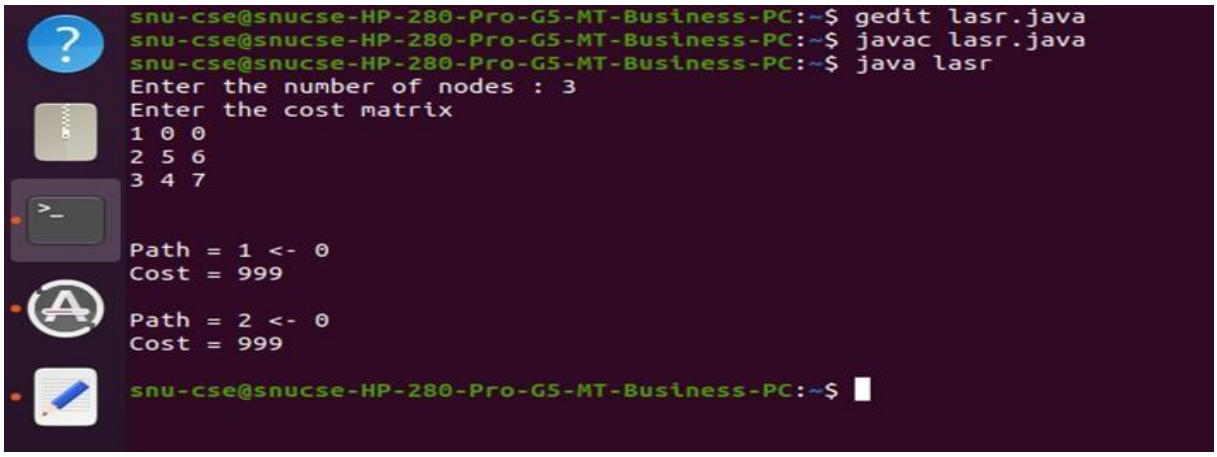
Program:

```
import java.util.*;
public class LSR{
     public static void main(String[] args){
          Scanner sc = new Scanner(System.in);
          System.out.print("Enter the number of nodes : ");
          int nodes = sc.nextInt();
        int[] preD = new int[nodes];
        int min = 999, nextNode = 0;
        int[] distance = new int[nodes];
        int[][] matrix = new int[nodes][nodes];
        int[] visited = new int[nodes];
        System.out.println("Enter the cost matrix");
```

```
for (int i = 0; i < distance.length; i++){
    visited[i] = 0;
    preD[i] = 0;
    for (int j = 0; j < distance.length; j++) {
        matrix[i][j] = sc.nextInt();
        if (matrix[i][j]==0)
            matrix[i][j] = 999;
    }
}
distance = matrix[0];
visited[0] = 1;
distance[0] = 0;
for (int counter = 0; counter < nodes; counter++) {
    min = 999;
    for (int i = 0; i < nodes; i++) {
        if (min > distance[i] && visited[i]!=1) {
            min = distance[i];
            nextNode = i;
        }
    }
    visited[nextNode] = 1;
    for (int i = 0; i < nodes; i++){
        if (visited[i]!=1){
            if (min+matrix[nextNode][i] < distance[i]){
                distance[i] = min+matrix[nextNode][i];
                preD[i] = nextNode;
            }
        }
    }
}
int j;
for (int i = 0; i < nodes; i++) {
    if (i!=0){
        System.out.print("Path = " + i);
        j = i;
        do{
            j = preD[j];
            System.out.print(" <- " + j);
        }
```

```
                while(j != 0);
                   System.out.println();
                   System.out.print("Cost = " + distance[i]);
            }
            System.out.println("\n");
        }
    }
}
```

Sample Input/Output:



Result: The code for simulation of Distance Vector and Link State Routing algorithm have been written and executed successfully in the Java environment.

Ex. No: 10	**Performance evaluation of Routing protocols using Simulation tool**
14.03.2023	

Aim:

To evaluate the performance of Routing protocols using NS2 Simulation tool with Ad-hoc On demand Distance Vector (AODV).

Algorithm:

1. The route discovery process involves ROUTE REQUEST (RREQ) and ROUTE REPLY (RREP) packets.
2. The source node initiates the route requested through the route discovery process using RREQ packets.
3. The generated route request is forwarded to the neighbours of the source node and this process is repeated till it reaches the destination.
4. On receiving a RREQ packet, an intermediate node with route to destination, it generates a RREP containing the number of hops required to reach the destination.
5. All intermediate nodes that participate in relaying this reply to the source node creates a forward route to destination.
6. AODV minimizes the number of packets involved in route discovery by establishing routes on-demand.
7. Prior to the establishment of communication between the source and receiver node, the routing protocol should be mentioned to find the route between them.
8. Data Transmission is established between nodes using UDP agent and CBR traffic.

