PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

(A CHRISITIAN MINORITY INSTITUTION)

JAISAKTHI EDUCATIONAL TRUST

ACCREDITED BY NATIONAL BOARD OF ACCREDITATION (NBA)

Bangalore Trunk Road, Varadharajapuram, Nasarathpettai,
Poonamallee, Chennai – 600 123

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

21CS1412 – NETWORKS LABORATORY

II CSE - IV SEMESTER

2022 - 2023

NAME :

REG NO :

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

(A CHRISITIAN MINORITY INSTITUTION)
JAISAKTHI EDUCATIONAL TRUST

ACCREDITED BY NATIONAL BOARD OF ACCREDITATION (NBA)

Bangalore Trunk Road, Varadharajapuram, Nasarathpettai,
Poonamallee, Chennai – 600 123

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is a Certified Bonafide Record Book of

Mr. /Ms. ____

Register Number ____

Submitted for End Semester Practical Examination held on __________________

in 21CS1412- NETWORKS LABORATORY during MAY/JUNE 2023.

Staff – In charge

Internal Examiner **External Examiner**

INDEX

EX.NO: 1A

DATE:

IMPLEMENTATION OF NETWORK COMMANDS

AIM

To study and implement network commands like tcpdump, netstat, ifconfig, nslookup and traceroute.

NETWORK COMMANDS

C:\tcpdump

tcpdump is a most powerful and widely used command-line packets sniffer or package analyzer tool which is used to capture or filter TCP/IP packets that received or transferred over a network on a specific interface.

C:\>netstat

Netstat displays a variety of statistics about a computers activeTCP/IP connections. This tool is most useful when you're having trouble with TCP/IP applications such as HTTP, and FTP.

C:\>ifconfig

The ifconfig command is used to get the information of active network-interfaces in a Unix-like operating system such as Linux, whereas ipconfig is used in the Windows OS.

C:\>nslookup

nslookup is used for diagnosing DNS problems. If you can access a resource by specifying an IP address but not it's DNS you have a DNS problem.

C:\>tracert

The tracert command displays a list of all the routers that a packet has to go through to get from the computer where tracert is run to any other computer on the internet.

RESULT
The network commands have been studied and implemented successfully.

EX.NO: 1B

DATE:

SIMULATION OF PING PROGRAM

AIM

To write a java program to simulate PING command.

ALGORITHM

1. Start the process
2. Using BufferedReader,get the IP address of the client to be pinged
3. Read the message from the buffer
4. Create a Runtime environment
5. Execute the ping command using the Process.
6. Using BufferedReader store the output of the ping command.
7. Read the message from the buffer
8. Print the number of packets sent and received
9. Stop the process

PROGRAM

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

class PingServer
{
public static void main(String args[])
{
try
{
String str;
System.out.println(" Enter the IP Address to be Ping : ");
BufferedReader buf1=new BufferedReader(new
InputStreamReader(System.in));
String  ip=buf1.readLine();
Runtime H=Runtime.getRuntime();
Process p=H.exec("ping " + ip);
InputStream in=p.getInputStream();
BufferedReader buf2=new BufferedReader(new InputStreamReader(in));
while((str=buf2.readLine())!=null)
{
System.out.println(" " + str);
}
}
catch(Exception  e)
{
System.out.println(e.getMessage());
}
}
}
```

OUTPUT:

```
D:\>javac PingServer.java

D:\>java PingServer

Enter the IP Address to be Ping: 132.147.163.44

Pinging 132.147.163.44 with 32 bytes of data:

Reply from 132.147.163.44: bytes=32 time<1ms TTL=128

Reply from 132.147.163.44: bytes=32 time<1ms TTL=128

Reply from 132.147.163.44: bytes=32 time<1ms TTL=128

Reply from 132.147.163.44: bytes=32 time<1ms TTL=128

Ping statistics for 132.147.163.44:

    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

RESULT

The java program to simulate PING command has been written, verified and executed successfully.

EX.NO: 1C

DATE:

SIMULATION OF TRACE ROUTE PROGRAM

AIM

To write a java program to simulate TRACEROUTE command.

ALGORITHM

1. Start the process
2. Using BufferedReader,get the address to be traced.
3. Read the message from the buffer
4. Create a Runtime environment
5. Execute the tracert command using the Process.
6. Using BufferedReader store the output of the tracert command.
7. Read the message from the buffer
8. Print the trout from the source to the destination.
9. Stop the process

PROGRAM

```
import java.io.*;
import java.net.*;
class TraceRoute
{
public static void main(String args[]) throws Exception
{
String str;
System.out.println(" Enter the address to be traced : ");
BufferedReader buf1=new BufferedReader(new
InputStreamReader(System.in));
String  ip=buf1.readLine();
Runtime H=Runtime.getRuntime();
Process p=H.exec("tracert " + ip);
InputStream in=p.getInputStream();
BufferedReader buf2=new BufferedReader(new InputStreamReader(in));
while((str=buf2.readLine())!=null)
{
  System.out.println(" " + str);
}
}
}
```

OUTPUT:

```
D:\>javac TraceRoute.java

D:\>java TraceRoute
Enter the address to be traced :
www.google.com
Tracing route to www.google.com [172.217.163.68]
 over a maximum of 30 hops:

   1     *        *        *     Request timed out.
   2    <1 ms    <1 ms    <1 ms 132-147-190-10.xinuos.com [132.147.190.10]
   3    <1 ms    <1 ms    <1 ms  115.111.10.1.static-chennai.vsnl.net.in
[115.11
1.10.1]
   4     *        *        *     Request timed out.
   5     2 ms     2 ms     2 ms  115.114.71.134.static-chennai.vsnl.net.in [115.
114.71.134]
   6     2 ms     2 ms     2 ms  172.31.167.46
   7     2 ms     2 ms     2 ms  121.240.1.46
   8     3 ms     3 ms     3 ms  74.125.242.129
   9     2 ms     3 ms     2 ms  216.239.42.243
  10     2 ms     2 ms     2 ms maa05s02-in-f4.1e100.net [172.217.163.68]
Trace complete.
```

RESULT

The java program to simulate TRACEROUTE command has been written, verified and executed successfully.

EX.NO:2

DATE:

HTTP FOR WEB PAGE UPLOAD AND DOWNLOAD

AIM

To implement HTTP for web page upload and download using java.

ALGORITHM

Client
1. Start the program, declare the variables
2. Create a socket using the socket structure socket
3. Set the socket family, IP address and the port using the server address
4. Establish the connection to the server
5. Read the image to be downloaded and send it to the server as a request
6. Receive the message from the server for a search result
7. Display the file which is downloaded
8. Terminate the connection and compile and execute

Server
1. Start the program, declare the variables
2. Create a socket using the socket structure socket
3. Set the socket family, IP address and the port using the server address
4. Bind and listen the socket structure
5. Accept the client connection using the socket descriptor and the server address
6. Open a file and make a displays the file size.

