JEPPIAAR INSTITUTE OF TECHNOLOGY

SELF BELIEF | SELF DISCIPLINE | SELF RESPECT

KUNNAM, SUNGUVARCHATRAM, SRIPERUMPUDHUR, CHENNAI - 631 604

DEPARTMENT OF INFORMATION TECHNOLOGY

CS8581-NETWORKS LABORATORY

MANUAL

NAME	**:**
REG NO	**:**
YEAR	**: III IT**
SEMESTER	**: 05**

JEPPIAAR INSTITUTE OF TECHNOLOGY

SELF BELIEF | SELF DISCIPLINE | SELF RESPECT

KUNNAM, SUNGUVARCHATRAM, SRIPERUMBUDUR, CHENNAI - 631 604

BONAFIDE CERTIFICATE

This is a certified Bonafide Record Work of Mr./Ms.____________________ Register No.________________ submitted for the Anna University Practical Examination held on________in **CS8581- NETWORKS LABORATORY** during the year 2022-2023.

Signature of the Lab In-charge   Head of the Department

Internal Examiner   **External Examiner**

Date _____

INSTITUTE VISION

Jeppiaar Institute of Technology aspires to provide technical education in futuristic technologies with the perspective of innovative, industrial and social application for the betterment of humanity.

INSTITUTE MISSION

- To produce competent and disciplined high-quality professionals with the practical skills necessary to excel as innovative professionals and entrepreneurs for the benefit of the society.
- To improve the quality of education through excellence in teaching and learning, research, leadership and by promoting the principles of scientific analysis, and creativethinking.
- To provide excellent infrastructure, serene and stimulating environment that is mostconducive to learning.
- To strive for productive partnership between the Industry and the Institute for researchand development in the emerging fields and creating opportunities for employability.
- To serve the global community by instilling ethics, values and life skills among the students needed to enrich their lives.

DEPARTMENT VISION

The department will be an excellent centre to impart futuristic and innovative technological education to facilitate the evolution of problem-solving skills along withknowledge application in the field of Information Technology, understanding industrial and global requirements and societal needs for the benefit of humanity.

DEPARTMET MISSION

- Produce competent and high-quality professional computing graduates in software developmentconsidering global requirements and societal needs thereby maximizing employability.
- Enhance evolution of professional skills and development of leadership traits among the studentsby providing favorable infrastructure and environment to grow into successful entrepreneurs.
- Training in multidisciplinary skills needed by Industries, higher educational institutions, research establishments and Entrepreneurship.
- Impart Human Values and Ethical Responsibilities in professional activities.

PEO'S

- To provide students with a fundamental knowledge in Science, mathematics and computing skills for creative and innovative application.
- To enable students competent and employable by providing excellent Infrastructure to learn andcontribute for the welfare of the society.
- To channelize the potentials of the students by offering state of the art amenities to undergo research and higher education.
- To evolve computing engineers with multi-disciplinary understanding and maximize Job Opportunities.
- To facilitate students, obtain profound understanding nature and social requirements and grow asprofessionals with values and integrity

PROGRAM OUTCOMES (PO'S):

Engineering Graduates will be able to:

1. **Engineering knowledge**: (K3) Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: (K4) Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: (K4) Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: (K5) Use research- based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of theinformation to provide valid conclusions.
5. **Modern tool usage**:(K3, K5, K6) Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineeringactivities with an understanding of the limitations.
6. **The engineer and society**: (A3) Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: (A2) Understand the impact of the professional engineering solutions in societal and environmentalcontexts, and demonstrate the knowledge of, and need for sustainableDevelopment.

8. **Ethics**: (A3) Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: (A3) Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** (A3) Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: (A3) Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, asa member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: (A2) recognize the need for and have the preparation and ability to engage in independent and life-long learningin the broadest context of technologicalchange.

PSO:

- Students are able to analyse, design, implement and test any software with the programming and testing skills they have acquired.
- Students are able to design and develop algorithms for real time problems, scientific and business applications through analytical,logical and problems solving skills.
- Students are able to provide security solution for network components and data storage and management which will enablethem to work efficiently in the industry.

CO-PO MAPPING

Subject Code & Name: CS8581 Networks Laboratory Department: IT Year/Sem: III/05

After successful completion of the course, the students should be able to

Course Outcome No.	Course Outcome	Highest Cognitive Level
CO308.1	Devise various protocols using TCP and UDP	K3
CO308.2	Compare the performance of different transport layer protocols	K3
CO308.3	Use simulation tools to analyze the performance of various network protocols	K3
CO308.4	Analyze various routing algorithms	K3
CO308.5	Implement error correction codes	K3
CO308.6	Illustrate Network simulator (NS) and Simulate Congestion Control Algorithms using NS	A3
CO308.7	Exhibit ethical principles in engineering practices	A3
CO308.8	Perform task as an individual and / or team member to manage the task in time	A3
CO308.9	Express the Engineering activities with effective presentation and report.	A3
CO3O8.10	Interpret the findings with appropriate technological / research citation.	A2