Program:

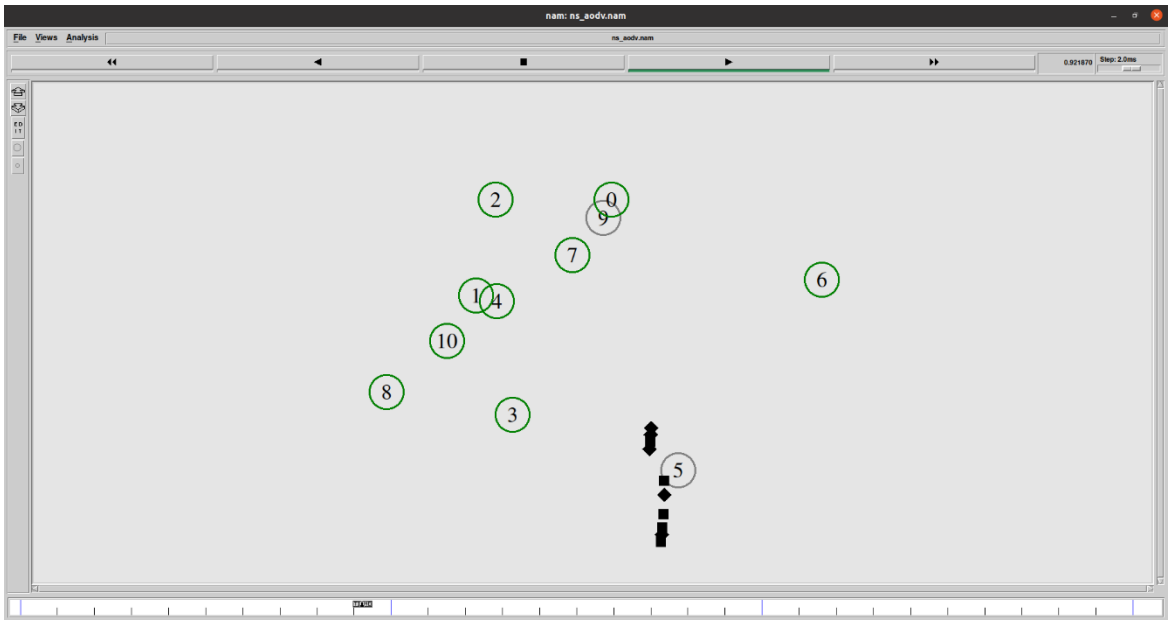
```
set val(chan) Channel/WirelessChannel;
set val(prop) Propagation/TwoRayGround;
set val(netif) Phy/WirelessPhy;
set val(mac) Mac/802_11;
set val(ifq) Queue/DropTail/PriQueue;
set val(ll) LL;
set val(ant) Antenna/OmniAntenna;
set val(ifqlen) 50;
```

```
set val(rp) AODV;
set val(nn) 11;
set val(x) 500;
set val(y) 400;
set val(stop) 3;
set val(energymodel) EnergyModel;
set val(initialenergy) 1000;

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

set nf [open ns_aodv.nam w]
$ns namtrace-all-wireless $nf $val(x) $val(y)

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

set chan_1_ [new $val(chan)]
$ns node-config -adhocRouting $val(rp) \
      -llType $val(ll) \
      -macType $val(mac) \
      -ifqType $val(ifq) \
      -ifqLen $val(ifqlen) \
      -antType $val(ant) \
      -propType $val(prop) \
      -phyType $val(netif) \
      -channel $chan_1_ \
      -topoInstance $topo \
      -agentTrace ON \
      -routerTrace ON \
      -macTrace OFF \
      -movementTrace ON \
      -energyModel $val(energymodel) \
      -initialEnergy $val(initialenergy) \
      -rxPower 0.4 \
      -txPower 1.0 \
      -idlePower 0.6 \
      -sleepPower 0.1 \
      -transitionPower 0.4 \
```

```
      -transitionTime 0.1

for {set i 0} {$i < $val(nn)} {incr i} {
      set node_($i) [$ns node]
      $node_($i) set X_ [ expr 10+round(rand()*480) ]
      $node_($i) set Y_ [ expr 10+round(rand()*380) ]
      $node_($i) set Z_ 0.0
}

for {set i 0} {$i < $val(nn)} {incr i} {
      $ns at [ expr 0.2+round(rand()) ] "$node_($i) setdest [ expr
10+round(rand()*480) ] [expr 10+round(rand()*380) ] [expr
60+round(rand()*30) ]"
}

set udp [new Agent/UDP]
$ns attach-agent $node_(5) $udp
set null [new Agent/Null]
$ns attach-agent $node_(2) $null

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packetSize_ 512
$cbr set interval_ 0.1
$cbr set rate_ 1mb
$cbr set maxpkts_ 10000
$ns connect $udp $null
$ns at 0.4 "$cbr start"

for {set i 0} {$i < $val(nn)} {incr i} {
        $ns initial_node_pos $node_($i) 30
}
for {set i 0} {$i < $val(nn)} {incr i} {
      $ns at $val(stop) "$node_($i) reset";
}

$ns at $val(stop) "finish"
$ns at 3.1  "puts \"end simulation\"; $ns halt"

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

```
        close $tf
        close $nf
        exec nam ns_aodv.nam &
        exit 0   }

puts "CBR packet size = [$cbr set packetSize_]"
puts "CBR interval = [$cbr set interval_]"

$ns run
```

Sample Input/Output:



Result: The performance of Routing protocols has been evaluated successfully using NS2 Simulation tool.