CS8581 **NETWORKS LABORATORY**

L	T	P	C
0	0	4	2

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

OUTCOMES:

Upon Completion of the course, the students will be able to:

- Implement various protocols using TCP and UDP.
- Compare the performance of different transport layer protocols.
- Use simulation tools to analyze the performance of various network protocols.
- Analyze various routing algorithms.
- Implement error correction codes.

LIST OF EQUIPMENT FOR A BATCH OF 30 STUDENTS:

LABORATORY REQUIREMENT FOR BATCH OF 30 STUDENTS:

HARDWARE:

1. Standalone desktops 30 Nos

SOFTWARE:

2. C / C++ / Java / Python / Equivalent Compiler 30

Network simulator like NS2/Glomosim/OPNET/ Packet Tracer / Equivalent

INDEX

Ex.No:1 Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute. Capture ping and traceroute PDUs using a network protocol analyzer and examine

AIM:

To write the java program for simulating Traceroute command

THEORY:

Traceroute is a network diagnostic tool used to track the pathway taken by a packet on an IP network from source to destination. Traceroute also records the time taken for each hop the packet makes during its route to the destination. Traceroute uses Internet Control Message Protocol (ICMP) echo packets with variable time to live (TTL) values. The response time of each hop is calculated. To guarantee accuracy, each hop is queried multiple times (usually three times) to better measure the response of that particular hop. Traceroute sends packets with TTL values that gradually increase from packet to packet, starting with TTL value of one. Routers decrement TTL values of packets by one when routing and discard packets whose TTL value has reached zero, returning the ICMP error message ICMP Time Exceeded. For the first set of packets, the first router receives the packet, decrements the TTL value and drops the packet because it then has TTL value zero. The router sends an ICMP Time Exceeded message back to the source. The next set of packets are given a TTL value of two, so the first router forwards the packets, but the second router drops them and replies with ICMP Time Exceeded. Proceeding in this way, traceroute uses the returned ICMP Time Exceeded messages to build a list of routers that packets traverse, until the destination is reached and returns an ICMP Echo Reply message. The timestamp values returned for each router along the path are the delay (latency) values, typically measured in milliseconds for each packet.The sender expects a reply within a specified number of seconds.

ALGORITHM:

1. Start
2. Create a UDP packet from the source to destination with a TTL = 1
3. The UDP packet reaches the first router where the router decrements the value of TTL by 1, thus making our UDP packet's TTL = 0 and hence the packet gets dropped.
4. As the packet got dropped, it sends an ICMP message (Time exceeded) back to the source.
5. Router's address and the time taken for the round-trip are noted.
6. It sends two more packets in the same way to get an average value of the round-trip time.
7. The steps 1 to 5 repeated until the destination has been reached. But for each time, the TTL is incremented by 1 when the UDP packet is to be sent to next router/host.
8. Once the destination is reached, an ICMP message (this time – Destination Unreachable) is sent back to the source .
9. Receiving Destination Unreachable message, display the details of the route traced.
10. Stop.

PROGRAM:

```
import java.net.*;
import java.io.*;
class Whois extends Thread
{
	public String traceCmdUnix = "tracert -h 10 ;
	private StringBuffer result = null;
	private void pingCmd (String command)
	{
		result = new StringBuffer ();
		try {
			Process p;
			p = Runtime.getRuntime().exec(command);
			readResult(p.getInputStream());
			p.destroy();
		}
		catch (Exception e) {
			e.printStackTrace();
		}
	}
	private void readResult (InputStream in)
	{
		String line = null;
		BufferedReader br = new BufferedReader(new InputStreamReader(in));
		try {
			while (!this.isInterrupted() && (line = br.readLine()) != null)
			{
				//this.doResultLine(line+"\n");
				System.out.println(line);
			}
		}
		catch (IOException e) {
			e.printStackTrace();
		}
		finally {
			try {
				br.close();
			}
			catch (IOException e) {}
		}
		if (this.isInterrupted())
			System.out.println("*** Canceld ***");
}
```

```
public static void main (String args[]) throws Exception
{
        Whois MyObj = new Whois();
        MyObj.traceCmdUnix = MyObj.traceCmdUnix + " " + "www.yahoo.com" ;
        MyObj.pingCmd(MyObj.traceCmdUnix);
}}
```

OUTPUT:

```
C:\>tracert -d 5 -w 600 www.yahoo.com
Tracing route to www.yahoo.akadns.net [68.142.226.39]
over a maximum of 30 hops:
1 <10 ms <10 ms <10 ms 10.129.210.100
2 * * * Request timed out.
3 * * * Request timed out.
4 * * * Request timed out.
5 * * * Request timed out.
6 * * * Request timed out.
7 * * * Request timed out.
8 * * * Request timed out.
9 * * * Request timed out.
10 * * * Request timed out.
```

POST LAB VIVA QUESTIONS:

1. Define point to point connection.
2. Define multipoint connection
3. Explain the difference between an unspecified passive open and a fully specified passive open
4. Define Subnetting.
5. Whar are the adaptive routing algorithms.
6. What is the main reason for IPV6 being developed?

RESULT:

Thus the program was implementing to simulate ping and traceroute commands executed successfully.

Ex.No: 2 Write a HTTP web client program to download a web page using TCP sockets

AIM:

To write a java program for socket for HTTP for web page upload and download .

THEORY:

HTTP means **H**yper**T**ext **T**ransfer **P**rotocol. HTTP is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page. The other main standard that controls how the World Wide Web works is HTML, which covers how Web pages are formatted and displayed. HTTP functions as a request–response protocol in the client–server computing model. A web browser, for example, may be the client and an application running on a computer hosting a website may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a responsemessage to the client. The response contains completion status information about the request and may also contain requested content in its message body.

PRE LAB VIVA QUESTIONS:

1. Compare HTTP and FTP
2. What do you mean by active web page?
3. What is web browser
4. What are the four Main properties of HTTP?
5. Describe why HTTP is designed as a stateless protocol.

ALGORITHM:

Client:

1. Start.
2. Create socket and establish the connection with the server.
3. Read the image to be uploaded from the disk
4. Send the image read to the server
5. Terminate the connection
6. Stop.

Server:

1. Start
2. Create socket, bind IP address and port number with the created socket and make server a listening server.
3. Accept the connection request from the client

4. Receive the image sent by the client.
5. Display the image.
6. Close the connection.
7. Stop.

PROGRAM

Client

```
import javax.swing.*;
import java.net.*;
import java.awt.image.*;
import javax.imageio.*;
import java.io.*;
import java.awt.image.BufferedImage; import
java.io.ByteArrayOutputStream; import
java.io.File;
import java.io.IOException; import
javax.imageio.ImageIO;
public class Client
{
        public static void main(String args[]) throws Exception
        {
                Socket soc;
                BufferedImage img = null;
                soc=new
            Socket("localhost",4000);
                System.out.println("Client is running.
        ");
                try {
                        System.out.println("Reading image from disk. ");
                        img = ImageIO.read(new File("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.

POST LAB VIVA QUESTIONS:

1. Define WWW?
2. What are the four groups of HTTP Headers?
3. What is URL.
4. What happens if ACK/NACK corrupted?
5. What is congestion
6. Write the services provided by the TCP?

RESULT:

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

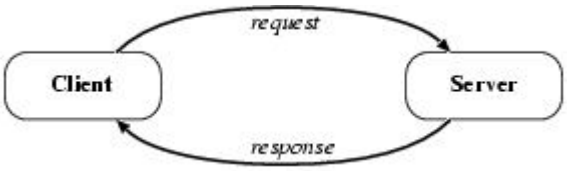
Ex.No: 3 Applications using TCP sockets like: Echo client and echo server, Chat and File Transfer

AIM

To write a java program for application using TCP Sockets Links

THEORY:

In the TCP Echo client a socket is created. Using the socket a connection is made to the server using the connect() function. After a connection is established, we send messages input from the user and display the data received from the server using send() and read() functions. In the TCP Echo server, we create a socket and bind to a advertized port number. After binding the process listens for incoming connections. Then an infinite loop is started to process the client requests for connections. After a connection is requested, it accepts the connection from the client machine and forks a new process. The new process receives data from the lient using recv() function and echoes the same data using the send() function. Please note hat this server is capable of handling multiple clients as it forks a new process for every client trying to connect to the server. TCP socket routines enable reliable IP communication using the transmission control protocol (TCP). This section describes the implementation of the Transmission Control Protocol (TCP) in the Network Component. TCP runs on top of the Internet Protocol (IP). TCP is a connection-oriented and reliable, full duplex protocol supporting a pair of byte streams, one for each direction. A TCP connection must be established before exchanging data. TCP retransmits data that do not reach the final destination due to errors or data corruption. Data is delivered in the sequence of its transmission



PRE LAB VIVA QUESTIONS:

1. Define – Socket
2. Define the three states of TCP **Connection** establishment and termination.
3. List the types of sockets
4. How a socket is uniquely identified?\
5. What is Active open
6. What is socket abstraction ?
7. Compare various socket address structures

ALGORITHM

Client

1. Start
2. Create the TCP socket
3. Establish connection with the server
4. Get the message to be echoed from the user
5. Send the message to the server
6. Receive the message echoed by the server
7. Display the message received from the server
8. Terminate the connection
9. Stop

Server

1. Start
2. Create TCP socket, make it a listening socket
3. Accept the connection request sent by the client for connection establishment
4. Receive the message sent by the client
5. Display the received message
6. Send the received message to the client from which it receives
7. Close the connection when client initiates termination and server becomes a listening server, waiting for clients.
8. Stop.

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:\Program Files\Java\jdk1.5.0\bin>javac EServer.java
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
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!
```

B.Chat

ALGORITHM

Client

1. Start
2. Create the UDP datagram socket
3. Get the request message to be sent from the user
4. Send the request message to the server
5. If the request message is "END" go to step 10
6. Wait for the reply message from the server
7. Receive the reply message sent by the server
8. Display the reply message received from the server
9. Repeat the steps from 3 to 8
10. Stop

Server

1. Start
2. Create UDP datagram socket, make it a listening socket
3. Receive the request message sent by the client
4. If the received message is "END" go to step 10
5. Retrieve the client's IP address from the request message received
6. Display the received message
7. Get the reply message from the user
8. Send the reply message to the client
9. Repeat the steps from 3 to 8.
10. Stop.

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
```

C. File Transfer

ALGORITHM

Client

1. Start
2. Create the TCP socket
3. Establish the connection with thw server
4. Get the name of the requested file from the user
5. Send the requested file name to the server
6. Get the new name for storing file content to be received from the user.
7. Create a file by new name in the write mode
8. Wait for the transfer of file from the server
9. Receive the file content sent by the server and store the contents in the file created in write mode
10. Display the file content received from the server
11. Stop

Server

1. Start
2. Create TCP socket, make it a listening socket
3. Accept the connection request from the client for connection establishment
4. Receive the name of the requested file from the client
5. Display the file content
6. Transfer the file content of the requestedvfile to the client
7. Close the connection when the client initiates connection termination
8. 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
```

POST LAB VIVA QUESTIONS:

1. State the differences between TCP and UDP
2. What is Passive open?
3. \What is half-close?
4. What are well-known ports?
5. What are Dynamic/ Private / Ephemeral ports?
6. What are registered ports?
7. What are Socket Address Structures?

RESULT:

Thus the java application program using TCP Sockets was developed and executed successfully.

Ex.No: 4 **Simulation of DNS using UDP sockets**

AIM

To write a java program for DNS application

THEORY:

The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols.

