



**VINAYAKA MISSION'S
RESEARCH FOUNDATION**

(Deemed to be University under section 3 of the UGC Act 1956)



**VINAYAKA MISSION'S KIRUPANANDA
VARIYAR ENGINEERING
COLLEGE**

**VINAYAKA MISSION'S KIRUPANANDA VARIYAR ENGINEERING
COLLEGE,
SALEM – 636 308**

**A Constituent College of VINAYAKA MISSION' S RESEARCH
FOUNDATION
(Deemed to be University)**

**DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING**

LABORATORY MANUAL

Name _____

Semester _____

Roll No _____ **Reg. No.** _____

Subject _____

CONTENTS

EX. NO.	DATE	TITLE OF THE PROGRAM	PAGE NO.	SIGNATURE
1.		IMPLEMENTATION OF STOP AND WAIT PROTOCOL AND SLIDING WINDOW PROTOCOL.		
2.		STUDY OF SOCKET PROGRAMMING AND CLIENT – SERVER MODEL		
3.		WRITE A CODE SIMULATING ARP /RARP PROTOCOLS.		
4.		WRITE A CODE SIMULATING PING AND TRACE ROUTE COMMANDS		
5.		STUDY OF NETWORK SIMULATOR (NS) AND SIMULATION OF CONGESTION CONTROL ALGORITHMS USING NS		
6.		SIMPLE TCP/IP CLIENT SERVER COMMUNICATION		
7.		UDP ECHO CLIENT SERVER COMMUNICATION		
8.		HALF DUPLEX CHAT USING TCP/IP		
9.		FULL DUPLEX CHAT USING TCP/IP		
10.		SIMULATION OF DISTANCE VECTOR/ LINK STATE ROUTING ALGORITHM		
11.		PERFORMANCE EVALUATION OF ROUTING PROTOCOLS USING SIMULATION TOOL.		
12.		SIMULATION OF ERROR CORRECTION CODE (LIKE CRC).		

EX.NO:1 IMPLEMENTATION OF STOP AND WAIT PROTOCOL

AIM

To write a java program to Implement of Stop and Wait Protocol and Sliding Window Protocol.

ALGORITHM:

1. Start the program.
2. Get the frame size from the user
3. To create the frame based on the user request.
- 4.To send frames to server from the client side.
- 5.If your frames reach the server it will send ACK signal to client otherwise it will send NACK signal to client.
- 6.Stop the program.

PROGRAM:

```
//Implementation of Stop and Wait Protocol  
//SENDER PROGRAM
```

```
import java.io.*;  
import java.net.*;  
import java.lang.String;  
import java.util.Date;  
  
class Sender{  
    Socket sender;  
    ObjectOutputStream out;  
    ObjectInputStream in;  
    String packet,ack,str,msg;  
    int n,i=0,sequence=0;  
    Sender(){ }  
    public void run()
```

```

{
try{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Waiting for Connection....");
sender = new Socket("localhost",2004);
sequence=0;
out=new ObjectOutputStream(sender.getOutputStream());
out.flush();
in=new ObjectInputStream(sender.getInputStream());
str=(String)in.readObject();
System.out.println("reciver > "+str);
System.out.println("Enter the data to send....");
packet=br.readLine();
n=packet.length();
do{
try{
if(i<n)
{
msg=String.valueOf(sequence);
msg=msg.concat(packet.substring(i,i+1));
}
else if(i==n)
{
msg="end";out.writeObject(msg);break;
}
out.writeObject(msg);
sequence=(sequence==0)?1:0;
out.flush();
System.out.println("data sent>"+msg);
ack=(String)in.readObject();
System.out.println("waiting for ack.....\n\n");
if(ack.equals(String.valueOf(sequence)))
{
i++;
System.out.println("receiver > "+" packet recieved\n\n");
}
else{
System.out.println("Time out resending data....\n\n");
sequence=(sequence==0)?1:0;

```

```

    }
    }catch(Exception e){ }
    }while(i<n+1);
    System.out.println("All data sent.exiting.");
    }catch(Exception e)
    {
    }
    finally
    {
    try{
    in.close();
    out.close();
    sender.close();
    }
    catch(Exception e){ }
    }
    }
    public static void main(String args[])throws Exception
    {
    Sender s=new Sender();
    s.run();
    }
    }

```

//Implementation of Stop and Wait Protocol
//RECEIVER PROGRAM

```

import java.io.*;
import java.net.*;
import java.lang.String;
import java.util.Date;

public class Receiver
{
    ServerSocket reciever;
    Socket connection=null;
    ObjectOutputStream out;
    ObjectInputStream in;

```

```

String packet,ack,data="";
int i=0,sequence=0;
Receiver(){ }
public void run(){
try{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
reciever = new ServerSocket(2004,10);
System.out.println("waiting for connection...");
connection=reciever.accept();
sequence=0;
System.out.println("Connection established :");

out=new ObjectOutputStream(connection.getOutputStream()); out.flush();

in=new ObjectInputStream(connection.getInputStream());
out.writeObject("connected ."); do{

try{
packet=(String)in.readObject();
if(Integer.valueOf(packet.substring(0,1))==sequence){
data+=packet.substring(1);
sequence=(sequence==0)?1:0;
System.out.println("\n\nreceiver>"+packet);
}
else
{
System.out.println("\n\nreceiver >"+packet + "duplicate data");
}
if(i<3)
{
out.writeObject(String.valueOf(sequence));i++;
}
else
{
out.writeObject(String.valueOf((sequence+1)%2));
i=0;
}
}
}

```

```
}  
catch(Exception e){ }  
}while(!packet.equals("end"));  
  
System.out.println("Data recived="+data);  
out.writeObject("connection ended.");  
}  
catch(Exception e){ }  
finally{  
try{  
in.close();  
out.close();  
reciever.close();  
}  
catch(Exception e){ }  
}  
}  
  
public static void main(String args[])throws Exception  
{  
Receiver s=new Receiver();  
while(true){  
s.run();  
}  
}  
}
```

INPUT:

```
C:\WINDOWS\system32\CMD.exe - java Sender
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Student>D:

D:\>cd jdk1.6\bin

D:\jdk1.6\bin>javac Sender.java

D:\jdk1.6\bin>java Sender
Waiting for Connection....

D:\jdk1.6\bin>java Sender
Waiting for Connection....
Receiver > connected .
Enter the data to send....
computer network

C:\WINDOWS\system32\CMD.exe - java Receiver
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Student>D:

D:\>cd jdk1.6\bin

D:\jdk1.6\bin>javac Receiver.java

D:\jdk1.6\bin>java Receiver
waiting for connection...
Connection established :
```

OUTPUT:

```
Select C:\WINDOWS\system32\CMD.exe
data sent>0c
waiting for ack.....

receiver > packet recieved

data sent>1o
waiting for ack.....

receiver > packet recieved

data sent>0m
waiting for ack.....

receiver > packet recieved

data sent>1p
waiting for ack.....

Time out resending data....

data sent>1p
waiting for ack.....

receiver > packet recieved

data sent>0u
waiting for ack.....

receiver > packet recieved

data sent>1t
waiting for ack.....

receiver > packet recieved

C:\WINDOWS\system32\CMD.exe - java Receiver
D:\jdk1.6\bin>java Receiver
waiting for connection...
Connection established :

receiver>0c

receiver>1o

receiver>0m

receiver>1p

receiver >1pduplicate data

receiver>0u

receiver>1t

receiver>0e

receiver >0eduplicate data

receiver>1r

receiver>0

receiver>1n

receiver >1nduplicate data

receiver>0e

receiver>1t
```



```
receiver > packet recieved
data sent>0e
waiting for ack.....
Time out resending data....
data sent>0e
waiting for ack.....
receiver > packet recieved
data sent>1r
waiting for ack.....
receiver > packet recieved
data sent>0e
waiting for ack.....
receiver > packet recieved
data sent>1n
waiting for ack.....
Time out resending data....
data sent>1n
waiting for ack.....
receiver > packet recieved
data sent>0e
waiting for ack.....

receiver>1t
receiver>0e
receiver >0eduplicate data
receiver>1r
receiver>0
receiver>1n
receiver >1nduplicate data
receiver>0e
receiver>1t
receiver>0w
receiver >0wduplicate data
receiver>1o
receiver>0r
receiver>1k
receiver >1kduplicate data
Data recived-computer network
waiting for connection...
```

```
receiver > packet recieved
data sent>0w
waiting for ack.....
Time out resending data....
data sent>0w
waiting for ack.....
receiver > packet recieved
data sent>1o
waiting for ack.....
receiver > packet recieved
data sent>0r
waiting for ack.....
receiver > packet recieved
data sent>1k
waiting for ack.....
Time out resending data....
data sent>1k
waiting for ack.....
receiver > packet recieved
All data sent.exiting.
```

RESULT:

Thus the program Implement of Stop and Wait Protocol and Sliding Window Protocol was executed successfully.

EX.NO:2 STUDY OF SOCKET PROGRAMMING AND CLIENT-SERVER MODE

AIM:

To write a java program to Study of Socket Programming and Client – Server model

ALGORITHM:

Server:

1. Create a server socket and bind it to port.
2. Listen for new connection and when a connection arrives accept it.
3. Send server's date and time to the client.
4. Read clients IP address sent by the client.
5. Display the client details.
6. Repeat steps 2-5 until the server is terminated.
7. Close all streams.
8. Close the server socket.
9. Stop.

Client:

1. Create a client socket and connect it to the server port number.
2. Retrieve its own IP address using built-in function.
3. Send its address to the server.
4. Display the date & time sent by the server.
5. Close the input and output streams.
6. Close the client socket.
7. Stop.

PROGRAM:

// A Java program for a Server

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

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

```
public class Server
```

```
{
```

```
    //initialize socket and input stream
```

```
private Socket      socket = null;
private ServerSocket server = null;
private DataInputStream in = null;

// constructor with port
public Server(int port)
{
    // starts server and waits for a connection
    try
    {
        server = new ServerSocket(port);
        System.out.println("Server started");

        System.out.println("Waiting for a client ...");

        socket = server.accept();
        System.out.println("Client accepted");

        // takes input from the client socket
        in = new DataInputStream(
            new BufferedInputStream(socket.getInputStream()));

        String line = "";

        // reads message from client until "Over" is sent
        while (!line.equals("Over"))
        {
            try
            {
                line = in.readUTF();
                System.out.println(line);

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

```
System.out.println("Closing connection");
```

```
// close connection
```

```
socket.close();
```

```
in.close();
```

```
}
```

```
catch(IOException i)
```

```
{
```

```
System.out.println(i);
```

```
}
```

```
}
```

```
public static void main(String args[])
```

```
{
```

```
Server server = new Server(5000);
```

```
}
```

```
}
```

```
// A Java program for a Client
```

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

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

```
import java.lang.String;
```

```
import java.util.Date;
```

```
public class Client {
```

```
// initialize socket and input output streams
```

```
private Socket socket = null;
```

```
private DataInputStream input = null;
```

```
private DataOutputStream out = null;
```

```
// constructor to put ip address and port
```

```
public Client(String address,int port)
```

```
{
```

```
// establish a connection
```

```
try {
```

```
socket = new Socket(address,port);
```

```
System.out.println("Connected");
```

```
// takes input from terminal
input = new DataInputStream(System.in);

// sends output to the socket
out = new DataOutputStream(socket.getOutputStream());
}
catch (UnknownHostException u) {
    System.out.println(u);
    return;
}
catch (IOException i)
{
    System.out.println(i);
    return;
}

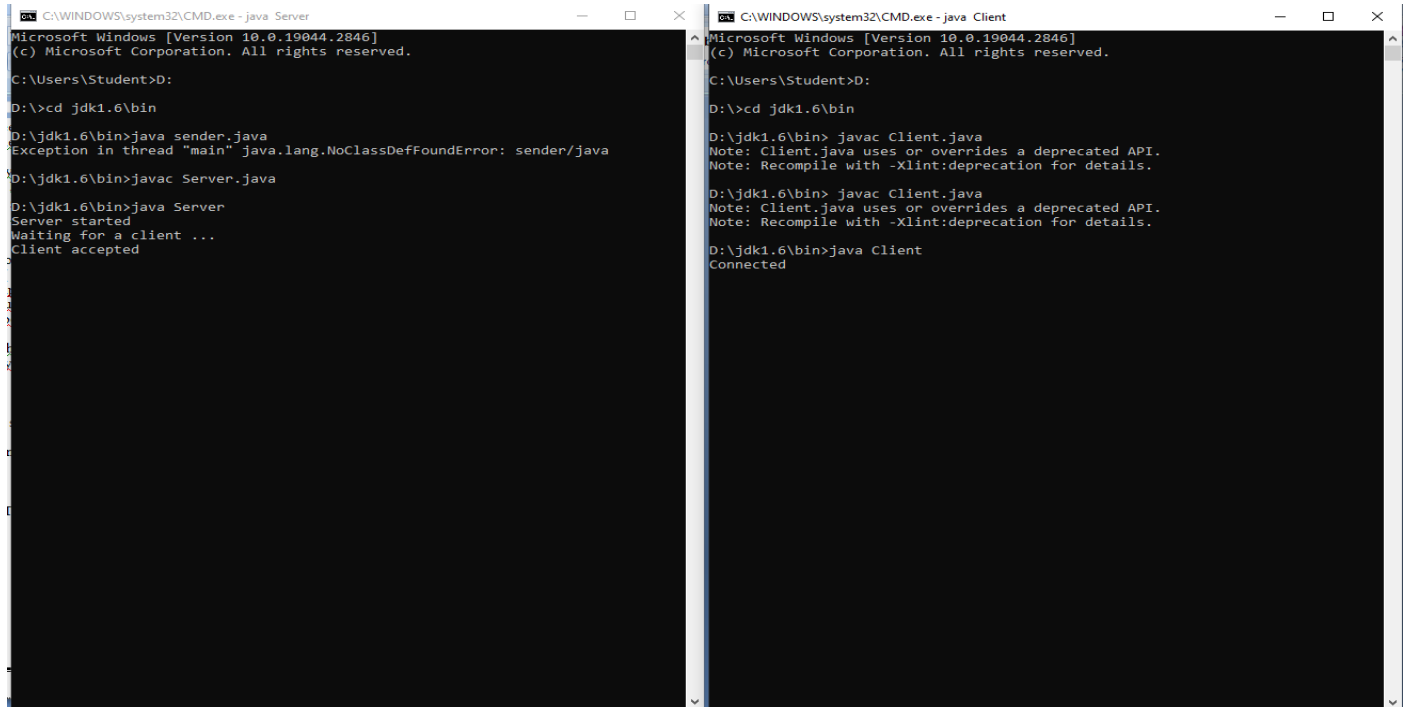
// string to read message from input
String line = "";

