DEPARTMENT COMPUTER SCIENCE AND ENGINEERING

ACADEMIC YEAR: 2022-2023

ODD SEMESTERS

LABMANUAL

(REGULATION-2017)

CS 8581-NETWORKS LABORATORY

5th SEMSTER

PreparedBy
Mrs.P.SATHYA,
AP/CSE
SSCET.

Vision and Mission of the Institute

Vision

To generate world class engineering professionals through innovations and quality research In various fields of Computer science and engineering with Ethical qualities.

Mission

1. Ensuring the academic growth by way of establishing centers of excellence and promoting collaborative learning
2. Strengthening the academic ambience through faculty training programs on contemporary technical education
3. Motivating the students to be successful, ethical and suitable for industry ready
4. Promoting research based projects in the emerging areas of technology convergence for the benefit of students and faculty

PROGRAMME OUTCOMES

The UG program in Computer Engineering will prepare students to attain:

 PO1 Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

 PO2 Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

 PO3 Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health, safety, cultural, societal and environmental considerations.

 PO4 Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis, and interpretation of data and synthesis of the information to provide valid conclusions.

 PO5 Modern tool usage: Create, select, apply appropriate techniques, resources, modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

 PO6 The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, cultural issues and the consequent responsibilities relevant to the professional engineering practice.

 PO7 Environment and sustainability: Understand the impact of the professional engineering solutions in societal, environmental contexts, demonstrate the knowledge and need for sustainable development.

 PO8 Ethics: Apply ethical principles, commit to professional ethics, responsibilities and norms of the engineering practice.

 PO9 Individual and team work: Function effectively as an individual, as a member or leader in diverse teams and in multidisciplinary settings.

 PO10 Communication: Communicate effectively on complex engineering activities with the engineering community with society at large being able to comprehend, write effective reports, design documentation, make effective presentations and receive clear instructions.

 PO11 Project management and finance: Demonstrate knowledge, understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

 PO12 Life-long learning: Recognize the need, ability to engage in independent and life-long learning in the broadest context of technological change.

PEO

PEO1. Basic Skills: Graduates work productively as successful Computer Professionals with problem solving skills, core computing skills and soft skills with social awareness.

PEO2. Technical Knowledge: Graduates engage in everlasting endeavor to promote research and development

PEO3. Technical Knowledge: Graduates communicate effectively, recognize and incorporate the societal needs in their profession by practicing their boundless skills with high regard to ethical responsibilities

Program Specific Outcomes

 PSO 1: An ability to analyze the real time problems and to develop solutions by applying appropriate mathematical logic, algorithms and data structures

 PSO 2: An ability to apply the software design and development concepts, methodologies and techniques to work in the industries

OBJECTIVES

- To learn and use network commands.
- To learn socket programming.
- To implement and analyze various network protocols.
- To learn and use simulation tools.
- To use simulation tools to analyze the performance of various network protocols

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:

    - Echo client and echo server
    - Chat
    - 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

LIST OF EQUIPMENTS FOR A BATCH OF 30 STUDENTS

HARDWARE

 Standalone desktops 30 No’s

SOFTWARE:

- C / C++ / Java / Python / Equivalent Compiler 30 Nos.
- Network simulator like NS2/Glomosim/OPNET/ Packet Tracer / Equivalent

JAVA

Java is a high-level programming language originally developed by Sun Microsystems. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic".

NS2

NS2 stands for Network Simulator Version 2. It is an open-source event-driven simulator designed specifically for research in computer communication networks. It provides substantial support to simulate bunch of protocols like TCP, FTP, UDP, https and DSR.It simulates wired and wireless network. It is primarily Unix based. Uses TCL as its scripting language.

CS8581.1	Implement various protocols using TCP and UDP.
CS8581.2	Compare the performance of different transport layer protocols.
CS8581.3	Use simulation tools to analyze the performance of various network protocols.
CS8581.4	Analyze various routing algorithms.
CS8581.5	Implement error correction codes.

CO-PO MATRIX

CO	PO1	PO2		PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CS8581.1	**3**			**2**	-	-	-	-	-	-	-	-	-
CS8581.2	**3**	**1**		**2**	-	-	-	-	-	-	-	-	-
CS8581.3						**2**	-	-	-	-	-	-	-
CS8581.4	**2**	**1**		**2**	-	-	-	-	-	-	-	-	-
CS8581.5	**3**	**1**		**1**	-	**2**	-	-	-	-	-	-	-
CS8581.6	**3**	**1**		**2**	-	**2**	-	-	-	-	-	--	-

CO-PO MATRIX

COURSE	PSO3
CS8581.1	**2**
CS8581.2	**3**
CS8581.3	**3**
CS8581.4	**3**
CS8581.5	**2**
CS8581.6	**3**

EVALUATION PROCEDURE FOR EACH EXPERIMENT

S.NO	Description	Mark
1	Aim & Pre-Lab discussion	20
2	Observation	20
3	Conduction and Execution	30
4	Output & Result	10
5	Viva	20
TOTAL		**100**

INTERNALASSESSMENTFORLABORATORY

S.NO	Description	Mark
1	Conduction & Execution of Experiment	25
2	Record	10
3	Model Test	15
TOTAL		**50**

NDEX DETAILS for Master File of Circulars and Processes	
Sl. No: 04	Format 03-Lab workbook format

LABORATORY WORKBOOK

Exercise/Experiment Number: 01

Lab Code / Lab : CS8581
Program / Branch : B.E/CSE
Title of the exercise / experiment: Basic networking commands

---

STEP 1: INTRODUCTION

a) OBJECTIVE OF THE EXERCISE/EXPERIMENT

Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute.Capture ping and traceroute PDUs using a network protocol analyzer and examine.

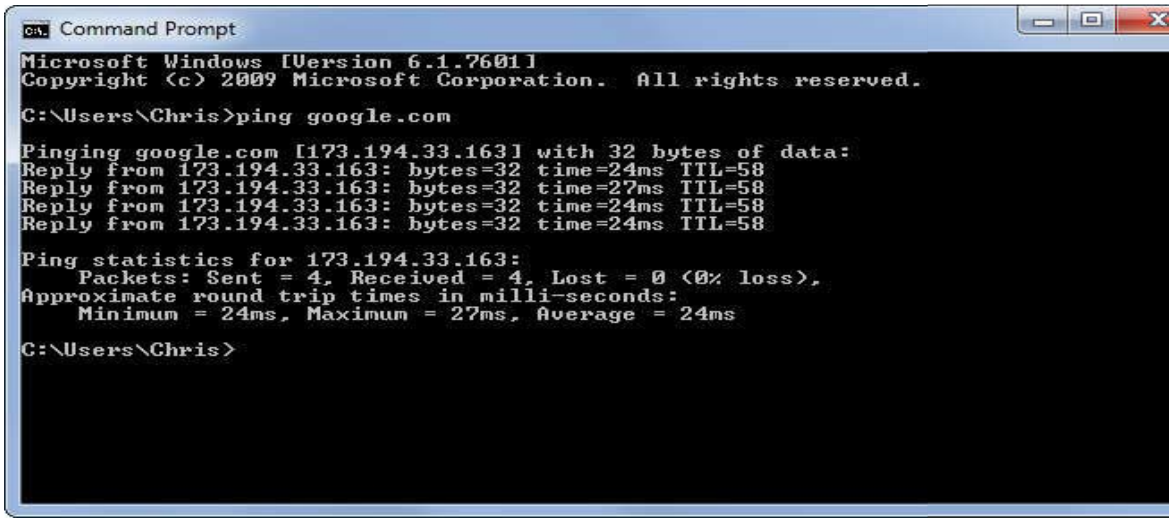
STEP 2: ACQUISITION

b) Facilities/material required to do the exercise/experiment:

Sl. No.	Facilities/material required	Quantity
1	Pentium IV with 2 GB RAM 160 GB HARD Disk Monitor 1024 x 768 colour	40
2	C / C++ / Java / Python / Equivalent Compiler	40

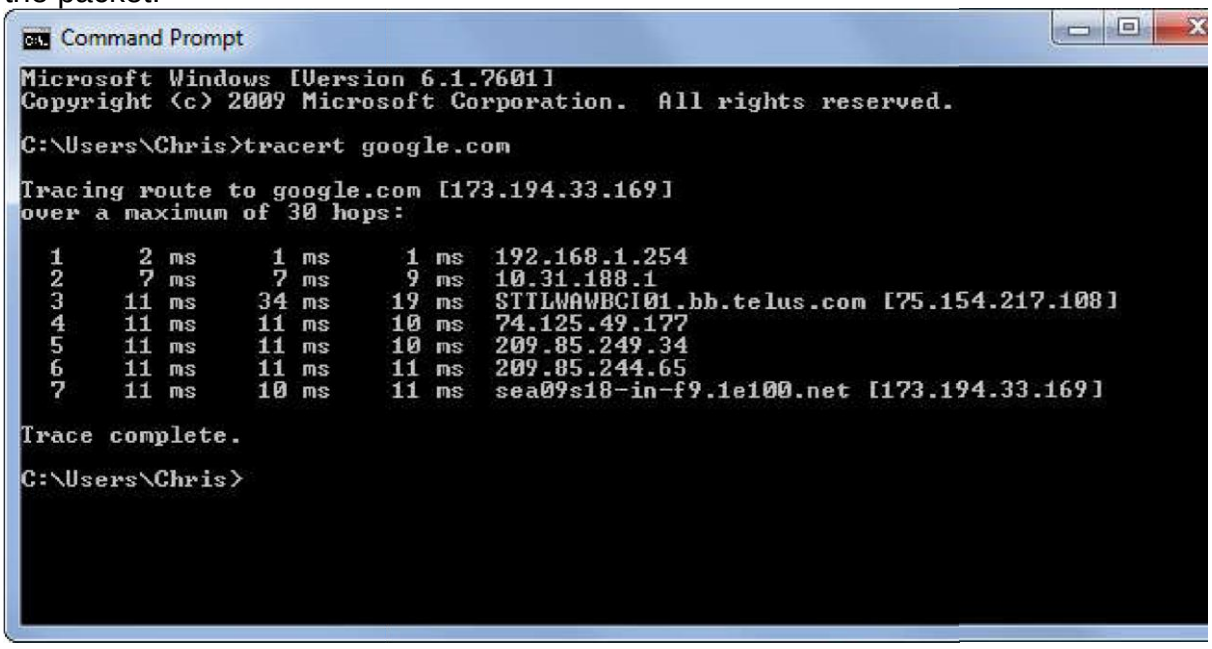
1. ping

The ping command sends ICMP echo request packets to a destination. For example, you could run ping google.com or ping 173.194.33.174 to ping a domain name or IP address. These packets ask the remote destination to reply. If the remote destination is configured to reply, it will respond with packets of its own.