PRE LAB VIVA QUESTIONS:

1. What is the Domain name System Responsible for?
2. Why do we need a Domain Name System?
3. What role does the DNS Resolver play in the DNS System.
4. How does a DNS Resolver bootstrap the Domain Name Look up Process.
5. What is a Post office Protocol/
6. List the Two types of DNS message.
7. Define SNMP?
8. What type of transport protocol is used for DNS.
9. What is VPN?

ALGORITHM

Server

1. Start
2. Create UDP datagram socket
3. Create a table that maps host name and IP address
4. Receive the host name from the client
5. Retrieve the client's IP address from the received datagram
6. Get the IP address mapped for the host name from the table.
7. Display the host name and corresponding IP address
8. Send the IP address for the requested host name to the client
9. Stop.

Client

1. Start
2. Create UDP datagram socket.
3. Get the host name from the client

4. Send the host name to the server
5. Wait for the reply from the server
6. Receive the reply datagram and read the IP address for the requested host name
7. Display the IP address.
8. Stop.

PROGRAM

DNS Server

```
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

```
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
java udpdnsclient
Enter the hostname : youtube.com
IP Address: Host Not Found
```

RESULT:

Thus the java application program using UDP Sockets to implement DNS was developed and executed successfully

Ex.No:5 Write a code simulating ARP /RARP protocols

AIM:

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

THEORY:

Address Resolution Protocol (ARP) is a low-level network protocol for translating network layer addresses into link layer addresses.ARP lies between layers 2 and 3 of the OSI model, although ARP was not included in the OSI framework and allows computers to introduce each other across a network prior to communication.Because protocols are basic network communication units, address resolution is dependent on protocols such as ARP, which is the only reliable method of handling required tasks. When configuring a new network computer, each system is assigned an Internet Protocol (IP) address for primary identification and communication. A computer also has a unique media access control (MAC) address identity. Manufacturers embed the MAC address in the local area network (LAN) card. The MAC address is also known as the computer's physical address.

PRE LAB VIVA QUESTIONS:

1. What is the function of ARP & RARP?
2. What are the functions of the IP?
3. Difference between ARP and RARP.
4. What is the benefits of DHCP?

ALGORITHM:

Client

1. Start the program
2. Create socket and establish connection with the server.
3. Get the IP address to be converted into MAC address from the user.
4. Send this IP address to server.
5. Receive the MAC address for the IP address from the server.
6. Display the received MAC address
7. Terminate the connection

Server

1. Start the program
2. Create the socket, bind the socket created with IP address and port number and make it a listening socket.
3. Accept the connection request when it is requested by the client.
4. Server maintains the table in which IP and corresponding MAC addresses are stored.

5. Receive the IP address sent by the client.
6. Retrieve the corresponding MAC address for the IP address and send it to the client.
7. Close the connection with the client and now the server becomes a listening server waiting for the connection request from other clients
8. Stop

PROGRAM

Client:

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

Server:

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

```
        {
        try{
                ServerSocket obj=new
                ServerSocket(139); Socket
                obj1=obj.accept();
                while(true)
                {
                        DataInputStream din=new DataInputStream(obj1.getInputStream());
                        DataOutputStream dout=new DataOutputStream(obj1.getOutputStream());
                        String str=din.readLine();
                        String ip[]={"165.165.80.80","165.165.79.1"};
                        String mac[]={"6A:08:AA:C2","8A:BC:E3:FA"};
                        for(int i=0;i<ip.length;i++)
                        {
                                if(str.equals(ip[i]))
                                {
                                        dout.writeBytes(mac[i]+'\n');
                                        break;
                                }
                        }
                        obj.close();
                }
        }
        catch(Exception e)
        {
                System.out.println(e);
        }}
}
```

Output:

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

(b) Program for Reverse Address Resolution Protocol (RARP) using UDP

AIM:

To write a java program for simulating RARP protocols using UDP.

THEORY:

Reverse Address Resolution Protocol (RARP) is an obsolete computer networking protocol used by a client computer to request its Internet Protocol(IPv4) address from a computer network, when all it has available is its link layer or hardware address, such as a MAC address RARP requires one or more server hosts to maintain a database of mappings of Link Layer addresses to their respective protocol addresses. Media Access Control (MAC) addresses need to be individually configured on the servers by an administrator. RARP is limited to serving only IP addresses.Reverse ARP differs from the Inverse Address Resolution Protocol which is designed to obtain the IP address associated with a local Frame Relay data link connection identifier. InARP is not used in Ethernet.

ALGORITHM:

Client

1. Start the program
2. Create datagram socket
3. Get the MAC address to be converted into IP address from the user.
4. Send this MAC address to server using UDP datagram.
5. Receive the datagram from the server and display the corresponding IP address.
6. Stop

Server

1. Start the program.
2. Server maintains the table in which IP and corresponding MAC addresses are stored.
3. Create the datagram socket
4. Receive the datagram sent by the client and read the MAC address sent.
5. Retrieve the IP address for the received MAC address from the table.
6. Display the corresponding IP address.
7. Stop

PROGRAM:

Client:

```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientrarp12
{
        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=newDatagramPacket(sendbyte,sendbyte.length,addr,1309);
                client.send(sender);
                DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);
                client.receive(receiver);
                String s=new String(receiver.getData());
                System.out.println("The Logical Address is(IP): "+s.trim());
                client.close();
        }
        catch(Exception e)
        {
                System.out.println(e);
        }}
}
```

Server:

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverrarp12
{
public static void main(String args[])
{
```

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

Output:

```
I:\ex>java Serverrarp12
I:\ex>java Clientrarp12
Enter the Physical address (MAC):
6A:08:AA:C2
The Logical Address is(IP): 165.165.80.80
```

POST LAB VIVA QUESTIONS:

1. What do you mean by MTU (Maximum transfer Unit)?
2. What is IPC? Name three techniques.
3. What is the multiplexing
4. What is ARP Cache?
5. What does TTL shows?

RESULT :

Thus the program for implementing to display simulating ARP and RARP protocols was executed successfully and output is verified.

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

AIM:

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

PRE LAB VIVA QUESTIONS:

1. What protocols does ns support?
2. What should you do to compile ns to reflect my changes if you've modified some .cc or .h files?
3. Name the factors that affect the performance of the network.
4. Define the terms : Unicast, multicast and Broadcast
5. What are the four files on the NS2 simulator?

NET WORK SIMULATOR (NS2)

Ns Overview

- Ns Status
- Periodical release (ns-2.26, Feb 2003)
- Platform support
- FreeBSD, Linux, Solaris, Windows and Mac

Ns functionalities

Routing, Transportation, Traffic sources,Queuing disciplines, QoS

Congestion Control Algorithms

- Slow start
- Additive increase/multiplicative decrease
- Fast retransmit and Fast recovery

Case Study: A simple Wireless network.

Ad hoc routing, mobile IP, sensor-MAC

Tracing, visualization and various utilitie

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. NetSim (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

Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their 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 transportlayer 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

Throughput 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. Throughput is usually measured inbitsper second (bit/s orbps), and sometimes in data packets per second or data packets per time slot. This measures 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 highbandwidths. 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.

Congestion control Algorithms

Slow-start is used in conjunction with other algorithms to avoid sending more data than the network is capable of transmitting, that is, to avoid causing network congestion. The additive increase/multiplicative decrease (AIMD) algorithm is a feedback control algorithm. AIMD combines linear growth of the congestion window with an exponential reduction when a congestion takes place. Multiple flows using AIMD congestion control will eventually converge

to use equal amounts of a contended link. Fast Retransmit is an enhancement to TCP that reduces the time a sender waits before retransmitting a lost segment.

POST LAB VIVA QUESTIONS:

1. What is the use of a tr file?
2. What does $1,$2,..... indicate in the awk file?
3. How do we increase the throughput?
4. What is a node?
5. What is point to point link?

Result:

Thus we have Studied Network simulator (NS) and Simulation of Congestion Control Algorithms using NS.

Ex.No: 7 Study of TCP/UDP performance using Simulation tool.

AIM:

To simulate the performance of TCP/UDP using NS2.

THEORY:

The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Therefore, the entire suite is commonly referred to as TCP/IP. TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating by an IP network. Major Internet applications such as the World Wide Web, email, remote administration, and file transfer rely on TCP. Applications that do not require reliable data stream service may use the User Datagram Protocol (UDP), which provides a connectionless datagram service that emphasizes reduced latency over reliability.

PRE LAB VIVA QUESTIONS:

1. What is TCP?
2. Explain the three way Handshake process?
3. Which layer is closer to a user?
4. Explain how TCP avoids a network meltdown?
5. What is the difference between flow control and Error control?

TCP Performance

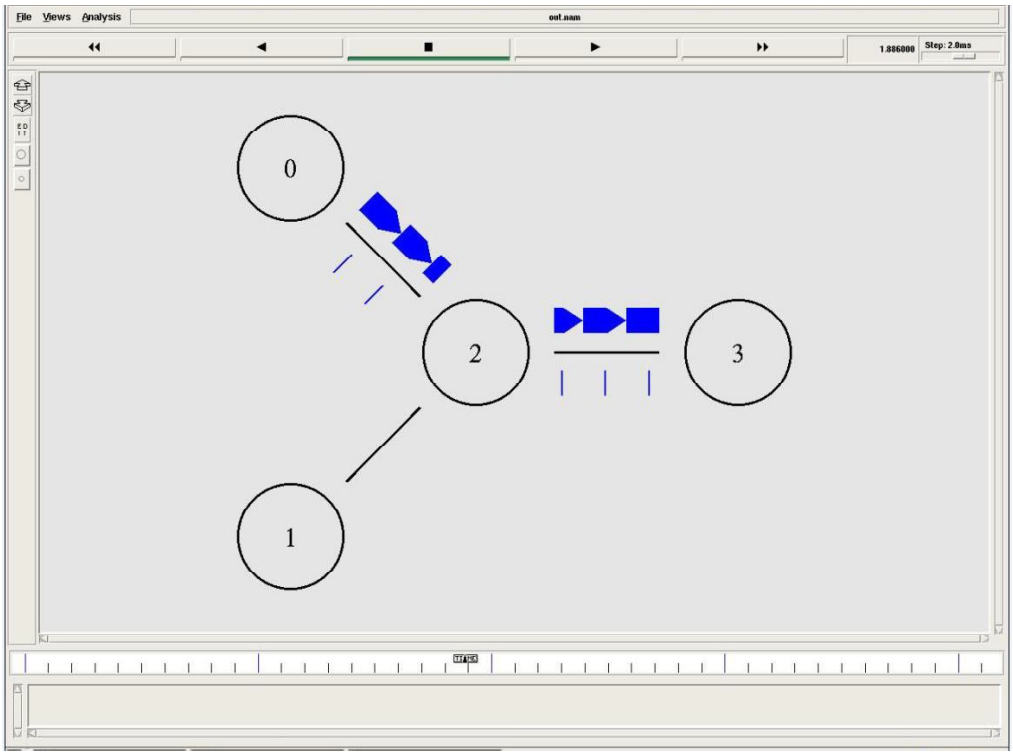
1. Algorithm:
2. Create a Simulator object.
3. Set routing as dynamic.
4. Open the trace and nam trace files.
5. Define the finish procedure.
6. Create nodes and the links between them.
7. Create the agents and attach them to the nodes.
8. Create the applications and attach them to the tcp agent.
9. Connect tcp and tcp sink.
10. 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

ALGORITHM :

1. Create a Simulator object.
2. Set routing as dynamic.
3. Open the trace and nam trace files.
4. Define the finish procedure.