// keep reading until "Over" is input
while (!line.equals("Over"))
{
    try {
        line = input.readLine();
        out.writeUTF(line);
    } catch (IOException i) { System.out.println(i); }
}

// close the connection
try {
    input.close();
    out.close();
    socket.close();
}
catch (IOException i) {
    System.out.println(i);
}
}
```

```
public static void main(String args[])throws Exception
{
    Client client = new Client("127.0.0.1",5000);
}
}
```

OUTPUT:



```
C:\WINDOWS\system32\CMD.exe - java Server
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Student>D:
D:\>cd jdk1.6\bin
D:\jdk1.6\bin>java sender.java
Exception in thread "main" java.lang.NoClassDefFoundError: sender/java
D:\jdk1.6\bin>javac Server.java
D:\jdk1.6\bin>java Server
Server started
Waiting for a client ...
Client accepted

C:\WINDOWS\system32\CMD.exe - java Client
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Student>D:
D:\>cd jdk1.6\bin
D:\jdk1.6\bin> javac Client.java
Note: Client.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
D:\jdk1.6\bin> javac Client.java
Note: Client.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
D:\jdk1.6\bin>java Client
Connected
```

RESULT:

Thus the program for Study of Socket Programming and Client – Server model was executed successfully.

EX.NO:3 Write a code simulating ARP/RARP Protocols.

Aim:

1. To write a java program for simulating ARP protocols using TCP

ALGORITHM:

Client

2. Start the program
3. Using socket connection is established between client and server.
4. Get the IPaddress to be converted into MAC address.
5. Send this IPaddress to server.
6. Server returns the MAC address to client.

Server

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

PROGRAM:

SERVERARP:

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

```

CLIENTARP:

```

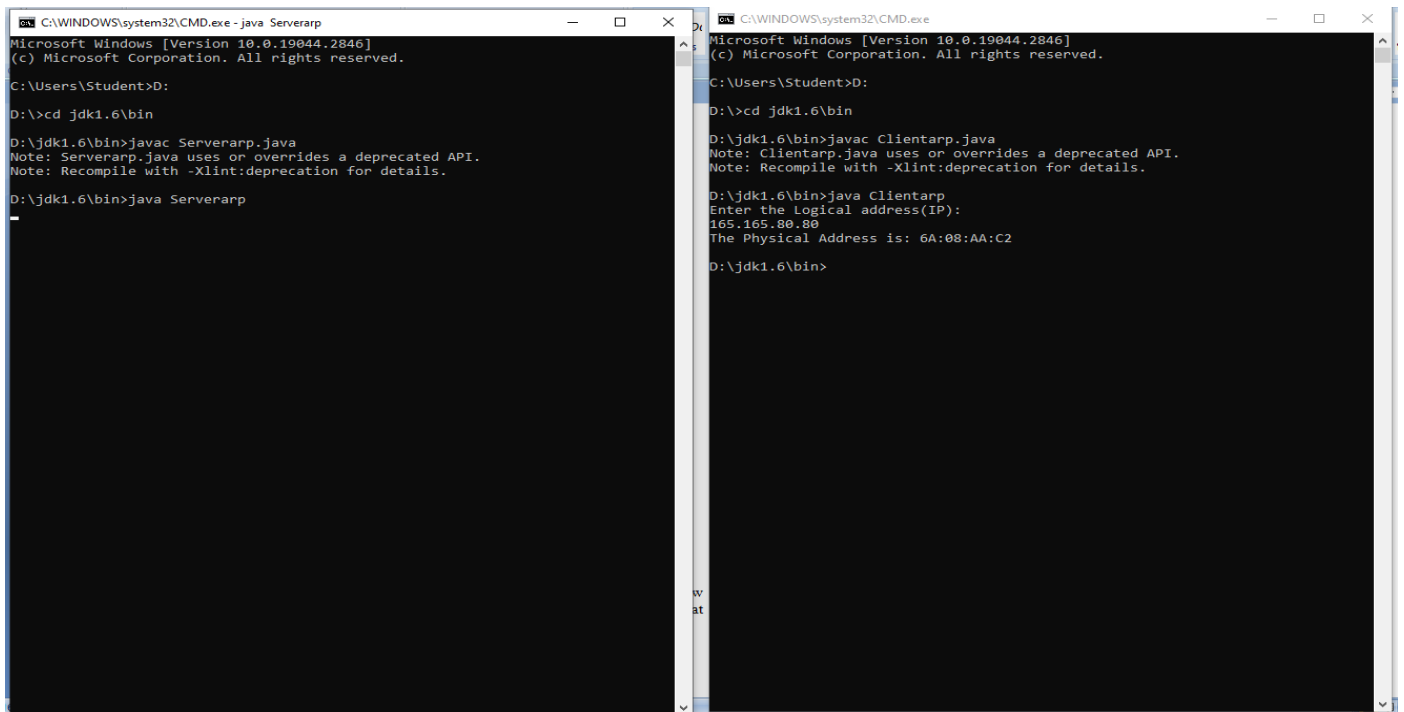
import java.io.*;
import java.net.*;
import java.util.*;
class Clientarp
{
public static void main(String args[])
{
try
{
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
Socket clsct=new Socket("127.0.0.1",139);
DataInputStream din=new DataInputStream(clsct.getInputStream());
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);  
}  
}  
}
```

OUTPUT:



The image shows two side-by-side screenshots of Windows Command Prompt windows. The left window shows the compilation and execution of a server program. The right window shows the compilation and execution of a client program that prompts for an IP address and displays the physical address.

```
C:\WINDOWS\system32\CMD.exe - java Serverarp  
Microsoft Windows [Version 10.0.19044.2846]  
(c) Microsoft Corporation. All rights reserved.  
C:\Users\Student>D:  
D:\>cd jdk1.6\bin  
D:\jdk1.6\bin>javac Serverarp.java  
Note: Serverarp.java uses or overrides a deprecated API.  
Note: Recompile with -Xlint:deprecation for details.  
D:\jdk1.6\bin>java Serverarp  
_
```

```
C:\WINDOWS\system32\CMD.exe  
Microsoft Windows [Version 10.0.19044.2846]  
(c) Microsoft Corporation. All rights reserved.  
C:\Users\Student>D:  
D:\>cd jdk1.6\bin  
D:\jdk1.6\bin>javac Clientarp.java  
Note: Clientarp.java uses or overrides a deprecated API.  
Note: Recompile with -Xlint:deprecation for details.  
D:\jdk1.6\bin>java Clientarp  
Enter the Logical address(IP):  
165.165.80.80  
The Physical Address is: 6A:08:AA:C2  
D:\jdk1.6\bin>
```

RESULT:

Thus a program for simulating ARP protocols using TCP was executed successfully.

EX-NO.4 . WRITE A CODE SIMULATING PING AND TRACE ROUTE COMMANDS

Aim:

To Write the java program for simulating ping command

Algorithm:

Step1:start the program.

Step2:Include necessary package in java.

Step3:To create a process object to implement the ping command.

Step4:declare one Buffered Reader stream class object.

Step5:Get the details of the server

5.1:length of the IP address

5.2:time required to get the details.

5.3:send packets,receive packet sand lost packets .5.4:minimum ,maximum and average times.

Step6:printthe results.

Step7:Stoptheprogram.

PROGRAM:

```
//PING SERVER
```

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

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

```
class pingserver
```

```
{
```

```
public static void main(String args[])
```

```
{
```

```
try
```

```
{
```

```
String str;
```

```

System.out.print(" 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());
}
}
}
//TRACEROUTE commands-
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class traceroutecmd
{
    public static void runSystemCommand(String command)
    {
        try
        {
            Process p = Runtime.getRuntime().exec(command);
            BufferedReader inputStream = new BufferedReader(
                new InputStreamReader(p.getInputStream()));
            String s = "";
            while ((s = inputStream.readLine()) != null)
                System.out.println(s);
        }
        catch (Exception e)
        {

```

```

    }
}

public static void main(String[] args)
{
    // String ip = "www.google.co.in";
    // String ip = "127.0.0.1";
    String ip = "www.cp-algorithms.com";
    runSystemCommand("tracert " + ip);
}
}

```

OUTPUT:

```

C:\WINDOWS\system32\CMD.exe
D:\jdk1.6\bin>javac Traceroutecmd.java
D:\jdk1.6\bin>java Traceroutecmd

Tracing route to cp-algorithms.firebaseio.com [199.36.158.100]
over a maximum of 30 hops:

  0  <1 ms    <1 ms    <1 ms    192.168.100.1
  1  1 ms      2 ms      1 ms     103.53.53.9
  2  2 ms      2 ms      2 ms     219.65.111.177.STATIC-Chennai.vsnl.net.in [219.
65.111.177]
  3  7 ms      7 ms      7 ms     172.19.249.170
  4  10 ms     6 ms      7 ms     172.31.167.58
  5  7 ms      7 ms      7 ms     14.141.123.226.static-Chennai.vsnl.net.in [14.1
41.123.226]
  6  25 ms     25 ms     24 ms     172.28.177.73
  7  26 ms     26 ms     26 ms     115.110.206.150.static-Mumbai.vsnl.net.in [115.
110.206.150]
  8  23 ms     24 ms     24 ms     199.36.158.100

Trace complete.
D:\jdk1.6\bin>

C:\WINDOWS\system32\CMD.exe
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Student>d:
D:\>cd jdk1.6\bin
D:\jdk1.6\bin>pingserver.java
D:\jdk1.6\bin>javac pingserver.java
D:\jdk1.6\bin>javac pingserver
error: Class names, 'pingserver', are only accepted if annotation pro
explicitly requested
1 error
D:\jdk1.6\bin>java pingserver
Enter the IP Address to be Ping : 192.168.0.1

Pinging 192.168.0.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.0.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
D:\jdk1.6\bin>

```

RESULT:

Thus the program the java program for simulating ping command was executed successfully.

EX-NO 05 STUDY OF NETWORK SIMULATOR (NS) AND SIMULATION OF CONGESTION CONTROL ALGORITHMS USING NS

Aim:

To Study of Network simulator(NS).and Simulation of Congestion Control Algorithms using NS

Algorithm:

There are several variants of flooding algorithm. Most work roughly as follows:

Each node acts as both a transmitter and a receiver.

Each node tries to forward every message to everyone of its neighbour's except the source node.

This results in every message eventually being delivered to all reachable parts of the network.

Algorithms may need to be more complex than this, since, in some case, precautions have to be taken to avoid wasted duplicate deliveries and infinite loops, and to allow messages to eventually expire from the system. A variant of flooding called selective flooding partially addresses these issues by only sending packets to routers in the same direction. In selective flooding the routers don't send every incoming packet on every line but only on those lines which are going approximately in the right direction

PROGRAM:

Sample Ns2 program for Congestion control

```
include <wifi_lte/wifi_lte_rtable.h>
```

```
    struct r_hist_entry *elm, *elm2;
```

```
int num_later = 1;
```

```
elm = STAILQ_FIRST(&r_hist_);
```

```
while (elm != NULL && num_later <= num_dup_acks_){  
    num_later;
```

```

    elm = STAILQ_NEXT(elm, linfo_);
}

if (elm != NULL){
    elm = findDataPacketInRecvHistory(STAILQ_NEXT(elm, linfo_));

    if (elm != NULL){
        elm2 = STAILQ_NEXT(elm, linfo_);
        while(elm2 != NULL){
            if (elm2->seq_num_ < seq_num && elm2->t_recv_ < time){
                STAILQ_REMOVE(&r_hist_, elm2, r_hist_entry, linfo_);
                delete elm2;
            } else
                elm = elm2;
            elm2 = STAILQ_NEXT(elm, linfo_);
        }
    }
}

void DCCPTFRCAgent::removeAcksRecvHistory(){
    struct r_hist_entry *elm1 = STAILQ_FIRST(&r_hist_);
    struct r_hist_entry *elm2;

    int num_later = 1;
    while (elm1 != NULL && num_later <= num_dup_acks_){
        num_later;
        elm1 = STAILQ_NEXT(elm1, linfo_);
    }

    if(elm1 == NULL)
        return;

    elm2 = STAILQ_NEXT(elm1, linfo_);
    while(elm2 != NULL){
        if (elm2->type_ == DCCP_ACK){

```

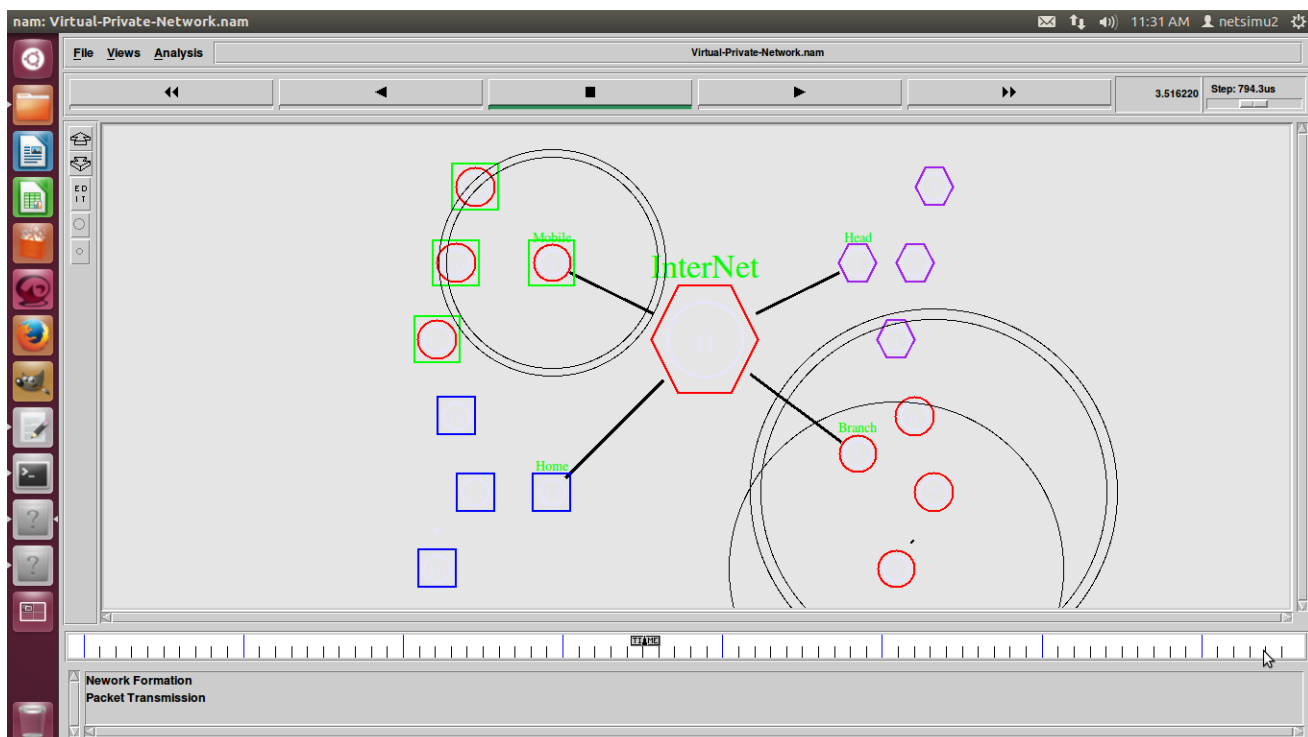
```

        STAILQ_REMOVE(&r_hist_,elm2,r_hist_entry,linfo_);
        delete elm2;
    } else {
        elm1 = elm2;
    }
    elm2 = STAILQ_NEXT(elm1, linfo_);
}
}

inline                                     r_hist_entry
*DCCPTFRCAgent::findDataPacketInRecvHistory(r_hist_entry *start){
while(start != NULL && start->type_ == DCCP_ACK)
    start = STAILQ_NEXT(start,linfo_);
return start;
}