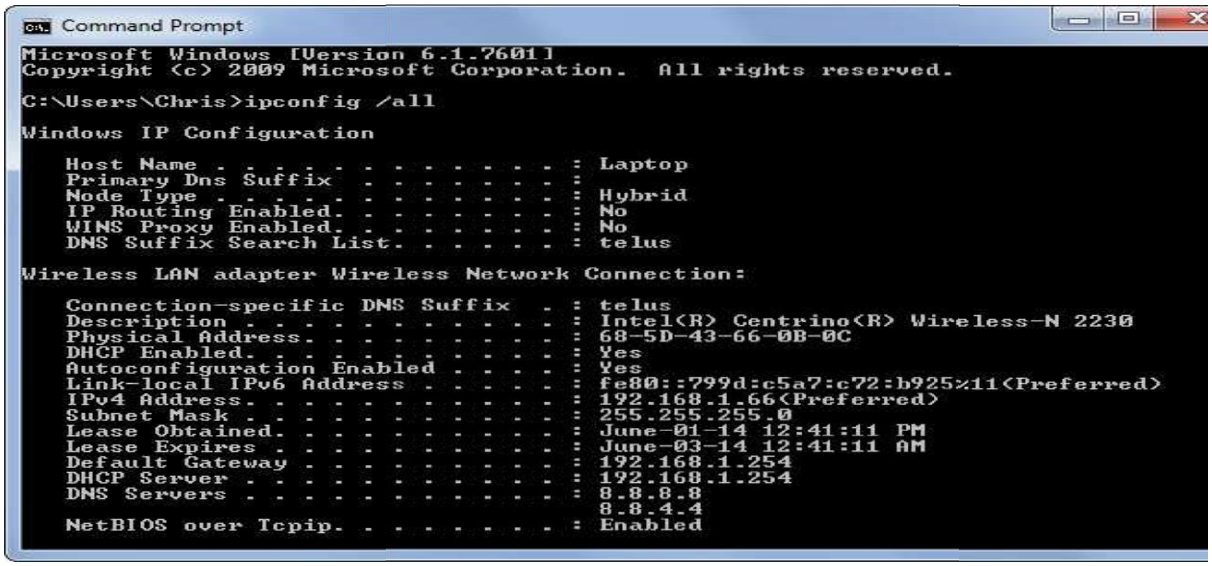
2. traceroute / tracert / tracepath

The traceroute, tracert, or tracepath command is similar to ping, but provides information about the path a packet takes. traceroute sends packets to a destination, asking each Internet router along the way to reply when it passes on the packet.



3. **ipconfig / ifconfig**

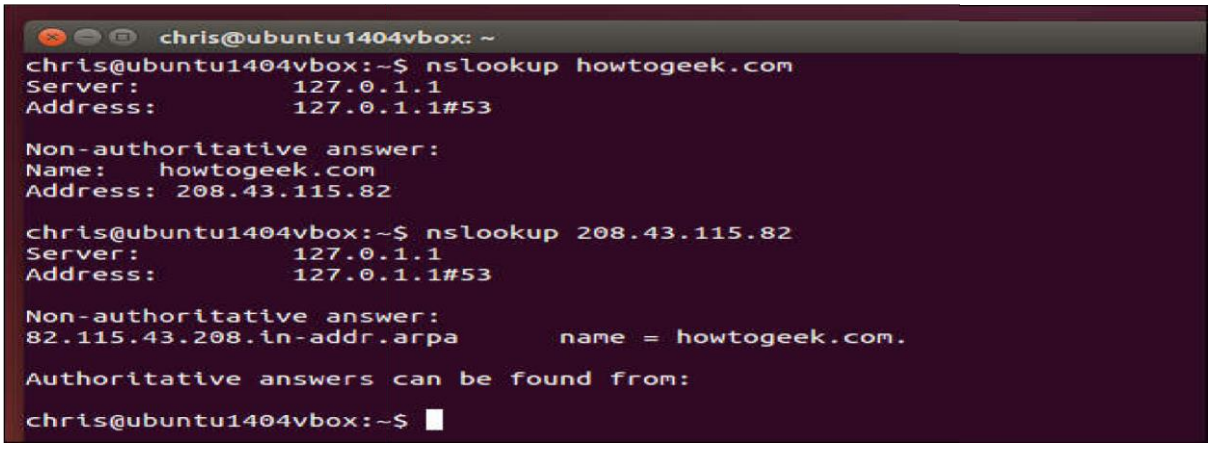
The ipconfig command is used on Windows, while the ifconfig command is used on Linux, Mac OS X, and other Unix-like operating systems. These commands allow you to configure your network interfaces and view information about them.



4.nslookup

The nslookup command will look up the IP addresses associated with a domain name. For example, you can run nslookup howtogeek.com to see the IP address of How-To Geek's server.

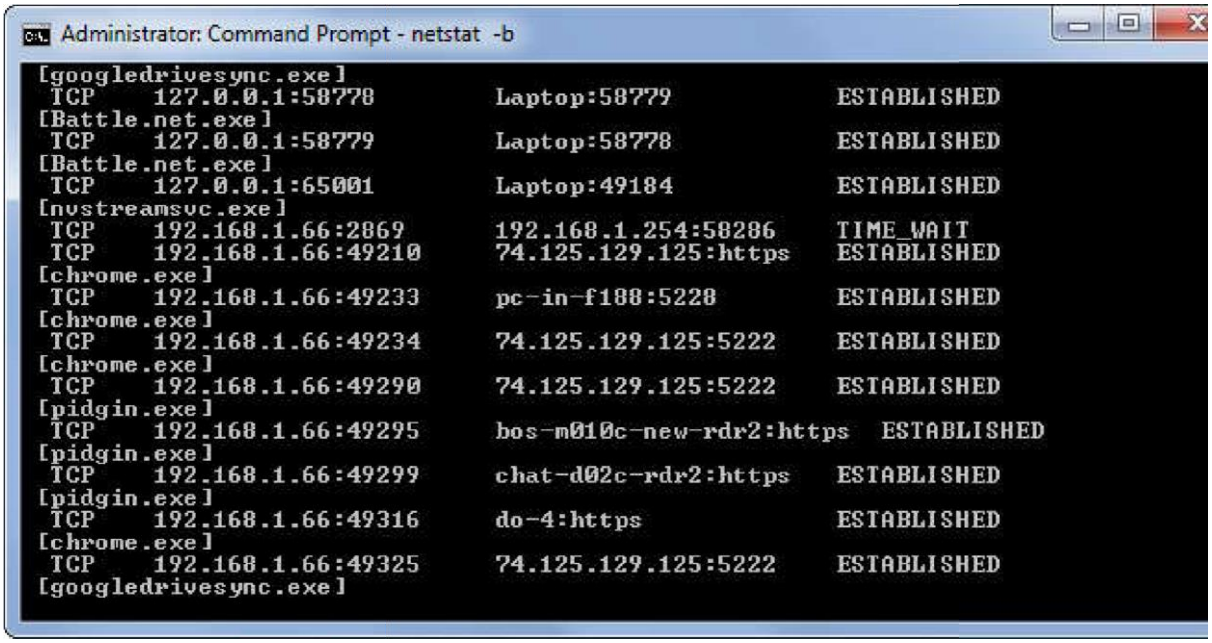
nslookup also allows you to perform a reverse lookup to find the domain name associated with an IP address.



5.netstat

netstat stands for network statistics. This command displays incoming and outgoing network connections as well as other network information. It's available on Windows, Mac, and Linux — each version has its own command-line options you can tweak to see different types of information.

The netstat utility can show open connections on your computer, which programs are making which connections, how much data is being transmitted, and other information.



STEP 3: PRACTICE/TESTING

Questions:

1.what is network?
2.What is the use of netstat command?
3.What is the nslookup command?
4.What is ping command?
What is traceroute command?

Result:

Thus the various networks commands like tcpdump, netstat, ifconfig, nslookup and traceroute ping are executed successfully

Sl. No: 04	Format 03-Lab workbook format

LABORATORY WORKBOOK

Exercise/Experiment Number: 02

Lab Code / Lab : CS85861
Program / Branch : B.E/CSE
Title of the exercise / experiment: HTTP web client program to download a web page using TCP sockets

---

STEP 1: INTRODUCTION

a) OBJECTIVE OF THE EXERCISE/EXPERIMENT

Develop a HTTP web client program to download a web page using TCP sockets.

STEP 2: ACQUISITION

b) Facilities/material required to do the exercise/experiment:

Sl. No.	Facilities/material required	Quantity
1	Pentium IV with 2 GB RAM 160 GB HARD Disk Monitor 1024 x 768 colour	40
2	C / C++ / Java / Python / Equivalent Compiler	40

c) Procedure for doing the exercise/experiment:

Steps	Description
1	Open command prombt
2	Import the java.net package.
3	Specify the host name which web page contents should be loaded.
4	Create a TCP socketfor making the HTTP connection to communicate with the web page
5	Download and display the web page contents.

Program:

Client

```
import javax.swing.*;

 import java.net.*;  import java.awt.image.*;  importjavax.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

```
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:

When you run the client code, following output screen would appear on client side.

Result:

Thus the socket program for HTTP for web page upload and download was developed and executed successfully.