CO & PO and PSO Mapping

Course No.	Level of CO	Program Outcomes												Program Specific Outcomes		
		K3	K4	K4	K5	K3,K5, K6	A3	A2	A3	A3	A3	A3	A2	PSO-1	PSO-2	PSO-3
		PO-1	PO-2	PO-3	PO-4	PO-5	PO-6	PO-7	PO-8	PO-9	PO-10	PO-11	PO-12			
CO308.1	K3	3	2	2	2	3	-	-	-	-	-	-	-	-	-	2
CO308.2	K3	3	2	2	2	3	-	-	-	-	-	-	-	-	-	2
CO308.3	K3	3	2	2	2	3	-	-	-	-	-	-	-	-	-	2
CO308.4	K3	3	2	2	2	3	-	-	-	-	-	-	-	-	-	2
CO308.5	K3	3	2	2	2	3	-	-	-	-	-	-	-	-	-	2
CO308.6	A3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2
CO308.7	A3	-	-	-	-	-	-	-	3	-	-	-	-	-	-	-
CO308.8	A3	-	-	-	-	-	-	-	-	3	-	3	-	-	-	-
CO308.9	A3	-	-	-	-	-	-	-	-	-	3	-	-	-	-	-
CO308.10	A2	-	-	-	-	-	-	-	-	-	-	-	3	-	-	-

ANNA UNIVERSITY –SYLLABUS

REGULATION -2017

CS8581 -NETWORKS LABORATORY

LIST OF EXPERIMENTS

1. Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute. Capture ping and traceroute PDUs using a network protocol analyzer and examine.
2. Write a HTTP web client program to download a web page using TCP sockets.
3. Applications using TCP sockets like:
   a) Echo client and echo server
   b) Chat
   c) File Transfer
4. Simulation of DNS using UDP sockets.
5. Write a code simulating ARP /RARP protocols.
6. Study of Network simulator (NS) and Simulation of Congestion Control Algorithms using NS.
7. Study of TCP/UDP performance using Simulation tool.
8. Simulation of Distance Vector/ Link State Routing algorithm.
9. Performance evaluation of Routing protocols using Simulation tool.
10. Simulation of error correction code (like CRC).

TOTAL: 60 PERIODS

INDEX

EX-NO:1 LEARN TO USE COMMANDS LIKE TCPDUMP, NETSTAT, IFCONFIG, NSLOOKUP AND TRACEROUTE. CAPTURE PINGAND TRACEROUTE PDUS USING A NETWORK PROTOCOL ANALYZER AND EXAMINE.

DATE:

AIM

To study the basic networking commands.

C:\>arp –a: ARP is short form of address resolution protocol, It will show the IP address of your computer along with the IP address and MAC address of your router.

C:\>hostname: This is the simplest of all TCP/IP commands. It simply displays the name of your computer.

C:\>ipconfig: The ipconfig command displays information about the host (the computer your sitting at)computer TCP/IP configuration.

C:\>ipconfig /all: This command displays detailed configuration information about your TCP/IP connection including Router, Gateway, DNS, DHCP, and type of Ethernet adapter in your system.

C:\>Ipconfig /renew: Using this command will renew all your IP addresses that you are currently (leasing) borrowing from the DHCP server. This command is a quick problem solver if you are having connection issues, but does not work if you have been configured with a static IP address.

C:\>Ipconifg /release: This command allows you to drop the IP lease from the DHCP server.

C:\>ipconfig /flushdns: This command is only needed if you're having trouble with your networks DNS configuration. The best time to use this command is after network configuration frustration sets in, and you really need the computer to reply with flushed.

C:\>nbtstat –a: This command helps solve problems with NetBIOS name resolution. (Nbt stands for NetBIOS over TCP/IP)

C:\>netdiag: Netdiag is a network testing utility that performs a variety of network diagnostic tests, allowing you to pinpoint problems in your network. Netdiag isn't installed by default, but can be installed from the Windows XP CD after saying no to the install. Navigate to the CD ROM drive letter and open the support\tools folder on the XP CD and click the setup.exe icon in the support\tools folder.

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

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:\>pathping: Pathping is unique to Window’s, and is basically a combination of the Ping and Tracert commands. Pathping traces the route to the destination address then launches a 25 second test of each router along the way, gathering statistics on the rate of data loss along each hop.

C:\>ping: Ping is the most basic TCP/IP command, and it’s the same as placing a phone call to your best friend. You pick up your telephone and dial a number, expecting your best friend to reply with “Hello” on the other end. Computers make phone calls to each other over a network by using a Ping command. The Ping commands main purpose is to place a phone call to another computer on the network, and request an answer. Ping has 2 options it can use to place a phone call to another computer on the network. It can use the computers name or IP address.

C:\>route: The route command displays the computers routing table. A typical computer, with a single network interface, connected to a LAN, with a router is fairly simple and generally doesn’t pose any network problems. But if you’re having trouble accessing other computers on your network, you can use the route command to make sure the entries in the routing table are correct.

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:

EX-NO: 2 WRITE A HTTP WEB CLIENT PROGRAM TO DOWNLOADA WEB PAGE USING TCP SOCKETS.

DATE:

AIM

To download a webpage using Java

ALGORITHM:

CLIENT SIDE:

4. Start the program.
5. Create a socket which binds the Ip address of server and the port address to acquire service.
6. After establishing connection send the url to server.
7. Open a file and store the received data into the file.
8. Close the socket.
9. End the program.

SERVER SIDE

4. Start the program.
5. Create a server socket to activate the port address.
6. Create a socket for the server socket which accepts the connection.
7. After establishing connection receive url from client.
8. Download the content of the url received and send the data to client.
9. Close the socket.
10. End the program.