```

OUTPUT:



EX-NO 6 SIMPLE TCP/IP CLIENT SERVER COMMUNICATION

Aim:

To write a java program for application using TCP Sockets Links

Algorithm:

- 1.Start the program.
 - 2.Get the frame size from the user
 - 3.To create the frame based on the user request.
 - 4.To send frames to server from the client side.
 - 5.If your frames reach the server it will send ACK signal to client otherwise it will send NACK signal to client.
- Stop the program

PROGRAM:

// Server Side

import java.net.*;

import java.io.*;

public class ServerSideSocket {

public void run() {

try {

int serverPort = 4020;

ServerSocket serverSocket = new ServerSocket(serverPort);

serverSocket.setSoTimeout(10000);

while(true) {

System.out.println("Waiting for client on port " +
serverSocket.getLocalPort() + "...");

Socket server = serverSocket.accept();

System.out.println("Just connected to " +
server.getRemoteSocketAddress());

PrintWriter toClient =


```

        new PrintWriter(server.getOutputStream(),true);
        BufferedReader fromClient =
            new BufferedReader(
                new
InputStreamReader(server.getInputStream()));
        String line = fromClient.readLine();
        System.out.println("Server received: " + line);
        toClient.println("Thank you for connecting to " +
server.getLocalSocketAddress() + "\nGoodbye!");
    }
}
catch(UnknownHostException ex) {
    ex.printStackTrace();
}
catch(IOException e){
    e.printStackTrace();
}
}

public static void main(String[] args) {
    ServerSideSocket srv = new ServerSideSocket();
    srv.run();
}
}

```

// Client Side

```

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

public class ClientSocket {
    public void run() {
        try {
            int serverPort = 4020;
            InetAddress host = InetAddress.getByName("localhost");
            System.out.println("Connecting to server on port " + serverPort);

            Socket socket = new Socket(host,serverPort);

```

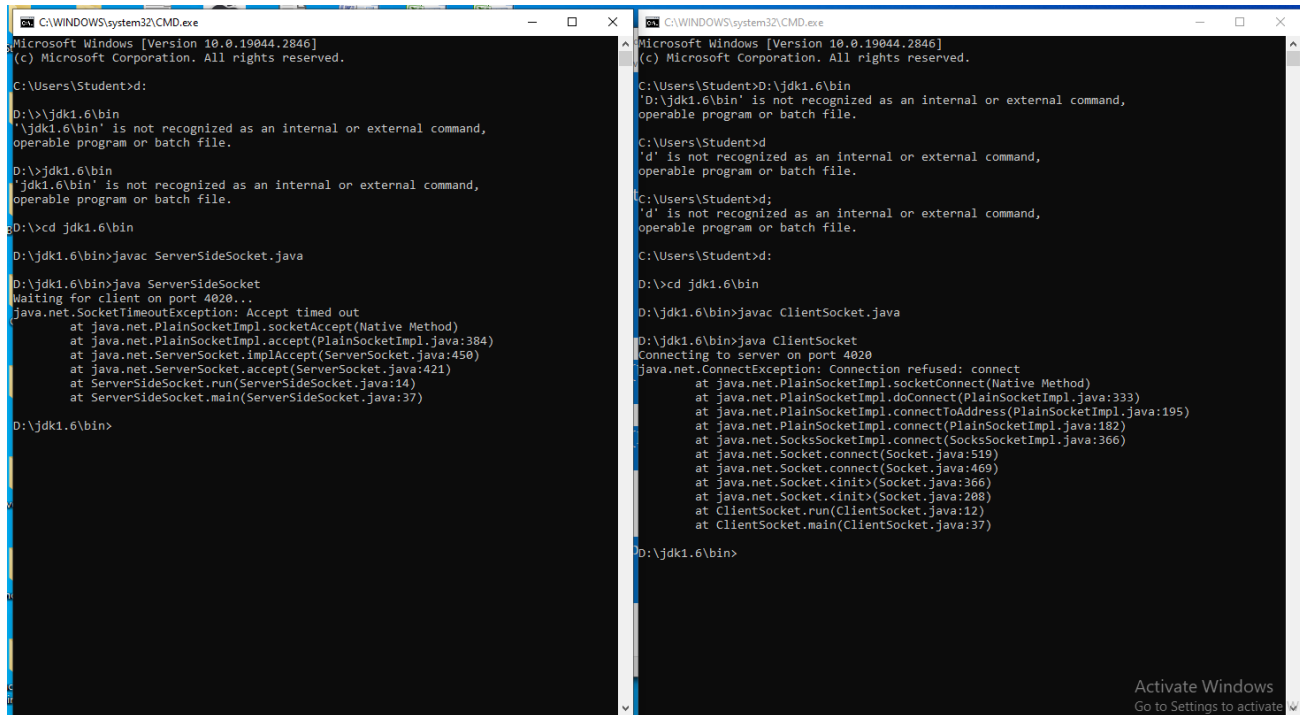
```

        //Socket socket = new Socket("127.0.0.1", serverPort);
        System.out.println("Just connected to " +
socket.getRemoteSocketAddress());
        PrintWriter toServer =
            new PrintWriter(socket.getOutputStream(),true);
        BufferedReader fromServer =
            new BufferedReader(
                new InputStreamReader(socket.getInputStream()));
        toServer.println("Hello from " + socket.getLocalSocketAddress());
        String line = fromServer.readLine();
        System.out.println("Client received: " + line + " from Server");
        toServer.close();
        fromServer.close();
        socket.close();
    }
    catch(UnknownHostException ex) {
        ex.printStackTrace();
    }
    catch(IOException e){
        e.printStackTrace();
    }
}

public static void main(String[] args) {
    ClientSocket client = new ClientSocket();
    client.run();
}
}

```

OUTPUT:



```
C:\WINDOWS\system32\CMD.exe
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Student>d:

D:\>\jdk1.6\bin
'\jdk1.6\bin' is not recognized as an internal or external command,
operable program or batch file.

D:\>\jdk1.6\bin
'\jdk1.6\bin' is not recognized as an internal or external command,
operable program or batch file.

D:\>\cd jdk1.6\bin

D:\jdk1.6\bin>javac ServerSideSocket.java

D:\jdk1.6\bin>java ServerSideSocket
Waiting for client on port 4020...
java.net.SocketTimeoutException: Accept timed out
    at java.net.PlainSocketImpl.socketAccept(Native Method)
    at java.net.PlainSocketImpl.accept(PlainSocketImpl.java:384)
    at java.net.ServerSocket.implAccept(ServerSocket.java:450)
    at java.net.ServerSocket.accept(ServerSocket.java:421)
    at ServerSideSocket.run(ServerSideSocket.java:14)
    at ServerSideSocket.main(ServerSideSocket.java:37)

D:\jdk1.6\bin>
```

```
C:\WINDOWS\system32\CMD.exe
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Student>D:\jdk1.6\bin
'D:\jdk1.6\bin' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Student>d
'd' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Student>d:
'd' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Student>d:

D:\>\cd jdk1.6\bin

D:\jdk1.6\bin>javac ClientSocket.java

D:\jdk1.6\bin>java ClientSocket
Connecting to server on port 4020
java.net.ConnectException: Connection refused: connect
    at java.net.PlainSocketImpl.socketConnect(Native Method)
    at java.net.PlainSocketImpl.doConnect(PlainSocketImpl.java:333)
    at java.net.PlainSocketImpl.connectToAddress(PlainSocketImpl.java:195)
    at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:182)
    at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:366)
    at java.net.Socket.connect(Socket.java:519)
    at java.net.Socket.connect(Socket.java:469)
    at java.net.Socket.<init>(Socket.java:366)
    at java.net.Socket.<init>(Socket.java:208)
    at ClientSocket.run(ClientSocket.java:12)
    at ClientSocket.main(ClientSocket.java:37)

D:\jdk1.6\bin>
```

RESULT:

Thus the program for application using TCP Sockets Links was executed successfully.

EX.NO 7:

PROGRAM FOR REVERSE ADDRESS RESOLUTION PROTOCOL (RARP) USING UDP

Aim:

To write a java program for simulating RARP protocols using UDP

ALGORITHM:

Client

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

PROGRAM:

CLIENT

```
import java.net.*;
import java.io.*;
public class UDPEchoClient
{
    public static class UDPEchoReader extends Thread
    {
        public UDPEchoReader(DatagramSocket socket)
        {
            datagramSocket = socket;
            active = true;
        }
        public void run()
        {
            byte[] buffer = new byte[1024];
            DatagramPacket incoming = new DatagramPacket(buffer, buffer.length);
            String receivedString;
            while(active)
            {
                try
                {
```

```

// listen for incoming datagram packet
datagramSocket.receive(incoming);
// print out received string
receivedString = new String(incoming.getData(),
0, incoming.getLength());
System.out.println("Received from server: "+receivedString);
}
catch(IOException e)
{
System.out.println(e);
active = false;
}

public boolean active;
public DatagramSocket datagramSocket;

public static void main(String[] args)
{
InetAddress address = null;
int port = 8000;
DatagramSocket datagramSocket = null;
BufferedReader keyboardReader = null;
// Create a Datagram Socket...
try
{
address = InetAddress.getByName("127.0.0.1");
datagramSocket = new DatagramSocket();
keyboardReader = new BufferedReader(new
InputStreamReader(System.in));
}
catch (IOException e)
{
System.out.println(e);
System.exit(1);
}

// Start the listening thread...
UDPEchoReader reader = new UDPEchoReader(datagramSocket);
reader.setDaemon(true);
reader.start();
System.out.println("Ready to send your messages...");
try
{
String input;
while (true)
{
// read input from the keyboard
input = keyboardReader.readLine();
// send datagram packet to the server
DatagramPacket datagramPacket = new DatagramPacket
(input.getBytes(), input.length(), address, port);
datagramSocket.send(datagramPacket);
}
}
catch(IOException e)
{
}
}

```

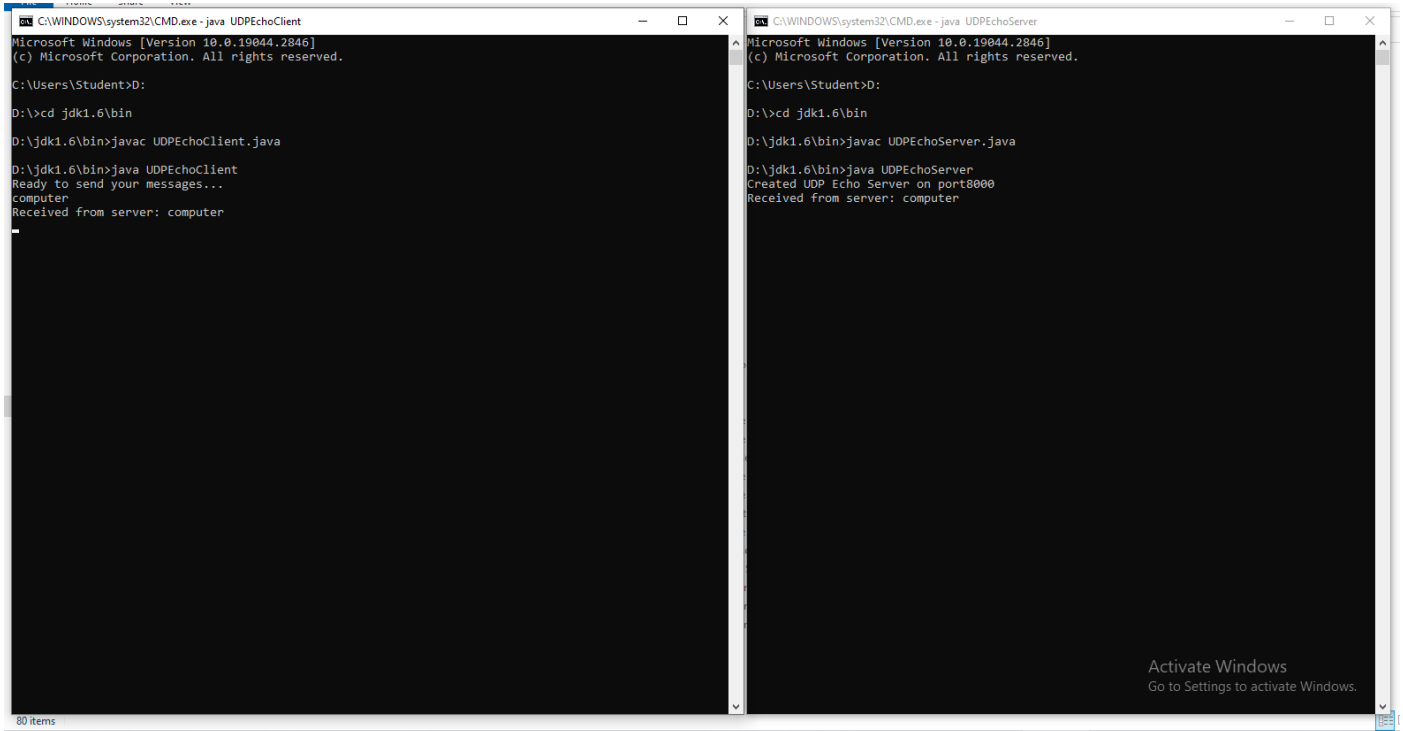
```
        System.out.println(e);
    }
}.
```

SERVER:

```
import java.net.*;
import java.io.*;
public class UDPEchoServer
{
    public static void main(String args[])
    {
        int port = 8000;
        // create the server...
        DatagramSocket serverDatagramSocket = null;
        try
        {
            serverDatagramSocket = new DatagramSocket(port);
            System.out.println("Created UDP Echo Server on port"+port);
        }
        catch(IOException e)
        {
            System.out.println(e);
            System.exit(1);
        }
        try
        {
            byte buffer[] = new byte[1024];
            DatagramPacket datagramPacket = new
            DatagramPacket(buffer, buffer.length);
            String input;
            while(true)
            {
                // listen for datagram packets
                serverDatagramSocket.receive(datagramPacket);
                input = new String(datagramPacket.getData(), 0,
                datagramPacket.getLength());
                System.out.println("Received from server: "+input);
                // send received packet back to the client
                serverDatagramSocket.send(datagramPacket);
            }
        }
    }
}
```

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

OUTPUT:



```
C:\WINDOWS\system32\CMD.exe - java UDPEchoClient  
Microsoft Windows [Version 10.0.19044.2846]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\Student>D:  
  
D:\>cd jdk1.6\bin  
  
D:\jdk1.6\bin>javac UDPEchoClient.java  
  
D:\jdk1.6\bin>java UDPEchoClient  
Ready to send your messages...  
computer  
Received from server: computer  
_
```

```
C:\WINDOWS\system32\CMD.exe - java UDPEchoServer  
Microsoft Windows [Version 10.0.19044.2846]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\Student>D:  
  
D:\>cd jdk1.6\bin  
  
D:\jdk1.6\bin>javac UDPEchoServer.java  
  
D:\jdk1.6\bin>java UDPEchoServer  
Created UDP Echo Server on port8000  
Received from server: computer
```

RESULT:

Thus the for simulating RARP protocols using UDP was executed successfully.

EX.NO:8 HALF DUPLEX CHAT USING TCP/IP

AIM:

To implement a chat server and client in java using TCP sockets in half duplex mode.

DESCRIPTION:

TCP Clients send requests to the server and the server will receive the request and response with acknowledgement. Every time either a client or a server can send and receive the messages

ALGORITHM:

Server

1. Create a server socket and bind it to the port.
2. Listen for new connections and when a connection arrives, accept it.
3. Read Client's message and display it
4. Get a message from user and send it to client
5. Repeat steps 3-4 until the client terminates
6. Close all streams
7. Close the server and client socket
8. Stop

Client:

1. Create a client socket and connect it to the server's port number
2. Get a message from user and send it to server
3. Read server's response and display it
4. Repeat steps 2-3 until chat is terminated with "exit" message
5. Close all input/output streams
6. Close the client socket
7. Stop

PROGRAM:

//Server

import java.io.*;

import java.net.*;

class Server_HalfDup {

public static void main(String args[])

throws Exception

{

// Create server Socket

ServerSocket ss = new ServerSocket(888);

// connect it to client socket

Socket s = ss.accept();

System.out.println("Connection established");

// to send data to the client

PrintStream ps

= new PrintStream(s.getOutputStream());

// to read data coming from the client

BufferedReader br

= new BufferedReader(
new InputStreamReader(
s.getInputStream()));

// to read data from the keyboard

BufferedReader kb = new BufferedReader(new
InputStreamReader(System.in));

// server executes continuously

while (true) {

String str, str1;

// repeat as long as the client

```

        // does not send a null string

        // read from client
        while ((str = br.readLine()) != null) {
            System.out.println("From Client:"+str);
            str1 = kb.readLine();

            // send to client
            ps.println(str1);
        }

        // close connection
        ps.close();
        br.close();
        kb.close();
        ss.close();
        s.close();

        // terminate application
        System.exit(0);

    } // end of while
}

```

//Client

```

import java.io.*;
import java.net.*;
class Client_HalfDup {

    public static void main(String args[])
        throws Exception
    {

        // Create client socket
        Socket s = new Socket("localhost", 888);
    }
}

```

```
// to send data to the server
DataOutputStream dos = new DataOutputStream(s.getOutputStream());

// to read data coming from the server
BufferedReader br
    = new BufferedReader(
        new InputStreamReader(
            s.getInputStream()));

// to read data from the keyboard
BufferedReader kb
    = new BufferedReader(
        new InputStreamReader(System.in));
String str, str1;

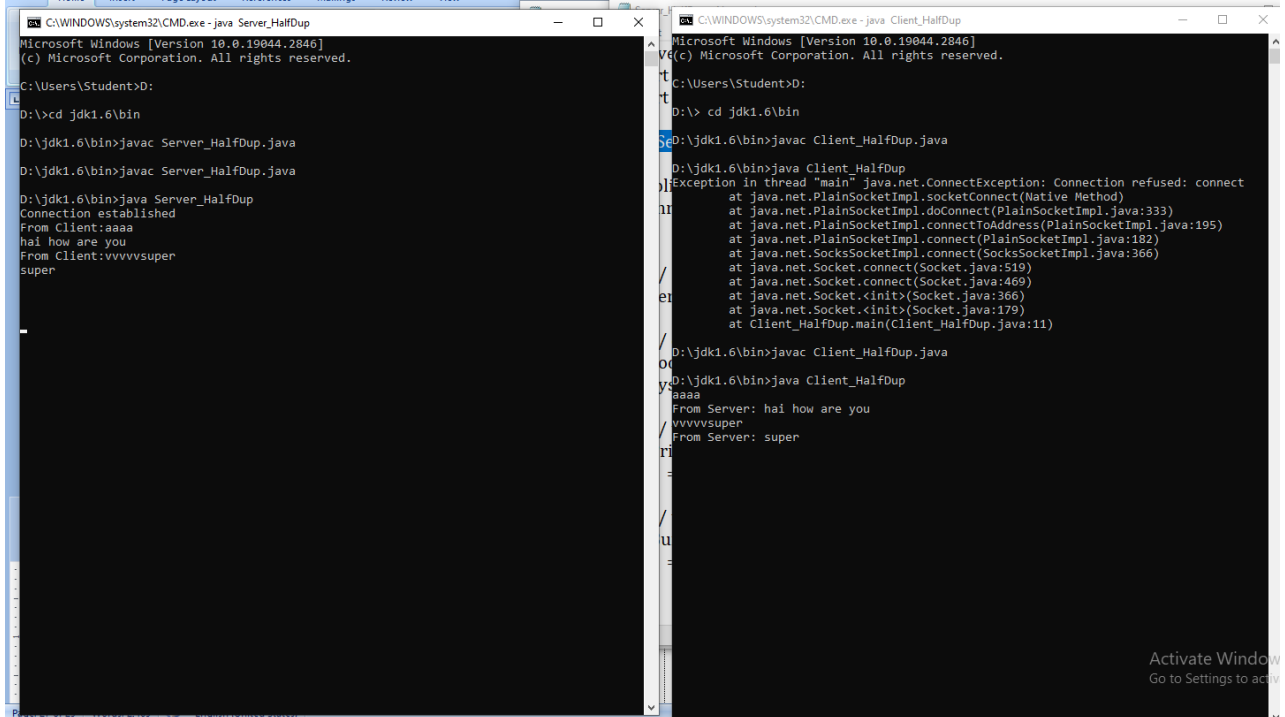
// repeat as long as exit
// is not typed at client
while (!(str = kb.readLine()).equals("exit"))
{
    // send to the server
    dos.writeBytes(str + "\n");

    // receive from the server
    str1 = br.readLine();

    System.out.println("From Server: "+str1);
}

// close connection.
dos.close();
br.close();
kb.close();
s.close();
}
}
```

OUTPUT:



```

C:\WINDOWS\system32\CMD.exe - java Server_HalfDup
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Student>D:

D:\>cd jdk1.6\bin

D:\jdk1.6\bin>javac Server_HalfDup.java

D:\jdk1.6\bin>javac Server_HalfDup.java

D:\jdk1.6\bin>java Server_HalfDup
Connection established
From Client:aaaa
hai how are you
From Client:vvvvsuper
super

C:\WINDOWS\system32\CMD.exe - java Client_HalfDup
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Student>D:

D:\> cd jdk1.6\bin

D:\jdk1.6\bin>javac Client_HalfDup.java

D:\jdk1.6\bin>java Client_HalfDup
Exception in thread "main" java.net.ConnectException: Connection refused: connect
    at java.net.PlainSocketImpl.socketConnect(Native Method)
    at java.net.PlainSocketImpl.doConnect(PlainSocketImpl.java:333)
    at java.net.PlainSocketImpl.connectToAddress(PlainSocketImpl.java:195)
    at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:182)
    at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:366)
    at java.net.Socket.connect(Socket.java:519)
    at java.net.Socket.connect(Socket.java:469)
    at java.net.Socket.<init>(Socket.java:366)
    at java.net.Socket.<init>(Socket.java:179)
    at Client_HalfDup.main(Client_HalfDup.java:11)

D:\jdk1.6\bin>javac Client_HalfDup.java

D:\jdk1.6\bin>java Client_HalfDup
aaaa
From Server: hai how are you
vvvvsuper
From Server: super

```

Activate Windows
Go to Settings to activate Windows.

EX.NO 9 FULL DUPLEX CHAT USING TCP/IP

AIM:

To implement a chat server and client in java using TCP sockets in half duplex mode.

DESCRIPTION:

TCP Clients send requests to the server and the server will receive the request and response with acknowledgement. Every time either a client or a server can send and receive the messages

ALGORITHM:

Server

1. Create a server socket and bind it to the port.
2. Listen for new connections and when a connection arrives, accept it.
3. Read Client's message and display it
4. Get a message from user and send it to client
5. Repeat steps 3-4 until the client terminates
6. Close all streams
7. Close the server and client socket
8. Stop

Client

1. Create a client socket and connect it to the server's port number
2. Get a message from user and send it to server
3. Read server's response and display it
4. Repeat steps 2-3 until chat is terminated with "exit" message
5. Close all input/output streams
6. Close the client socket
7. Stop

Server:

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

```
class Server2 {
```

```
public static void main(String args[])
    throws Exception
{

    // Create server Socket
    ServerSocket ss = new ServerSocket(888);

    // connect it to client socket
    Socket s = ss.accept();
    System.out.println("Connection established");

    // to send data to the client
    PrintStream ps
        = new PrintStream(s.getOutputStream());

    // to read data coming from the client
    BufferedReader br
        = new BufferedReader(
            new InputStreamReader(
                s.getInputStream()));

    // to read data from the keyboard
    BufferedReader kb
        = new BufferedReader(
            new InputStreamReader(System.in));

    // server executes continuously
    while (true) {

        String str, str1;

        // repeat as long as the client
        // does not send a null string

        // read from client
        while ((str = br.readLine()) != null) {
            System.out.println(str);
            str1 = kb.readLine();
```

```
        // send to client
        ps.println(str1);
    }

    // close connection
    ps.close();
    br.close();
    kb.close();
    ss.close();
    s.close();

    // terminate application
    System.exit(0);

} // end of while
}
```

// Client2 class that
// sends data and receives also

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

class Client2 {

    public static void main(String args[])
        throws Exception
    {

        // Create client socket
        Socket s = new Socket("localhost", 888);

        // to send data to the server
        DataOutputStream dos
            = new DataOutputStream(
                s.getOutputStream());
```

```
// to read data coming from the server
```

```
BufferedReader br
```

```
    = new BufferedReader(  
        new InputStreamReader(  
            s.getInputStream()));
```

```
// to read data from the keyboard
```

```
BufferedReader kb
```

```
    = new BufferedReader(  
        new InputStreamReader(System.in));
```

```
String str, str1;
```

```
// repeat as long as exit
```

```
// is not typed at client
```

```
while (!(str = kb.readLine()).equals("exit")) {
```

```
    // send to the server
```

```
    dos.writeBytes(str + "\n");
```

```
    // receive from the server
```

```
    str1 = br.readLine();
```

```
    System.out.println(str1);
```

```
}
```

```
// close connection.
```

```
dos.close();
```

```
br.close();
```

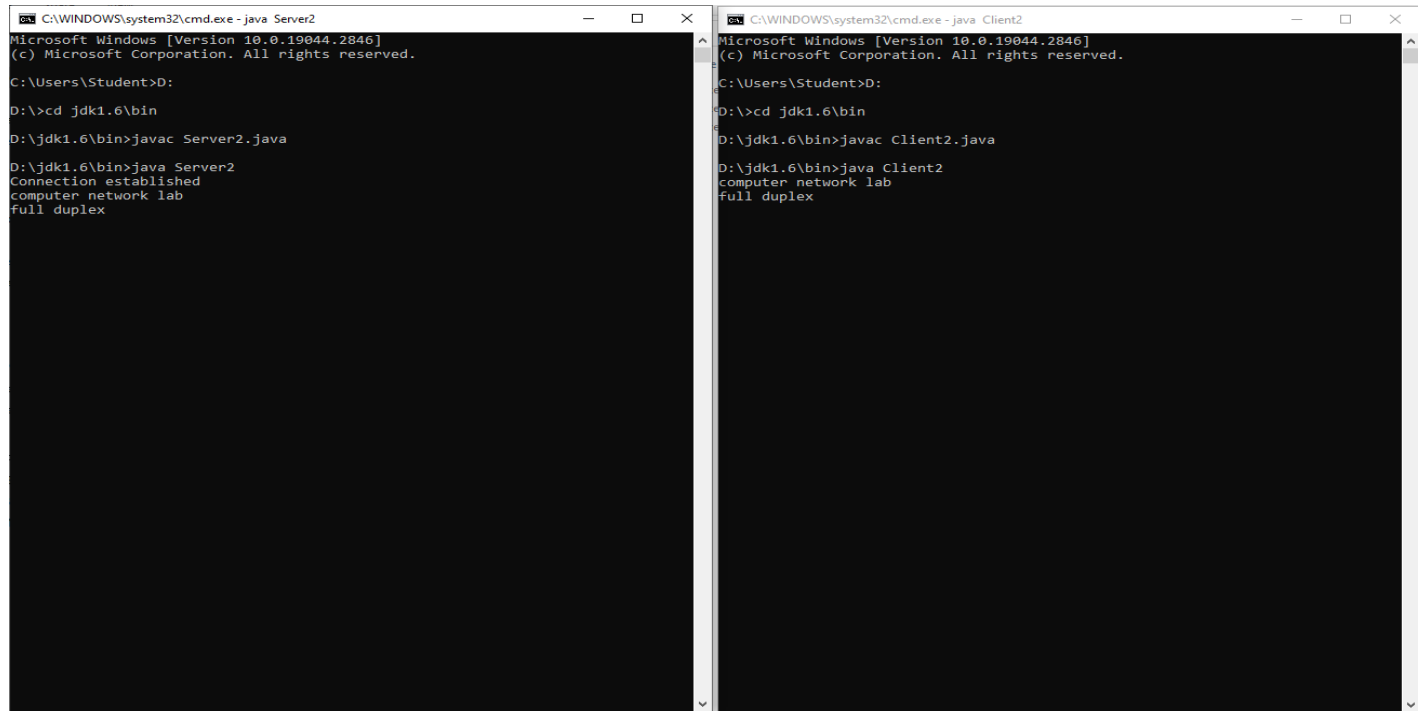
```
kb.close();
```