INDEX DETAILS for Master File of Circulars and Processes	
Sl. No: 04	Format 03-Lab workbook format

LABORATORY WORKBOOK

Exercise/Experiment Number: 03

Lab Code / Lab : CS85861
Program / Branch : B.E/CSE
Title of the exercise / experiment: Creation of Applications using TCP sockets

---

STEP 1: INTRODUCTION

a) OBJECTIVE OF THE EXERCISE/EXPERIMENT

Develop the following java Applications using TCP sockets:

a. Echo client and echo server

b. Chat

c. File Transfer

STEP 2: ACQUISITION

b) Facilities/material required to do the exercise/experiment:

Sl. No.	Facilities/material required	Quantity
1	Pentium IV with 2 GB RAM 160 GB HARD Disk Monitor 1024 x 768 colour	40
2	C / C++ / Java / Python / Equivalent Compiler	40

c) Procedure for doing the exercise/experiment:

Steps	Description
1	Open command prombt
2	Import the java.net package.
3	Create sockets in both server and client program for making the connection between them
4	Create objects for DataInputStream and DataOutputStream to send and receive the messages between client and server.
5	Open the socket connection and wait for the acceptance from the

	server.
6	Perform the communication
7	Close the sockets after the message transmission ends.

Programs:

a. Echo client and echo server:

echoserver.java

```
import java.io.*;
import java.net.*
class echoserver{
        public static void main(String args[])throws Exception{
        ServerSocket ss=new ServerSocket(3333);
        Socket s=ss.accept();
        DataInputStream din=new DataInputStream(s.getInputStream());
        DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));


        String str="",str2="";
        while(!str.equals("stop")){
        str=din.readUTF();
System.out.println("client says: "+str);
str2=br.readLine();
dout.writeUTF(str2);   dout.flush();
        }
        din.close();
        s.close();
        ss.close();
```

echoclient.java

```
import java.net.*;
import java.io.*;
class echoclient{
        public static void main(String args[])throws Exception{
        Socket s=new Socket("localhost",3333);
        DataInputStream din=new DataInputStream(s.getInputStream());
        DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));


        String str="",str2="";
        while(!str.equals("stop")){
        str=br.readLine();
```

```
dout.writeUTF(str);
dout.flush();
str2=din.readUTF();
System.out.println("Serve
r says: "+str2);
}


dout.close();
s.close();
}}
```

Output:

echoserver.java	**echoclient.java**
client says: Hi Server Hello Client client says: I am Larry. Your name? This is Steve.	Hi Server Server says: Hello Client I am Larry. Your name? Server says: This is Steve.

b. Chat:

chatserver.java

```
import java.io.*;
 import java.net.*;
public class chatserver
        {
          public static void main(String[] args) throws Exception
          {
             ServerSocket sersock = new ServerSocket(3000);
             System.out.println("Server  ready for chatting");
             Socket sock = sersock.accept( );
                                // reading from keyboard (keyRead object)
             BufferedReader keyRead = new BufferedReader(new
        InputStreamReader(System.in));
                                  // sending to client (pwrite object)
             OutputStream ostream = sock.getOutputStream();
             PrintWriter pwrite = new PrintWriter(ostream, true);


                                // receiving from server ( receiveRead  object)
             InputStream istream = sock.getInputStream();
             BufferedReader receiveRead = new BufferedReader(new
        InputStreamReader(istream));
```

```
        String receiveMessage, sendMessage;
      while(true)
        {
          if((receiveMessage = receiveRead.readLine()) != null)
          {
            System.out.println(receiveMessage);
          }
          sendMessage = keyRead.readLine();
      pwrite.println(sendMessage);
      pwrite.flush();
        }
       }
      }
```

chatclient.java

```
 import java.io.*;
import java.net.*;
public class chatclient
      {
        public static void main(String[] args) throws Exception
        {
          Socket sock = new Socket("localhost", 3000);
                          // reading from keyboard (keyRead object)
      BufferedReader keyRead = new BufferedReader(new
      InputStreamReader(System.in));
                         // sending to client (pwrite object)
          OutputStream ostream = sock.getOutputStream();
          PrintWriter pwrite = new PrintWriter(ostream, true);


                         // receiving from server ( receiveRead  object)
          InputStream istream = sock.getInputStream();
          BufferedReader  receiveRead =  new  BufferedReader(new
      InputStreamReader(istream));


          System.out.println("Start the chitchat, type and press Enter key");


          String receiveMessage, sendMessage;
      while(true)
          {
            sendMessage = keyRead.readLine();  // keyboard reading
      pwrite.println(sendMessage);       // sending to server
      pwrite.flush();                    // flush the data
            if((receiveMessage = receiveRead.readLine()) != null) //receive from
      server
            {
              System.out.println(receiveMessage); // displaying at DOS prompt
            }
```

```
    }
  }
}
```

Output:

chatserver.java	**chatclient.java**
Server ready for chatting	Start the chitchat, type and press Enter key
Hello Mr.Bean, Hi Mr.Donald..	Hello Mr.Bean, Hi Mr.Donald..
How are you	How are you
I am fine. What about you.	I am fine. What about you.
Ya. I am good. What are u doing now.	Ya. I am good. What are u doing now.
I am a business man.	I am a business man.

c. File Transfer:

FtpClient.java

```
import java.io.*;
import java.net.*;
import java.lang.*;
public class FtpClient{
        public static void main(String
        args[]){  try{
        Socket s= new Socket("localhost",4000);
        BufferedReader      br=new        BufferedReader(new
        InputStreamReader(System.in));          BufferedReader br1=new
        BufferedReader(new InputStreamReader(s.getInputStream()));
        PrintWriter pw=new
        PrintWriter(s.getOutputStream(),true);//
        String msg1,msg2;           char c;
                System.out.println("Connection Established between Client and
        Server....\n");
                System.out.println("Enter the file
        name:\n");          msg1=br.readLine();
        pw.println(msg1);       msg2=br1.readLine();
                  System.out.println(msg2);
```

```
            FileOutputStream fout=new
        FileOutputStream("NewFile.txt");            int len=msg2.length();


         for(int i=0;i<len;i++)
                        {
                         c=msg2.charAt(i);
                    fout.write(c);
                        }
                   }
              catch(Exception e){
                   System.out.println(e);
        }
         }
        }
```

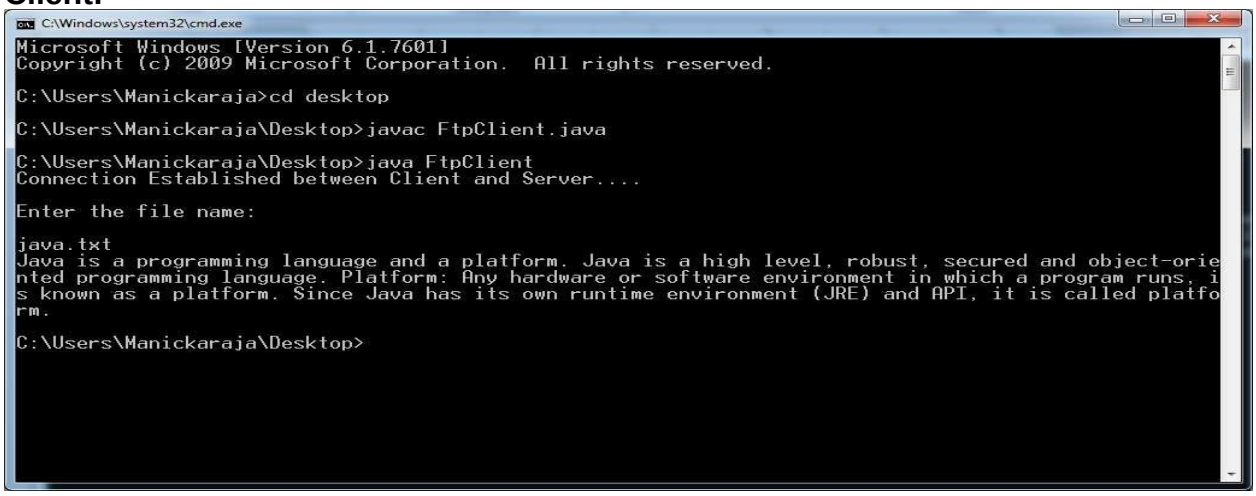
FtpServer.java

```
import java.io.*;
import java.net.*;
 import java.lang.*;
        public class
        FtpServer{
            public static void main(String
        args[]){        try{
        System.out.println("Waiting for client connection.....");
                    ServerSocket ss= new ServerSocket(4000);
                    Socket s=ss.accept();
                   String
        msg1,msg="";          char
        c;
                  int a[]=new int[1000];
        BufferedReader       br=new          BufferedReader(new
        InputStreamReader(s.getInputStream()));
        msg1=br.readLine();
                    FileInputStream fin=new FileInputStream(msg1);
                    PrintWriter pw=new
        PrintWriter(s.getOutputStream(),true);         int
        leng=fin.available();            for(int i=0;i<leng;i++)
                        {
                          a[i]=fin.read();
                        }
                      for(int i=0;i<leng;i++)
                        {
                          c=(char)a[i];
                          msg=msg+c;
                      }
                      System.out.println(msg);
                     pw.println(msg);
```

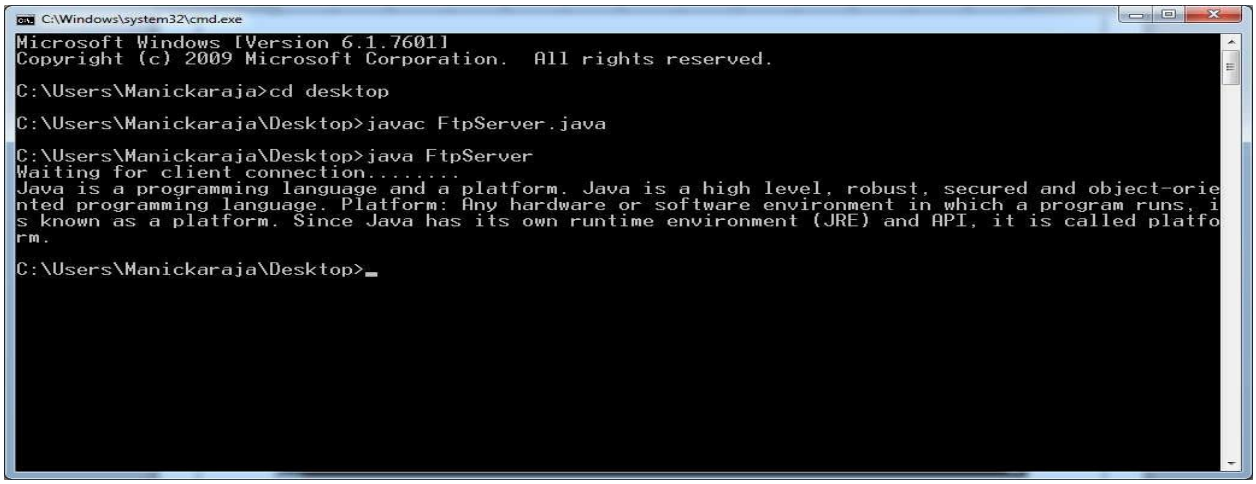
```
                }
            catch (Exception e){
                System.out.println(e);
                }
          }
        }
```

Output:

Client:



Server:



Result:

Thus the java program for Creation of Applications using TCP sockets was verified successfully.