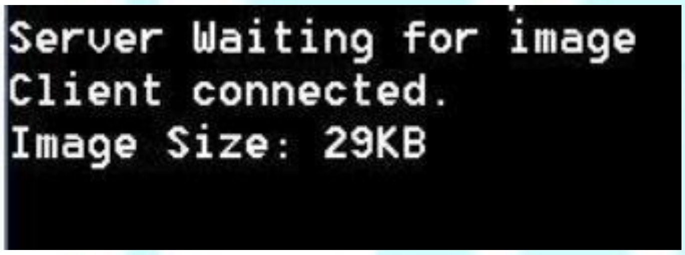
PROGRAM

```
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("digital_image_processing.jpg")); 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(); }}
```

SERVER PROGRAM

```
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); }}
```

OUTPUT



```
Server Waiting for image
Client connected.
Image Size: 29KB
```

RESULT

EX-NO 3A APPLICATIONS USING TCP SOCKET

DATE :

A. Echo client and Echo server

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.
6. Stop the program

EchoServer.java

```
import java.net.*;
import java.io.*;
public class EServer
{
public static void main(String args[])
{
ServerSocket s=null;
String line;
DataInputStream is;
PrintStream ps;
Socket c=null;
try
{
s=new ServerSocket(9000);
}
catch(IOException e)

{
System.out.println(e);
}
try
{
c=s.accept();
is=new DataInputStream(c.getInputStream());
ps=new PrintStream(c.getOutputStream());
while(true)
{
line=is.readLine();
ps.println(line);
}

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

EClient.java

```
import java.net.*;
import java.io.*;
public class EClient
{
public static void main(String arg[])
{
Socket c=null;
String line;
DataInputStream is,is1;
PrintStream os;
try
{
InetAddress ia = InetAddress.getLocalHost();
c=new Socket(ia,9000);
}
catch(IOException e)
{
System.out.println(e);
}
try
{
os=new PrintStream(c.getOutputStream());
is=new DataInputStream(System.in);
is1=new DataInputStream(c.getInputStream());
while(true)
{
System.out.println("Client:");
line=is.readLine();
os.println(line);
System.out.println("Server:" + is1.readLine());
}
}
catch(IOException e)
{
```

```
System.out.println("Socket Closed!");
}
}
}
```

OUTPUT

Server

```
C:\ProgramFiles\Java\jdk1.5.0\bin>javac EServer.java Note: EServer.java
uses or overrides a deprecated API. Note: Recompile with -deprecation for
details.
C:\Program Files\Java\jdk1.5.0\bin>java EServer C:\Program
Files\Java\jdk1.5.0\bin>
```

Client

```
C:\Program Files\Java\jdk1.5.0\bin>javac EClient.java Note: EClient.java uses
or overrides a deprecated API. Note: Recompile with -deprecation for details.
C:\Program Files\Java\jdk1.5.0\bin>java EClient
Client:
Hai Server
Server:Hai Server
Client:
Hello
Server:Hello
Client:
end
Server:end
Client:
ds
Socket Closed!
```

RESULT:

EX-NO: 3B **CHAT USING TCP SOCKETS**

DATE :

AIM

Write a Program client –server application for chat using TCP Sockets

ALGORITHM

CLIENT

1. Start the program
2. Include necessary package in java
3. To create a socket in client to server.
4. The client establishes a connection to the server.
5. The client accept the connection and to send the data from client to server.
6. The client communicates the server to send the end of the message
7. Stop the program.

SERVER

1. Start the program
2. Include necessary package in java
3. To create a socket in server to client
4. The server establishes a connection to the client.
5. The server accept the connection and to send the data from server to client and
6. vice versa
7. The server communicate the client to send the end of the message.
8. Stop the program.

Program

UDPserver.java

```
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 InputStreamReader(System.in));
InetAddress ia=InetAddress.geyLocalHost(); 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);
System.out.println("Server:"); String str=dis.readLine();
if(str.equals("end"))
break;
buffer=str.getBytes();
ds.send(new
DatagramPacket(buffer,str.length(),ia,serverport));
}
}
}
```

UDPclient.java

```
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 InputStreamReader(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:

Server

```
C:\Program Files\Java\jdk1.5.0\bin>javac UDPserver.java

C:\Program Files\Java\jdk1.5.0\bin>java
UDPserver press ctrl+c to quit the
program Client:Hai Server
Server:
Hello Client
Client:How are You
Server:
I am Fine
```

Client

```
C:\Program Files\Java\jdk1.5.0\bin>javac UDPclient.java
C:\Program Files\Java\jdk1.5.0\bin>java UDPclient
server waiting

Client:
Hai Server
Server:Hello Clie
Client:
How are You
Server:I am Fine
Client:
end
```

RESULT:

EX NO: 3C FILE TRANSFER USING TCP SOCKET

AIM

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

ALGORITHM

CLIENT SIDE

1. Start.
2. Establish a connection between the Client and Server.
3. Socket ss=new Socket(InetAddress.getLocalHost(),1100);
4. Implement a client that can send two requests.
   i) To get a file from the server.
   ii) To put or send a file to the server.
5. After getting approval from the server ,the client either get file from the server or send
6. file to the server.