8. Send the corresponding result to the client, Terminate the connection and compile and execute

PROGRAM:

SERVER:

```
import java.net.*;
import java.io.*;
import java.awt.image.*;
import javax.imageio.*;
import  javax.swing.*;
class Server {
public static void main(String args[]) throws Exception{
ServerSocket server=null;
Socket socket;
server=new ServerSocket(4000);
System.out.println("Server Waiting for image");
socket=server.accept();
System.out.println("Client connected.");
InputStream in = socket.getInputStream();
DataInputStream dis = new DataInputStream(in);
int len = dis.readInt();
System.out.println("Image Size: " + len/1024 + "KB");
byte[] data = new byte[len];
dis.readFully(data);
dis.close();
in.close();
InputStream ian = new ByteArrayInputStream(data);
BufferedImage bImage = ImageIO.read(ian);
JFrame f = new JFrame("Server");
ImageIcon icon = new ImageIcon(bImage);
JLabel l = new JLabel();
l.setIcon(icon);
f.add(l);
f.pack();
f.setVisible(true);
}
}
```

CLIENT:

```
import javax.swing.*;
import  java.net.*;
import java.awt.image.*;
import javax.imageio.*;
import java.io.*;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
public class Client{
public static void main(String args[]) throws Exception{
Socket soc;
BufferedImage img = null;
soc=new Socket("localhost",4000);
System.out.println("Client is running. ");
try {
System.out.println("Reading image from disk. ");
img = ImageIO.read(new File("tiger.jpeg"));
ByteArrayOutputStream baos = new ByteArrayOutputStream();
ImageIO.write(img, "jpg", baos);
baos.flush();
byte[] bytes = baos.toByteArray();
baos.close();
System.out.println("Sending image to server. ");
OutputStream out = soc.getOutputStream();
DataOutputStream dos = new DataOutputStream(out);
dos.writeInt(bytes.length);
dos.write(bytes, 0, bytes.length);
System.out.println("Image sent to server. ");
dos.close();
out.close();
}catch (Exception e) {
System.out.println("Exception: " + e.getMessage());
soc.close();
}
soc.close();
}
}
```

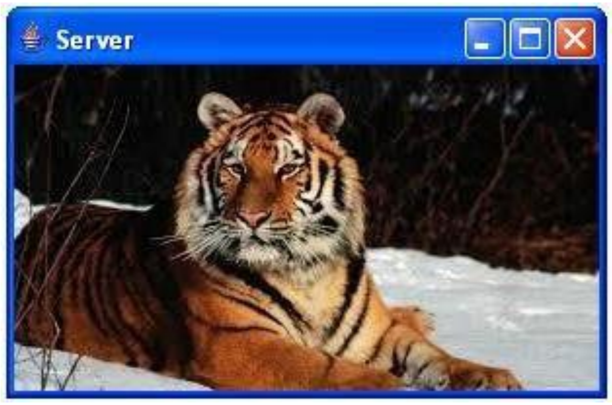
OUTPUT

CLIENT:

```
Z:\http>javac Client.java
Z:\http>java Client
Client is running.
Reading image from disk.
Sending image to server.
Image sent to server.
```

SERVER:

```
Z:\http>javac Server.java
Z:\http>java Server
Server Waiting for image

Client connected.
Image Size: 11KB
```



RESULT:

Thus , the HTTP for web page upload and download was executed and output is verified successfully.

EX.NO: 3A

DATE:

TCP ECHO CLIENT AND SERVER

AIM

To implement echo client and server in java using TCP sockets.

ALGORITHM

Server

1. Create a server socket.
2. Wait for client to be connected.
3. Read text from the client
4. Echo the text back to the client.
5. Repeat steps 4-5 until ‘bye’ or ‘null’ is read.
6. Close the I/O streams
7. Close the server socket
8. Stop

Client

1. Create a socket and establish connection with the server
2. Get input from user.
3. If equal to bye or null, then go to step 7.
4. Send text to the server.
5. Display the text echoed by the server
6. Repeat steps 2-4
7. Close the I/O streams
8. Close the client socket
9. Stop

PROGRAM:

ECHOSERVER:

```
import java.io.*;
import java.net.*;
public class EchoServer
{
public static void main(String args[]) throws Exception
{
try
{
int Port;
BufferedReader Buf =new BufferedReader(new
InputStreamReader(System.in));
System.out.print(" Enter the Port Address : " );
Port=Integer.parseInt(Buf.readLine());
ServerSocket sok =new ServerSocket(Port);
System.out.println(" Server is Ready To Receive a Message. ");
System.out.println(" Waiting......... ");
Socket so=sok.accept();
if(so.isConnected()==true)
        System.out.println(" Client Socket is Connected Successfully. ");
InputStream in=so.getInputStream();
OutputStream ou=so.getOutputStream();
PrintWriter pr=new PrintWriter(ou);
BufferedReader buf=new BufferedReader(new
InputStreamReader(in));
String str=buf.readLine();
System.out.println(" Message Received From Client : " + str);
System.out.println(" This Message is Forwarded To Client. ");
pr.println(str);
pr.flush();
}
```

```
 catch(Exception  e)
  {
  System.out.println(" Error : " + e.getMessage());
  }
}
}
```

ECHOCLIENT:

```
import java.io.*;
import java.net.*;
public class EchoClient
{
public static void main(String args[]) throws Exception
{
try {
int Port;
BufferedReader Buf =new BufferedReader(new
InputStreamReader(System.in));
System.out.print(" Enter the Port Address : " );
Port=Integer.parseInt(Buf.readLine());
Socket sok=new Socket("localhost",Port);
if(sok.isConnected()==true)
        System.out.println(" Server Socket is Connected Successfully. ");
InputStream in=sok.getInputStream();
OutputStream ou=sok.getOutputStream();
PrintWriter pr=new PrintWriter(ou);
BufferedReader buf1=new BufferedReader(new
InputStreamReader(System.in));
BufferedReader buf2=new BufferedReader(new
InputStreamReader(in));
String str1,str2;
System.out.print(" Enter the Message : ");
str1=buf1.readLine();
pr.println(str1);
pr.flush();
System.out.println(" Message Send Successfully. ");
str2=buf2.readLine();
System.out.println(" Message From Server : " + str2);
   }
```

```
  catch(Exception  e)
  {
   System.out.println(" Error : " + e.getMessage());
  }
}
}
```

OUTPUT:

ECHOSERVER:

```
Z:\>javac EchoServer.java
Z:\>java     EchoServer
 Enter the Port Address : 5432
 Server is Ready To Receive a Message.
 Waiting .....
 Client Socket is Connected Successfully.
 Message Received From Client : good morning
 This Message is Forwarded To Client.
```

ECHOCLIENT:

```
Z:\>javac EchoClient.java
Z:\>java     EchoClient
 Enter the Port Address : 5432
 Server Socket is Connected Successfully.
 Enter the Message : good morning
 Message Send Successfully.
 Message From Server : good morning
```

RESULT:

Thus , the echo client and server using TCP socket was executed and output is verified successfully.

EX.NO:3B

DATE:

TCP CHAT CLIENT AND SERVER

AIM

To implement chat client and server in java using TCP sockets.