INDEX DETAILS for Master File of Circulars and Processes	
Sl. No: 04	Format 03-Lab workbook format

LABORATORY WORKBOOK

Exercise/Experiment Number: 04

Lab Code / Lab : CS8581
Program / Branch : B.E/CSE
Title of the exercise / experiment: Simulation of DNS using UDP sockets

---

STEP 1: INTRODUCTION

d) OBJECTIVE OF THE EXERCISE/EXPERIMENT

Develop a java program for simulation of DNS using UDP sockets.

STEP 2: ACQUISITION

e) Facilities/material required to do the exercise/experiment:

Sl. No.	Facilities/material required	Quantity
1	Pentium IV with 2 GB RAM 160 GB HARD Disk Monitor 1024 x 768 colour	40
2	C / C++ / Java / Python / Equivalent Compiler	40

f) Procedure for doing the exercise/experiment:

Steps	Description
1	Open command prombt
2	Import the java.net package.
3	In Server, create a UDP socket and accepts the connection from the client.
4	Store the Domain names and it's relevant IP addresses in the server program.
5	In Client, create a UDP socket and make the communication to the server.
6	Send Domain name to the server for getting IP address (or) send IP address to the server for getting Domain name.

7	Close the sockets after the message transmission ends.

Program:

dnsserver.java

```
import java.io.*;
import java.net.*;
import java.util.*;
class dnsserver
	{
		public static void main(String args[])

	{
	try
		{
			DatagramSocket server=new DatagramSocket(1309);
	while(true)
			{
				byte[] sendbyte=new
	byte[1024];				byte[]
	receivebyte=new byte[1024];
	DatagramPacket receiver=new
	DatagramPacket(receivebyte,receivebyte.length);
				server.receive(receiver);
				String str=new String(receiver.getData());
				String s=str.trim();
				//System.out.println(s);
				InetAddress
	addr=receiver.getAddress();				int
	port=receiver.getPort();
				String ip[]={"165.165.80.80","165.165.79.1"};
				String
	name[]={"www.aptitudeguru.com","www.downloadcyclone.blogspot.com"};
				for(int i=0;i<ip.length;i++)
				{
					if(s.equals(ip[i]))
					{
						sendbyte=name[i].getBytes();
						DatagramPacket sender=new
	DatagramPacket(sendbyte,sendbyte.length,addr,port);
						server.send(sender);
	break;
					}
					else if(s.equals(name[i]))
					{
						sendbyte=ip[i].getBytes();
						DatagramPacket sender=new
	DatagramPacket(sendbyte,sendbyte.length,addr,port);
```

```
                                        server.send(sender);
        break;

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

dnsclient.java

```
import java.io.*;
importjava.et.*;
import java.util.*;
class dnsclient
        {
            public static void main(String args[])

        {
        try
            {
                DatagramSocket client=new DatagramSocket();
        InetAddress addr=InetAddress.getByName("localhost");
        byte[] sendbyte=new byte[1024];                byte[]
        receivebyte=new byte[1024];
                BufferedReader in=new BufferedReader(new
        InputStreamReader(System.in));
                System.out.println("Enter the DOMAIN NAME or IP
        address:");             String str=in.readLine();
        sendbyte=str.getBytes();
                DatagramPacket sender=new
        DatagramPacket(sendbyte,sendbyte.length,addr,1309);
                client.send(sender);
                DatagramPacket receiver=new
        DatagramPacket(receivebyte,receivebyte.length);
                client.receive(receiver);
                String s=new String(receiver.getData());
                System.out.println("IP address or DOMAIN NAME:
        "+s.trim());                client.close();
            }
            catch(Exception e)
            {
                System.out.println(e);
```

```
            }
            }
    }
```

Output:

```
Enter the DOMAIN NAME or IP address:

165.165.79.1

IP address or DOMAIN NAME: www.downloadcyclone.blogspot.com
```

Result:

Thus the java program for simulation of DNS using UDP sockets was executed successfully.

INDEX DETAILS for Master File of Circulars and Processes	
Sl. No: 04	Format 03-Lab workbook format

LABORATORY WORKBOOK

Exercise/Experiment Number: 05 a

Lab Code / Lab : CS8581
Program / Branch : B.E/CSE
Title of the exercise / experiment: Simulation of Address Resolution Protocol (ARP)

STEP 1: INTRODUCTION

a) OBJECTIVE OF THE EXERCISE/EXPERIMENT

Develop a java program for simulating Address Resolution Protocol (ARP).

STEP 2: ACQUISITION

b) Facilities/material required to do the exercise/experiment:

Sl. No.	Facilities/material required	Quantity
1	Pentium IV with 2 GB RAM 160 GB HARD Disk Monitor 1024 x 768 colour	40
2	C / C++ / Java / Python / Equivalent Compiler	40

c) Procedure for doing the exercise/experiment:

Client:

Steps	Description
1	Open command prombt
2	Import the java.net package.
3	Using socket connection is established between client and server.
4	Get the IP address to be converted into MAC address.

5	Send this IP address to server.
6	Server returns the MAC address to client.

Server:

Steps	Description
1	Open command prombt
2	Import the java.net package.
3	Accept the socket which is created by the client.
4	Server maintains the table in which IP and corresponding MAC addresses are stored
5	Read the IP address which is send by the client.
6	Map the IP address with its MAC address and return the MAC address to client.

Program:

Clientarp.java

```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientarp
        {
        public static void main(String args[])
        {

        t
        r
        y

        {

        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);
}
}
}
```

Serverarp.java

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverarp
{
public static void main(String args[])
{

t
r
y

{
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);
}
}
}
```

Output:

Enter the Logical address(IP):

165.165.79.1

The Physical Address is: 8A:BC:E3:FA

Result:

Thus the java program for simulating Address Resolution Protocol (ARP) was executed successfully.

INDEX DETAILS for Master File of Circulars and Processes	
Sl. No: 04	Format 03-Lab workbook format

LABORATORY WORKBOOK

Exercise/Experiment Number: 05 b

Lab Code / Lab : CS8581
Program / Branch : B.E/CSE
Title of the exercise / experiment: Simulation of Reverse Address Resolution Protocol (RARP)

STEP 1: INTRODUCTION

a) OBJECTIVE OF THE EXERCISE/EXPERIMENT

Develop a java program for simulating Reverse Address Resolution Protocol (RARP).

STEP 2: ACQUISITION

b) Facilities/material required to do the exercise/experiment:

Sl. No.	Facilities/material required	Quantity
1	Pentium IV with 2 GB RAM 160 GB HARD Disk Monitor 1024 x 768 colour	40
2	C / C++ / Java / Python / Equivalent Compiler	40

c) Procedure for doing the exercise/experiment:

Client:

Steps	Description
1	Open command prombt
2	Import the java.net package.
3	Using socket connection is established between client and server.
4	Get the MAC address to be converted into IP address

5	Send this MAC address to server.
6	Server returns the IP address to client.

Server :

Steps	Description
1	Open command prombt
2	Import the java.net package.
3	Server maintains the table in which MAC and corresponding IP addresses are stored.
4	Read the MAC address which is send by the client.
5	Map the MAC address with its IP address and return the IP address to client.

Program:

Clientrarp.java

```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientrarp
        {
public static void main(String args[])
{
try
        {
        DatagramSocket client=new DatagramSocket();
        InetAddress
        addr=InetAddress.getByName("127.0.0.1"); byte[]
        sendbyte=new byte[1024]; byte[]
        receivebyte=new byte[1024];
        BufferedReader in=new BufferedReader(new
        InputStreamReader(System.in));
        System.out.println("Enter the Physical address
        (MAC):"); String str=in.readLine();
        sendbyte=str.getBytes();
        DatagramPacket sender=new
        DatagramPacket(sendbyte,sendbyte.length,addr,1309);
        client.send(sender);
        DatagramPacket receiver=new
```

```
DatagramPacket(receivebyte,receivebyte.length);
client.receive(receiver);
String s=new String(receiver.getData());
System.out.println("The Logical Address is(IP): "+s.trim());
client.close();
}
catch(Exception e)
{
System.out.println(e);
}
}
}
```

Serverrarp.java

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

{
if(s.equals(mac[i]))
{
sendbyte=ip[i].getBytes();
DatagramPacket sender=new
```