SERVER SIDE

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

Program

File Client

```
import java.io.*;
import  java.net.*;
import java.util.*;
class Clientfile
{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 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);
}}}
```

Server

```
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

```
File content
Computer networks
jhfcgsauf
jbsdava
jbvuesagv
client
Enter the file name:
sample.txt
server
Computer networks
jhfcgsauf
jbsdava
jbvuesagv
client
Enter the new file name:
net.txt
Computer networks
jhfcgsauf
jbsdava
jbvuesagv
Destination file
Computer networks
jhfcgsauf
jbsdava
jbvuesagv
```

RESULT:

EX-NO: 4 SIMULATION OF DNS USING UDP SOCKETS

DATE :

AIM

To write a java program for DNS application

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

UDP DNS Server Udpdnsserver

```
java import java.io.*;
import java.net.*;
public class udpdnsserver
{
private static int indexOf(String[] array, String str)
{
str = str.trim();
for (int i=0; i < array.length; i++)
{
if (array[i].equals(str)) return i;
}
return -1;
}
public static void main(String arg[])throws IOException
{
String[] hosts = {"yahoo.com", "gmail.com","cricinfo.com", "facebook.com"};
String[] ip = {"68.180.206.184", "209.85.148.19","80.168.92.140","69.63.189.16"};
 System.out.println("Press Ctrl + C to Quit");
while (true)
{
DatagramSocket serversocket=new
DatagramSocket(1362); byte[] senddata = new
byte[1021];
byte[] receivedata = new byte[1021];
DatagramPacket recvpack = new DatagramPacket(receivedata,
receivedata.length); serversocket.receive(recvpack);
String sen = new String(recvpack.getData());
InetAddress ipaddress = recvpack.getAddress();
int port = recvpack.getPort();
String capsent;
System.out.println("Request for host " + sen);
if(indexOf (hosts, sen) != -1)
capsent = ip[indexOf (hosts, sen)];
else
capsent = "Host Not Found";
senddata = capsent.getBytes();
 DatagramPacket pack = new DatagramPacket (senddata,
senddata.length,ipaddress,port); serversocket.send(pack);
serversocket.close();
}
}
}
```

//**UDP DNS Client** –

```
Udpdnsclient
java import java.io.*;
import java.net.*;
public class udpdnsclient
{
public static void main(String args[])throws IOException
{
 BufferedReader br = new BufferedReader(new
InputStreamReader(System.in)); DatagramSocket clientsocket =
new DatagramSocket();
InetAddress ipaddress;
if (args.length == 0)
ipaddress =InetAddress.getLocalHost();
else
ipaddress = InetAddress.getByName(args[0]);
byte[] senddata = new byte[1024];
byte[] receivedata = new byte[1024];
int portaddr = 1362;
System.out.print("Enter the hostname : ");
String sentence = br.readLine();
Senddata = sentence.getBytes();
DatagramPacket pack = new
DatagramPacket(senddata,senddata.length,ipaddress,portaddr);
clientsocket.send(pack);
DatagramPacket recvpack =new
DatagramPacket(receivedata,receivedata.length);
clientsocket.receive(recvpack);
String modified = new
String(recvpack.getData());
System.out.println("IP Address: " +
modified); clientsocket.close();
}
}
```

OUTPUT

Server

```
javac udpdnsserver.java
java udpdnsserver

Press Ctrl + C to Quit Request for host yahoo.com
Request for host cricinfo.com
Request for host youtube.com
```

Client

```
javac udpdnsclient.java
java udpdnsclient
Enter the hostname : yahoo.com
IP Address: 68.180.206.184
java udpdnsclient
Enter the hostname : cricinfo.com
IP Address: 80.168.92.140
```

RESULT:

EX-NO: 5 SIMULATION OF ARP /RARP PROTOCOLS

DATE :

AIM

To implement Address Resolution Protocol .

ALGORITHM

CLIENT SIDE

1. Establish a connection between the Client and Server.
   Socket ss=new Socket(InetAddress.getLocalHost(),1100);
2. Create instance output stream writer
   PrintWriter ps=new PrintWriter(s.getOutputStream(),true);
3. Get the IP Address to resolve its physical address.
4. Send the IPAddress to its output Stream.ps.println(ip);
5. Print the Physical Address received from the server.

SERVER SIDE

1. Accept the connection request by the client.
   ServerSocket ss=new ServerSocket(2000);Socket s=ss.accept();
2. Get the IPaddress from its inputstream.
   BufferedReader br1=new
   BufferedReader(newInputStreamReader(s.getInputStream())); ip=br1.readLine();
3. During runtime execute the processRuntime
   r=Runtime.getRuntime(); Process p=r.exec("arp -a "+ip);
4. Send the Physical Address to the client.

PROGRAM

ARP CLIENT

```
import java.io.*;

import java.net.*;

class ArpClient

{

public static void main(String args[])throws IOException

{try{

Socket ss=new Socket(InetAddress.getLocalHost(),1100);
PrintStream ps=new PrintStream(ss.getOutputStream());

BufferedReader br=new BufferedReader(newInputStreamReader(System.in));
 String ip;

 System.out.println("Enter the IPADDRESS:");

 ip=br.readLine();

 ps.println(ip);

 String str,data;

 BufferedReader br2=new
 BufferedReader(newInputStreamReader(ss.getInputStream()));
 System.out.println("ARP From Server::"); do

 {

 str=br2.readLine();

 System.out.println(str);

 }

 while(!(str.equalsIgnoreCase("end")));

 }

 catch(IOException e)

 {

 System.out.println("Error"+e);


 }}}
```

ARP SERVER

```
import java.io.*;

import java.net.*;

class ArpServer

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

{

try

{

ServerSocket ss=new ServerSocket(1100);
Socket s=ss.accept();
PrintStream ps=new PrintStream(s.getOutputStream());
BufferedReader br1=new BufferedReader(newInputStreamReader(s.getInputStream()));

String ip;
ip=br1.readLine();
Runtime r=Runtime.getRuntime();
Process p=r.exec("arp -a "+ip);
BufferedReader br2=new BufferedReader(newInputStreamReader(p.getInputStream()));