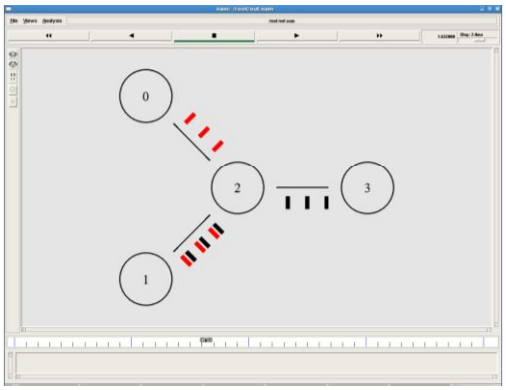
5. Create nodes and the links between them.
6. Create the agents and attach them to the nodes.
7. Create the applications and attach them to the UDP agent.
8. Connect udp and null agents.
9. Run the simulation.

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.

Ex.No: 8 Simulation of Distance Vector/ Link State Routing algorithm.

AIM:

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

THEORY:

LINK STATE ROUTING

Routing is the process of selecting best paths in a network. In the past, the term routing was also used to mean forwarding network traffic among networks. However this latter function is much better described as simply forwarding. Routing is performed for many kinds of networks, including the telephone network (circuit switching), electronic data networks (such as the Internet), and transportation networks. This article is concerned primarily with routing in electronic data networks using packet switching technology.

In packet switching networks, routing directs packet forwarding (the transit of logically addressed network packets from their source toward their ultimate destination) through intermediate nodes. Intermediate nodes are typically network hardware devices such as routers, bridges, gateways, firewalls, or switches. General-purpose computers can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time. Multipath routing techniques enable the use of multiple alternative paths.

In case of overlapping/equal routes, the following elements are considered in order to decide which routes get installed into the routing table (sorted by priority):

1. *Prefix-Length*: where longer subnet masks are preferred (independent of whether it is within a routing protocol or over different routing protocol)
2. *Metric*: where a lower metric/cost is preferred (only valid within one and the same routing protocol)
3. *Administrative distance*: where a lower distance is preferred (only valid between different routing protocols)

Routing, in a more narrow sense of the term, is often contrasted with bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Structured addresses allow a single routing table entry to represent the route to a group of devices. In large networks, structured addressing (routing, in the narrow sense) outperforms unstructured addressing (bridging). Routing has become the dominant form of addressing on the Internet. Bridging is still widely used within localized environments.

b. Flooding

Flooding is a simple routing algorithm in which every incoming packet is sent through every outgoing link except the one it arrived on. Flooding is used in bridging and in systems such as Usenet and peer-to-peer file sharing and as part of some routing protocols, including OSPF, DVMRP, and those used in ad-hoc wireless networks.There are generally two types of flooding available, Uncontrolled Flooding and Controlled Flooding. Uncontrolled Flooding is the fatal law of flooding. All nodes have neighbours and route packets indefinitely. More than two neighbours creates a broadcast storm.

Controlled Flooding has its own two algorithms to make it reliable, SNCF (Sequence Number Controlled Flooding) and RPF (Reverse Path Flooding). In SNCF, the node attaches its own address and sequence number to the packet, since every node has a memory of addresses and sequence numbers. If it receives a packet in memory, it drops it immediately while in RPF, the node will only send the packet forward. If it is received from the next node, it sends it back to the sender.

c. Distance vector

In computer communication theory relating to packet-switched networks, a **distance-vector routing protocol** is one of the two major classes of routing protocols, the other major class being the link-state protocol. Distance-vector routing protocols use the Bellman–Ford algorithm, Ford–Fulkerson algorithm, or DUAL FSM (in the case of Cisco Systems's protocols) to calculate paths.

A distance-vector routing protocol requires that a router informs its neighbors of topology changes periodically. Compared to link-state protocols, which require a router to inform all the nodes in a network of topology changes, distance-vector routing protocols have less computational complexity and message overhead.

The term *distance vector* refers to the fact that the protocol manipulates *vectors* (arrays) of distances to other nodes in the network. The vector distance algorithm was the original ARPANET routing algorithm and was also used in the internet under the name of RIP (Routing Information Protocol).Examples of distance-vector routing protocols include RIPv1 and RIPv2 and IGRP.

Method

Routers using distance-vector protocol do not have knowledge of the entire path to a destination. Instead they use two methods:

1. Direction in which router or exit interface a packet should be forwarded.
2. Distance from its destination

Distance-vector protocols are based on calculating the direction and distance to any link in a network. "Direction" usually means the next hop address and the exit interface. "Distance" is a measure of the cost to reach a certain node. The least cost route between any two nodes is the route with minimum distance. Each node maintains a vector (table) of minimum distance to

every node. The cost of reaching a destination is calculated using various route metrics. RIP uses the hop count of the destination whereas IGRP takes into account other information such as node delay and available bandwidth.

Updates are performed periodically in a distance-vector protocol where all or part of a router's routing table is sent to all its neighbors that are configured to use the same distance-vector routing protocol. RIP supports cross-platform distance vector routing whereas IGRP is a Cisco Systems proprietary distance vector routing protocol. Once a router has this information it is able to amend its own routing table to reflect the changes and then inform its neighbors of the changes. This process has been described as ‗routing by rumor' because routers are relying on the information they receive from other routers and cannot determine if the information is actually valid and true. There are a number of features which can be used to help with instability and inaccurate routing information.

EGP and BGP are not pure distance-vector routing protocols because a distance-vector protocol calculates routes based only on link costs whereas in BGP, for example, the local route preference value takes priority over the link cost.

Count-to-infinity problem

The Bellman–Ford algorithm does not prevent routing loops from happening and suffers from the **count-to-infinity problem**. The core of the count-to-infinity problem is that if A tells B that it has a path somewhere, there is no way for B to know if the path has B as a part of it. To see the problem clearly, imagine a subnet connected like A–B–C–D–E–F, and let the metric between the routers be "number of jumps". Now suppose that A is taken offline. In the vector-update-process B notices that the route to A, which was distance 1, is down – B does not receive the vector update from A. The problem is, B also gets an update from C, and C is still not aware of the fact that A is down – so it tells B that A is only two jumps from C (C to B to A), which is false. This slowly propagates through the network until it reaches infinity (in which case the algorithm corrects itself, due to the relaxation property of Bellman–Ford).

PRE LAB VIVA QUESTIONS:

1. Explain RIP.
2. Describe the process of routing packets.
3. What are some of the merits used by routing protocols?
4. Whare is routing table is maintained? Also State the of maintaining a routing table.
5. Distinguish between bridges and routers

ALGORITHM:

1. Create a Simulator object.
2. Set routing as dynamic.
3. Open the trace and nam trace files.
4. Define the finish procedure.

5. Create nodes and the links between them.
6. Create the agents and attach them to the nodes.
7. Create the applications and attach them to the udp agent.
8. Connect udp and null..
9. At 1 sec the link between node 1 and 2 is broken.
10. At 2 sec the link is up again.
11. Run the simulation.

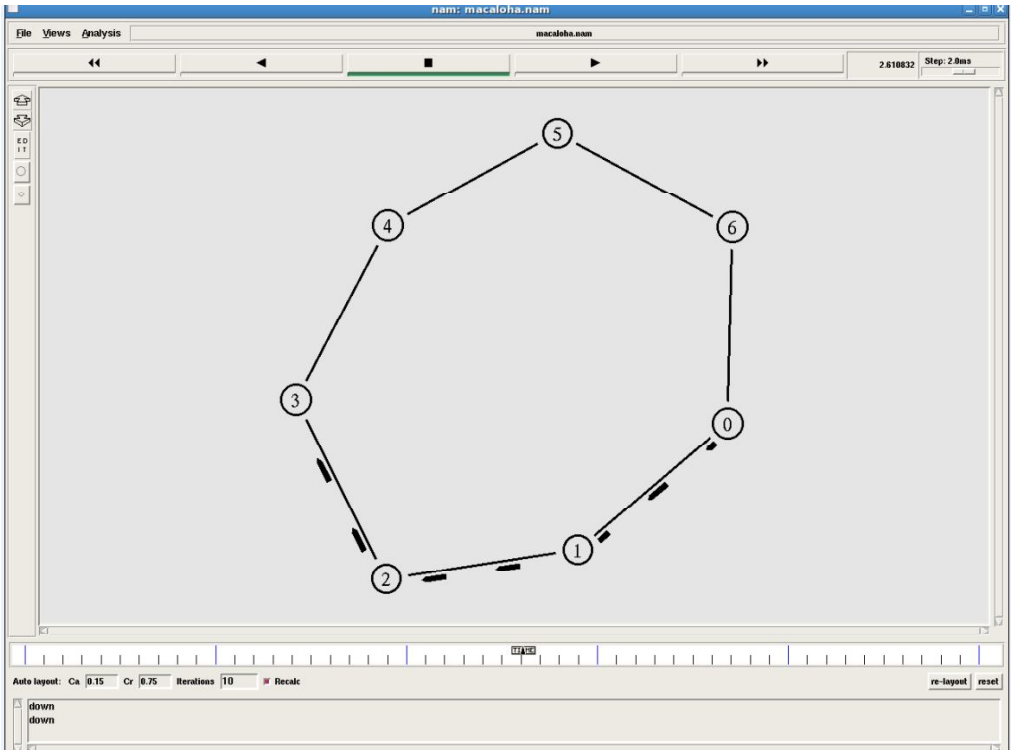
LINK STATE ROUTING PROTOCOL

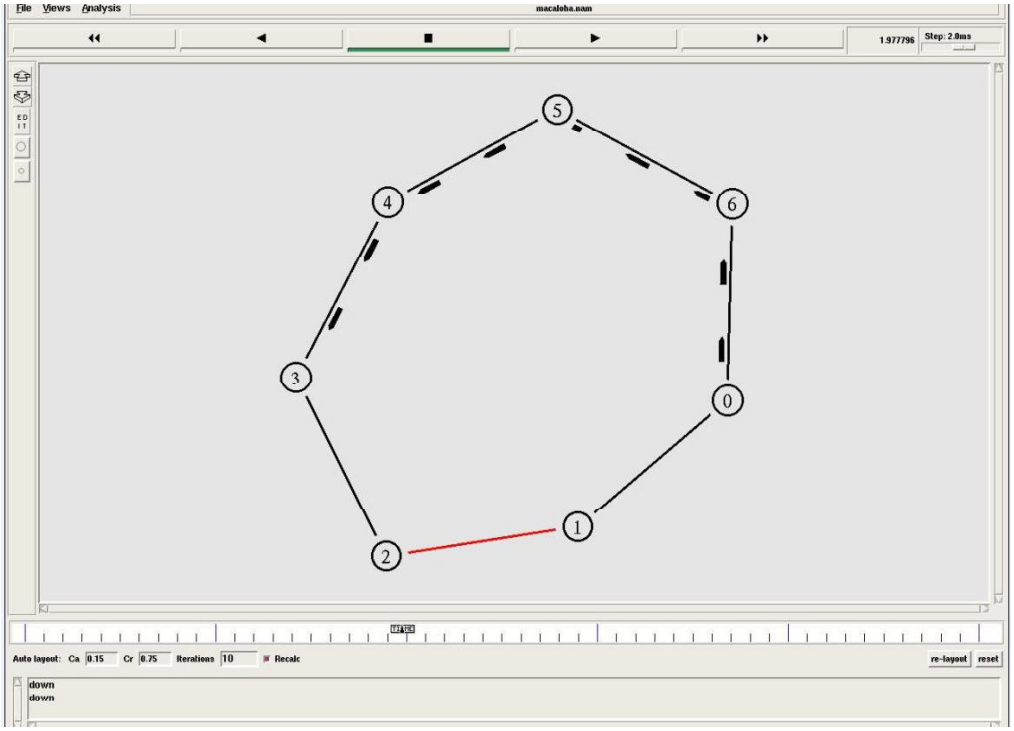
PROGRAM

```
set ns [new Simulator]
$ns rtproto LS
set nf [open linkstate.nam w]
$ns namtrace-all $nf
set f0 [open linkstate.tr w]
$ns trace-all $f0
proc finish {} {
        global ns f0 nf
        $ns flush-trace
        close $f0
        close $nf
        exec nam linkstate.nam &
        exit 0
}
for {set i 0} {$i <7} {incr i} {
        set n($i) [$ns node]
}
for {set i 0} {$i <7} {incr i} {
        $ns duplex-link $n($i) $n([expr ($i+1)%7]) 1Mb 10ms DropTail
}
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

```
set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0
$ns connect $udp0 $null0
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

Output





DISTANCE VECTOR ROUTING ALGORITHM

ALGORITHM:

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

15. View the simulated events and trace file analyze it
16. Stop

PROGRAM

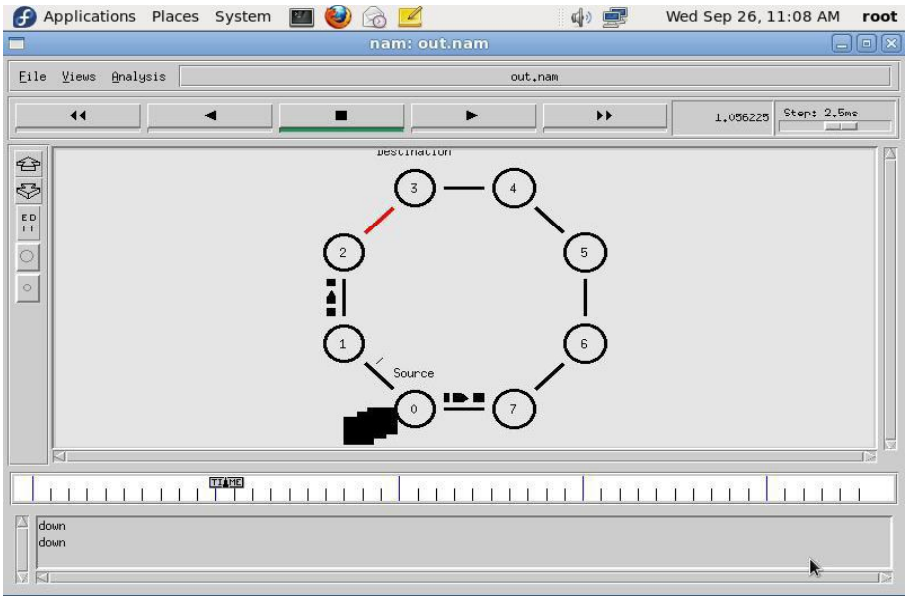
```
#Distance vector routing protocol – distvect.tcl
#Create a simulator object
set ns [new Simulator]
#Use distance vector routing
$ns rtproto DV
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
# Open tracefile
set nt [open trace.tr w]
$ns trace-all $nt
#Define 'finish' procedure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

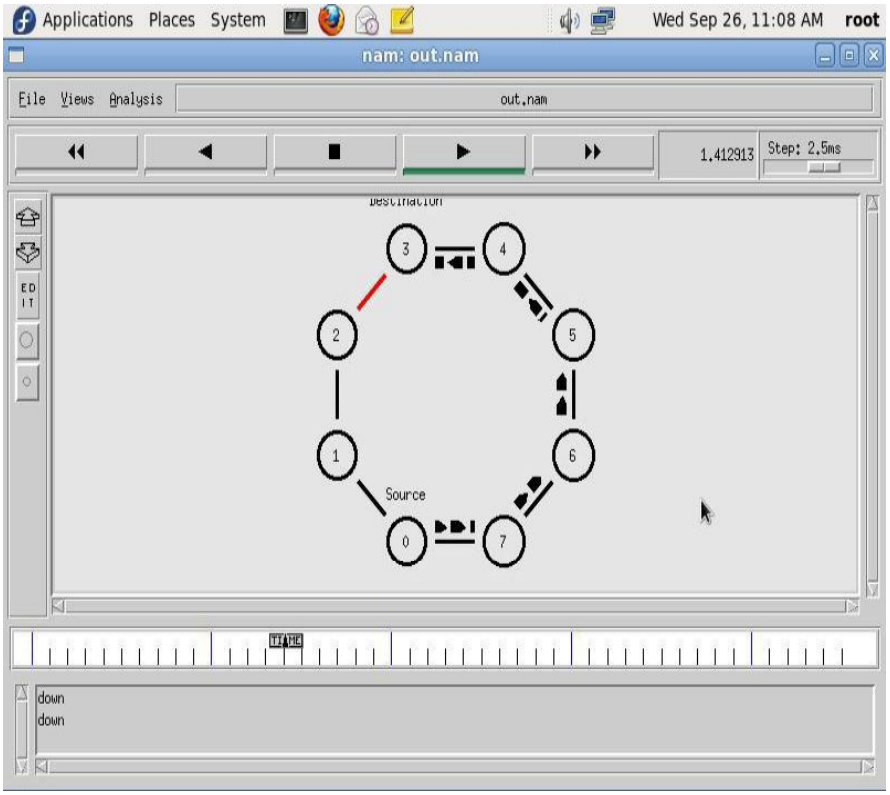
```
$ns duplex-link $n6 $n7 1Mb 10ms DropTail
$ns duplex-link $n7 $n8 1Mb 10ms DropTail
$ns duplex-link $n8 $n1 1Mb 10ms DropTail
# specify layout as a octagon
$ns duplex-link-op $n1 $n2 orient left-up
$ns duplex-link-op $n2 $n3 orient up
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $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
```

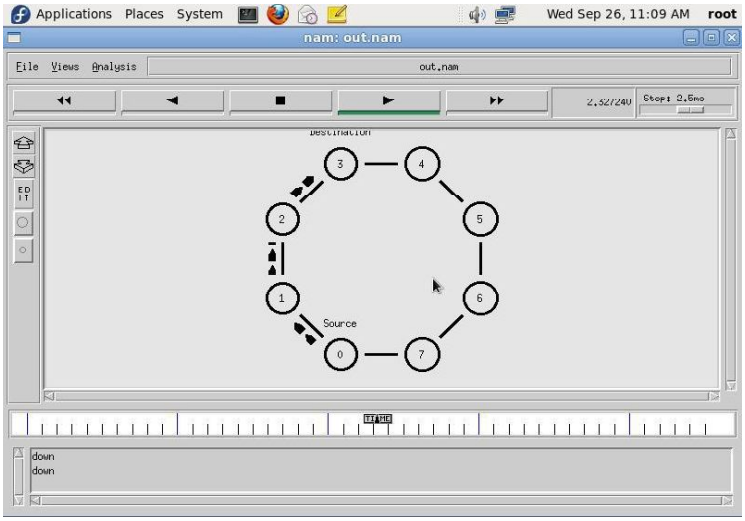
OUTPUT

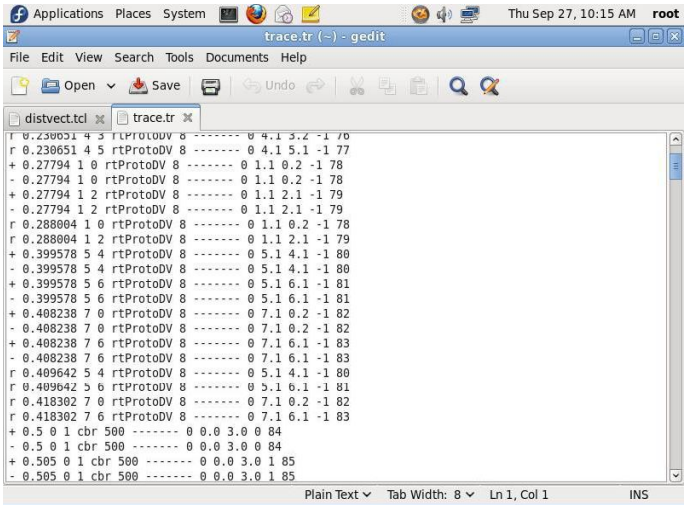
$ ns distvect.tc

l









POST LAB VIVA QUESTIONS:

1. What do you mean by the term “no. of hops”?
2. List the basic functions of routers.
3. Explain multicast routing.
4. Explain the difference between interior and exterior neighbor gateways.

RESULT:

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

Ex.No:9 Performance evaluation of Routing protocols using Simulation tool.

AIM:

To study the link state routing.

LINK STATE ROUTING

a. Link State routing

Routing is the process of selecting best paths in a network. In the past, the term routing was also used to mean forwarding network traffic among networks. However this latter function is much better described as simply forwarding. Routing is performed for many kinds of networks, including the telephone network (circuit switching), electronic data networks (such as the Internet), and transportation networks. This article is concerned primarily with routing in electronic data networks using packet switching technology.

In packet switching networks, routing directs packet forwarding (the transit of logically addressed network packets from their source toward their ultimate destination) through intermediate nodes. Intermediate nodes are typically network hardware devices such as routers, bridges, gateways, firewalls, or switches. General-purpose computers can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time. Multipath routing techniques enable the use of multiple alternative paths.

In case of overlapping/equal routes, the following elements are considered in order to decide which routes get installed into the routing table (sorted by priority):

4. *Prefix-Length*: where longer subnet masks are preferred (independent of whether it is within a routing protocol or over different routing protocol)
5. *Metric*: where a lower metric/cost is preferred (only valid within one and the same routing protocol)
6. *Administrative distance*: where a lower distance is preferred (only valid between different routing protocols)

Routing, in a more narrow sense of the term, is often contrasted with bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Structured addresses allow a single routing table entry to represent the route to a group of devices. In large networks, structured addressing (routing, in the narrow sense) outperforms unstructured addressing (bridging). Routing has become the dominant form of addressing on the Internet. Bridging is still widely used within localized environments.

b. Flooding

Flooding is a simple routing algorithm in which every incoming packet is sent through every outgoing link except the one it arrived on. Flooding is used in bridging and in systems such

as Usenet and peer-to-peer file sharing and as part of some routing protocols, including OSPF, DVMRP, and those used in ad-hoc wireless networks.There are generally two types of flooding available, Uncontrolled Flooding and Controlled Flooding. Uncontrolled Flooding is the fatal law of flooding. All nodes have neighbours and route packets indefinitely. More than two neighbours creates a broadcast storm.

Controlled Flooding has its own two algorithms to make it reliable, SNCF (Sequence Number Controlled Flooding) and RPF (Reverse Path Flooding). In SNCF, the node attaches its own address and sequence number to the packet, since every node has a memory of addresses and sequence numbers. If it receives a packet in memory, it drops it immediately while in RPF, the node will only send the packet forward. If it is received from the next node, it sends it back to the sender.

Algorithm

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

1. Each node acts as both a transmitter and a receiver.
2. Each node tries to forward every message to every one of its neighbours except the source node.

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

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

Advantages

- Since flooding naturally utilizes every path through the network, it will also use the shortest path.
- This algorithm is very simple to implement.

Disadvantages

- Flooding can be costly in terms of wasted bandwidth. While a message may only have one destination it has to be sent to every host. In the case of a ping flood or a denial of service attack, it can be harmful to the reliability of a computer network.
- Messages can become duplicated in the network further increasing the load on the networks bandwidth as well as requiring an increase in processing complexity to disregard duplicate messages.
- Duplicate packets may circulate forever, unless certain precautions are taken:
- Use a hop count or a time to live count and include it with each packet. This value should take into account the number of nodes that a packet may have to pass through on the way to its destination.
- Have each node keep track of every packet seen and only forward each packet once
- Enforce a network topology without loops

d. Distance vector

In computer communication theory relating to packet-switched networks, a **distance-vector routing protocol** is one of the two major classes of routing protocols, the other major class being the link-state protocol. Distance-vector routing protocols use the Bellman–Ford algorithm, Ford–Fulkerson algorithm, or DUAL FSM (in the case of Cisco Systems's protocols) to calculate paths.

A distance-vector routing protocol requires that a router informs its neighbors of topology changes periodically. Compared to link-state protocols, which require a router to inform all the nodes in a network of topology changes, distance-vector routing protocols have less computational complexity and message overhead.

The term *distance vector* refers to the fact that the protocol manipulates *vectors* (arrays) of distances to other nodes in the network. The vector distance algorithm was the original ARPANET routing algorithm and was also used in the internet under the name of RIP (Routing Information Protocol).Examples of distance-vector routing protocols include RIPv1 and RIPv2 and IGRP.

Method

Routers using distance-vector protocol do not have knowledge of the entire path to a destination. Instead they use two methods:

3. Direction in which router or exit interface a packet should be forwarded.
4. Distance from its destination

Distance-vector protocols are based on calculating the direction and distance to any link in a network. "Direction" usually means the next hop address and the exit interface. "Distance" is a measure of the cost to reach a certain node. The least cost route between any two nodes is the route with minimum distance. Each node maintains a vector (table) of minimum distance to every node. The cost of reaching a destination is calculated using various route metrics. RIP uses

the hop count of the destination whereas IGRP takes into account other information such as node delay and available bandwidth.

Updates are performed periodically in a distance-vector protocol where all or part of a router's routing table is sent to all its neighbors that are configured to use the same distance-vector routing protocol. RIP supports cross-platform distance vector routing whereas IGRP is a Cisco Systems proprietary distance vector routing protocol. Once a router has this information it is able to amend its own routing table to reflect the changes and then inform its neighbors of the changes. This process has been described as _routing by rumor‘ because routers are relying on the information they receive from other routers and cannot determine if the information is actually valid and true. There are a number of features which can be used to help with instability and inaccurate routing information.

EGP and BGP are not pure distance-vector routing protocols because a distance-vector protocol calculates routes based only on link costs whereas in BGP, for example, the local route preference value takes priority over the link cost.

Count-to-infinity problem

The Bellman–Ford algorithm does not prevent routing loops from happening and suffers from the **count-to-infinity problem**. The core of the count-to-infinity problem is that if A tells B that it has a path somewhere, there is no way for B to know if the path has B as a part of it. To see the problem clearly, imagine a subnet connected like A–B–C–D–E–F, and let the metric between the routers be "number of jumps". Now suppose that A is taken offline. In the vector-update-process B notices that the route to A, which was distance 1, is down – B does not receive the vector update from A. The problem is, B also gets an update from C, and C is still not aware of the fact that A is down – so it tells B that A is only two jumps from C (C to B to A), which is false. This slowly propagates through the network until it reaches infinity (in which case the algorithm corrects itself, due to the relaxation property of Bellman–Ford).

(a) UNICAST ROUTING PROTOCOL

AIM:

To write a ns2 program for implementing unicast routing protocol.

THEORY:

When a device has multiple paths to reach a destination, it always selects one path by preferring it over others. This selection process is termed as Routing. Routing is done by special network devices called routers or it can be done by means of software processes.The software based routers have limited functionality and limited scope.A router is always configured with some default route. A default route tells the router where to forward a packet if there is no route found for specific destination. In case there are multiple path existing to reach the same destination, router can make decision based on the following information.Routes can be statically configured or dynamically learnt. One route can be configured to be preferred over others.Most of the traffic on the internet and intranets known as unicast data or unicast traffic is sent with specified destination. Routing unicast data over the internet is called unicast routing. It is the simplest form of routing because the destination is already known. Hence the router just has to look up the routing table and forward the packet to next hop

PRE LAB VIVA QUESTIONS:

1. What type of protocol is BGP?
2. What type of protocol is OSPF?
3. What is Distance Vector Routing.
4. What is flooding.
5. Describe the difference between static and dynamic routing/
6. Define an autonomous system.
7. What are the adaptive routing algorithms.
8. Dintinguish between adaptive and non adaptive routing algorithms.
9. What is routing

ALGORITHM:

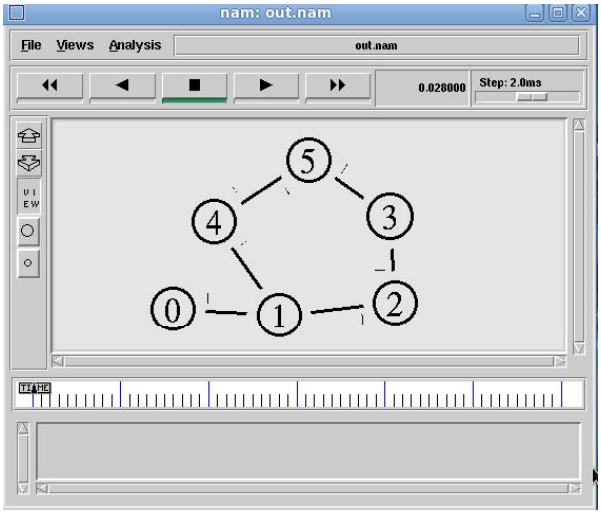
1. Start the program.
2. Declare the global variables ns for creating a new simulator.
3. Set the color for packets.
4. Open the network animator file in the name of file2 in the write mode.
5. Open the trace file in the name of file 1 in the write mode.
6. Set the unicast routing protocol to transfer 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.

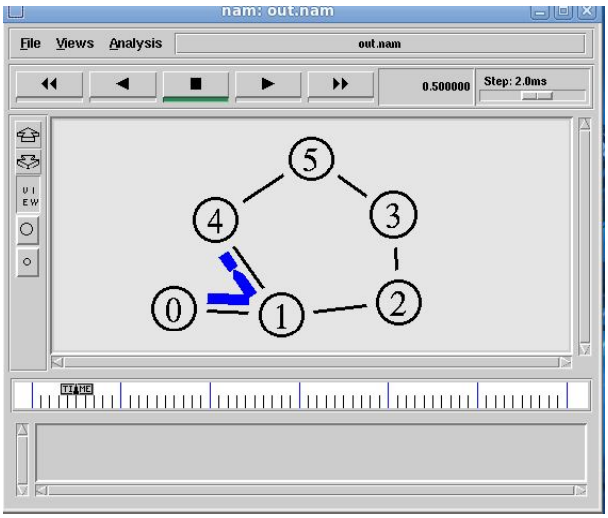
11. Set the destination node using tcp sink.
12. Setup a ftp connection over the tcp connection.
13. Down the connection between any nodes at a particular time.
14. Reconnect the downed connection at a particular time.
15. Define the finish procedure.
16. In the definition of the finish procedure declare 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.

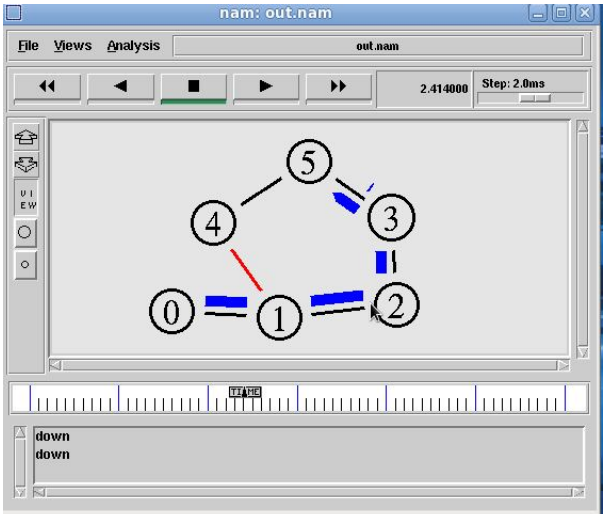
PROGRAM:

```
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
#Open the Trace 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
```


nam: out.nam
File Views Analysis
out.nam
0.028000
Step: 2.0ms
5
4
3
0
1
2


nam: out.nam
File Views Analysis
out.nam
0.500000
Step: 2.0ms
5
4
3
0
1
2


nam: out.nam
File Views Analysis
out.nam
2.414000
Step: 2.0ms
5
4
3
0
1
2
down
down

(b) MULTICASTING ROUTING PROTOCOL

AIM:

To write a ns2 program for implementing multicasting routing protocol.

ALGORITHM:

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 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:

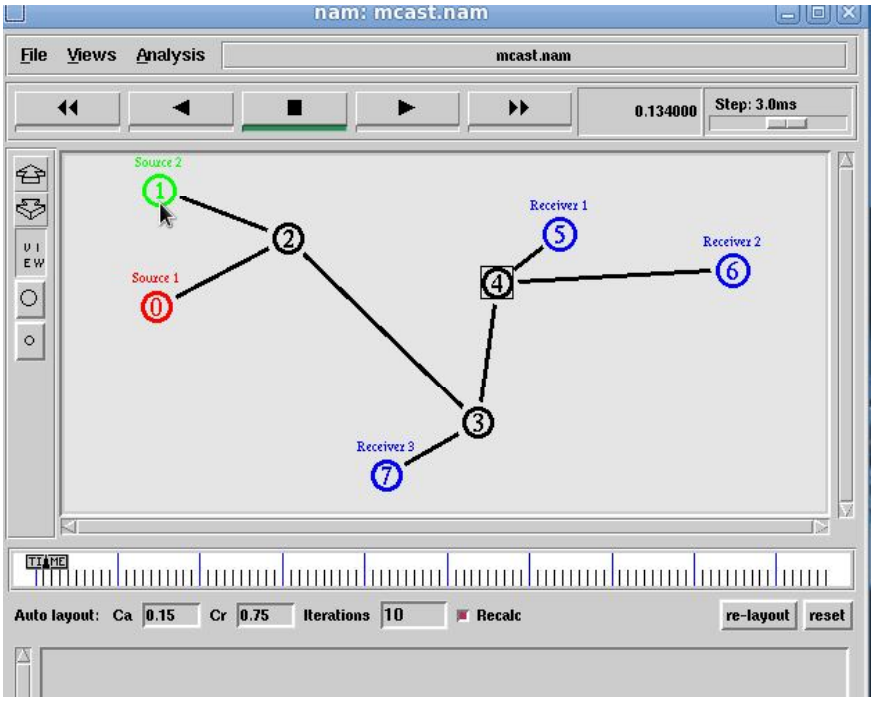
```
# Create scheduler
#Create an event scheduler wit multicast turned on
set ns [new Simulator -multicast on]
#$ns multicast
#Turn on Tracing
set tf [open output.tr w]
$ns trace-all $tf
# Turn on nam Tracing
set fd [open mcast.nam w]
$ns namtrace-all $fd
```

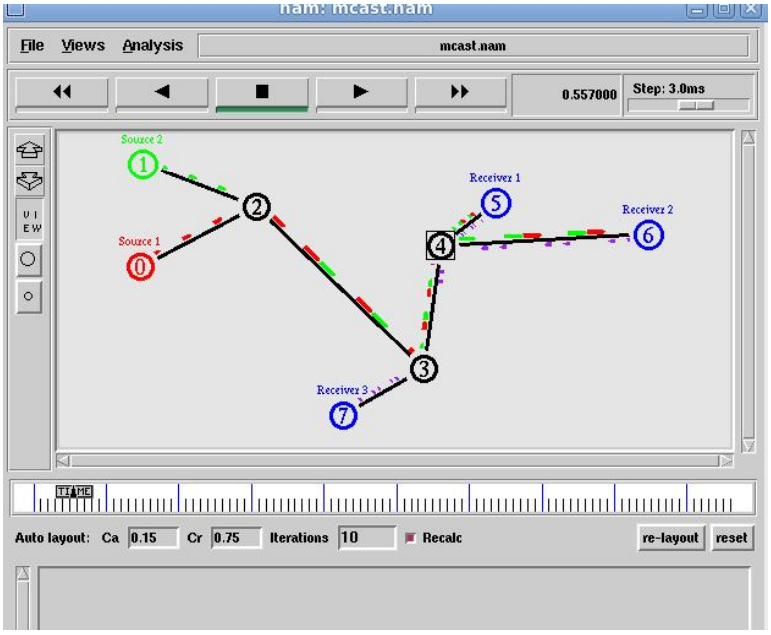
```
# Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]

# 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
```

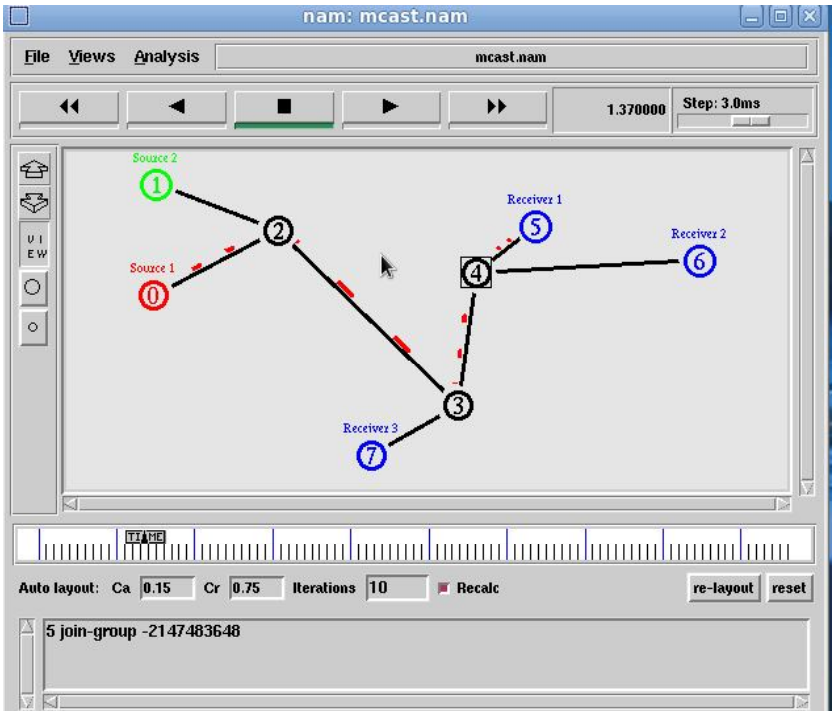
```
$udp1 set dst_port_ 0
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp1

# 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 {}
{
```

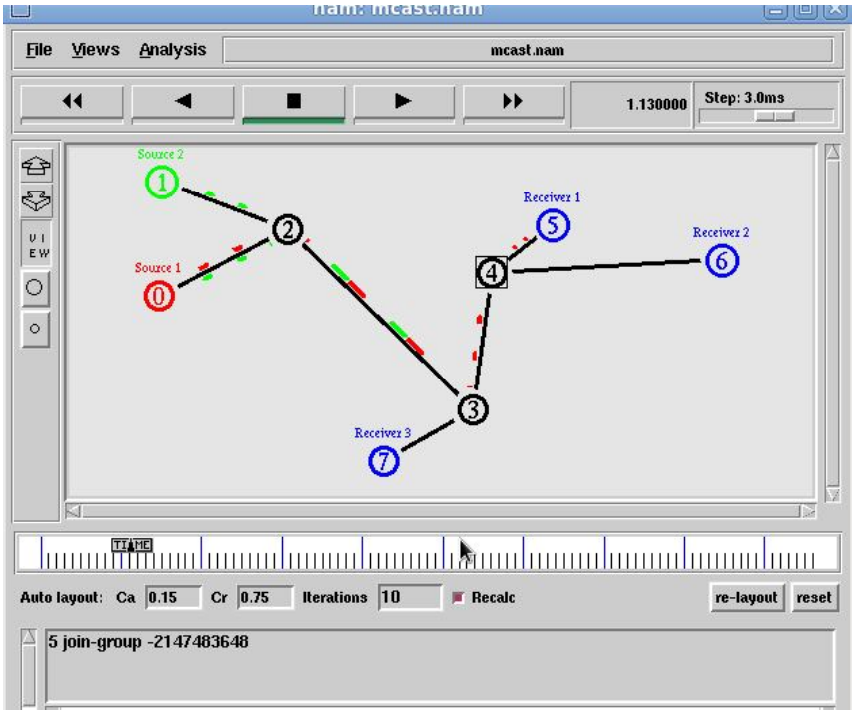
```
        global ns tf
        $ns flush-trace
        close $tf
        exec nam mcast.nam &
        exit 0
}

# 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
```






nam: mcast.nam
File Views Analysis
mcast.nam
1.370000
Step: 3.0ms
Source 2
Source 1
Receiver 1
Receiver 2
Receiver 3
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout reset
5 join-group -2147483648


File Views Analysis
mcast.nam
1.130000
Step: 3.0ms
Source 2
Source 1
Receiver 1
Receiver 2
Receiver 3
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout reset
5 join-group -2147483648

(c) DISTANCE VECTOR ROUTING PROTOCOL

AIM:

To simulate a link failure and to observe distance vector routing protocol in action.

ALGORITHM:

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

PROGRAM

```
#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
```

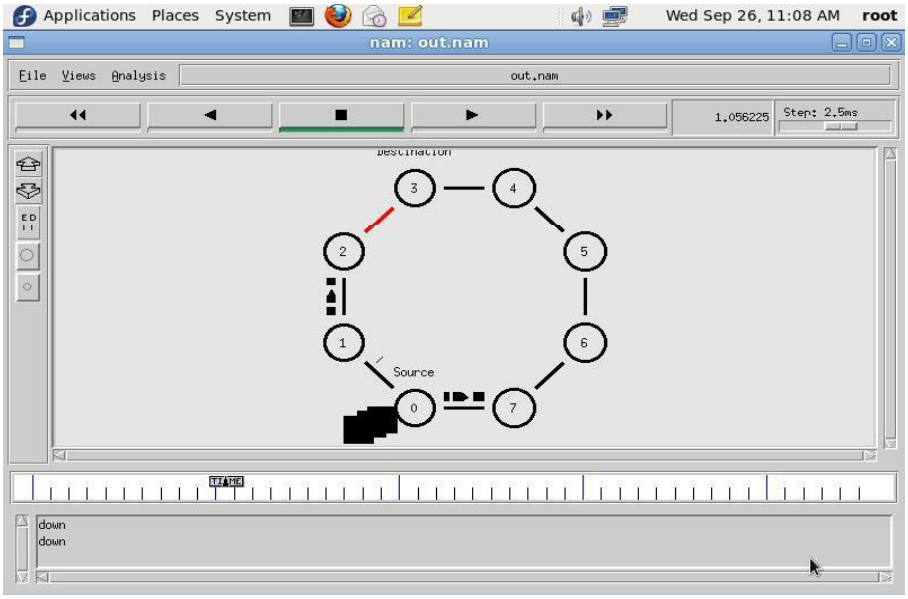
```
proc finish {}
{
        global ns nf
        $ns flush-trace
        #Close the trace file
        close $nf
        #Execute nam on the trace file
        exec nam -a out.nam &
        exit 0
}
# Create 8 nodes
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns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

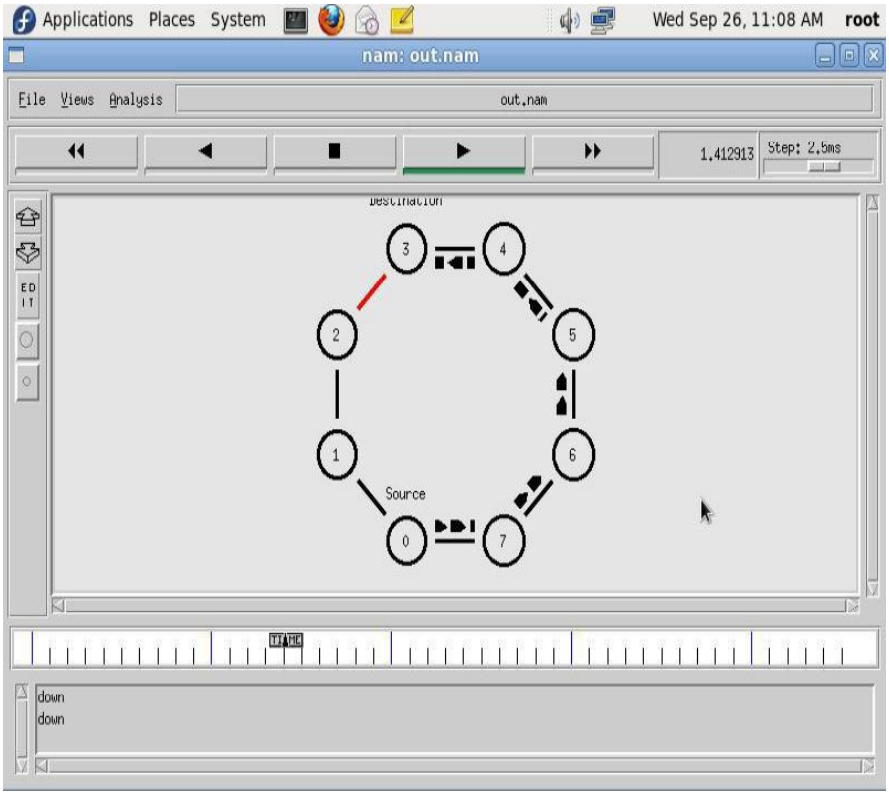
```
#Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
#Create a Null agent (a traffic sink) and attach it to node n4
set null0 [new Agent/Null]
$ns attach-agent $n4 $null0
#Connect the 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
```

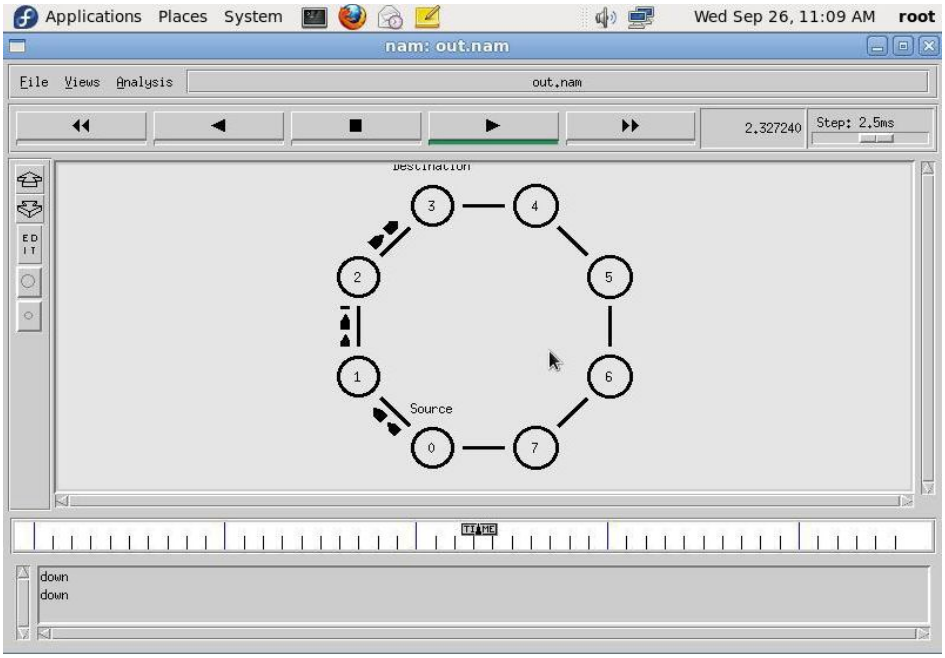
OUTPUT

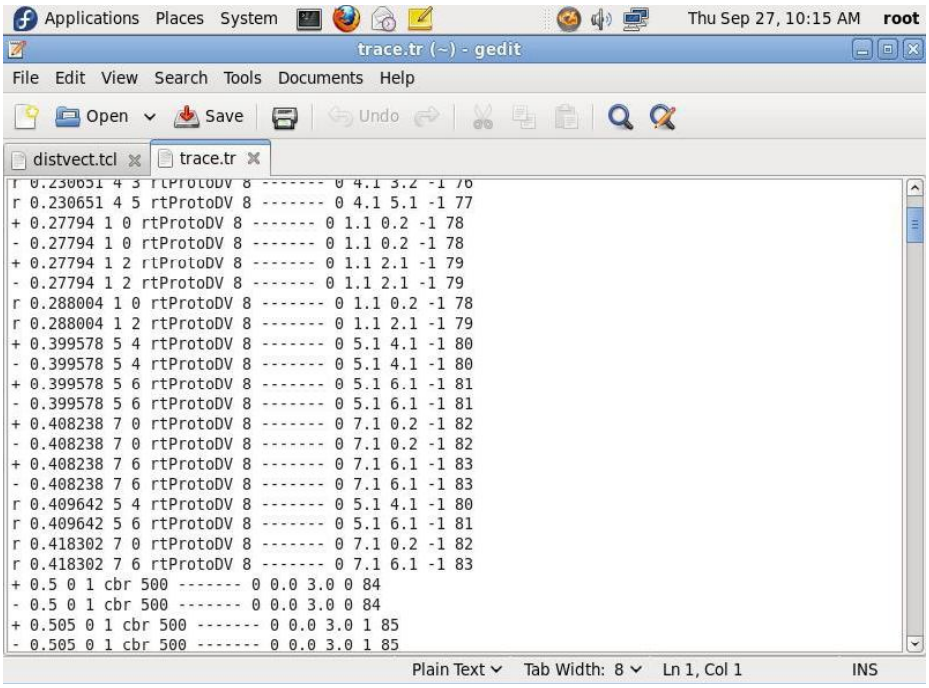
$ ns distvect.tc

l




Applications Places System
Wed Sep 26, 11:08 AM root
nam: out.nam
File Views Analysis
out.nam
1.412913
Step: 2.5ms
Destination
Source
down
down


Applications Places System
Wed Sep 26, 11:09 AM root
nam: out.nam
File Views Analysis
out.nam
2.327240
Step: 2.5ms
Destination
Source
down
down



POST LAB VIVA QUESTIONS:

1. What do you mean by the term "no. of hops"?
2. List the basic functions of routers.
3. Explain multicast routing.
4. Explain the difference between interior and exterior neighbor gateways.
5. Explain RIP.
6. Describe the process of routing packets.
7. What are some of the merits used by routing protocols?
8. Whare is routing table is maintained? Also State the of maintaining a routing table.
9. Distinguish between bridges and routers

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.

Ex.No:10 Simulation of Error Correction Code (like CRC)

AIM:

To implement error checking code using java.

THEORY:

The cyclic redundancy check, or CRC, is a technique for detecting errors in digital data, but not for making corrections when errors are detected. It is used primarily in data transmission.

In the CRC method, a certain number of check bits, often called a checksum, are appended to the message being transmitted. The receiver can determine whether or not the check bits agree with the data, to ascertain with a certain degree of probability whether or not an error occurred in transmission.

It does error checking via polynomial division. In general, a bit string
b b b …b b b
n-1 n-2 n-3 2 1 0
As
bn-1Xn-1 + bn-2 Xn-2 + bn-3 Xn-3 + …b2 X2 + b1 X1 + b0
Ex: -
10001000000100001 As
X16 + X12 + X5 +1
All computations are done in modulo 2

PRE LAB VIVA QUESTIONS:

1. What is flow control?
2. What is meant by error control?
3. Which is operated in the Data Link and the Network layer?
4. State the purpose of layering.
5. What are the two types of line configuration?

ALGORITHM:

1. Start the Program
2. Given a bit string, append 0S to the end of it (the number of 0s is the same as the degree of the generator polynomial) let B(x) be the polynomial corresponding to B.
3. Divide B(x) by some agreed on polynomial G(x) (generator polynomial) and determine the remainder R(x). This division is to be done using Modulo 2 Division.

4. Define T(x) = B(x) –R(x)
5. (T(x)/G(x) => 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.readLine());
 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("Error");
break;
}
if(i==rem.length-1)
```

```
System.out.println("No 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 :

```
run:

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.... :)
BUILD SUCCESSFUL (total time: 1 minute 34 seconds)

POST LAB VIVA QUESTIONS:

1. What are headers and trailers and how do they get added and removed?
2. What are protocols?
3. 8.Which one of the following condition is used to transmit two packets over a medium at the same time?
    - Contention
    - Collision
    - Synchronous
4. What are the responsibilities of data link layer?
5. What are the functions of MAC?

RESULT:

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

Ex.no.11

TOPIC BEYOND SYLLABUS

a. UDP CHAT SERVER AND CLIENT

AIM:

To Develop a Client/Server Application for a chat session using UDP.

THEORY:

User Datagram Protocol (UDP) is one of the core members of the Internet protocol suite. The protocol was designed by David P. Reed in 1980 and formally defined in RFC 768. With UDP, computer applications can send messages, in this case referred to as datagrams, to other hosts on an Internet Protocol (IP) network. Prior communications are not required in order to set udp communication channels or data paths. UDP uses a simple connectionless communication model with a minimum of protocol mechanism. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network; There is no guarantee of delivery, ordering, or duplicate protection.

PRE LAB VIVA QUESTIONS:

1. What is CSMA?
2. What is meant by non-persistent CSMA ?
3. What is meant by p-persistent CSMA?
4. What u meant by Exponatonal Back OFF?
5. Define Collision?
6. What is the access method used by wireless LAN?

ALGORITHM:

Server

1. Create a new socket.
2. Bind the newly created socket (Assign local protocol address).
3. Server ready to receive data from client
4. Receive data from the client and Reply (CHAT).

Client

1. Create a new socket.
2. Start to send messages to the server
3. Receive the messages sent by the server.
4. Close the connection.

PROGRAM:

SERVER :

```
#include<stdio.h>
#include<string.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<sys/times.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#define SA struct sockaddr
void dgecho(int sockfd,SA *pcliaddr,socklen_t clilen)
{
	int n,i,n1;
	socklen_t len;
	char msg[500],msg1[500],msg2[]={'b','y','e','\0'};
	for(;;)
	{
		len=clilen;
		n=recvfrom(sockfd,msg,500,0,pcliaddr,&len);
		printf("\nCLIENT ");
		if((msg[0]=='b')&&(msg[1]=='y')&&(msg[2]=='e'))
		{
			printf("bye\n");
			printf("\nClient terminated");
			return;
		}
		for(i=0;i<n;i++)
			printf("%c",msg[i]);
		printf("\nSERVER: ");
		fgets(msg1,500,stdin);
		sendto(sockfd,msg1,strlen(msg1),0,pcliaddr,len);
		if((msg[0]=='b')&&(msg[1]=='y')&&(msg[2]=='e'))
			exit(0);
	}
}
int main(int agrc,char *argv)
{
	int sockfd;
	struct sockaddr_in servaddr,cliaddr;
	sockfd=socket(AF_INET,SOCK_DGRAM,0);
```

```
	bzero(&servaddr,sizeof(servaddr));
	servaddr.sin_family=AF_INET;
	servaddr.sin_port=htons(9003);
	servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
	bind(sockfd,(SA*)&servaddr,sizeof(servaddr));
	printf("\nSERVER connected\n");
	dgecho(sockfd,(SA*)&cliaddr,sizeof(cliaddr));
}
```

CLIENT:

```
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<string.h>
#include<unistd.h>
#include<sys/times.h>
#define SA struct sockaddr
void dgcli(FILE *fp,int sockfd,const SA *pservaddr,socklen_t servlen)
{
	int n,n1;
	char sendline[500],recvline[500],msg[]={'b','y','e','\0'};
	for(;;)
	{
		printf("\nCLIENT : ");
		sendline[0]='\0';
		fgets(sendline,500,stdin);
		sendto(sockfd,sendline,strlen(sendline),0,pservaddr,servlen);
		if((sendline[0]=='b')&&(sendline[1]=='y')&&(sendline[2]=='e'))
			return;
		recvline[0]='\0';
		n=recvfrom(sockfd,recvline,500,0,NULL,NULL);
		recvline[n]=0;
		if((recvline[0]=='b')&&(recvline[1]=='y')&&(recvline[2]=='e'))
			return;
		recvline[n]=0;
		printf("\nSERVER ");
		fputs(recvline,stdout);
		printf("\n");
```

```
        }
}
int main(int argc,char *argv[])
{
        int sockfd;
        struct sockaddr_in servaddr;
        bzero(&servaddr,sizeof(servaddr));
        servaddr.sin_family=AF_INET;
        servaddr.sin_port=htons(9003);
        inet_pton(AF_INET,argv[1],&servaddr.sin_addr);
        sockfd=socket(AF_INET,SOCK_DGRAM,0);
        printf("\nClient connected to... %s\n",argv[1]);
        dgcli(stdin,sockfd,(SA*)&servaddr,sizeof(servaddr));
        exit(0);
}
```

OUTPUT:

Server Side:

```
SERVER connected
CLIENT: hai
SERVER: hello
CLIENT: how are u
SERVER: fine..you?
CLIENT: fine
SERVER: ok take care..bye.
CLIENT: bye
Client terminated[it5003@vecit it5003]$
```

Client Side:

```
Client connected to... 172.16.0.5
CLIENT : hai
SERVER hello
CLIENT : how are u
SERVER fine..you?
CLIENT : fine
SERVER ok take care..bye.
CLIENT : bye
[it5003@vecit it5003]$
```

b.CARRIER SENSE MULTIPLE ACCESS

AIM:

To write a ns2 program for implementing carrier sense multiple access.

THEORY:

Carrier Sensed Multiple Access (CSMA) : CSMA is a network access method used on shared network topologies such as Ethernet to control access to the network. Devices attached to the network cable listen (carrier sense) before transmitting. If the channel is in use, devices wait before transmitting. MA (Multiple Access) indicates that many devices can connect to and share the same network. All devices have equal access to use the network when it is clear.In other words, a station that wants to communicate "listen" first on the media communication and awaits a "silence" of a preset time (called the Distributed Inter Frame Space or DIFS). After this compulsory period, the station starts a countdown for a random period considered. The maximum duration of this countdown is called the collision window (Window Collision, CW). If no equipment speaks before the end of the countdown, the station simply deliver its package. However, if it is overtaken by another station, it stops immediately its countdown and waits for the next silence. She then continued his account countdown where it left off. This is summarized in Figure. The waiting time random has the advantage of allowing a statistically equitable distribution of speaking time between the various network equipment, while making little unlikely (but not impossible) that both devices speak exactly the same time. The countdown system prevents a station waiting too long before issuing its package. It's a bit what place in a meeting room when no master session (and all the World's polite) expected a silence, then a few moments before speaking, to allow time for someone else to speak. The time is and randomly assigned, that is to say, more or less equally.

ALGORITHM:

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 write mode.
5. Open the trace file and the win file in the write mode.
6. Transfer the packets in network.
7. Create the 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 tcp connection for source node.
11. Set the destination node using tcp sink.
12. Set the window size and the packet size for the tcp.
13. Set up the ftp over the tcp connection.

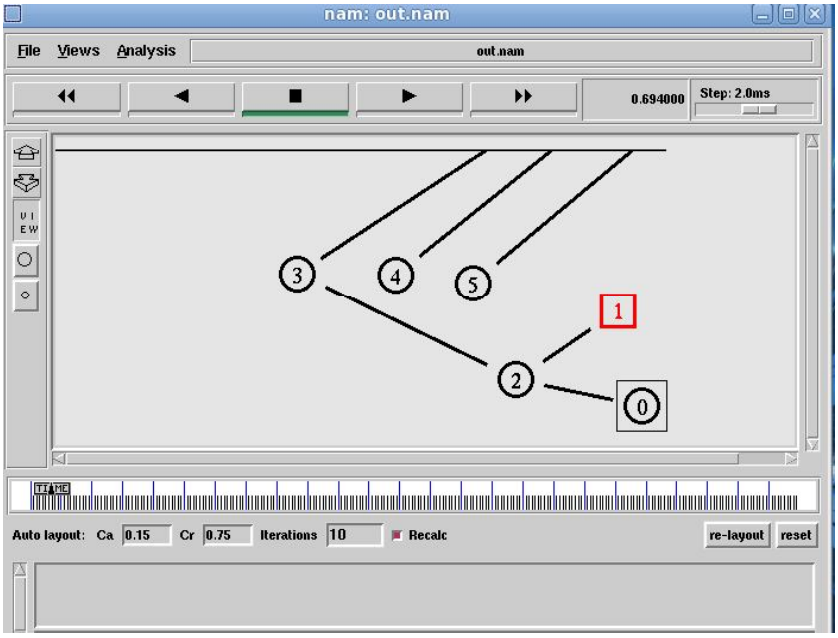
14. Set the udp and tcp connection for the source and destination.
15. Create the traffic generator CBR for the source and destination files.
16. Define the plot window and finish procedure.
17. In the definition of the finish procedure declare the global variables.
18. Close the trace file and namefile and execute the network animation file.
19. At the particular time call the finish procedure.
20. Stop the program.

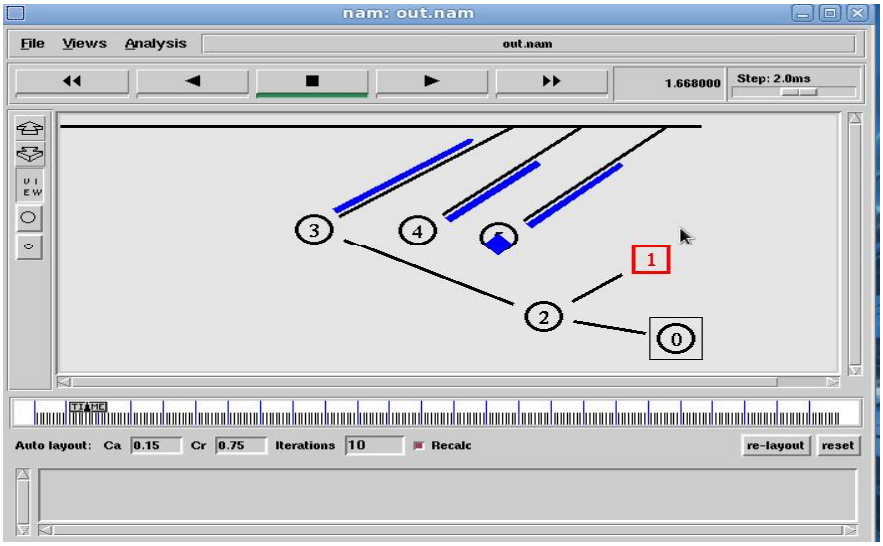
PROGRAM:

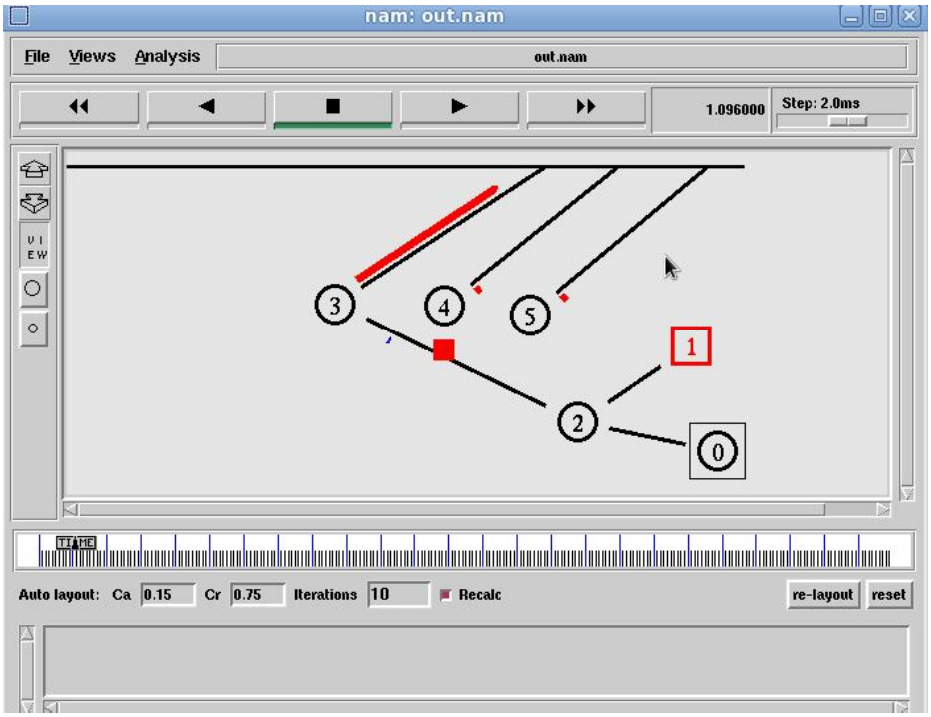
```
set ns [new Simulator]
$ns color 1 blue
$ns color 2 red
set fi1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $fi1
set fi2 [open out.nam w]
$ns namtrace-all $fi2
proc finish {}
{
        global ns fi1 fi2
        $ns flush-trace
        close $fi1
        close $fi2
        exec nam out.nam &
        exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n1 color red
$n1 shape box
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]
set tcp [new Agent/TCP/Newreno]
```

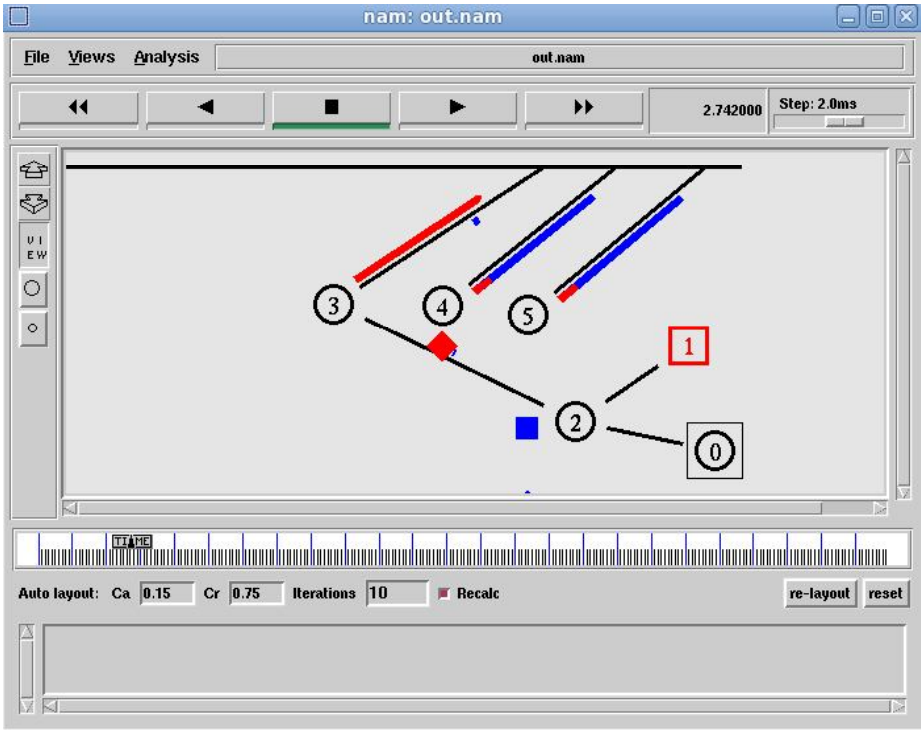
```
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetsize_ 552
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 24.0 "$ftp stop"
$ns at 24.5 "$cbr stop"
proc plotwindow { tcpSource file }
{
        global ns
        set time 0.1
        set now [$ns now]
        set cwnd [$tcpSource set cwnd_]
        set wnd [$tcpSource set window_]
        puts $file "$now $cwnd"
        $ns at [expr $now+$time] "plotwindow $tcpSource $file"
}
$ns at 1.0 "plotwindow $tcp $winfile"
$ns at 5 "$ns trace-annotate \"packet drop\""
$ns at 125.0 "finish"
$ns run
```

OUTPUT:


nam: out.nam
File Views Analysis
out.nam
0.694000
Step: 2.0ms
3
4
5
1
2
0
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout
reset


nam: out.nam
File Views Analysis
out.nam
1.668000
Step: 2.0ms
3
4
5
1
2
0
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout
reset


nam: out.nam
File
Views
Analysis
out.nam
1.096000
Step: 2.0ms
3
4
5
1
2
0
TIME
Auto layout:
Ca
0.15
Cr
0.75
Iterations
10
Recalc
re-layout
reset


nam: out.nam
File
Views
Analysis
out.nam
2.742000
Step: 2.0ms
3
4
5
1
2
0
TIME
Auto layout:
Ca
0.15
Cr
0.75
Iterations
10
Recalc
re-layout
reset

POST LAB VIVA QUESTIONS:

1. What is CSMA/CD?
2. Define the term Medium Access control Mechnism?
3. What is Collision Detection?
4. Mention different random access Techniques/
5. What is ALOHA Protocol.
6. What do you understand CSMA Protocol.

RESULT

Thus the UDP chat server and client application and carrier sense multiple access are implemented and executed successfully.