```
DatagramPacket(sendbyte,sendbyte.length,add
r,port); server.send(sender); break;
}
}
break;
}
}
catch(Exception e)
{
System.out.println(e);
}
}
}
```

Output:

Enter the Physical address (MAC):

6A:08:AA:C2

The Logical Address is(IP): 165.165.80.80

Result:

Thus the java program for simulating Reverse Address Resolution Protocol (RARP) was executed successfully.

INDEX DETAILS for Master File of Circulars and Processes	
Sl. No: 04	Format 03-Lab workbook format

LABORATORY WORKBOOK

Exercise/Experiment Number: 06

Lab Code / Lab : CS8581
Program / Branch : B.E/CSE
Title of the exercise / experiment: Study of Network simulator (NS)

STEP 1: INTRODUCTION

a) OBJECTIVE OF THE EXERCISE/EXPERIMENT

study about Network Simulator – NS2.

STEP 2: ACQUISITION

Network Simulator (NS):

NS (from network simulator) is a name for a series of discrete event network simulators, specifically ns-1, ns-2 and ns-3. All of them are discrete-event computer network simulators, primarily used in research and teaching. ns-3 is free software, publicly available under the GNU GPLv2 license for research, development, and use.

The goal of the ns-3 project is to create an open simulation environment for computer networking research that will be preferred inside the research community:

- It should be aligned with the simulation needs of modern networking research.
- It should encourage community contribution, peer review, and validation of the software.

Since the process of creation of a network simulator that contains a sufficient number of high-quality validated, tested and actively maintained models requires a lot of work, ns-3 project spreads this workload over a large community of users and developers. **History ns-1**

The first version of ns, known as ns-1, was developed at Lawrence Berkeley National Laboratory (LBNL) in the 1995-97 timeframe by Steve McCanne, Sally Floyd, Kevin Fall, and other contributors. This was known as the LBNL Network Simulator, and derived from an earlier simulator known as REAL by S. Keshav. The core of the simulator was written in C++, with Tcl-based scripting of simulation scenarios.

ns-2

NS began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995 ns development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI. Currently ns development is support through DARPA with SAMAN and through NSF with CONSER, both in collaboration with other researchers including ACIRI. Ns has always included substantial contributions from other researchers, including wireless code from the UCB Daedelus and CMU Monarch projects and Sun Microsystems. For documentation on recent changes, see the version 2 change log. **ns-3**

A team led by Tom Henderson, George Riley, Sally Floyd, and Sumit Roy, applied for and received funding from the U.S. National Science Foundation (NSF) to build a replacement for ns-2, called ns-3. This team collaborated with the Planete project of INRIA at Sophia Antipolis, with Mathieu Lacage as the software lead, and formed a new open source project.

In the process of developing ns-3, it was decided to completely abandon backward-compatibility with ns-2. The new simulator would be written from scratch, using the C++ programming language. Development of ns-3 began in July 2006.

The first release, ns-3.1 was made in June 2008, and afterwards the project continued making quarterly software releases, and more recently has moved to three releases per year. ns-3 made its twenty first release (ns-3.21) in September 2014.

Current status of the three versions is:

- ns-1 is no longer developed nor maintained,
- ns-2 is not actively maintained (active development stopped in 2010 and is not accepted for publications anymore)
- ns-3 is actively developed (but not compatible for work done on ns-2). **Design**

ns-3 is built using C++ and Python with scripting capability. The ns-3 library is wrapped by Python thanks to the pybindgen library which delegates the parsing of the ns-3 C++ headers to gccxml and pygccxml to automatically generate the corresponding C++ binding glue. These automatically-generated C++ files are finally compiled into the ns-3 Python module to allow users to interact with the C++ ns-3 models and core through Python scripts. The ns-3 simulator features an integrated attribute-based system to manage default and per-instance values for simulation parameters. All of the configurable default values for parameters are managed by this system, integrated with command-

line argument processing, Doxygen documentation, and an XML-based and optional GTK-based configuration subsystem.

Simulation workflow

The general process of creating a simulation can be divided into several steps:

Topology definition: To ease the creation of basic facilities and define their interrelationships, ns-3 has a system of containers and helpers that facilitates this process.

Model development: Models are added to simulation (for example, UDP, IPv4, point-to-point devices and links, applications); most of the time this is done using helpers.

Node and link configuration: models set their default values (for example, the size of packets sent by an application or MTU of a point-to-point link); most of the time this is done using the attribute system.

Execution: Simulation facilities generate events, data requested by the user is logged.

Performance analysis: After the simulation is finished and data is available as a time-stamped event trace. This data can then be statistically analysed with tools like R to draw conclusions.

Graphical Visualization: Raw or processed data collected in a simulation can be graphed using tools like Gnuplot, matplotlib or XGRAPH.

Result:

Thus the study about Network Simulator – NS2 was done.

EX DETAILS for Master File of Circulars and Processes	
Sl. No: 04	Format 03-Lab workbook format

LABORATORY WORKBOOK

Exercise/Experiment Number: 07

Lab Code / Lab : CS8581
Program / Branch : B.E/CSE
Title of the exercise / experiment : Study of TCP/UDP performance using NS-2

STEP 1: INTRODUCTION

a) OBJECTIVE OF THE EXERCISE/EXPERIMENT

Study about Network Simulator – NS2.

STEP 2: ACQUISITION

b) Facilities/material required to do the exercise/experiment:

Sl. No.	Facilities/material required	Quantity
1	Pentium IV with 2 GB RAM 160 GB HARD Disk Monitor 1024 x 768 colour	40
2	C / C++ / Java / Python / Equivalent Compiler	40

TCP PERFORMANCE

a) Procedure for doing the exercise/experiment:

Client:

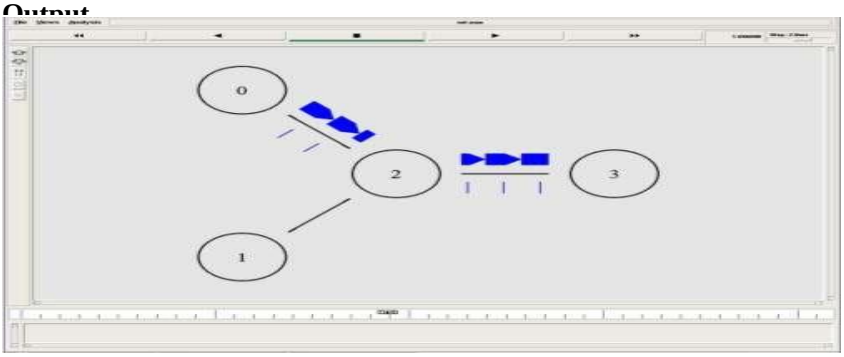
Steps	Description
1	Create a Simulator object.
2	Set routing as dynamic.
3	Open the trace and nam trace files.
4	Define the finish procedure.
5	Create nodes and the links between them.
6	Create the agents and attach them to the nodes.
7	Create the applications and attach them to the tcp agent.
8	Connect tcp and tcp sink.
9	Run the simulation.

PROGRAM:

```
set ns [new Simulator]
$ns color 0 Blue
$ns color 1 Red
$ns color 2 Yellow
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set f [open tcpout.tr w]
$ns trace-all $f
set nf [open tcpout.nam w]
$ns namtrace-all $nf
$ns duplex-link $n0 $n2 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 5Mb 2ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
$ns duplex-link-op $n0 $n2 orient right-up
$ns duplex-link-op $n1 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n2 $n3 queuePos 0.5
set tcp [new Agent/TCP]
$tcp set class_ 1
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $tcp
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 1.2 "$ftp start"
$ns at 1.35 "$ns detach-agent $n1 $tcp ; $ns detach-agent $n3 $sink"
$ns at 3.0
"finish" proc
finish {} {
        global ns f nf
 $ns flush-trace         close $f
        close $nf        puts "Running
```

```
nam.."  exec xgraph tcpout.tr -geometry
600x800 &       exec nam tcpout.nam &

        exit 0
}
$ns run
```

Output



UDP Performance

a) Procedure for doing the exercise/experiment:

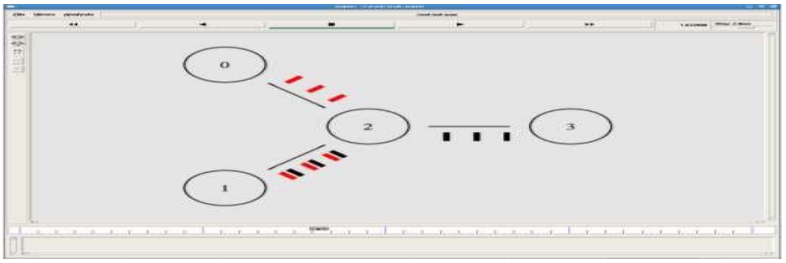
Steps	Description
1	Create a Simulator object.
2	Set routing as dynamic.
3	Open the trace and nam trace files.
4	Define the finish procedure.
5	Create nodes and the links between them.
6	Create the agents and attach them to the nodes.
7	Create the applications and attach them to the UDP agent.
8	Connect udp and null agents.
9	Run the simulation.

PROGRAM:

```
set ns [new Simulator]
$ns color 0 Blue
$ns color 1 Red
$ns color 2 Yellow
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set f [open udpout.tr w]
$ns trace-all $f
set nf [open udpout.nam w]
$ns namtrace-all $nf
$ns duplex-link $n0 $n2 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 5Mb 2ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
$ns duplex-link-op $n0 $n2 orient right-up
$ns duplex-link-op $n1 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n2 $n3 queuePos 0.5
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new
Application/Traffic/CBR]
$cbr0 attach-agent $udp0
set udp1 [new Agent/UDP]
$ns attach-agent $n3 $udp1
$udp1 set class_ 0 set cbr1
[new
Application/Traffic/CBR]
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
set null1 [new Agent/Null]
```