String str;
while((str=br2.readLine())!=null)
{
ps.println(str);
}}
catch(IOException e)
{
  System.out.println("Error"+e); }}}
```

OUTPUT

```
C:\Networking Programs>java ArpServer
C:\Networking Programs>java ArpClient
Enter the IPADDRESS:
192.168.11.58
ARP From Server::
Interface: 192.168.11.57 on Interface 0x1000003
Internet Address Physical Address     Type
192.168.11.58    00-14-85-67-11-84 dynamic
```

RESULT

EX-NO: 6 STUDY OF NETWORK SIMULATOR (NS).AND SIMULATION OF CONGESTION CONTROL ALGORITHMS USING NS

DATE :

AIM:

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

NET WORK SIMULATOR (NS2)

Ns overview

Ns programming: A Quick start

Case study I: A simple Wireless network

Case study II: Create a new agent in Ns

Ns overview

Ns Status

Periodical release (ns-2.26, Feb 2003)

Platform support

FreeBSD, Linux, Solaris, Windows and Mac

NS2 Functionalities

Routing, Transportation, Traffic sources,
Queuing disciplines, QoS

Wireless

Ad hoc routing, mobile IP, sensor-MAC
Tracing, visualization and various utilities
NS(Network Simulators)

Most of the commercial simulators are GUI driven, while some network simulators are CLI driven. The network model / configuration describes the state of the network (nodes,routers, switches, links) and the events (data transmissions, packet error etc.). An important output of simulations are the trace files. Trace files log every packet, every event that occurred in the simulation and are used for analysis. Network simulators can also provide other tools to facilitate visual analysis of trends and potential trouble spots.

Most network simulators use discrete event simulation, in which a list of pending "events" is stored, and those events are processed in order, with some events triggering future events— such as the event of the arrival of a packet at one node triggering the event of the arrival of that packet at a downstream node.

Simulation of networks is a very complex task. For example, if congestion is high, then estimation of the average occupancy is challenging because of high variance. To estimate the likelihood of a buffer overflow in a network, the time required for an accurate answer can be extremely large. Specialized techniques such as "control variates" and "importance sampling" have been developed to speed simulation.

Examples of network simulators There are many both free/open-source and proprietary network simulators. Examples of notable network simulation software are, ordered after how often they are mentioned in research papers:

1. ns (open source)
2. OPNET (proprietary software)
3. Net Sim (proprietary software)

Uses of network simulators

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive. They allow engineers, researchers to test scenarios that might be particularly difficult or expensive to emulate using real hardware - for instance, simulating a scenario with several nodes or experimenting with a new protocol in the network. Network simulators are particularly useful in allowing researchers to test new networking protocols or changes to existing protocols in a controlled and reproducible environment.

A typical network simulator encompasses a wide range of networking technologies and can help the users to build complex networks from basic building blocks such as a variety of nodes and links. With the help of simulators, one can design hierarchical networks using various types of nodes like computers, hubs, bridges, routers, switches, links, mobile units etc.

Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP etc. and Local Area Network (LAN) technologies like Ethernet, token rings etc., can all be simulated with a typical simulator and the user can test, analyze various standard results apart from devising some novel protocol or strategy for routing etc. Network simulators are also widely used to simulate battlefield networks in Network-centric warfare

There are a wide variety of network simulators, ranging from the very simple to the very complex. Minimally, a network simulator must enable a user to represent a network topology, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to handle traffic in a network. Graphical applications allow users to easily visualize the workings of their simulated environment. Text-based applications may provide a less intuitive interface, but may permit more advanced forms of customization.

Packet loss

occurs when one or more packets of data travelling across a computer network fail to reach heir destination. Packet loss is distinguished as one of the three main error types encountered in digital communications; the other two being bit error and spurious packets caused due to noise. Packets can be lost in a network because they may be dropped when a queue in the network node overflows. The amount of packet loss during the steady state is another important property of a congestion control scheme. The larger the value of packet loss, the more difficult it is for transport layer protocols to maintain high bandwidths, the sensitivity to loss of individual packets, as well as to frequency and patterns of loss among longer packet sequences is strongly dependent on the application itself.

Throughput

This is the main performance measure characteristic, and most widely used. In communication networks, such as Ethernet or packet radio, throughput or network throughput is the average rate of successful message delivery over a communication channel. The throughput is usually measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per time slot This measure how soon the receiver is able to get a certain amount of data send by the sender. It is determined as the ratio of the total data received to the end to end delay. Throughput is an important factor which directly impacts the network performance

Delay

Delay is the time elapsed while a packet travels from one point e.g., source premise or network ingress to destination premise or network degrees. The larger the value of delay, the more difficult it is for transport layer protocols to maintain high bandwidths. We will calculate end to end delay

Queue Length

A queuing system in networks can be described as packets arriving for service, waiting for service if it is not immediate, and if having waited for service, leaving the system after being served. Thus queue length is very important characteristic to determine that how well the active queue management of the congestion control algorithm has been working

RESULT:

EX-NO: 7 STUDY OF TCP/UDP PERFORMANCE USINGSIMULATION TOOL

DATE :

AIM

To simulate the performance of TCP and UDP protocol using Network simulator

ALGORITHM :

Step 1: start the program.
Step 2: declare the global variables ns for creating a new simulator.
Step 3: set the color for packets.
Step 4: open the network animator file in the name of file2 in the write mode.
Step 5: open the trace file in the name of file 1 in the write mode.
Step 6: set the multicast routing protocol to transfer the packets in network.
Step 7: create the multicast capable no of nodes.
Step 8: create the duplex-link between the nodes including the delay time,bandwidth and dropping

Program :

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

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

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

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

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