```
s.close();
```

```
}
```

```
}
```

OUTPUT:



The image shows two side-by-side Windows command prompt windows. The left window is titled 'C:\WINDOWS\system32\cmd.exe - java Server2' and the right window is titled 'C:\WINDOWS\system32\cmd.exe - java Client2'. Both windows show the same sequence of commands and output:

```
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Student>D:
D:\>cd jdk1.6\bin
D:\jdk1.6\bin>javac Server2.java
D:\jdk1.6\bin>java Server2
Connection established
computer network lab
full duplex
```

RESULT:

Thus the program for Full Duplex Chat Using TCP/IP was executed successfully.

Ex.No: 10 SIMULATION OF DISTANCE VECTOR/ LINK STATE ROUTING ALGORITHM.

AIM:

To simulate the Distance vector and link state routing protocols using NS2

ALGORITHM:

1. Create a Simulator object.
2. Set routing as dynamic.
3. Open the trace and nam trace files
4. Define the finish procedure.
5. Create nodes and the links between them.
6. Create the agents and attach them to the nodes
7. Create the applications and attach them to the udp agent.
8. Connect udp and null..
9. At 1 sec the link between node 1 and 2 is broken
10. At 2 sec the link is up again.
11. Run the simulation.

LINK STATE ROUTING PROTOCOL

PROGRAM

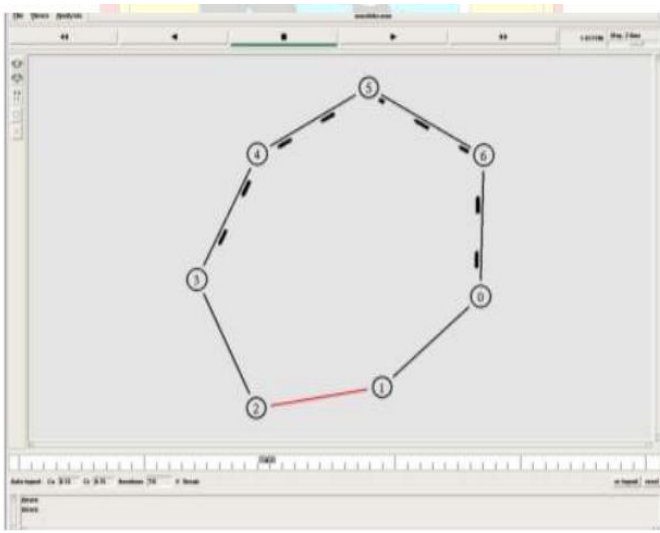
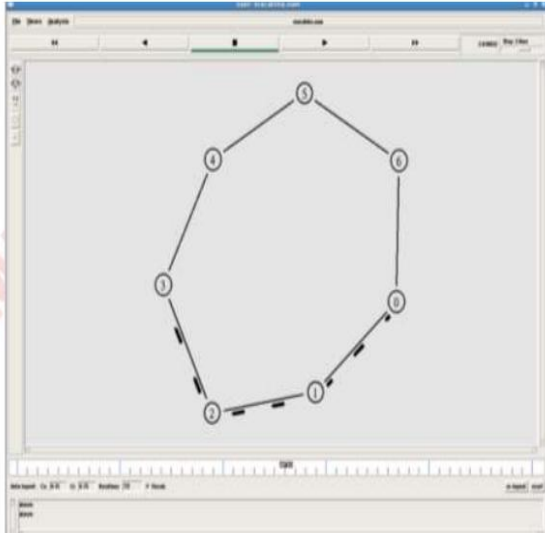
```
set ns [new Simulator]
$ns rtproto LS
set nf [open linkstate.nam w]
$ns namtrace-all $nf
set f0 [open linkstate.tr w]
$ns trace-all $f0
proc finish { } {
    global ns f0 nf
    $ns flush-trace
    close $f0
    close $nf
    exec nam linkstate.nam &
    exit 0
}
for {set i 0} {$i < 7} {incr i} {
    set n($i) [$ns node]
}
for {set i 0} {$i < 7} {incr i} {
```

```

$ns duplex-link $n($i) $n([expr ($i+1)%7]) 1Mb 10ms DropTail
}
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $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 $n(3) $null0
$ns connect $udp0 $null0
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run

```

OUTPUT:



EX.no 10(b) DISTANCE VECTOR ROUTING ALGORITHM

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 node
- s 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:

```
#Distance vector routing protocol – distvect.tcl
#Create a simulator object
set ns [new Simulator]
#Use distance vector routing
$ns rtproto DV
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
# Open tracefile
set nt [open trace.tr w]
$ns trace-all $nt
#Define 'finish' procedure
```

Downloaded from: annauniversityedu.blogspot.com

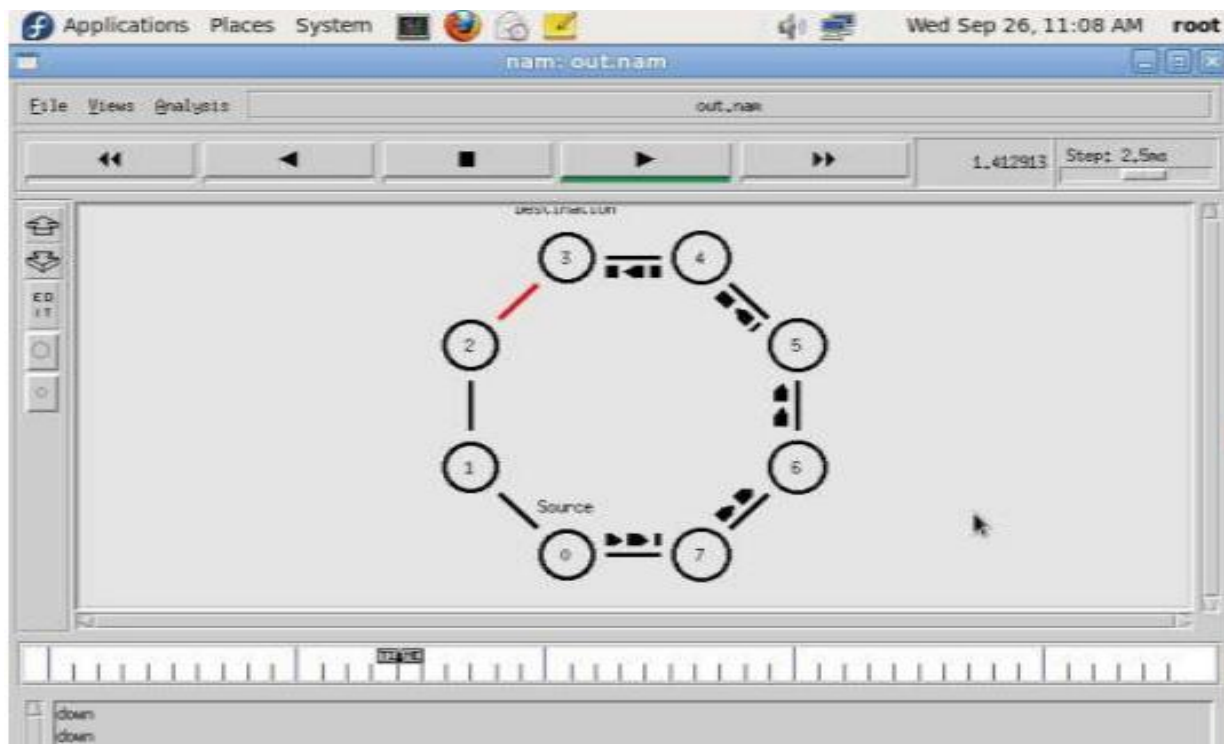
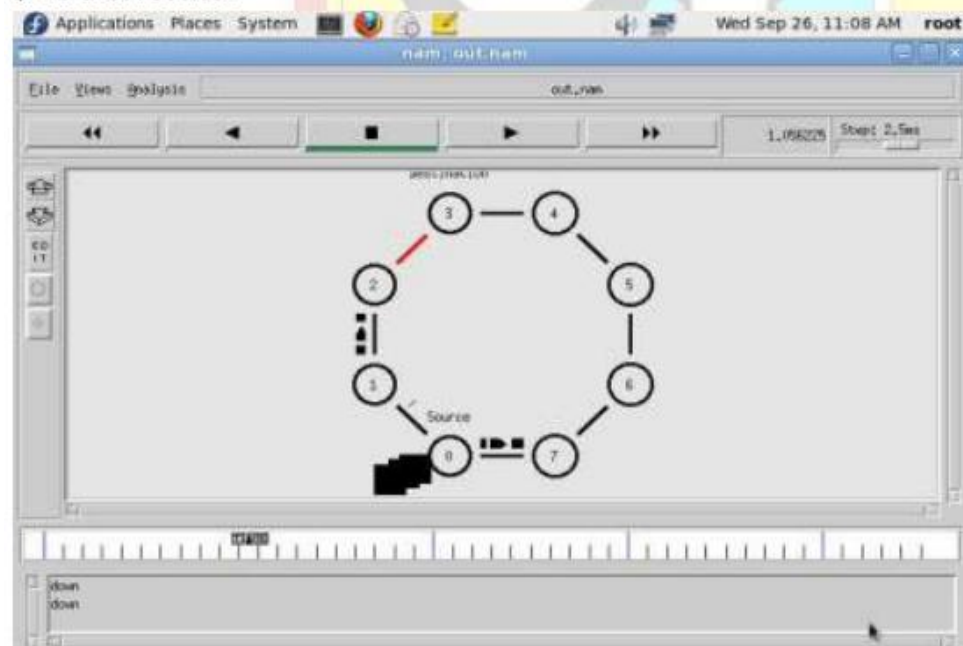
56

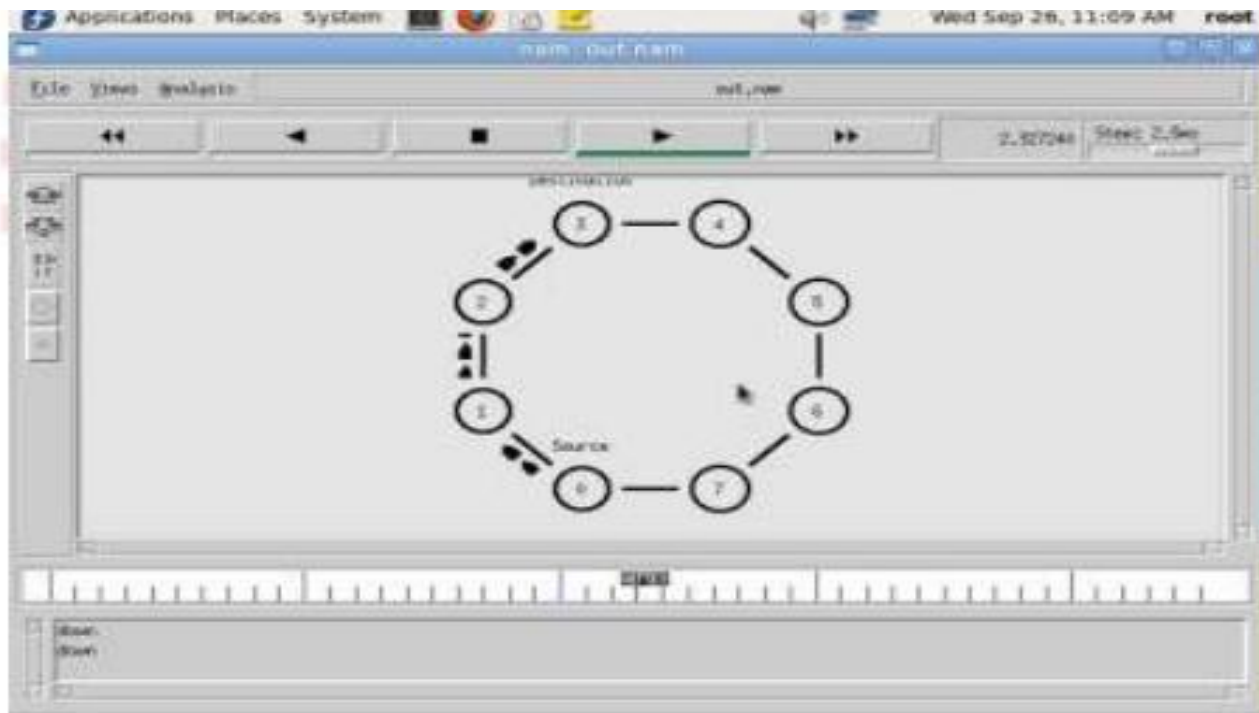
```
proc finish { }
{
  global ns nf
  $ns flush-trace
  #Close the trace file
  close $nf
  #Execute nam on the trace file
  exec nam -a out.nam &
  exit 0
}
# Create 8 nodes
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
# Specify link characteristics
$ns duplex-link $n1 $n2 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 $n5 $n6 1Mb 10ms DropTail
$ns duplex-link $n6 $n7 1Mb 10ms DropTail
$ns duplex-link $n7 $n8 1Mb 10ms DropTail
$ns duplex-link $n8 $n1 1Mb 10ms DropTail
# specify layout as a octagon
$ns duplex-link-op $n1 $n2 orient left-up
$ns duplex-link-op $n2 $n3 orient up
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n5 $n6 orient right-down
$ns duplex-link-op $n6 $n7 orient down
$ns duplex-link-op $n7 $n8 orient left-down
```

```
$ns duplex-link-op $n8 $n1 orient left
#Create a UDP agent and attach it to node n1
set udp0 [new Agent/UDP]
$ns attach-agent $n1 $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
#Create a Null agent (a traffic sink) and attach it to node n4
set null0 [new Agent/Null]
$ns attach-agent $n4 $null0
#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0
#Schedule events for the CBR agent and the network dynamics
$ns at 0.0 "$n1 label Source"
$ns at 0.0 "$n4 label Destination"
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n3 $n4
$ns rtmodel-at 2.0 up $n3 $n4
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

OUTPUT:

\$ ns distvect.tcl





The image shows the NS2 console output for a simulation. The interface is titled 'trace.tr - qedit'. The output displays a series of network events, including route updates and packet transmissions. The events are timestamped and include node IDs and protocol names. The output is as follows:

```

t 0.239031 4 2 rtProtoDv 0 ----- 0 4.1 3.2 -1 76
r 0.239031 4 3 rtProtoDv 0 ----- 0 4.1 3.1 -1 77
+ 0.27794 1 0 rtProtoDv 0 ----- 0 1.1 0.2 -1 78
- 0.27794 1 0 rtProtoDv 0 ----- 0 1.1 0.2 -1 78
+ 0.27794 1 2 rtProtoDv 0 ----- 0 1.1 2.1 -1 79
- 0.27794 1 2 rtProtoDv 0 ----- 0 1.1 2.1 -1 79
r 0.288004 1 0 rtProtoDv 0 ----- 0 1.1 0.2 -1 78
r 0.288004 1 2 rtProtoDv 0 ----- 0 1.1 2.1 -1 79
+ 0.399578 5 4 rtProtoDv 0 ----- 0 5.1 4.1 -1 80
- 0.399578 5 4 rtProtoDv 0 ----- 0 5.1 4.1 -1 80
+ 0.399578 5 6 rtProtoDv 0 ----- 0 5.1 6.1 -1 81
- 0.399578 5 6 rtProtoDv 0 ----- 0 5.1 6.1 -1 81
+ 0.408238 7 6 rtProtoDv 0 ----- 0 7.1 6.1 -1 83
- 0.408238 7 6 rtProtoDv 0 ----- 0 7.1 6.1 -1 83
+ 0.408238 7 0 rtProtoDv 0 ----- 0 7.1 0.2 -1 82
- 0.408238 7 0 rtProtoDv 0 ----- 0 7.1 0.2 -1 82
+ 0.408238 7 6 rtProtoDv 0 ----- 0 7.1 6.1 -1 83
- 0.408238 7 6 rtProtoDv 0 ----- 0 7.1 6.1 -1 83
r 0.409642 5 4 rtProtoDv 0 ----- 0 5.1 4.1 -1 80
r 0.409642 5 6 rtProtoDv 0 ----- 0 5.1 6.1 -1 81
r 0.418382 7 0 rtProtoDv 0 ----- 0 7.1 0.2 -1 82
r 0.418382 7 6 rtProtoDv 0 ----- 0 7.1 6.1 -1 83
+ 0.5 0 1 cbr 500 ----- 0 0.0 3.0 0 84
- 0.5 0 1 cbr 500 ----- 0 0.0 3.0 0 84
+ 0.505 0 1 cbr 500 ----- 0 0.0 3.0 1 85
- 0.505 0 1 cbr 500 ----- 0 0.0 3.0 1 85

```

RESULT:

Thus the program for simulate the Distance vector and link state routing protocols using NS2 was executed successfully.

Ex.no.11 PERFORMANCE EVALUATION OF ROUTING PROTOCOLS USING SIMULATION TOOL.

A) UNICAST ROUTING PROTOCOL

Aim:

To write a program for implementing unicast routing protocol.

Algorithm:

Step1: start the program.

Step2: declare the global variables ns for creating a new simulator.

Step3: set the color for packets.

Step4: open the network animator file in the name of file2 in the write mode.

Step5: open the trace file in the name of file 1 in the write mode.

Step6: set the unicast routing protocol to transfer the packet in network. Step7: create the required no of nodes.

Step8: create the duplex-link between the nodes including the delay time, bandwidth and dropping queue mechanism.

Step 9: give the position for the links between the nodes.

Step10: set a tcp reno connection for source node.

Step11: set the destination node using tcp sink.

Step12: setup after connection over the tcp connection.

Step13: down the connection between any nodes at a particular time.

Step14: reconnect the downed connection at a particular time.

Step15: define the finish procedure.

Step16: in the definition of the finish procedure declare the global variables ns, file1, file2.

Step17: close the tracefile and namefile and execute the network animation file.

Step18:at
theparticartimecallthefinishprocedure.

Step19: stop the program.

PROGRAM:

setns [new Simulator]

#Definedifferent colors fordata flows (forNAM)

\$nscolor1 Blue

\$nscolor 2 Red

#OpentheTracefil

esetfile1[openout

.trw]

\$nstrace-all\$file1

Open the NAM

trace

filesetfile2[openout

.namw]

\$nsnamtrace-all\$file2

#Definea'finish'proc

edureprocfinish {} {

globalnsfile1file2

\$ns flush-

traceclose

\$file1close

\$file2

execnamout.na

m&exit 3

}

#Next line should becommented out tohavethestatic routing

\$nsrtproto DV

#Create six

nodeset n0

[\$ns

node]set n1

[\$ns

```
node]set n2
[$ns
node]set n4
[$ns
node]set n4
[$ns
node]set n5
[$ns node]
```

```
#Createlinksbetweenthe nodes
```

```
$nsduplex-link $n0 $n1 10.3Mb 10ms DropTail
$nsduplex-link $n1 $n2 20.3Mb 10ms DropTail
$nsduplex-link $n2 $n3 30.3Mb 10ms DropTail
$nsduplex-link $n1 $n4 40.3Mb 10ms DropTail
$nsduplex-link $n3 $n5 50.5Mb 10ms DropTail
$nsduplex-link $n4 $n5 50.5Mb 10ms DropTail
```

```
#Givenodeposition (for NAM)
```

```
$nsduplex-link-op $n0 $n1 orient right
$nsduplex-link-op $n1 $n2 orient right
$nsduplex-link-op $n2 $n3 orient up
$nsduplex-link-op $n1 $n4 orient up-left
$nsduplex-link-op $n3 $n5 orient left-up
$nsduplex-link-op $n4 $n5 orient right-up
```

```
#Set up a TCP connection
```

```
settcp[newAgent/TCP/Newreno]
$nsattach-agent $n0 $tcp
setsink[newAgent/TCP/Sink/DelAck]
$nsattach-agent $n5 $sink
$nsconnect $tcp $sink
$tcpset fid_1
```

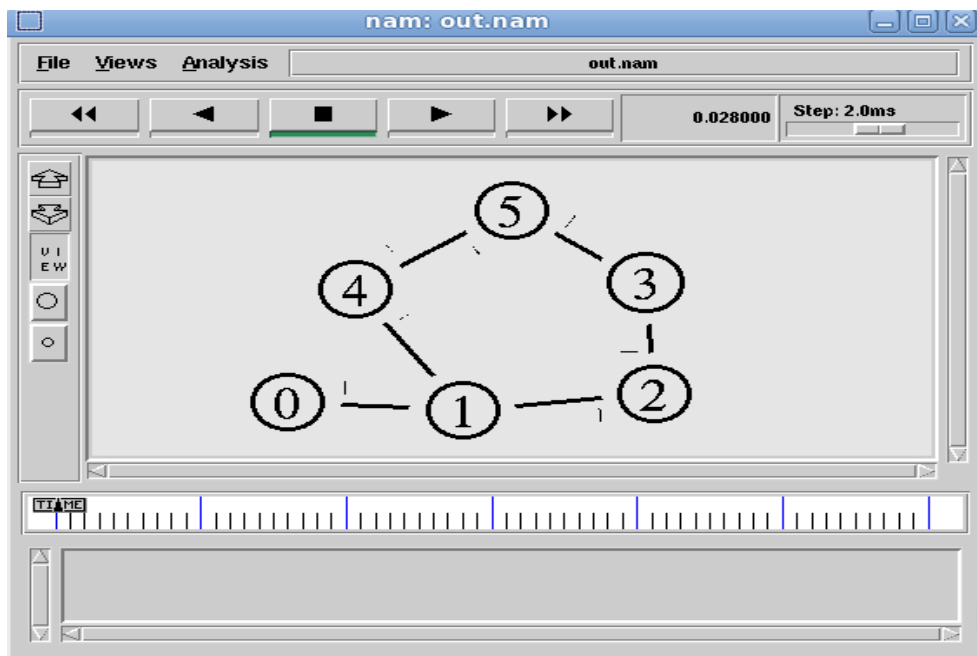
```
#SetupaFTPOverTCPconn  
ectionsetftp  
[newApplication/FTP]  
$ftpattach-agent$tcp  
$ftpsettype_FTP
```

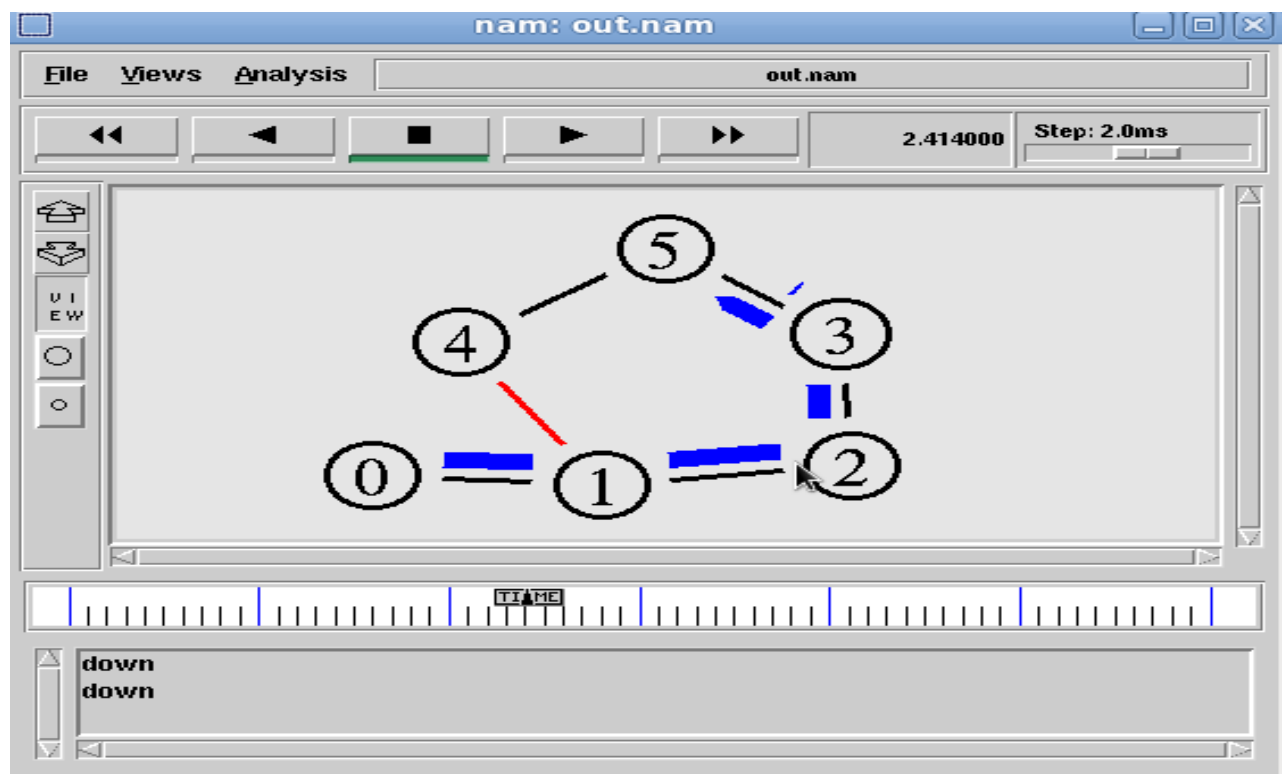
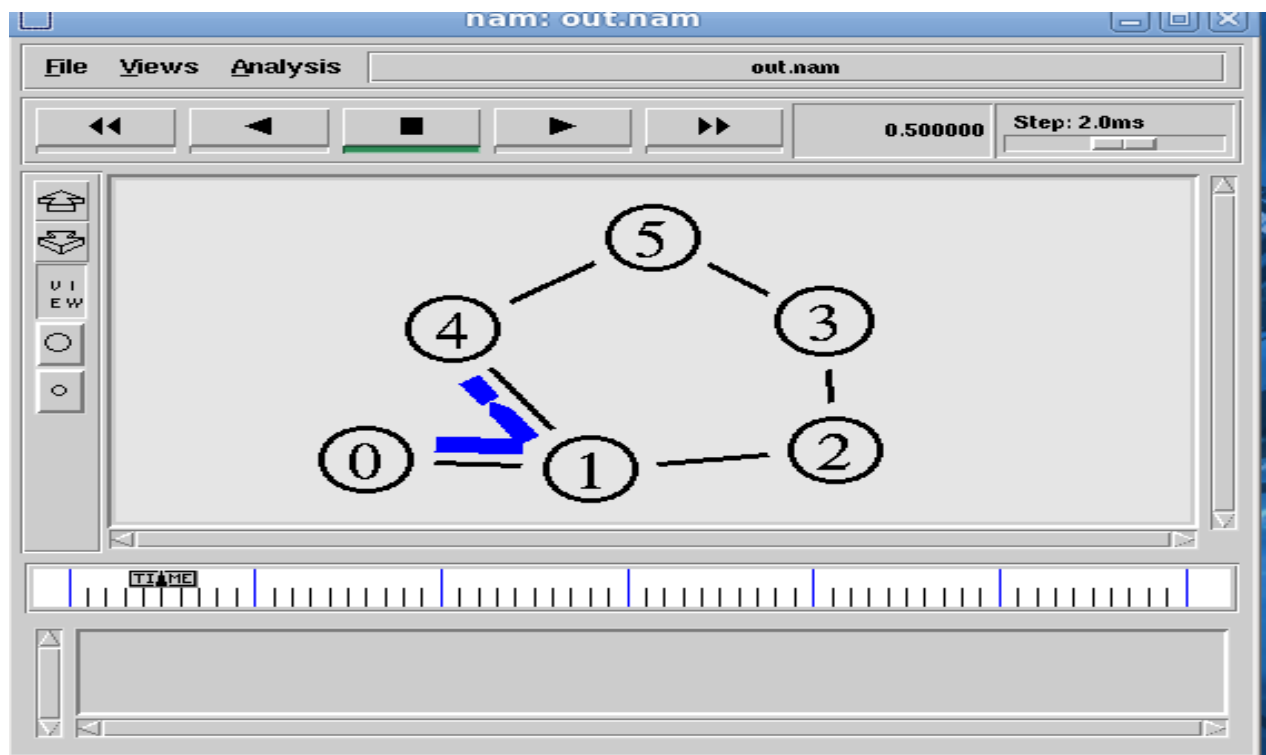
```
$nsrtmodel-at 1.0 down$n1 $n4  
$nsrtmodel-at 4.5 up$n1 $n4
```

```
$nsat0.1 "$ftp start"
```

```
$nsat 6.0 "finish"
```

```
$nsrun
```





Ex.no.11(B).MULTICASTINGROUTINGPROTOCOL

Aim:

To write ans2 program forimplementingmulticastingroutingprotocol.

Algorithm:

Step1:starttheprogram.

Step2:declarethe

globalvariablesnsforcreatinganew

simulator.Step3: set thecolor forpackets.

Step4:open thenetworkanimatorfileinthenameof file2in

thewritemode.Step5: open the tracefilein thename of file 1in the writemode.

Step6:set themulticast routingprotocolto transferthepacketsin network.Step7: create themulticast capable no ofnodes.

Step8: create theduplex-link between the nodes includingthe delaytime,bandwidth and dropping queuemechanism.

Step9: give theposition for thelinks between the nodes.

Step10:set audpconnection forsourcenode.

Step 11: set the destination node ,port and random false for the source and destination files.Step12: setup atrafficgenerator CBR forthe sourceanddestination files.

Step13: downtheconnectionbetweenanynodesat aparticular time.

Step14: createthereceiveagentfor joiningand leavingifthenodes inthegroup.Step15: definethe finish procedure.

Step 16: in the definition of the finish procedure declare the global variables.Step17: closethe tracefile andnamefileandexecute thenetworkanimationfile.Step18: at theparticular time call the finish procedure.

Step19:stoptheprogram.

Program:

```
#Createscheduler
```

```
#Create an event scheduler wit  
multicast turned onsets [new  
Simulator -multicast on]
```

```
#$ns
multicast#Tu
rnonTracing
settf[open output.trw]
$ns trace-all $tf
```

```
#TurnonnamTracin
gsetfd[openmcast.n
amw]
$ns namtrace-all $fd
```

```
#      Create
nodeset 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]
```

```
#Createlinks
$ns duplex-link $n0 $n1 21.5Mb 10ms DropTail
```

```
$nsduplex-link $n1 $n2 1.5Mb 10ms DropTail
$nsduplex-link $n2 $n3 1.5Mb 10ms DropTail
$nsduplex-link $n3 $n4 1.5Mb 10ms DropTail
$nsduplex-link $n3 $n7 1.5Mb 10ms DropTail
$nsduplex-link $n4 $n5 1.5Mb 10ms DropTail
$nsduplex-link $n4 $n6 1.5Mb 10ms DropTail
```

```
#Routingprotocol:saydistanc
evector#Protocols: CtrMcast,
DM, ST, BSTsetmproto DM
setmrthandle[$ns mrtproto $mproto{ }]
```

```
# Allocate group
addressessetgroup1[
Nodeallocaddr]setgr
oup2[Nodeallocaddr
]
```

```
#UDPTTransportagentforthetraffi
sourcesetudp0 [new
Agent/UDP]
$nsattach-agent$n0 $udp0
$udp0set dst_addr_$group1
$udp0 set dst_port_ 0
setcbr1[newApplication/Traffic/CBR]
$cbr1attach-agent$udp0
```

```
#Transportagentforthetraffic
sourcesetudp1 [new
Agent/UDP]
$nsattach-agent$n1 $udp1
$udp1set dst_addr_$group2
$udp1set dst_port_ 0
setcbr2[newApplication/Traffic/CBR]
$cbr2attach-agent$udp1
```

```
#Createreceiver
setrcvr1[newAgent/Null]
```


\$nsattach-agent\$N5\$rcvr1
\$nsat1.0"\$N5join-
group\$rcvr1\$group1"setrcvr2
[new Agent/Null]
\$nsattach-agent\$N6\$rcvr2
\$nsat1.5"\$N6join-
group\$rcvr2\$group1"setrcvr3
[new Agent/Null]
\$nsattach-agent\$N7\$rcvr3
\$nsat2.0"\$N7join-
group\$rcvr3\$group1"setrcvr4
[new Agent/Null]
\$nsattach-agent\$N5\$rcvr1
\$nsat2.5"\$N5join-
group\$rcvr4\$group2"setrcvr5
[new Agent/Null]
\$nsattach-agent\$N6\$rcvr2
\$nsat3.0"\$N6join-
group\$rcvr5\$group2"setrcvr6
[new Agent/Null]
\$nsattach-agent\$N7\$rcvr3

\$nsat3.5"\$N7ioin-groun\$rcvr6\$groun2"
\$nsat4.0"\$N5leave-group\$rcvr1 \$group1"
\$nsat4.5"\$N6leave-group\$rcvr2 \$group1"
\$nsat5.0"\$N7leave-group\$rcvr3 \$group1"
\$nsat5.5"\$N5leave-group\$rcvr4 \$group2"
\$nsat6.0"\$N6leave-group\$rcvr5 \$group2"
\$nsat6.5"\$N7leave-group\$rcvr6 \$group2"

#Scheduleevents

\$nsat 0.5"\$cbr1 start"
\$nsat 9.5"\$cbr1 stop"
\$nsat 0.5"\$cbr2 start"
\$nsat 9.5"\$cbr2 stop"

#post-processing

\$ns at 10.0

```
"finish"procfi  
nish { } {
```

```
globalnstf  
$ns flush-  
traceclose$  
tf  
exec      nam  
mcast.nam &exit  
0  
}
```

```
#Fornam  
#Colorsforpackets fromtwomcast groups  
$nscolor10red  
$nscolor 11green  
$nscolor 30purple  
$nscolor31 green
```

```
#Manuallayout:orderofthelinkissign  
ificant!#$nsduplex-link-op $n0$n1  
orientright  
#$ns duplex-link-op $n0 $n2  
orient right-up#$nsduplex-link-  
op$n0$n3orientright-down#Show  
queueon simplexlink n0->n1  
#$nsduplex-link-op $n2$n3 queuePos 0.5
```

```
#Group0 source  
$udp0set fid_ 10  
$n0colorred  
$n0label "Source1"
```

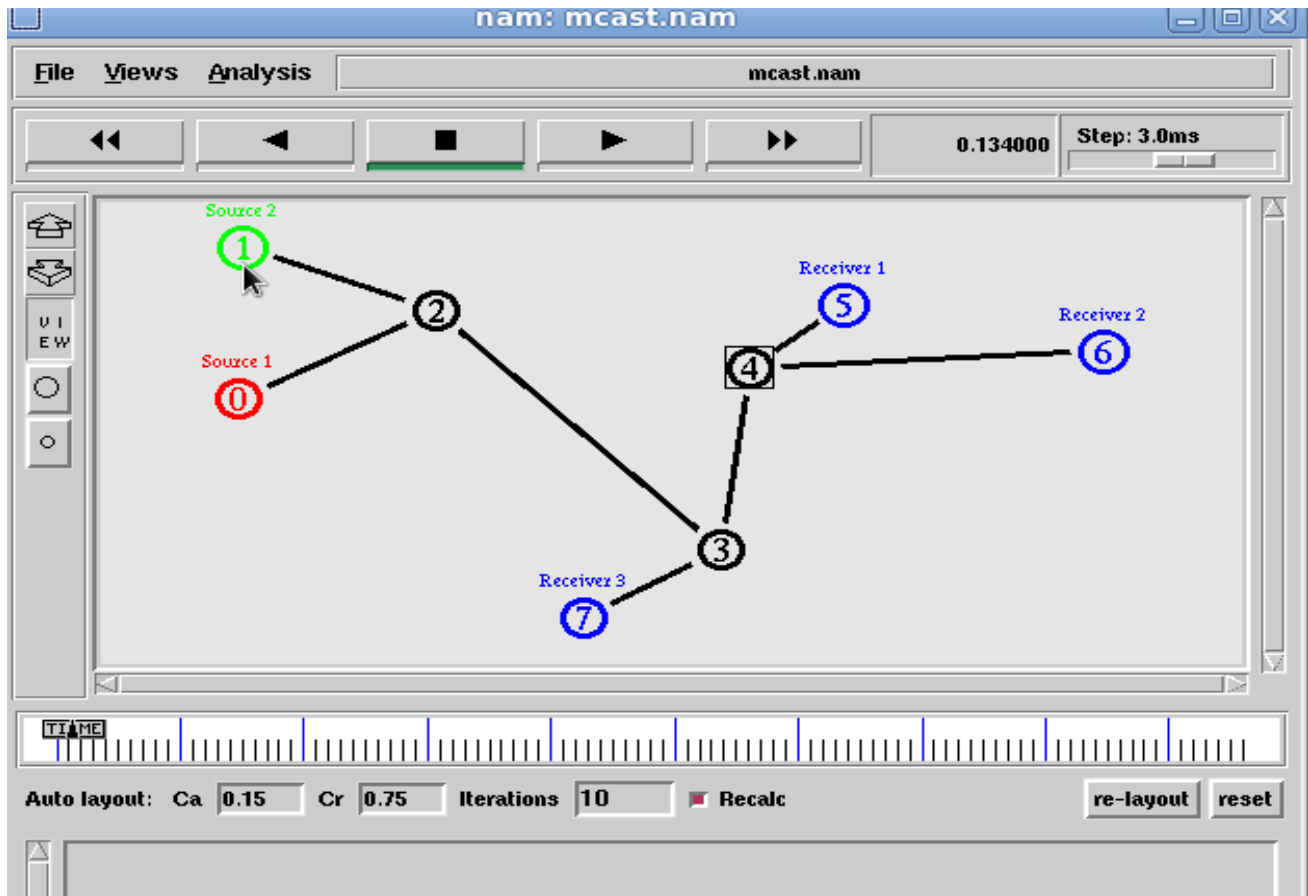
```
#Group1 source  
$udp1set fid_ 11  
$n1color green
```

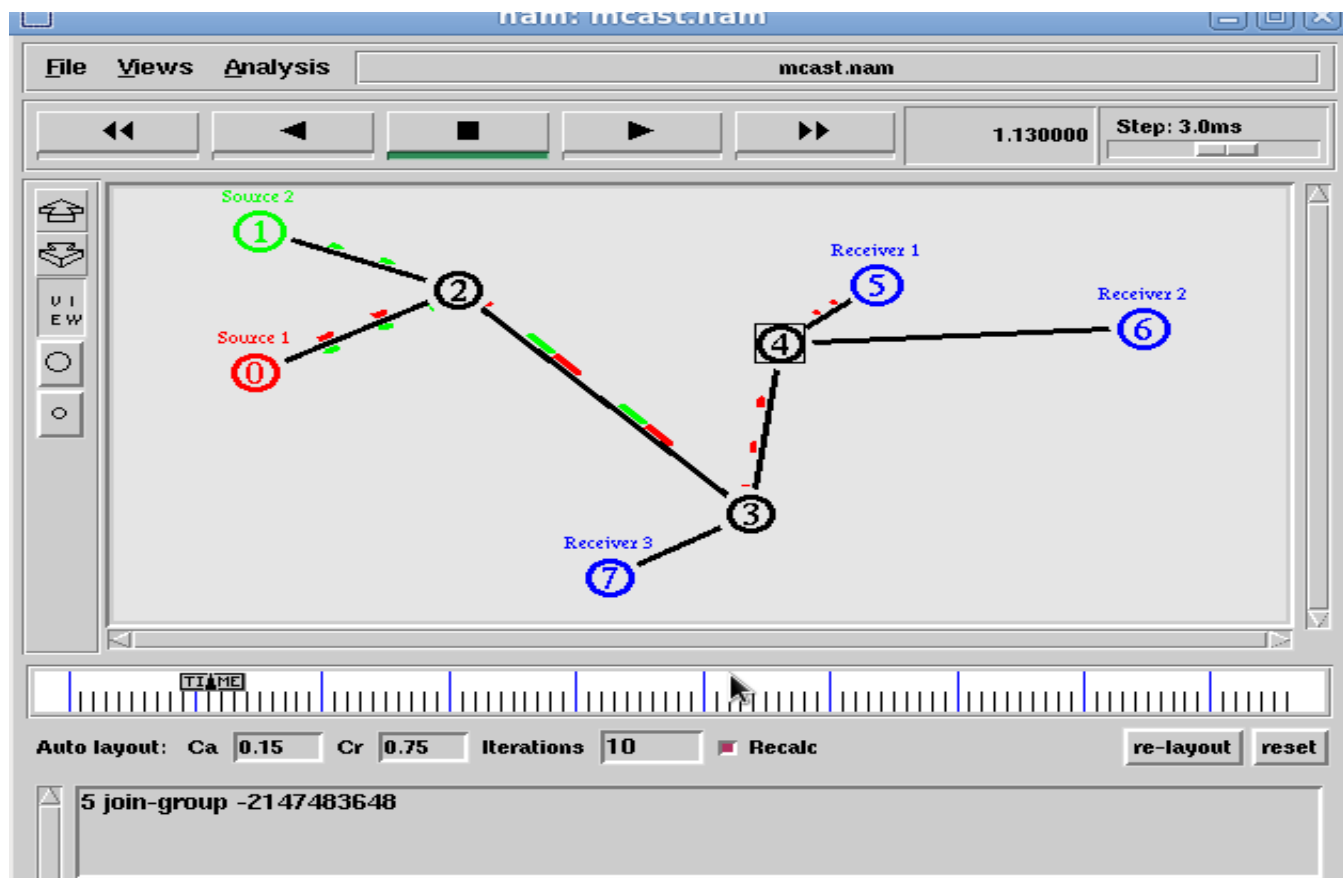
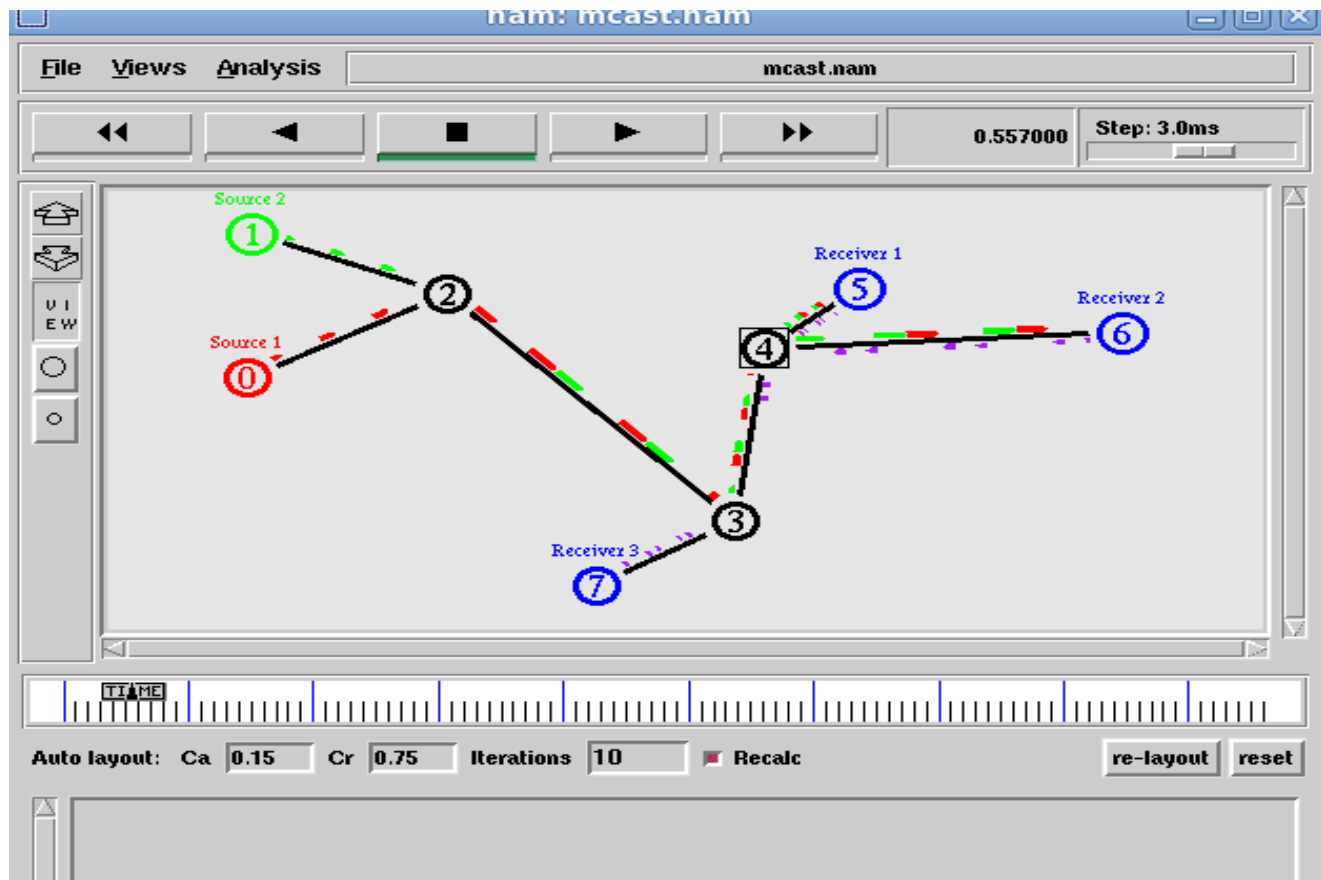
```
$n1label "Source2"  
$n5label "Receiver1"  
$n5color blue  
$n6label "Receiver2"  
$n6color blue  
$n7label "Receiver3"  
$n7color blue
```

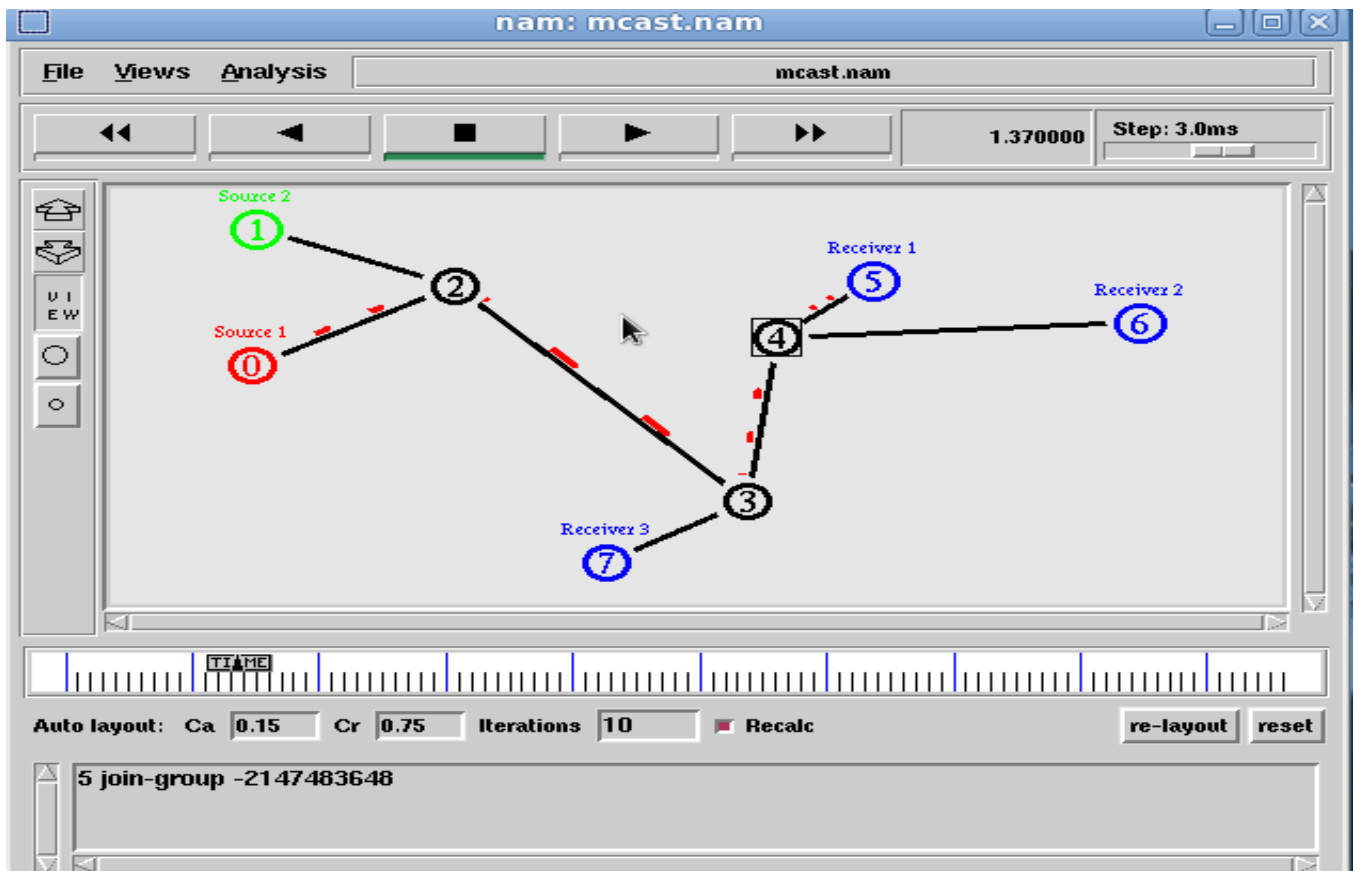
```
#$n2add-  
markm0red#$n2  
delete-markm0"
```

```
#Animationrate  
$nsset-animation-rate3.0ms  
$nsrun
```

OUTPUT:







RESULT:

Thus the program for performance evaluation of routing protocols using simulation tool.

Ex.no 12 SIMULATION OF ERROR CORRECTION CODE (LIKE CRC).

AIM:

To implement error checking code using java.

ALGORITHM:

1. Start the Program
2. Given a bit string, append 0s to the end of it (the number of 0s is the same as the degree of the generator polynomial) let $B(x)$ be the polynomial corresponding to B.
3. Divide $B(x)$ by some agreed on polynomial $G(x)$ (generator polynomial) and determine the remainder $R(x)$. This division is to be done using Modulo 2 Division.
4. Define $T(x) = B(x) - R(x)$
5. $(T(x)/G(x) \Rightarrow \text{remainder } 0)$
6. Transmit T, the bit string corresponding to $T(x)$.
7. Let T' represent the bit stream the receiver gets and $T'(x)$ the associated polynomial. The receiver divides $T1(x)$ by $G(x)$. If there is a 0 remainder, the receiver concludes $T = T'$ and no error occurred otherwise, the receiver concludes an error occurred and requires a retransmission
8. Stop the Program

PROGRAM:

```
//package javaTpoint.MicrosoftJava;
import java.util.*;
// create CRCEXample class to demonstrate the working of Cyclic Redundancy
Check
class CRCEXample {
    // main() method start
    public static void main(String args[]) {
        // create scanner class object to take input from user
        Scanner scan = new Scanner(System.in);
        // declare n for the size of the data
        int size;
        // take the size of the data from the user
```

```

System.out.println("Enter the size of the data array: ");
size = scan.nextInt();
// declaration of the data array
int data[] = new int[size];
// take bits of the data from the user
System.out.println("Enter data bits in the array one by one: ");
for(int i = 0 ; i < size ; i++) {
    System.out.println("Enter bit " + (size-i) + ":");
    data[i] = scan.nextInt();
}
// take the size of the divisor from the user
System.out.println("Enter the size of the divisor array:");
size = scan.nextInt();
// declaration of the divisor array
int divisor[] = new int[size];
System.out.println("Enter divisor bits in the array one by one: ");
for(int i = 0 ; i < size ; i++) {
    System.out.println("Enter bit " + (size-i) + ":");
    divisor[i] = scan.nextInt();
}
// Divide the input data by the input divisor and store the result in the rem array
int rem[] = divideDataWithDivisor(data, divisor);
// iterate rem using for loop to print each bit
for(int i = 0; i < rem.length-1; i++) {
    System.out.print(rem[i]);
}
System.out.println("\nGenerated CRC code is: ");

for(int i = 0; i < data.length; i++) {
    System.out.print(data[i]);
}
for(int i = 0; i < rem.length-1; i++) {
    System.out.print(rem[i]);
}
System.out.println();
// we create a new array that contains the original data with its CRC code
// the size of the sendData array with be equal to the sum of the data and the

```

rem arrays length

```
int sendData[] = new int[data.length + rem.length - 1];
System.out.println("Enter bits in the array which you want to send: ");
for(int i = 0; i < sendData.length; i++) {
    System.out.println("Enter bit " + (sendData.length - 1) + ":");
    sendData[i] = scan.nextInt();
}
receiveData(sendData, divisor);
}
// create divideDataWithDivisor() method to get CRC
static int[] divideDataWithDivisor(int oldData[], int divisor[]) {
    // declare rem[] array
    int rem[] = new int[divisor.length];
    int i;
    int data[] = new int[oldData.length + divisor.length];
    // use system's arraycopy() method for copying data into rem and data arrays
    System.arraycopy(oldData, 0, data, 0, oldData.length);
    System.arraycopy(data, 0, rem, 0, divisor.length);
    // iterate the oldData and exor the bits of the remainder and the divisor
    for(i = 0; i < oldData.length; i++) {
        System.out.println((i+1) + ".) First data bit is : "+ rem[0]);
        System.out.print("Remainder : ");
        if(rem[0] == 1) {
            // We have to exor the remainder bits with divisor bits
            for(int j = 1; j < divisor.length; j++) {
                rem[j-1] = exorOperation(rem[j], divisor[j]);
                System.out.print(rem[j-1]);
            }
        }
        else {
            // We have to exor the remainder bits with 0
            for(int j = 1; j < divisor.length; j++) {
                rem[j-1] = exorOperation(rem[j], 0);
                System.out.print(rem[j-1]);
            }
        }
    }
    // The last bit of the remainder will be taken from the data
```



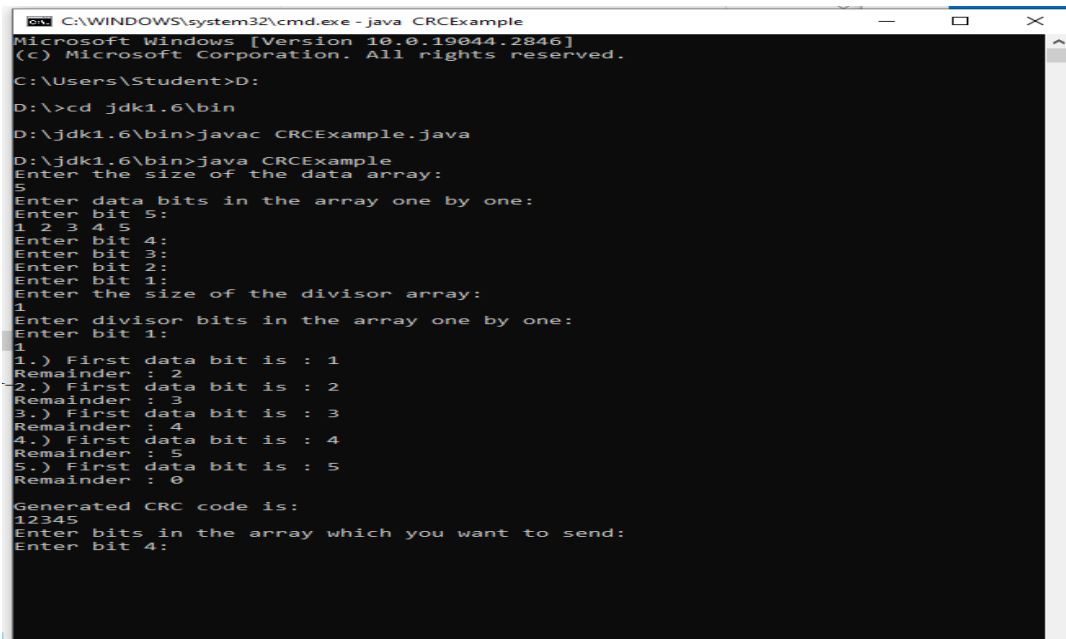
```

        // This is the 'carry' taken from the dividend after every step
        // of division
        rem[divisor.length-1] = data[i+divisor.length];
        System.out.println(rem[divisor.length-1]);
    }
    return rem;
}
// create exorOperation() method to perform exor data
static int exorOperation(int x, int y) {
    // This simple function returns the exor of two bits
    if(x == y) {
        return 0;
    }
    return 1;
}
// method to print received data
static void receiveData(int data[], int divisor[]) {

    int rem[] = divideDataWithDivisor(data, divisor);
    // Division is done
    for(int i = 0; i < rem.length; i++) {
        if(rem[i] != 0) {
            // if the remainder is not equal to zero, data is corrupted
            System.out.println("Corrupted data received...");
            return;
        }
    }
    System.out.println("Data received without any error.");
}
}

```

OUTPUT:



```
C:\WINDOWS\system32\cmd.exe - java CRCEXample
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Student>D:
D:\>cd jdk1.6\bin
D:\jdk1.6\bin>javac CRCEXample.java
D:\jdk1.6\bin>java CRCEXample
Enter the size of the data array:
5
Enter data bits in the array one by one:
Enter bit 5:
1 2 3 4 5
Enter bit 4:
Enter bit 3:
Enter bit 2:
Enter bit 1:
Enter the size of the divisor array:
1
Enter divisor bits in the array one by one:
Enter bit 1:
1
1.) First data bit is : 1
Remainder : 2
2.) First data bit is : 2
Remainder : 3
3.) First data bit is : 3
Remainder : 4
4.) First data bit is : 4
Remainder : 5
5.) First data bit is : 5
Remainder : 0

Generated CRC code is:
12345
Enter bits in the array which you want to send:
Enter bit 4:
```

RESULT:

Thus the program for Simulation of error correction code (like CRC) was executed successfully.