```
$ns attach-agent $n1 $null1
$ns connect $udp0 $null0
$ns connect $udp1 $null1
$ns at 1.0 "$cbr0 start"
$ns at 1.1 "$cbr1 start"
puts [$cbr0 set packetSize_]
puts [$cbr0 set interval_]
$ns at 3.0 "finish"
proc finish {} {
        global ns f nf
        $ns flush-trace
        close $f
        close $nf
        puts "Running nam.."
        exec nam udpout.nam &
        exit 0
}
$ns run
```

Output:



RESULT :

Thus the study of TCP/UDP performance is done successfully.

INDEX DETAILS for Master File of Circulars and Processes	
Sl. No: 04	Format 03-Lab workbook format

LABORATORY WORKBOOK

Exercise/Experiment Number: 08

Lab Code / Lab : CS8581
Program / Branch : B.E/CSE
Title of the exercise / experiment : Simulation of Distance Vector/ Link State Routing algorithm.

STEP 1: INTRODUCTION

c) OBJECTIVE OF THE EXERCISE/EXPERIMENT

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

STEP 2: ACQUISITION

d) Facilities/material required to do the exercise/experiment:

Sl. No.	Facilities/material required	Quantity
1	Pentium IV with 2 GB RAM 160 GB HARD Disk Monitor 1024 x 768 colour	40
2	C / C++ / Java / Python / Equivalent Compiler	40

PERFORMANCE

b) Procedure for doing the exercise/experiment:

Client:

Steps	Description
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.

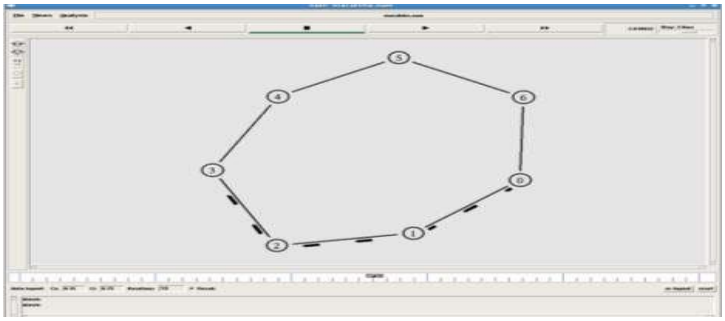
LINK STATE ROUTING PROTOCOL

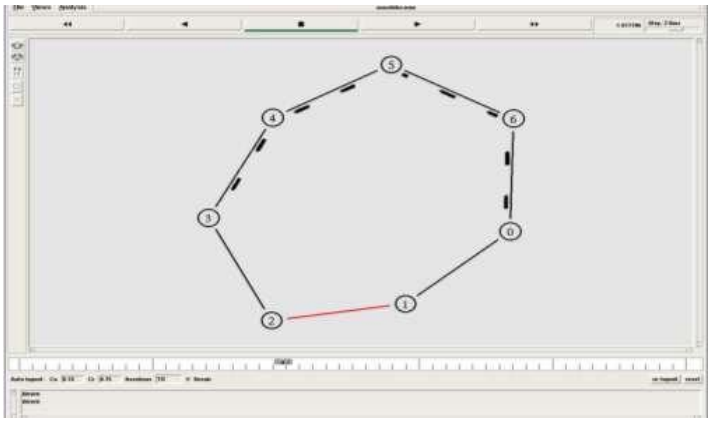
PROGRAM

```
set ns [new Simulator]
```

```
$ns rtproto LS
$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:





DISTANCE VECTOR ROUTING ALGORITHM

PERFORMANCE

a) Procedure for doing the exercise/experiment:

Client:

Steps	Description
1	Create a simulator object
2	Set routing protocol to Distance Vector routing
3	Trace packets on all links onto NAM trace and text trace file
4	Define finish procedure to close files, flush tracing and run NAM
5	Create eight nodes

6	Specify the link characteristics between nodes
7	Describe their layout topology as a octagon
8	Add UDP agent for node n1
9	Create CBR traffic on top of UDP and set traffic parameters.
10	Add a sink agent to node n4
11	Connect source and the sink 12. Schedule events as follows: a. Start traffic flow at 0.5 b. Down the link n3-n4 at 1.0 c. Up the link n3-n4 at 2.0 d. Stop traffic at 3.0 e. Call finish procedure at 5.0
12	Start the scheduler
13	Observe the traffic route when link is up and down
14	View the simulated events and trace file analyze it
15	Stop

PROGRAM

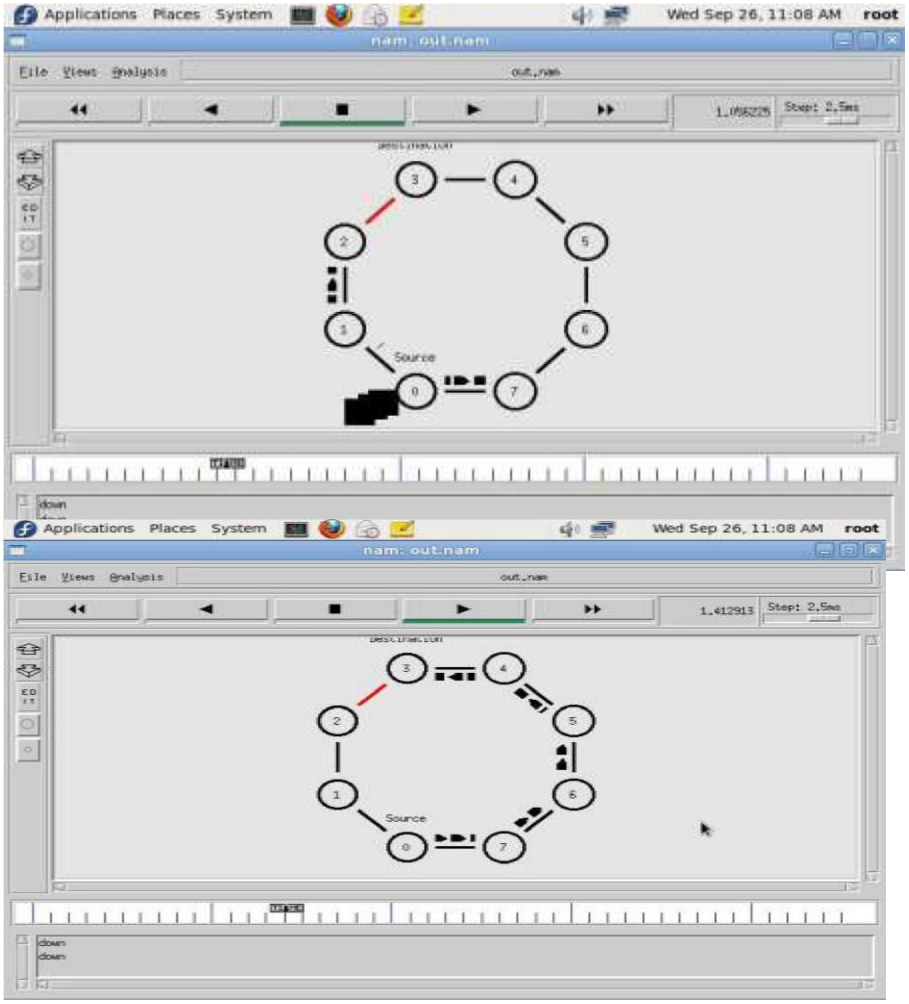
```
#Distance vector routing protocol – distvect.tcl
#Create a
simulator object
set ns [new
Simulator] #Use
distance vector
routing
$ns rtproto DV
#Open the nam
trace file set nf
[open out.nam
w]
$ns
namtrace-all
$nf # Open
tracefile set
nt [open
trace.tr w]
$ns trace-all
$nt
#Define 'finish' procedure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

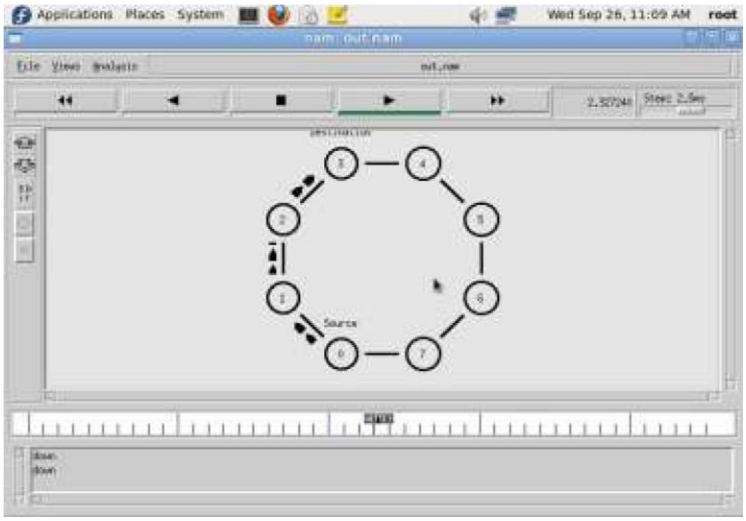
```
proc finish {}
{ global ns nf $ns
        flush-trace
        #Close the
        trace file
        close $nf
        #Execute nam on the trace file
        exec nam -a out.nam &
        exit 0
}
# Create 8 nodes
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
# Specify link characterestics
$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
```

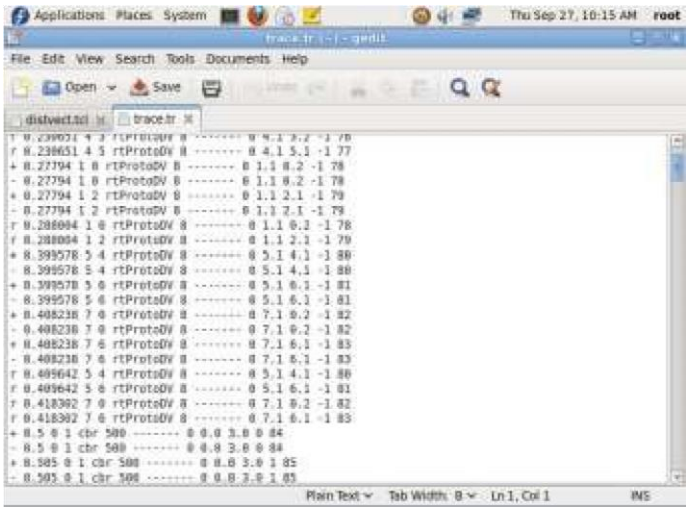
#Create a Null agent (a traffic sink) and attach it
to node n4 set null0 [new Agent/Null]
$ns attach-agent $n4 $null0
#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0
#Schedule events for the CBR agent and the network dynamics
$ns at 0.0 "$n1 label Source"
$ns at 0.0 "$n4 label 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







RESULT:

Thus the simulation for Distance vector and link state routing protocols was done using NS2.

INDEX DETAILS for Master File of Circulars and Processes	
Sl. No: 04	Format 03-Lab workbook format

LABORATORY WORKBOOK

Exercise/Experiment Number: 09

Lab Code / Lab : CS8581
Program / Branch : B.E/CSE

Title of the exercise / experiment : Performance evaluation of Routing protocols using Simulation tool.

STEP 1: INTRODUCTION

e) OBJECTIVE OF THE EXERCISE/EXPERIMENT

To write a ns2 program for implementing unicast routing protocol and multicasting routing protocol.

STEP 2: ACQUISITION

f) Facilities/material required to do the exercise/experiment:

Sl. No.	Facilities/material required	Quantity
1	Pentium IV with 2 GB RAM 160 GB HARD Disk Monitor 1024 x 768 colour	40
2	C / C++ / Java / Python / Equivalent Compiler	40

PERFORMANCE

A)UNICAST ROUTING PROTOCOL

c) Procedure for doing the exercise/experiment:

Steps	Description
1	Start the program.
2	Declare the global variables ns for creating a new simulator.
3	Set the color for packets.

4	Open the network animator file in the name of file2 in the write mode.
5	Open the trace file in the name of file 1 in the write mode.
6	Set the unicast routing protocol to transfer the packets in network.
7	Create the required no of nodes
8	Create the duplex-link between the nodes including the delay time,bandwidth and dropping queue mechanism.
9	Give the position for the links between the nodes.
10	Set a tcp reno connection for source node.
11	Set the destination node using tcp sink.
12	Setup a ftp connection over the tcp connection.
13	Down the connection between any nodes at a particular time.
14	Reconnect the downed connection at a particular time.
15	Define the finish procedure.
16	In the definition of the finish procedure declare the global variables ns, file1, and file2
17	Close the trace file and name file and execute the network animation file.
18	At the particular time call the finish procedure.
19	Stop the program.

PROGRAM:

```
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2
Red #Open
theTrace file
set file1 [open
out.tr w] $ns
trace-all $file1
#Open the NAM
trace file set file2
[open out.nam w]
$ns namtrace-all
$file2 #Define a
'finish' procedure
proc finish {}
{
```

```
        global ns
file1 file2
$ns flush-trace
close $file1
close $file2
exec nam
out.nam &
exit 3
}
# Next line should be commented out to have the static routing
$ns
rtproto
DV
#Create
six
nodes
set n0
[$ns
node]
set n1 [$ns node]
set n2 [$ns node]
set n4 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
#Create links between the nodes
$ns duplex-link $n0 $n1 0.3Mb 10ms DropTail
$ns duplex-link $n1 $n2 0.3Mb 10ms DropTail
$ns duplex-link $n2 $n3 0.3Mb 10ms DropTail
$ns duplex-link $n1 $n4 0.3Mb 10ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 10ms DropTail
$ns duplex-link $n4 $n5 0.5Mb 10ms DropTail

#Give node position (for NAM)
$ns duplex-link-op  $n0 $n1 orient right
$ns duplex-link-op  $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient up
$ns duplex-link-op $n1 $n4 orient up-left
$ns duplex-link-op  $n3 $n5 orient left-up
$ns duplex-link-op  $n4 $n5 orient right-up

#Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n5 $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

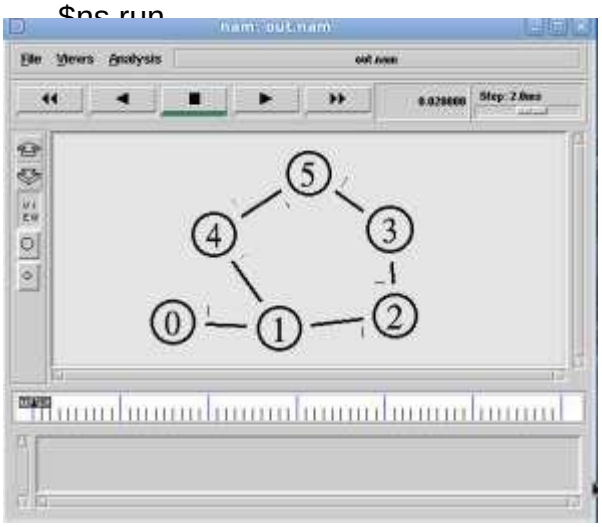
$ns rtmodel-at 1.0 down $n1 $n4

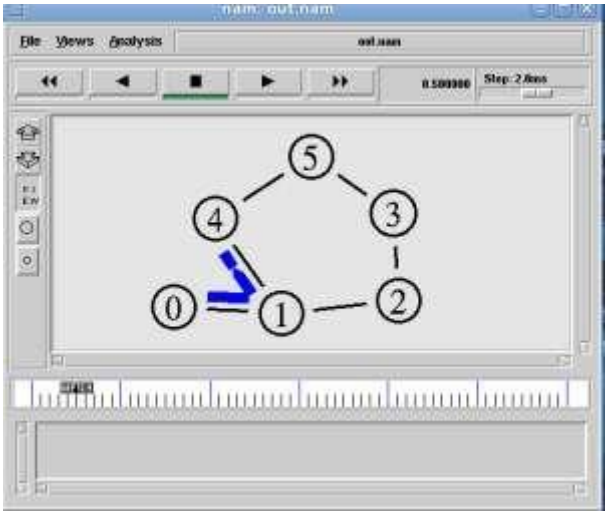
$ns rtmodel-at 4.5 up $n1 $n4

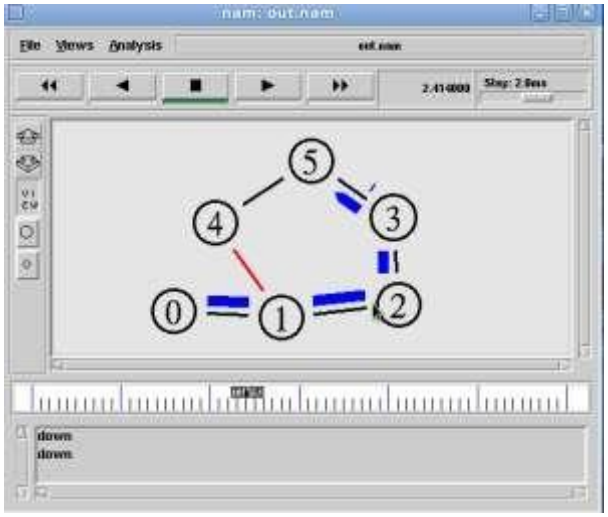
$ns at 0.1 "$ftp start"

$ns at 6.0 "finish"

$ns run







B)MULTICASTING ROUTING PROTOCOL

PERFORMANCE

d) Procedure for doing the exercise/experiment:
Client:

Steps	Description
1	Start the program.
2	Declare the global variables ns for creating a new simulator.
3	Set the color for packets.
4	Open the network animator file in the name of file 2 in the write mode.
5	Open the trace file in the name of file 1 in the write mode.
6	Set the multicast routing protocol to transfer the packets in network.
7	Create the multicast capable no of nodes.
8	Create the duplex link between the nodes including the delay time ,bandwidth and dropping queue mechanism.
9	Give the position for the links between the nodes.

10	Set a udp connection for source node.
11	Set the destination node ,port and random false for the source and destination files.
12	Setup a traffic generator CBR for the source and destination files.
13	Down the connection between any nodes at a particular time.
14	Create the receive agent for joining and leaving if the nodes in the group.
15	Define the finish procedure.
16	In the definition of the finish procedure declare the global variables.
17	Close the trace file and namefile and execute the network animation file.
18	At the particular time call the finish procedure.
19	Stop the program.

PROGRAM:

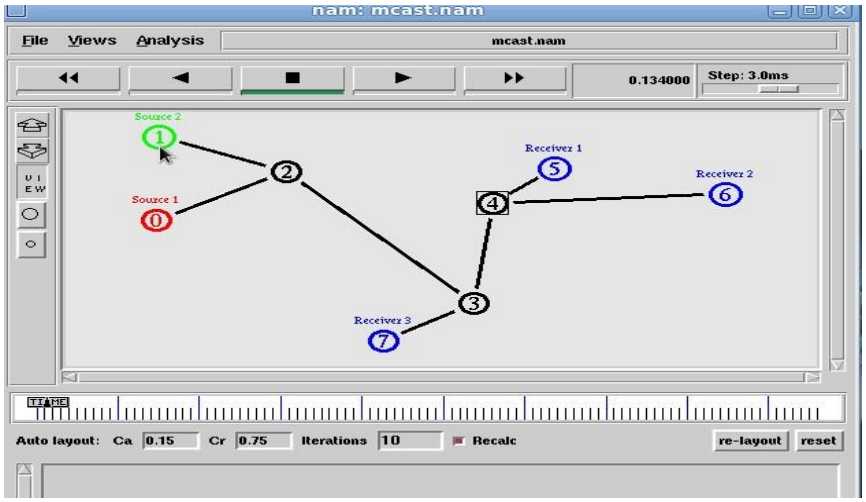
```
# Create scheduler
#Create an event scheduler wit multicast
turned on set ns [new Simulator -
multicast on]
#$ns multicast
#Turn on Tracing
```

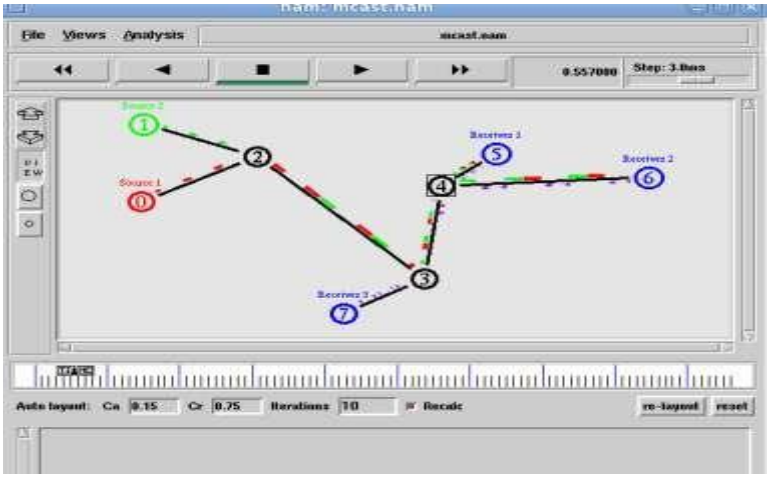
```
set tf [open output.tr w]
$ns trace-all $tf
# Turn on nam Tracing
set fd [open mcast.nam w]
$ns namtrace-all $fd
# Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]

# Create links
$ns duplex-link $n0 $n2 1.5Mb 10ms DropTail
$ns duplex-link $n1 $n2 1.5Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
$ns duplex-link $n3 $n4 1.5Mb 10ms DropTail
$ns duplex-link $n3 $n7 1.5Mb 10ms DropTail
$ns duplex-link $n4 $n5 1.5Mb 10ms DropTail
$ns duplex-link $n4 $n6 1.5Mb 10ms DropTail

# Routing protocol: say distance vector
#Protocols: CtrMcast, DM, ST, BST
set mproto DM
set mrthandle [$ns mrtproto $mproto {}]

# Allocate group
addresses set
group1 [Node
allocaddr] set
group2 [Node
allocaddr]

# UDP Transport agent for the
traffic source set udp0 [new
Agent/UDP]
$ns attach-agent $n0 $udp0
$udp0 set dst_addr_
$group1 $udp0 set dst_port_
0 set cbr1 [new
Application/Traffic/CBR]
```

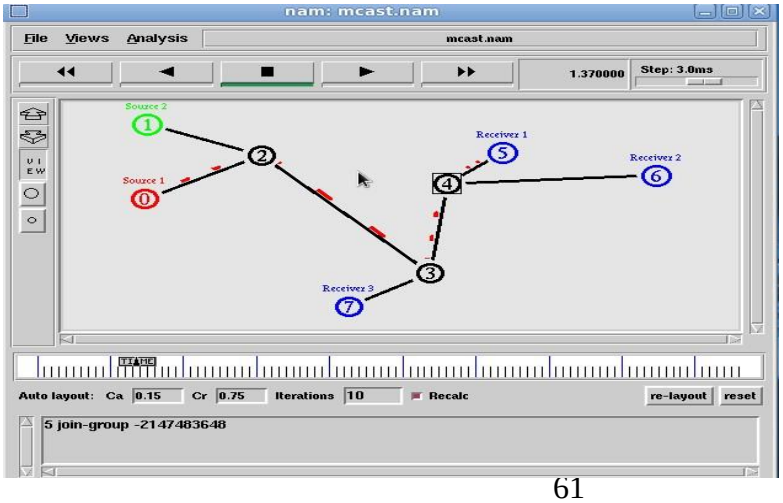
```
$cbr1 attach-agent $udp0

# Transport agent for the traffic source
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
$udp1 set dst_addr_ $group2
$udp1 set dst_port_ 0
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp1

# Create receiver
set rcvr1 [new Agent/Null]
$ns attach-agent $n5 $rcvr1
$ns at 1.0 "$n5 join-group $rcvr1 $group1"
set rcvr2 [new Agent/Null]
$ns attach-agent $n6 $rcvr2
$ns at 1.5 "$n6 join-group $rcvr2 $group1"
set rcvr3 [new Agent/Null]
$ns attach-agent $n7 $rcvr3
$ns at 2.0 "$n7 join-group $rcvr3 $group1"
set rcvr4 [new Agent/Null]
$ns attach-agent $n5 $rcvr1
$ns at 2.5 "$n5 join-group $rcvr4 $group2"
set rcvr5 [new Agent/Null]
$ns attach-agent $n6 $rcvr2
$ns at 3.0 "$n6 join-group $rcvr5 $group2"
set rcvr6 [new Agent/Null]
$ns attach-agent $n7 $rcvr3
$ns at 3.5 "$n7 join-group $rcvr6 $group2"
$ns at 4.0 "$n5 leave-group $rcvr1 $group1"
$ns at 4.5 "$n6 leave-group $rcvr2 $group1"
$ns at 5.0 "$n7 leave-group $rcvr3 $group1"
$ns at 5.5 "$n5 leave-group $rcvr4 $group2"
$ns at 6.0 "$n6 leave-group $rcvr5 $group2"
$ns at 6.5 "$n7 leave-group $rcvr6 $group2"

# Schedule events
$ns at 0.5
"$cbr1 start" $ns
at 9.5 "$cbr1
stop" $ns at 0.5
"$cbr2 start"
$ns at 9.5 "$cbr2 stop"
```

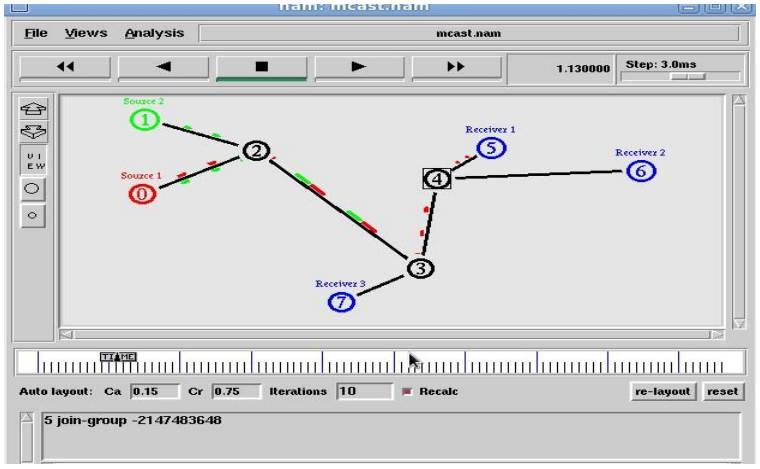
```
#post-processing
$ns at 10.0 "finish"
proc finish {}
{

        global ns tf
        $ns flush-trace
        close $tf
        exec nam mcast.nam &
        exit 0
}

# For nam
#Colors for packets from two mcast groups
$ns color 10 red
$ns color 11 green
$ns color 30 purple
$ns color 31 green

# Manual layout: order of the link is significant!
#$ns duplex-link-op $n0 $n1 orient right
#$ns duplex-link-op $n0 $n2 orient right-up
#$ns duplex-link-op $n0 $n3 orient right-down
# Show queue on simplex link n0->n1
#$ns duplex-link-op $n2 $n3 queuePos 0.5

# Group 0 source
$udp0 set fid_ 10
$n0 color red
$n0 label "Source 1"
# Group 1
source $udp1
set fid_ 11
$n1 color green
$n1 label "Source 2"
$n5 label "Receiver 1" $n5 color blue
$n6 label "Receiver 2"
$n6 color blue
$n7 label "Receiver 3"
$n7 color blue
```

```
#$n2 add-mark m0 red
#$n2 delete-mark m0"

# Animation rate
$ns set-animation-rate 3.0ms
$ns run
```









RESULT:

Thus the case study about the different routing algorithms to select the network path with its optimum and economical during data transfer is done.

INDEX DETAILS for Master File of Circulars and Processes	
Sl. No: 04	Format 03-Lab workbook format

LABORATORY WORKBOOK

Exercise/Experiment Number: 10

Lab Code / Lab **:** CS8581
Program / Branch **:** B.E/CSE
Title of the exercise / experiment ***:*** Simulation of error correction code (like CRC).

STEP 1: INTRODUCTION

g) OBJECTIVE OF THE EXERCISE/EXPERIMENT

To implement error checking code using java.

STEP 2: ACQUISITION

h) Facilities/material required to do the exercise/experiment:

Sl. No.	Facilities/material required	Quantity
1	Pentium IV with 2 GB RAM 160 GB HARD Disk Monitor 1024 x 768 colour	40
2	C / C++ / Java / Python / Equivalent Compiler	40

PERFORMANCE

A)UNICAST ROUTING PROTOCOL

e) Procedure for doing the exercise/experiment:

Steps	Description
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) => 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:

```
 Import java.io.*;
class crc_gen
{
public static void main(String args[]) throws IOException {
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in)); int[] data; int[] div; int[] divisor;
int[] rem; int[] crc;
int data_bits, divisor_bits, tot_length;
System.out.println("Enter number of data bits : ");
data_bits=Integer.parseInt(br.readLine()); data=new int[data_bits];
System.out.println("Enter data bits
: "); for(int i=0; i<data_bits; i++)
data[i]=Integer.parseInt(br.readLine());
System.out.println("Enter number of bits in divisor : ");
divisor_bits=Integer.parseInt(br.readLine()); divisor=new int[divisor_bits];
System.out.println("Enter Divisor bits
: "); for(int i=0; i<divisor_bits; i++)
divisor[i]=Integer.parseInt(br.re
adLine());
System.out.print("Data bits are
: "); for(int i=0; i< data_bits; i++)
System.out.print(data[i]);
System.out.println();
System.out.print("divisor bits are
: "); for(int i=0; i< divisor_bits;
i++) System.out.print(divisor[i]);
```

```
System.out.println();
 */
tot_length=data_bits+divisor
_bits-1; div=new
int[tot_length];
rem=new int[tot_length];
crc=new int[tot_length];
/*------------------ CRC GENERATION-----------------------*/
for(int i=0;i<data.length;i++)
 div[i]=data[i];
System.out.print("Dividend (after appending 0's) are : "); for(int i=0; i< div.length; i++)
System.out.print(div[i]);
System.out.println();
for(int j=0; j<div.length; j++){
rem[j] = div[j];
}
rem=divide(div, divisor, rem);
 for(int i=0;i<div.length;i++)
{

//append dividend and remainder

crc[i]=(div[i]^rem[i]);
}
 System.out.println();
System.out.println("CRC code : ");
for(int i=0;i<crc.length;i++)
System.out.print(crc[i]);

/*--------------------ERROR DETECTION----------------------*/
System.out.println();
System.out.println("Enter CRC code of "+tot_length+" bits : "); for(int i=0; i<crc.length; i++)
crc[i]=Integer.parseInt(br.readLine());
System.out.print("crc bits are : ");
for(int i=0; i< crc.length; i++)
System.out.print(crc[i]);
System.out.println();
for(int j=0; j<crc.length; j++){
rem[j] = crc[j];



} rem=divide(crc,
divisor, rem); for(int
i=0; i< rem.length;
i++)
```

```
{
if(rem
[i]!=0)
{
System.out.println("Er
ror"); break; }
if(i==rem.length-1)
System.out.println("N
o Error");
}
System.out.println("THANK YOU.... :)");
}
static int[] divide(int div[],int divisor[], int rem[])
{
int cur=0;
while(true)
{
for(int i=0;i<divisor.length;i++)
rem[cur+i]=(rem[cur+i]^divisor[i]);
while(rem[cur]==0 && cur!=rem.length-1)
cur++;
if((rem.length-cur)<divisor.length)
break;
}
return rem;
}
}
```

OUTPUT :

```
Enter number of data bits :
7
Enter data bits :
1
0
1
1
0
0
1
Enter number of bits in divisor :
3
Enter Divisor bits :
1
0
1
Dividend (after appending 0's) are : 101100100
```

CRC code :
101100111
Enter CRC code of 9 bits :
1
0
1
1
0
0
1
0 1 crc bits are
: 101100101

Error Thank you

Result:

Thus the above program for error checking code using was executed successfully.