```
#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail

$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

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

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

#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5
#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
```

```
$cbr set random_ false
#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
#Detach tcp and sink agents (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"
#Run the simulation
$ns run
```

Output :

```
  Packet Received

-AD=120

Infinite root(reaches 16 hopcount)
```

RESULT:

EX-NO: 8 SIMULATION OF DISTANCE VECTOR/ LINKSTATE ROUTING PROTOCOL

DATE :

AIM :

To simulate and study the Distance Vector routing algorithm using simulation.

ALGORITHM:

1. Create a simulator object

2. Define different colors for different data flows

3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.

4. Create n number of nodes using for loop

5. Create duplex links between the nodes

6. Setup UDP Connection between n(0) and n(5)

7. Setup another UDP connection between n(1) and n(5)

8. Apply CBR Traffic over both UDP connections

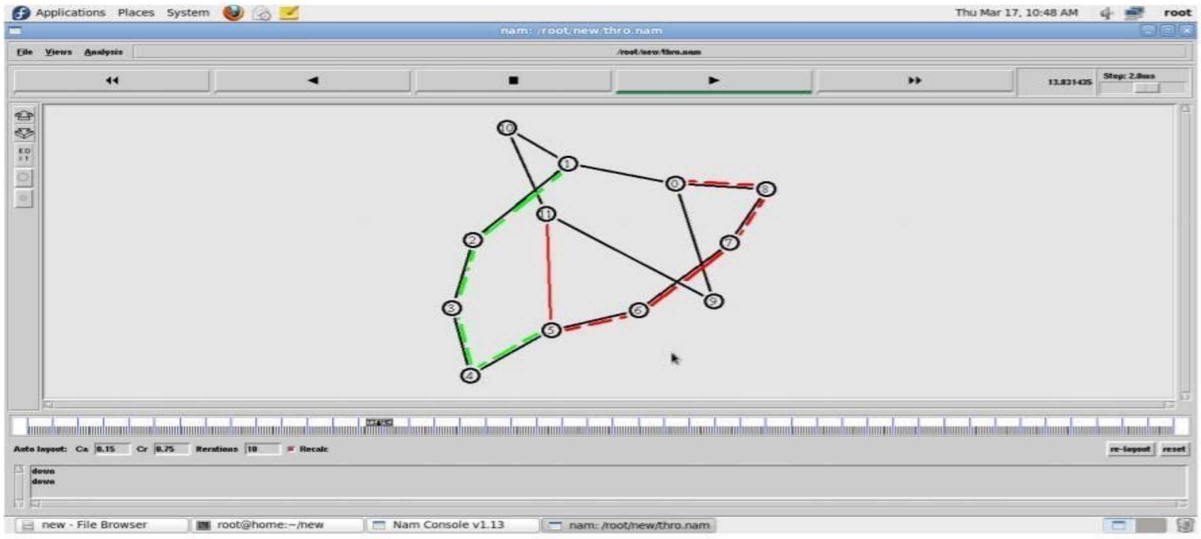
9. Choose distance vector routing protocol to transmit data from sender to receiver.

10. Schedule events and run the program.

PROGRAM:

```
set ns [new Simulator]

set nr [open thro.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

}

 for { set i 0 } { $i < 12} { incr i 1 } {

set n($i) [$ns node]}

for {set i 0} {$i < 8} {incr i} {

 $ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }

$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail

$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail

$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail

$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail

$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail

$ns duplex-link $n(11) $n(5) 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 $n(5)$null0

 $ns connect $udp0 $null0

set udp1 [new Agent/UDP]

$ns attach-agent $n(1) $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 $n(5) $null0

$ns connect $udp1 $null0

$ns rtproto DV

$ns rtmodel-at 10.0 down $n(11) $n(5)

$ns rtmodel-at 15.0 down $n(7) $n(6)

$ns rtmodel-at 30.0 up $n(11) $n(5)

$ns rtmodel-at 20.0 up $n(7) $n(6)

$udp0 set fid_ 1 $udp1

set fid_ 2

$ns color 1 Red

$ns color 2 Green

$ns at 1.0 "$cbr0 start"

$ns at 2.0 "

$cbr1 start"

$ns at 45 "finish"