ALGORITHM

Server

1. Create a serversocket
2. Wait for client to beconnected.
3. Read Client's message and displayit
4. Get a message from user and send it to client
5. Repeatsteps3-4untiltheclientsends"end"
6. Close all streams
7. Close the server and client socket
8. Stop

Client

1. Create a client socket and establishconnection with the server
2. Get a message from user and send it to server
3. Read server's response and displayit
4. Repeat steps 2-3 until chat isterminated with "end" message
5. Close all input/output streams
6. Close the clientsocket
7. Stop

PROGRAM:

CHATSERVER:

```
import java.io.*;
import java.net.*;
public class ChatServer
{
  public static void main(String args[]) throws Exception
  {
         ServerSocket  sersock=new  ServerSocket(9999);
         System.out.println("Server ready for chatting");
         Socket sock= sersock.accept();
         BufferedReader  sendRead=  new  BufferedReader(new
         InputStreamReader(System.in));
         OutputStream   ofstream=sock.getOutputStream();
         PrintWriter pwrite=new PrintWriter(ofstream,true);
         InputStream istream=sock.getInputStream();
         BufferedReader receivedread=new BufferedReader(new
         InputStreamReader(istream));
         String receivemessage,sendmessage;
         while(true)
         {
           if((receivemessage=receivedread.readLine())!=null)
           {
              System.out.println(receivemessage);
           }
           sendmessage=sendRead.readLine();
           pwrite.println(sendmessage);
           System.out.flush();
         }
    }
}
```

CHATCLIENT:

```
import  java.io.*;
import java.net.*;
public  class  ChatClient
{
  public static void main(String args[]) throws Exception
  {
         Socket sock=new Socket("localhost",9999);
         BufferedReader  sendRead=  new  BufferedReader(new
         InputStreamReader(System.in));
         OutputStream   ofstream=sock.getOutputStream();
         PrintWriter pwrite=new PrintWriter(ofstream,true);
         InputStream istream=sock.getInputStream();
         BufferedReader receivedread=new BufferedReader(new
         InputStreamReader(istream));
         System.out.println("client ready for chatting");
         String receivemessage,sendmessage;
         while(true)
         {
           sendmessage=sendRead.readLine();
           pwrite.println(sendmessage);
           System.out.flush();
            if((receivemessage=receivedread.readLine())!=null)
           {
              System.out.println(receivemessage);
           }
         }
    }
}
```

OUTPUT:

CHATSERVER:

```
Z:\>javac ChatServer.java
Z:\>java ChatServer
Server ready for chatting
welcome
Hello
```

CHATCLIENT:

```
Z:\>javac ChatClient.java
Z:\>java ChatClient
client ready for chatting
welcome
Hello
```

RESULT:

Thus , the chat client and server using TCP socket was executed and output is verified successfully.

EX.NO: 3C

DATE:

FILE TRANSFER USING TCP

AIM

To implement file transfer in java using TCP sockets.

ALGORITHM

Server

1. Start the program.
2. Declare the variables and structure for the socket.
3. Create a socket using socket functions
4. The socket is binded at the specified port.
5. Using the object the port and address are declared.
6. After the binding is executed the file is specified.
7. Then the file is specified.
8. Execute the client program.

Client

1. Start the program.
2. Declare the variables and structure.
3. Socket is created and connects function is executed.
4. If the connection is successful then server sends the message.
5. The file name that is to be transferred is specified in the client side.
6. The contents of the file is verified from the server side.

PROGRAM:

FILECLIENT:

```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientfile
{
public static void main(String args[])
{
try
{
BufferedReader =new BufferedReader(new InputStreamReader(System.in));
Socket clsct=new Socket("127.0.0.1",139);
DataInputStream din=new DataInputStream(clsct.getInputStream());
DataOutputStream dout=new DataOutputStream(clsct.getOutputStream());
System.out.println("Enter the file name:");
String str=in.readLine();
dout.writeBytes(str+'\n');
System.out.println("Enter the new file name:");
String str2=in.readLine();
String str1,ss;
FileWriter f=new FileWriter(str2);
char buffer[];
while(true)
{
str1=din.readLine();
if(str1.equals("-1"))
break;
System.out.println(str1);
buffer=new char[str1.length()];
str1.getChars(0,str1.length(),buffer,0);
f.write(buffer);
}
f.close();
clsct.close();
}
catch (Exception e)
{
System.out.println(e);
 }
 }
 }
```

FILESERVER:

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverfile
{
public static void main(String args[])
{
try
{
 ServerSocket obj=new ServerSocket(139);
while(true) {
Socket obj1=obj.accept();
DataInputStream din=new DataInputStream(obj1.getInputStream());
DataOutputStream dout=new DataOutputStream(obj1.getOutputStream());
String str=din.readLine();
FileReader f=new FileReader(str);
BufferedReader b=new BufferedReader(f);
String s;
while((s=b.readLine())!=null)
{
System.out.println(s);
dout.writeBytes(s+'\n');
}
f.close();
dout.writeBytes("-1\n");         } }
catch(Exception e) { System.out.println(e);
}
 }
 }
```

OUTPUT:

NET.TXT

```
computer lab
dept of CSE
panimalar
```

FILESERVER:

```
Z:\>javac Serverfile.java
Note: Serverfile.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Z:\>java Serverfile
computer  lab
dept of CSE
panimalar
```

FILECLIENT:

```
Z:\>javac Clientfile.java
Note: Clientfile.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Z:\>java Clientfile
Enter the file name:
net.txt

Enter the new file name:
sample.txt
computer lab
dept of CSE
panimalar
```

RESULT:

Thus , the echo client and server using TCP socket was executed and output is verified successfully.

EX.NO: 4

DATE:

SIMULATION OF DNS USING UDP SOCKETS

AIM

To implement the simulation of DNS in java using UDP sockets.

ALGORITHM

Server

1. Define a array of hosts and its corresponding IP address in another array
2. Create a datagram socket.
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. Construct a datagram packet to send response back to the client
8. Repeat steps 3-7 to resolve further requests from clients
9. Close the server socket
10. Stop

Client

1. Create a datagram socket
2. Get domain name from user
3. Construct a datagram packet to send domain name to the server
4. Create a datagram packet to receive server message
5. If it contains IP address then display it, else display "Domain does not exist"
6. Close the client socket
7. Stop

PROGRAM:

UDPSERVER:

```
import java.io.*;
import java.net.*;
class UDPServer
{
public static DatagramSocket ds;
public static byte buffer[]=new byte[1024];
public static int clientport=789,serverport=790;
public static void main(String args[])throws Exception
{
ds=new DatagramSocket(clientport);
System.out.println("press ctrl+c to quit the program");
BufferedReader dis=new BufferedReader(new Input Stream Reader
                                                  (System.in));
InetAddress ia=InetAddress.getLocalHost();
while(true)
{
DatagramPacket p=new DatagramPacket(buffer,buffer.length);
ds.receive(p);
String psx=new String(p.getData(),0,p.getLength());
System.out.println("Client:" + psx);
InetAddress ib=InetAddress.getByName(psx);
System.out.println("Server output:"+ib);
String str=dis.readLine();
if(str.equals("end"))
break;
buffer=str.getBytes();
ds.send(new DatagramPacket(buffer,str.length(),ia,serverport));
}
}
}
```

UDPCLIENT:

```
import java .io.*;
import java.net.*;
class  UDPClient
{
public static DatagramSocket ds;
public static int clientport=789,serverport=790;
public static void main(String args[])throws Exception
{
byte buffer[]=new  byte[1024];
ds=new DatagramSocket(serverport);
BufferedReader dis=new BufferedReader(new Input Stream Reader
                                                        (System.in));
System.out.println("server waiting");
InetAddress ia=InetAddress.getLocalHost();
while(true)
{
System.out.println("Client:");
String str=dis.readLine();
if(str.equals("end"))
break;
buffer=str.getBytes();
ds.send(new     DatagramPacket(buffer,str.length(),ia,clientport));
DatagramPacket p=new DatagramPacket(buffer,buffer.length);
ds.receive(p);
String psx=new String(p.getData(),0,p.getLength());
System.out.println("Server:" + psx);
}
}
}
```

OUTPUT:

UDPserver

```
Z:\javac UDPserver.java
Z:\java UDPserver
Press ctrl+c to quit the program
Client:www.yahoo.com
Server output:www.yahoo.com/106.10.170.115
```

UDPclient

```
Z:\javac UDPclient.java
Z:\java UDPclient
Server waiting
Client:www.yahoo.com
```

RESULT:

Thus , the simulation of DNS using UDP socket was executed and output is verified successfully.

EX.NO:5A

DATE:

SIMULATION OF ADDRESS RESOLUTION PROTOCOL (ARP)

AIM

To implement the simulation of address resolution protocol (ARP) using java.

ALGORITHM

Server

1. Create a server socket.
2. Accept client connection.
3. Read IP address from the client request
4. Check its configuration file and compare with its logical address.
4. If there is a match, send the host physical address.
5. Stop

Client

1. Create a socket.
2. Send IP address to the target machine
3. Receive target's response
4. If it is a MAC address then display it and go to step 6
5. Display "Host not found"
6. Stop

PROGRAM:

ARPCLIENT:

```
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",139);
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);
}
}
}
```

ARPSERVER:

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverarp
{
public static void main(String args[])
{
try
{
ServerSocket obj=new ServerSocket(139);
Socket obj1=obj.accept();
while(true)
{
DataInputStream din=new DataInputStream(obj1.getInputStream());
DataOutputStream dout=new DataOutputStream(obj1.getOutputStream());
String str=din.readLine();
System.out.println("Processing....")
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();
}
}
catch(Exception  e)  {
}
System.out.println("Physical  Address  Retrieval  Processing");
}
}
```

OUTPUT:

ARPSERVER:

```
Z:\>java Serverarp
Physical Address Retrieval Processing
```

ARPCLIENT:

```
Z:\>javac Clientarp.java
Note: Clientarp.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

Z:\>java Clientarp
Enter the Logical address(IP):
165.165.80.80
The Physical Address is: 6A:08:AA:C2
```

RESULT:

Thus , the simulation of address resolution protocol (ARP) was executed and output is verified successfully.

EX.NO: 5B

DATE:

SIMULATION OF REVERSE ADDRESS RESOLUTION PROTOCOL (RARP)

AIM

To implement the simulation of reverse address resolution protocol (RARP) using java.

ALGORITHM

Server

1. Create a server socket.
2. Accept client connection.
3. Read MAC address from the client request
4. Check its configuration file and compare with its physical address.
5. If there is a match, send the host logical address.
6. Stop

Client

1. Create a socket.
2. Send physical address to the target machine
3. Receive target's response
4. If it is a IP address then display it and go to step 6
5. Display "Host not found"
6. Stop

PROGRAM:

RARPCLIENT:

```
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);
}
}
}
```

RARPSERVER:

```
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("Logical Address Retrieval Processing");
}
}
```

OUTPUT:

RARPSERVER:

```
Z:\>java Serverrarp
Logical Address Retrieval Processing
```

RARPCLIENT:

```
Z:\>javac Clientarp.java
Note: Clientrarp.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

Z:\>java Clientrarp
Enter the Physical address (MAC): 6A:08:AA:C2

The Logical address is(IP):
165.165.80.80
```

RESULT:

Thus , the simulation of reverse address resolution protocol (RARP) was executed and output is verified successfully.

EX.NO:6

DATE:

STUDY OF NETWORK SIMULATOR (NS)

NS2 SIMULATOR

A simulator is a device, software or system which behaves or operates like a given system when provided with a set of controlled inputs. The need for simulators is:

- Provide users with practical feedback such as accuracy, efficiency, cost, etc., when designing real world systems.
- Permit system designers to study at several different levels of abstraction
- Simulation can give results that are not experimentally measurable with our current level of technology.
- Simulations take the building/rebuilding phase out of the loop by using the model already created in the design phase.
- Effective means for teaching or demonstrating concepts to students.

A few popular network simulators are NS-2, OPNET, GLOMOSIM, etc. Network Simulator NS2

NS2 is an object-oriented, discrete event driven network simulator developed at UC Berkley written in C++ and OTcl. NS is primarily useful for simulating local and wide area networks. NS2 is an open-source simulation tool that primarily runs on Linux (cygwin for Windows). The features of NS2 are

- Is a discrete event simulator for networking research
- Works at packet level.
- Provide support to simulate bunch of protocols like TCP, UDP, FTP, etc.
- Simulate wired and wireless network.
- Is a standard experiment environment in research community.

How to start:

First of all, you need to create a simulator object. This is done with the command

Now we open a file for writing that is going to be used for the nam trace data.

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

The first line opens the file 'out.nam' for writing and gives it the file handle 'nf'. In thesecond line we tell the simulator object that we created above to write all simulation data that is going to be relevant for nam into this file.The next step is to add a 'finish' procedure that closes the trace file and starts nam.

```
proc finish {} {
global ns nf
$ns flush-trace
close $nf
exec nam out.nam &
exit 0}
```

The next line tells the simulator object to execute the 'finish' procedure after 5.0 seconds of simulation time.

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

ns provides you with a very simple way to schedule events with the 'at' command. The last line finally starts the simulation.

```
$ns run
```

We can actually save the file now and try to run it with 'ns example1.tcl'. We are going to get an error message like 'nam: empty trace file out.nam' though, because until now we haven't defined any objects (nodes, links, etc.) or events.

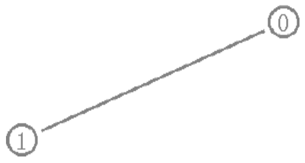
```
#Create a simulator object

set ns [new Simulator]

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

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

```
# Insert your own code for topology creation
# and agent definitions, etc. here

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

#Run the simulation
$ns run
set n0 [$ns node]
set n1 [$ns node]
```

A new node object is created with the command '$ns node'. The above code creates two nodes and assigns them to the handles 'n0' and 'n1'. The next line connects the two nodes.

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

This line tells the simulator object to connect the nodes n0 and n1 with a duplex link with the bandwidth 1Megabit, a delay of 10ms and a DropTail queue. Now you can save your file and start the script with 'ns example1.tcl'. nam will be started automatically and you should see an output that resembles the picture below.



Commands to create a link between nodes:

$ns_ simplex-link <node1> <node2> <bw> <delay> <qtype> <args>

This command creates an unidirectional link between node1 and node2 with specified bandwidth (BW) and delay characteristics. The link uses a queue type of <qtype> and depending on the queue type different arguments are passed through <args>.

$ns_ duplex-link <node1> <node2> <bw> <delay> <qtype> <args>

This creates a bi-directional link between node1 and node2. This procedure essentially creates a duplex-link from two simplex links, one from node1 to node2 and the other from node2 to node1. The syntax for duplex- link is same as that of simplex-link described above.

$ns_ simplex-link-op <n1> <n2> <op> <args>

This is used to set attributes for a simplex link. The attributes may be the orientation, color, label, or queue-position.

$ns_ duplex-link-op <n1> <n2> <op> <args>

This command is used to set link attributes (like orientation of thelinks, color, label, or queue-position) for duplex links.

Sending data:

The next step is to send some data from node n0 to node n1. In ns, data is always being sent from one 'agent' to another. So the next step is to create an agent object that sends data from node n0, and another agent object that receives the data on node n1.

```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

These lines create a UDP agent and attach it to the node n0, then attach a CBR traffic generator to the UDP agent. CBR stands for 'constantbit rate'. Line 7 and 8 should be self-explaining. The packet Size is being setto 500 bytes and a packet will be sent every 0.005 seconds (i.e. 200 packets per second).
The next lines create a Null agent which acts as traffic sink and attach it to node n1.

```
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
```

Now the two agents have to be connected with each other.

```
$ns connect $udp0 $null0
```

And now we have to tell the CBR agent when to send data and when to stop sending. Note: It's probably best to put the following lines just before the line '$ns at 5.0 "finish"'.

```
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

This code should be self-explaining again. Now you can save the file and start the simulation again. When you click on the 'play' button in the nam window, you will see that after 0.5 simulation seconds,node 0 startssending data packets to node 1. You might want to slow nam down thenwith the 'Step' slider.

Now you start some experiments with nam and the Tcl script. You can click on any packet in the nam window to monitor it, and you can also click directly on the link to get some graphs with statistics.Try to change the 'packetsize_' and 'interval_' parameters in the Tcl script to see what happens.

Agents

Agents represent endpoints where network-layer packets are constructed or consumed, and are used in the implementation of protocols at various layers.

ns_ attach-agent <node> <agent>

This command attaches the <agent> to the <node>. We assume here that the <agent> has already been created. An agent is typically created by set agent [new Agent/AgentType] where Agent/AgentType defines the class definiton of the specified agent type.

Topology

You will always have to create a simulator object, you will always have to start the simulation with the same command, and if you want to run nam automatically, you will always have to open a trace file, initialize it, and define a procedure which closes it and starts nam.

Now insert the following lines into the code to create four nodes.

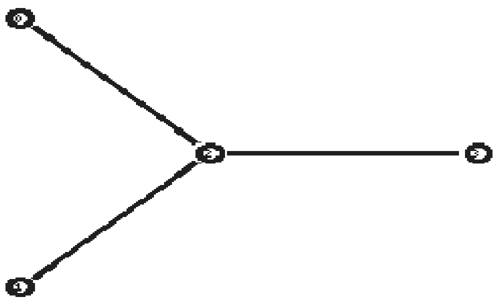
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

The following piece of Tcl code creates three duplex links between the nodes.

$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms DropTail

You can save and start the script now. You might notice that the topology looks a bit awkward in nam. You can hit the 're-layout' button to make it look better, but it would be nice to have some more control over the layout. Add the next three lines to your Tcl script and start it again.

$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



```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
```

After successful installation, path setting and validation, execute NS2 programs.

```
  $ ns filename.tcl
```

EX.NO:7

DATE:

SIMULATION OF CONGESTION CONTROL ALGORITHM (RED)

AIM

To implement congestion control algorithm(RED)using Network Simulator.

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.

PROGRAM:

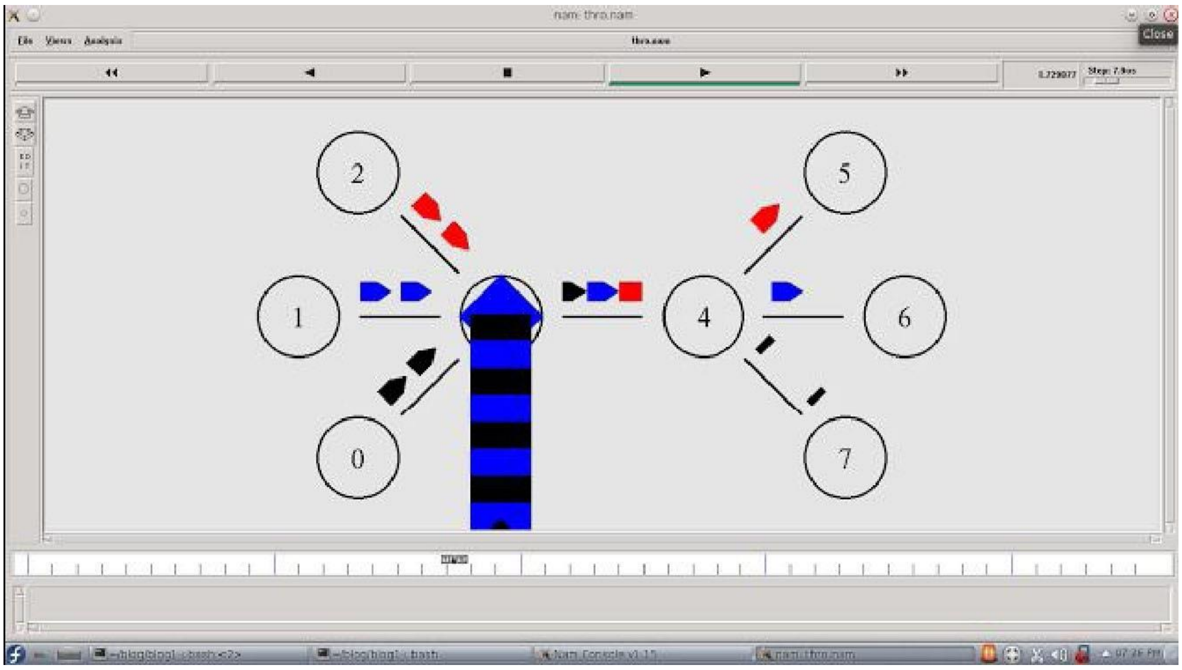
RED.TCL

```
set ns [new Simulator]
set nr [open thro_red.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 RED
$ns duplex-link $n1 $n3 1Mb 10ms RED
$ns duplex-link $n2 $n3 1Mb 10ms RED
$ns duplex-link $n3 $n4 1Mb 10ms RED
$ns duplex-link $n4 $n5 1Mb 10ms RED
$ns duplex-link $n4 $n6 1Mb 10ms RED
$ns duplex-link $n4 $n7 1Mb 10ms RED

$ns duplex-link-op $n0 $n3 orient right-up
$ns duplex-link-op $n3 $n4 orient middle
$ns duplex-link-op $n2 $n3 orient right-down
$ns duplex-link-op $n4 $n5 orient right-up
$ns duplex-link-op $n4 $n7 orient right-down
$ns duplex-link-op $n1 $n3 orient right
$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 udp2 [new Agent/UDP]
$ns attach-agent $n0 $udp2
set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 500
$cbr2 set  interval_ 0.005
$cbr2 attach-agent $udp2
set null0 [new Agent/Null]
$ns attach-agent $n7 $null0
$ns connect $udp2 $null0

$udp0 set fid_ 1
$udp1 set fid_ 2
$udp2 set fid_ 3

$ns color 1 Red
$ns color 2 Green
$ns color 2 blue

$ns at 0.1 "$cbr0 start"
$ns at 0.2 "$cbr1 start"
```

```
$ns at 0.5 "$cbr2 start"
$ns at 4.0 "$cbr2 stop"
$ns at 4.2 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

OUTPUT:



RESULT:

Thus , the congestion control algorithm(RED)using Network Simulator was executed and output is verified successfully.

EX.NO: 8

DATE:

TCP/UDP PERFORMANCE USING NS2

AIM

To implement the study of TCP/UDP performance using Network Simulator.

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 analyze it
10. Start the scheduler

PROGRAM:

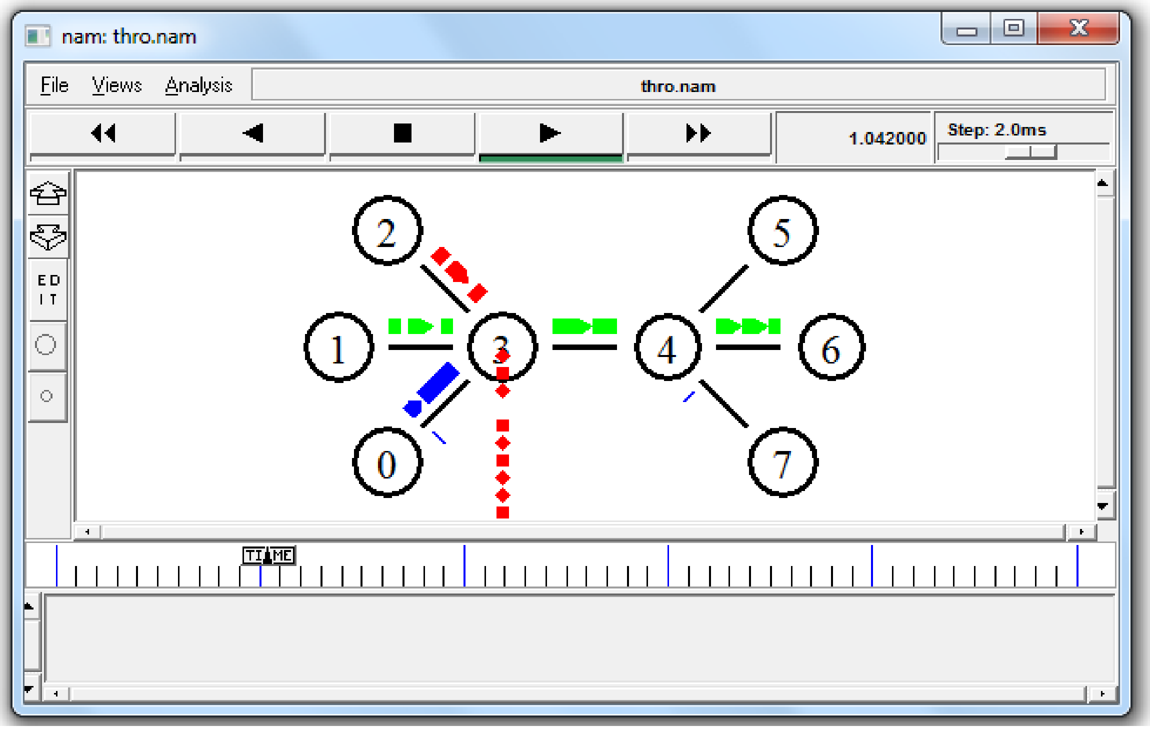
UDPTCP.TCL

```
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
```

OUTPUT:



RESULT:

Thus , the performance of TCP/UDP using Network Simulator was studied and output is verified successfully.

EX.NO : 9

DATE:

SIMULATION OF DISTANCE VECTOR ROUTING

AIM:

To implement the simulation of distance vector routing using Network Simulator.

ALGORITHM:

1. Create a simulator object
2. Set routing protocol to Distance Vector routing
3. Trace packets on all links onto NAM trace and text trace file
4. Define finish procedure to close files, flush tracing and run NAM
5. Create eight nodes
6. Specify the link characteristics between nodes
7. Describe their layout topology as a octagon
8. Add UDP agent for node n1
9. Create CBR traffic on top of UDP and set traffic parameters.
10. Add a sink agent to node n4
11. Connect source and the sink
12. Schedule events as follows:
    a. Start traffic flow at 0.5
    b. Down the link n3-n4 at 1.0
    c. Up the link n3-n4 at 2.0
    d. Stop traffic at 3.0
    e. Call finish procedure at 5.0
13. Start the scheduler
14. Observe the traffic route when link is up and down
15. View the simulated events and trace file analyze it
16. Stop

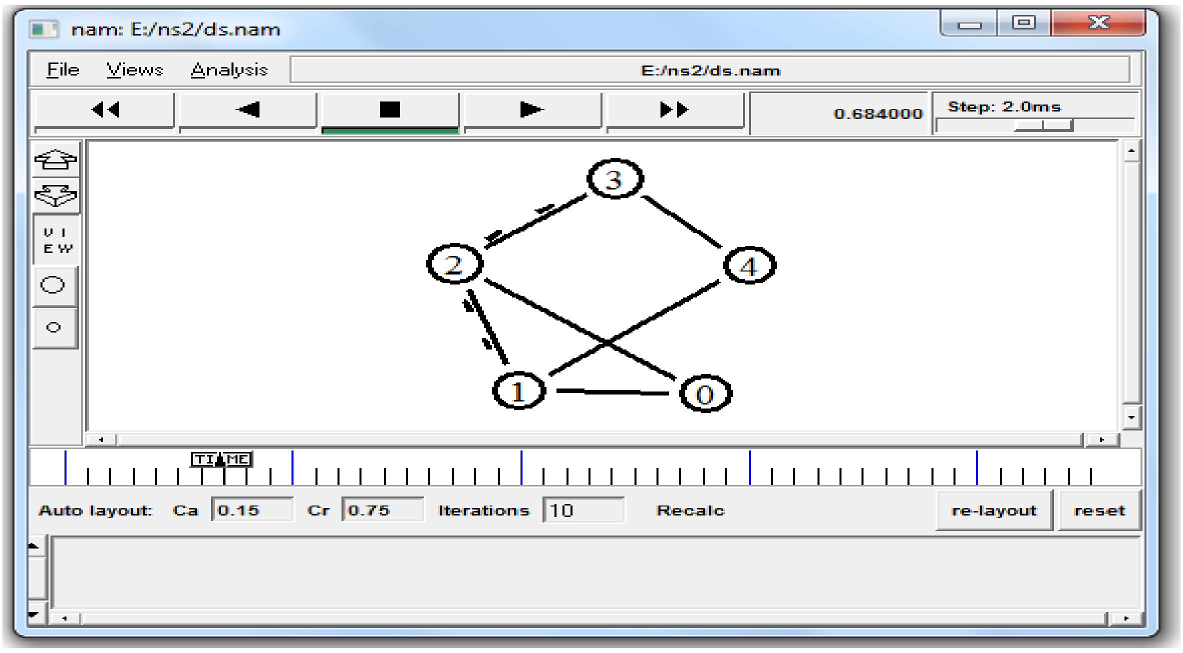
PROGRAM:

DVR.TCL

```
set ns1 [new Simulator]
set nr1 [open ds.tr w]
$ns1 trace-all $nr1
set nf1 [open ds.nam w]
$ns1 namtrace-all $nf1

set n0 [$ns1 node]
set n1 [$ns1 node]
set n2 [$ns1 node]
set n3 [$ns1 node]
set n4 [$ns1 node]

$ns1 duplex-link $n0 $n1  1Mb 10ms DropTail
$ns1 duplex-link $n0 $n2  1Mb 10ms DropTail
$ns1 duplex-link $n1 $n2  1Mb 10ms DropTail
$ns1 duplex-link $n1 $n4  1Mb 10ms DropTail
$ns1 duplex-link $n2 $n3  1Mb 10ms DropTail
$ns1 duplex-link $n3 $n4  1Mb 10ms DropTail

$ns1 duplex-link-op $n0 $n1 orient left-up
$ns1 duplex-link-op $n0 $n2 orient left-down
$ns1 duplex-link-op $n4 $n1 orient right-up
$ns1 duplex-link-op $n3 $n2 orient right-down

$ns1 cost $n0 $n1 1
$ns1 cost $n0 $n2 5

$ns1 cost $n1 $n0 1
$ns1 cost $n1 $n4 9
$ns1 cost $n1 $n2 3

$ns1 cost $n2 $n0 5
$ns1 cost $n2 $n1 3
$ns1 cost $n2 $n3 4

$ns1 cost $n3 $n2 4
$ns1 cost $n3 $n4 2
```

```
$ns1 cost $n4 $n3 2
$ns1 cost $n4 $n1 9
set udp0 [new Agent/UDP]
$ns1 attach-agent $n1 $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetsize_ 1000
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set Null0 [new Agent/Null]
$ns1 attach-agent $n4 $Null0
$ns1 connect $udp0 $Null0
$ns1 rtproto DV
$ns1 at 0.5 "$cbr0 start"
$ns1 at 4.5 "$cbr0 stop"
proc finish { } {
        global ns1 nr1 nf1
        $ns1 flush-trace
        close $nf1
        close $nr1
        exec nam ds.nam &
        exit 0   }
$ns1 at 5.0 "finish"
$ns1 run
```

OUTPUT:



RESULT:

Thus , the simulation of distance vector routing using Network Simulator was executed and output is verified successfully.

EX NO:10

DATE:

SIMULATION OF LINK STATE ROUTING

AIM:

To implement the simulation of link state routing using Network Simulator.

ALGORITHM:

1. Create a simulator object
2. Set routing protocol to Link State routing
3. Trace packets on all links onto NAM trace and text trace file
4. Define finish procedure to close files, flush tracing and run NAM
5. Create eight nodes
6. Specify the link characteristics between nodes
7. Describe their layout topology as a octagon
8. Add UDP agent for node n1
9. Create CBR traffic on top of UDP and set traffic parameters.
10. Add a sink agent to node n4
11. Connect source and the sink
12. Schedule events as follows:
    a. Start traffic flow at 0.5
    b. Down the link n3-n4 at 1.0
    c. Up the link n3-n4 at 2.0
    d. Stop traffic at 3.0
    e. Call finish procedure at 5.0
13. Start the scheduler
14. Observe the traffic route when link is up and down
15. View the simulated events and trace file analyze it
16. Stop

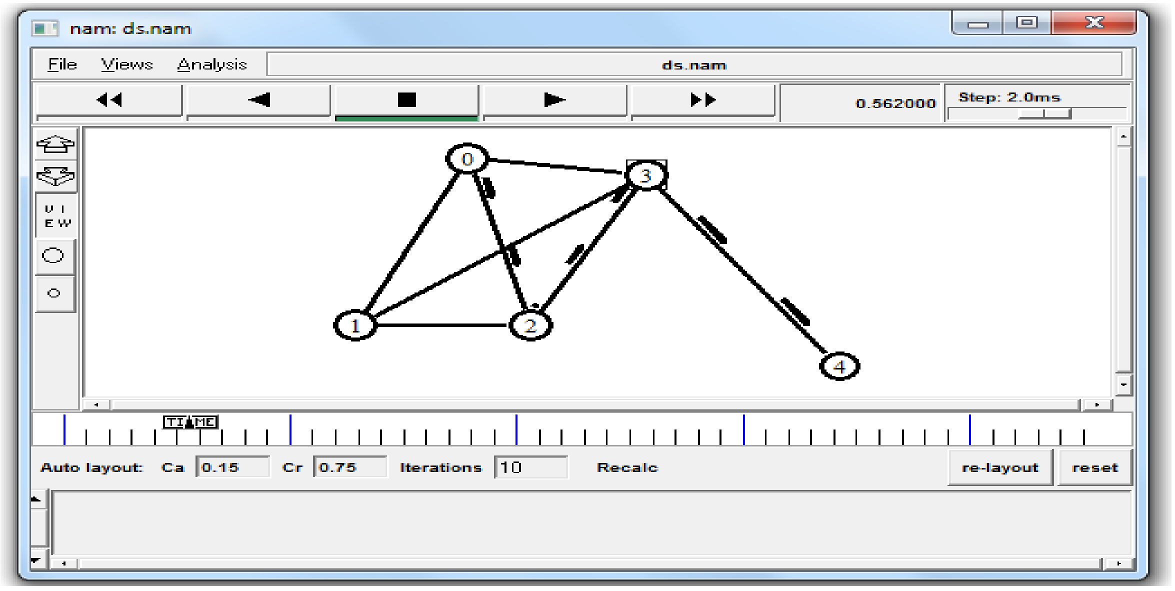
PROGRAM:

LINKSTATE.TCL

```
set ns1 [new Simulator]
set nr1 [open ds.tr w]
$ns1 trace-all $nr1
set nf1 [open ds.nam w]
$ns1 namtrace-all $nf1

set n0 [$ns1 node]
set n1 [$ns1 node]
set n2 [$ns1 node]
set n3 [$ns1 node]
set n4 [$ns1 node]

$ns1 duplex-link $n0 $n1  1Mb 10ms DropTail
$ns1 duplex-link $n0 $n2  1Mb 10ms DropTail
$ns1 duplex-link $n1 $n3  1Mb 10ms DropTail
$ns1 duplex-link $n1 $n2  1Mb 10ms DropTail
$ns1 duplex-link $n0 $n3  1Mb 10ms DropTail
$ns1 duplex-link $n3 $n4  1Mb 10ms DropTail
$ns1 duplex-link $n2 $n3  1Mb 10ms DropTail

$ns1 cost $n0 $n1 2
$ns1 cost $n0 $n2 1
$ns1 cost $n0 $n3 3

$ns1 cost $n1 $n0 2
$ns1 cost $n1 $n2 2
$ns1 cost $n1 $n3 3

$ns1 cost $n2 $n0 1
$ns1 cost $n2 $n1 2
$ns1 cost $n2 $n3 1

$ns1 cost $n3 $n1 3
$ns1 cost $n3 $n2 1
$ns1 cost $n3 $n4 2
$ns1 cost $n3 $n0 3
```

```
$ns1 cost $n4 $n3 2


set udp0 [new Agent/UDP]
$ns1 attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetsize_ 1000
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set Null0 [new Agent/Null]
$ns1 attach-agent $n4 $Null0
$ns1 connect $udp0 $Null0
$ns1 rtproto LS
$ns1 at 0.5 "$cbr0 start"
$ns1 at 4.5 "$cbr0 stop"
proc finish { } {
        global ns1 nr1 nf1
        $ns1 flush-trace
        close $nf1
        close $nr1
        exec nam ds.nam &
        exit 0   }
$ns1 at 5.0 "finish"
$ns1 run
```

OUTPUT:



RESULT:

Thus , the simulation of link state routing using Network Simulator was executed and output is verified successfully.

EX.NO: 11

DATE:

SIMULATION OF ERROR CORRECTION CODE (CRC)

AIM

To implement simulation of Error correction Code(CRC) using java.

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 zero’s. 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:

CRC.JAVA

```
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());
 }
}
```

OUTPUT:

```
Enter Generator:
1011
Enter Data:
1001010
The transmitted Code Word is: 1001010111
Please enter the received Code Word:
1110110111
The received code word contains errors.
```

RESULT:

Thus , the simulation of Error correction Code(CRC) was executed and output is verified successfully.

Ex.No:12

Date:

STOP AND WAIT PROTOCOL

PROGRAM:

SENDER

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
class SWSender
{
public static void main(String args[])throws Exception
{
SWSender sws=new SWSender();
sws.run();
}
public void run()throws Exception
{
Scanner sc=new Scanner(System.in);
Socket myskt=new Socket("localhost",9999);
PrintStream out=new PrintStream(myskt.getOutputStream());
BufferedReader in=new BufferedReader(new InputStreamReader(myskt.getInputStream()));
String msg,ack;
while(true)
{
System.out.println("input message to send?");
msg=sc.next();
out.println(msg);
if(msg.equalsIgnoreCase("quit"))
break;
System.out.println("Waiting for Acknowledgement...");
Thread.sleep(2000);
ack=in.readLine();
if(ack!=null)
System.out.println("Acknowledgement:"+ack);
}
}
}
```

RECEIVER

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
class SWReciever
{
public static void main(String args[])throws Exception
{
SWReciever swr=new SWReciever();
swr.run();
}
public void run()throws Exception
{
String msg;
ServerSocket myss=new ServerSocket(9999);
Socket skt=myss.accept();
BufferedReader in=new BufferedReader(new InputStreamReader(skt.getInputStream()));
PrintStream out=new PrintStream(skt.getOutputStream());
while(true)
{
Thread.sleep(1000);
msg=in.readLine();
if (msg.equalsIgnoreCase("quit"))
break;
System.out.println(msg+"was received");
System.out.println("acknowledgement sent...");
out.println(msg+"received...");
System.out.println("ALL FRAMES WERE RECIEVED SUCCESSFULLY");
}
}
}
```

OUTPUT:

SENDER

```
Z:\>javac SWSender.java
Z:\>java SWSender
input message to send?
HELLO
```

```
Waiting for Acknowledgement...
Acknowledgement:HELLO recieved...
input message to send?
HI
Waiting for Acknowledgement...
Acknowledgement:HIrecieved...
input message to send?
QUIT
```

RECIEVER

```
Z:\>javac SWReceiver.java
Z:\>java SWReceiver
HELLO was received
acknowledgement sent...
ALL FRAMES WERE RECIEVED SUCCESSFULLY
HI was received
acknowledgement sent...
ALL FRAMES WERE RECIEVED SUCCESSFULLY
```

Ex.No:13
Date :

SLIDING WINDOW PROTOCOL

PROGRAM:

SENDER:

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class SlideSender
{
public void send()throws Exception
{
try{
int LFS=0,SWS=0,LAR=0;
String msg,ack;
Socket myskt=new Socket("localhost",9999);
PrintStream out=new PrintStream(myskt.getOutputStream());
BufferedReader in=new BufferedReader(new InputStreamReader(myskt.getInputStream()));
Scanner sc=new Scanner(System.in);
System.out.println("Sliding window size?");
SWS=sc.nextInt();
while(true)
{
if(LFS<SWS)
{
System.out.println("\nInput message to send?");
msg=sc.next();
LFS++;
out.println(msg);
if(msg.equalsIgnoreCase("quit"))
break;
Thread.sleep(300);
}
else
{
System.out.println("\nWindow is full.Waiting for acknowledgement...");
while(LAR<LFS)
{
ack=in.readLine();
System.out.println("Acknowledgement:"+ack+"recieved");
SWS++;
LAR++;}}
}
}catch(Exception e){}
}
public static void main(String args[])throws Exception
{
SlideSender ss=new SlideSender();
ss.send();
}
}
```

RECIEVER:

```
import java.io.*;
import java.net.*;
public class SlideReciever
{
public static void main(String args[])throws Exception
{
SlideReciever sr=new SlideReciever();
sr.run();
}
public void run()throws Exception
{
String msg;
ServerSocket myss=new ServerSocket(9999);
Socket skt=myss.accept();
BufferedReader in=new BufferedReader(new InputStreamReader(skt.getInputStream()));
PrintStream out=new PrintStream(skt.getOutputStream());
int LFR=0;
while(true)
{
Thread.sleep(1000);
msg=in.readLine();
if(msg!=null)
LFR++;
if(msg.equalsIgnoreCase("quit"))
break;
System.out.println(msg+"was recieved");
System.out.println("Acknowledgement sent for
frame"+LFR); out.println(Integer.toString(LFR));
}
System.out.println("ALL FRAMES WERE RECIEVED SUCCESSFULLY"); } }
```

OUTPUT:

SENDER:

```
Z:\>javac SlideSender.java
Z:\>java SlideSender
Sliding window size?
4
Input message to send?
COMPUTER
Input message to send?
PENDRIVE
Input message to send?
CD
Input message to send?
DVD

Window is full.Waiting for acknowledgement...
Acknowledgement:1recieved
Acknowledgement:2recieved
Acknowledgement:3recieved
Acknowledgement:4recieved
Input message to send?
HI
Input message to send?
quit
```

RECIEVER:

```
Z:\>javac SlideReciever.java
Z:\>java SlideReciever
COMPUTER was recieved
Acknowledgement sent for frame1
PENDRIVE was recieved
Acknowledgement sent for frame2
CD was recieved
Acknowledgement sent for frame3
DVD was recieved
Acknowledgement sent for frame4
HI was recieved
Acknowledgement sent for frame5
ALL FRAMES WERE RECIEVED SUCCESSFULLY
```