$ns run
```


Applications Places System
Thu Mar 17, 10:48 AM root
nam: /root/new/thro.nam
File Views Analysis
/root/new/thro.nam
13.831435
Step: 2.0ms
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout reset
down
down
new - File Browser
root@home:~/new
Nam Console v1.13
nam: /root/new/thro.nam

Result :

EX-NO: 9 PERFORMANCE EVALUATION OF ROUTING PROTOCOLS USING SIMULATION TOOL

DATE:

AIM:

To simulate and study the link state routing algorithm using simulation.

ALGORITHM:

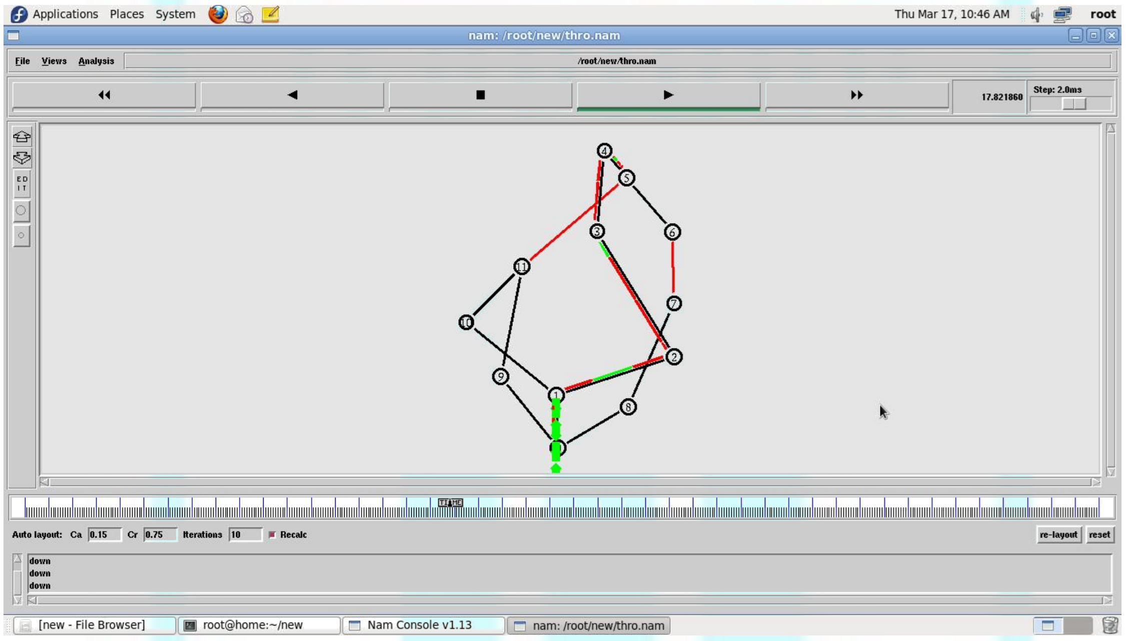
1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create n number of nodes using for loop
5. Create duplex links between the nodes
6. Setup UDP Connection between n(0) and n(5)
7. Setup another UDP connection between n(1) and n(5)
8. Apply CBR Traffic over both UDP connections
9. Choose Link state routing protocol to transmit data from sender to receiver.
10. Schedule events and run the program.

```
set ns [new Simulator]

set nr [open thro.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

    }

or { set i 0 } { $i < 12} { incr i 1 } {
```

```
set n($i) [$ns node]}

for {set i 0} {$i < 8} {incr i} {

$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail  }

$ns duplex-link $n(0) $n(8) 1Mb 10ms
DropTail $ns duplex-link $n(1) $n(10) 1Mb
10ms DropTail $ns duplex-link $n(0) $n(9)
1Mb 10ms DropTail $ns duplex-link $n(9)
$n(11) 1Mb 10ms DropTail $ns duplex-link
$n(10) $n(11) 1Mb 10ms DropTail

$ns duplex-link $n(11) $n(5) 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(5) $null0

$ns connect $udp0 $null0

set udp1 [new Agent/UDP]

$ns attach-agent $n(1) $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 $n(5) $null0

$ns connect $udp1 $null0

$ns rtproto LS
```

```
$ns rtmodel-at 10.0 down $n(11) $n(5)

$ns rtmodel-at 15.0 down $n(7) $n(6)
$ns rtmodel-at 30.0 up $n(11) $n(5)

$ns rtmodel-at 20.0 up $n(7) $n(6)

$udp0 set fid_ 1

$udp1 set fid_ 2

$ns color 1 Red

$ns color 2 Green

$ns at 1.0 "$cbr0 start"

$ns at 2.0 "$cbr1 start"

$ns at 45 "finish"

$ns run
```

Output



```
down
down
down
```

RESULT:

EX-NO: 10 IMPLEMENTATION OF ERROR DETECTION AND ERROR CORRECTION TECHNIQUES

DATE :

AIM:

To implement error detection and error correction techniques.

ALGORITHM:

1. Open Turbo c++ software and type the program for error detection
2. Get the input in the form of bits.
3. Append 16 zeros as redundancy bits.
4. Divide the appended data using a divisor polynomial.
5. The resulting data should be transmitted to the receiver.
6. At the receiver the received data is entered.
7. The same process is repeated at the receiver.
8. If the remainder is zero there is no error otherwise there is some error in the received bits
9. Run the program.

PROGRAM

```
#include<stdio.h>

char m[50],g[50],r[50],q[50],temp[50];

void caltrans(int);

void crc(int);

void calram();

void shiftl();

int main()

{

int n,i=0;

char ch,flag=0;

printf("Enter the frame bits:"); while((ch=getc(stdin))!='\n') m[i++]=ch;

n=i;

for(i=0;i<16;i++)

m[n++]='0';
```

```
m[n]='\0';
printf("Message after appending 16 zeros:%s",m); for(i=0;i<=16;i++)
g[i]='0';
g[0]=g[4]=g[11]=g[16]='1';g[17]='\0';
printf("\ngenerator:%s\n",g);
crc(n);
rintf("\n\nquotient:%s",q);
caltrans(n);
printf("\ntransmitted frame:%s",m);
printf("\nEnter transmitted frame:");
scanf("\n%s",m);
printf("CRC checking\n");
crc(n);
printf("\n\nlast remainder:%s",r);
for(i=0;i<16;i++)
if(r[i]!='0')
flag=1;
else
continue;
if(flag==1)
printf("Error during transmission");
else
printf("\n\nReceived freme is correct");
}
void crc(int n)
{
```

```
int i,j;
for(i=0;i<n;i++)
temp[i]=m[i];
for(i=0;i<16;i++)
r[i]=m[i];
printf("\nintermediate remainder\n");
for(i=0;i<n-16;i++)
{
if(r[0]=='1')
{
q[i]='1';
calram();
}
else
{
q[i]='0';
shiftl();
}
r[16]=m[17+i];
r[17]='\0';
printf("\nremainder %d:%s",i+1,r);
for(j=0;j<=17;j++)
temp[j]=r[j];
}
q[n-16]='\0';
}
```

```
void calram()
{
int i,j;
for(i=1;i<=16;i++)
r[i-1]=((int)temp[i]-48)^((int)g[i]-48)+48;
}
void shiftl()
{int i;
for(i=1;i<=16;i++)
r[i-1]=r[i];
}
void caltrans(int n)
int i,k=0;
for(i=n-16;i<n;i++)
m[i]=((int)m[i]-48)^((int)r[k++]-48)+48;
m[i]='\0';
}
```

OUTPUT:

```
Enter the Frame Bits:
1011
The msg after appending 16 zeros:
10110000000000000000
The Transmitted frame is:10111011000101101011
Enter the transmitted Frame
10111011000101101011
Received msg:10111011000101101011
The Remainder is:0000000000000000
Received frame is correct.
```

RESULT:

EX-NO: 11 IMPLEMENTATION OF REMOTE COMMAND EXECUTION

DATE:

AIM

To implement Remote Command Execution(RCE).

ALGORITHM

CLIENT SIDE

1. Establish a connection between the Client and Server. Socket client=new Socket("127.0.0.1",6555);
2. Create instances for input and output streams.
   Print Stream ps=new Print Stream(client.getOutputStream());

3. BufferedReader br=new BufferedReader(newInputStreamReader(System.in));
   4. Enter the command in Client Window. Send the message to its output str=br.readLine();ps.println(str);

SERVER SIDE

1. Accept the connection request by the client. ServerSocket server=new ServerSocket(6555); Socket s=server.accept();
2. Get the IPaddress from its inputstream.
   BufferedReader br1=new
   BufferedReader(newInputStreamReader(s.getInputStream())); ip=br1.readLine();

3. During runtime execute the process Runtime r=Runtime.getRuntime(); Process p=r.exec(str);

CLIENT PROGRAM

```
import java.io.*;

import java.net.*;

class clientRCE

{

public static void main(String args[]) throws IOException

{

try

{

String str;Socket client=new Socket("127.0.0.1",6555);
PrintStream ps=new PrintStream(client.getOutputStream());
BufferedReader br=new BufferedReader(newInputStreamReader(System.in));
System.out.println("\t\t\t\tCLIENT WINDOW\n\n\t\tEnter TheCommand:");
str=br.readLine();
ps.println(str);
}
catch(IOException e)
{
System.out.println("Error"+e);     }}}
```

<u>SERVER PROGRAM</u>:

```
import java.io.*;
import java.net.*;
class serverRCE
{
public static void main(String args[]) throws IOException
{
  try
{
String str;
ServerSocket server=new ServerSocket(6555);
Socket s=server.accept();
BufferedReader br=new
BufferedReader(newInputStreamReader(s.getInputStream())); str=br.readLine();
Runtime r=Runtime.getRuntime();
Process p=r.exec(str);
}
catch(IOException e)
{
```

```
System.out.println("Error"+e);
}
}}
```

OUTPUT

```
C:\Networking Programs>java serverRCE
C:\Networking Programs>java clientRCE
```

CLIENT WINDOW

```
Enter The Command:
Notepad
```

RESULT:

EX-NO: 12 CLIENT SERVER APPLICATION USING UDP

DATE:

AIM

To write a program to implement simple client-server application using UDP.

ALGORITHM

CLIENT SIDE

1. Create a datagram socket with server's IP address.
2. Create datagram packets with data, data length and the port address.
3. Send the datagram packets to server through datagram sockets
4. Receive the datagram packets from server through datagram sockets
5. Close the socket.

SERVER SIDE

1. Create a datagram socket with port address.
2. Create datagram packets with data, data length and the port address.
3. Send the datagram packets to client through datagram sockets
4. Receive the datagram packets from client through datagram sockets
5. Close the socket.

UDPserver.java

```
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 InputStreamReader(System.in));
 InetAddress ia=InetAddress.getByName("localhost"); 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);
 System.out.println("Server:");String str=dis.readLine();
 if(str.equals("end"))
 break;
 buffer=str.getBytes();
ds.send(new DatagramPacket(buffer,str.length(),ia,serverport));
}}}
```

UDP CLIENT.JAVA

```
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 InputStreamReader(System.in));
System.out.println("server waiting");InetAddress ia=InetAddress.getByName("10.0.200.36");
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

Server

```
C:\Program Files\Java\jdk1.5.0\bin>javac UDPserver.java
C:\Program Files\Java\jdk1.5.0\bin>java UDPserver press
ctrl+c to quit the program Client:Hai Server

Server:Hello Client

Client:How are You

Server:I am Fine what about you
```

CLIENT

```
C:\Program Files\Java\jdk1.5.0\bin>javac UDPclient.java

C:\Program Files\Java\jdk1.5.0\bin>java UDPclientserver

Waiting

Client:Hai Server

Server:Hello Clie

Client:How are YouServer:I am Fine

Client:end
```

RESULT: