

## **19CSE501- NETWORKS LABORATORY**

### **LIST OF EXPERIMENTS:**

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

**TOTAL: 15 PERIODS**

### TABLE OF CONTENTS

S.NO.	DATE	EXPERIMENT TITLE	MARKS	SIGN.

**DATE:**

## **1. NETWORKING COMMANDS**

**AIM:**

To study the basic networking commands.

### **NETWORKING COMMANDS:**

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

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

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

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

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

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

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

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

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

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

C:\>nslookup: Nslookup is used for diagnosing DNS problems. If you can access a resource by specifying an IP address but not its DNS you have a DNS problem.

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

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

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

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

## **RESULT:**

Thus the above list of primitive has been studied.

**DATE:**                      **2. Write a HTTP web client program to download a web page using TCP sockets**

**AIM:**

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

**ALGORITHM:**

**CLIENT SIDE:**

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

**SERVER SIDE**

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

**PROGRAM**

```
import javax.swing.*;
import java.net.*;
import
java.awt.image.*;
import
javax.imageio.*;
import java.io.*;
import
java.awt.image.BufferedImage;
import
java.io.ByteArrayOutputStream;
```

```

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

```

## SERVER PROGRAM

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

## OUTPUT

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

## RESULT:

The webpage is successfully downloaded and the contents are displayed and verified.



**DATE:**

### **3. a. SOCKET PROGRAM FOR ECHO**

**AIM:**

To write a socket program for implementation of echo.

**ALGORITHM:**

#### **CLIENT SIDE**

1. Start the program.
2. Create a socket which binds the Ip address of server and the port address to acquireservice.
3. After establishing connection send a data to server.
4. Receive and print the same data from server.
5. Close the socket.
6. End the program.

#### **SERVER SIDE**

1. Start the program.
2. Create a server socket to activate the port address.
3. Create a socket for the server socket which accepts the connection.
4. After establishing connection receive the data from client.
5. Print and send the same data to client.
6. Close the socket.
7. End the program.

**PROGRAM:**

#### **ECHO CLIENT**

```
import java.io.*;
import java.net.*;
public class
eclient
{
public static void main(String args[])
{
Socket
c=null;String
line;
DataInputStream
```

```

is,is1;PrintStream
os;
try
{
c=new Socket("localhost",8080);
}
catch(IOException e)
{
System.out.println(e);
}
try
{
os=new
PrintStream(c.getOutputStream());
is=new DataInputStream(System.in);
is1=new
DataInputStream(c.getInputStream());do
{
System.out.println("client");
line=is.readLine();
os.println(line);
if(!line.equals("exit"))
System.out.println("server:"+is1.readLine());
}while(!line.equals("exit"));
}
catch(IOException e)
{
System.out.println("socket closed");
}}}

```

### **Echo Server:**

```

import java.io.*;
import java.net.*;
import
java.lang.*; public
class eserver

```

```

{
    public static void main(String args[])throws IOException
    {
        ServerSocket
        s=null;String line;
        DataInputStream
        is; PrintStream ps;
        Socket c=null;
        try
        {
            s=new ServerSocket(8080);
        }
        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();
                System.out.println("msg received and sent back to
                client");ps.println(line);
            }
        }
        catch(IOException e)
        {
            System.out.println(e);
        }
    }
}

```

**OUTPUT:****CLIENT**

Enter the IP address  
127.0.0.1 CONNECTION  
ESTABLISHED  
Enter the data  
CSE Client  
received CSE

**SERVER**

CONNECTION ACCEPTED  
Server received CSE

**RESULT:**

Thus the program for simulation of echo server was written & executed.

**DATE:**

### **3.B. CLIENT- SERVER APPLICATION FOR CHAT**

**AIM:**

To write a client-server application for chat using TCP

**ALGORITHM:**

**CLIENT**

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

**SERVER**

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

**PROGRAM:**

**TCPserver1.java**

```
import
java.net.*;
import java.io.*;
public class TCPserver1
{
    public static void main(String arg[])
    {
        ServerSocket
        s=null;String line;
        DataInputStream
```

```

is=null,is1=null;PrintStream
os=null;
Socket
c=null;try
{
s=new ServerSocket(9999);
}
catch(IOException e)
{
System.out.println(e);
}
try
{
c=s.accept();
is=new
DataInputStream(c.getInputStream());
is1=new DataInputStream(System.in);
os=new
PrintStream(c.getOutputStream()); do
{
line=is.readLine();
System.out.println("Client:"+lin
e);
System.out.println("Server:");
line=is1.readLine();
os.println(line);
}
while(line.equalsIgnoreCase("quit")!=fals
e);is.close();
os.close();
}
catch(IOException e)
{
System.out.println(e);
}
}

```

```
}
```

### **TCPclient1.jav**

```
import
java.net.*;
import java.io.*;
public class TCPclient1
{
    public static void main(String arg[])
    {
        Socket
        c=null;String
        line;
        DataInputStream
        is,is1;PrintStream
        os;
        try
        {
            c=new Socket("10.0.200.36",9999);
        }
        catch(IOException e)
        {
            System.out.println(e);
        }
        try
        {
            os=new
            PrintStream(c.getOutputStream());
            is=new DataInputStream(System.in);
            is1=new
            DataInputStream(c.getInputStream());do
            {
                System.out.println("Client:"
                );line=is.readLine();
                os.println(line);
                System.out.println("Server:" + is1.readLine());
```

```

}
while(line.equalsIgnoreCase("quit")==false);
is1.close();
os.close();
}
catch(IOException e)
{
System.out.println("Socket Closed!Message Passing is over");
}}

```

## **OUT PUT :**

### **SERVER**

C:\Program Files\Java\jdk1.5.0\bin>javac TCPserver1.java Note:

TCPserver1.java uses or overrides a deprecated API. Note:

Recompile with -deprecation for details

C:\Program Files\Java\jdk1.5.0\bin>java TCPserver1

Client: Hai Server

Server: Hai Client

Client: How are

you Server: Fine

Client: quit

Server: quit

### **CLIENT**

C:\Program Files\Java\jdk1.5.0\bin>javac TCPclient1.java Note: TCPclient1.java uses or overrides a deprecated API. Note: Recompile with -deprecation for details. C:\Program

Files\Java\jdk1.5.0\bin>java TCPclient1

Client: Hai Server

Server: Hai Client

Client: How are

you Server: Fine

Client: quit

Server: quit

## **RESULT:**

Thus the above program a client-server application for chat using TCP / IP was executed and successfully.



**DATE:**

### **3.C. FILE TRANSFER IN CLIENT & SERVER**

**AIM:**

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

**ALGORITHM:**

#### **CLIENT SIDE**

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

#### **SERVER SIDE**

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

**PROGRAM:**

#### **CLIENT SIDE**

```
import
java.net.*;
import java.io.*;
public class FileClient{
    public static void main (String [] args ) throws IOException
    { int filesize=6022386; // filesize temporary hardcoded
      long start = System.currentTimeMillis();
      int
      bytesRead;
      int current =
      0;
      // localhost for testing
```

```

Socket sock = new
Socket("127.0.0.1",13267);
System.out.println("Connecting...");
// receive file
byte [] mybytearray = new byte [filesize];
InputStream is = sock.getInputStream();
FileOutputStream fos = new FileOutputStream("source-
copy.pdf");BufferedOutputStream bos = new
BufferedOutputStream(fos); bytesRead =
is.read(mybytearray,0,mybytearray.length);
current = bytesRead;
// thanks to A. Cádiz for the bug fix
do {bytesRead =
    is.read(mybytearray, current, (mybytearray.length-
current));if(bytesRead >= 0) current += bytesRead;
} while(bytesRead > -1);
bos.write(mybytearray, 0 ,
current);bos.flush();
long end =
System.currentTimeMillis();
System.out.println(end-start);
bos.close();
sock.close();
}}

```

## **SERVER SIDE**

```

import
java.net.*;
import java.io.*;
public class FileServer
{
    public static void main (String [] args ) throws IOException {
        ServerSocket servsock = new ServerSocket(13267); while
        (true)
        {

            System.out.println("Waiting..."); Socket

```

```

sock =servsock.accept();
System.out.println("Acceptedconnection : "
+ sock); File myFile = new File
("source.pdf");
byte [] mybytearray = new byte [(int)myFile.length()];
FileInputStream fis = new FileInputStream(myFile);
BufferedInputStream bis = new
BufferedInputStream(fis);
bis.read(mybytearray,0,mybytearray.length);
OutputStream os = sock.getOutputStream();
System.out.println("Sending...");
os.write(mybytearray,0,mybytearray.length);
os.flush();
sock.close();
}}}

```

## OUTPUT:

### SERVER OUTPUT

C:\Program Files\Java\jdk1.6.0\bin>javac

FServer.javaC:\Program

Files\Java\jdk1.6.0\bin>java FServer

Waiting for clients...

Connection Established

Client wants

file:network.txt

### CLIENT OUTPUT

C:\Program Files\Java\jdk1.6.0\bin>javac FClient.javaC:\Program Files\Java\jdk1.6.0\bin>java

FClient

Connection request.Connected

Enter the filename: network.txt

Computer networks: A computer network, often simply referred to as a network, is a collection of computers and devices connected by communications channels that facilitates communications among users and allows users to shareresources with other user

## RESULT:

Thus the File transfer Operation is done & executed successfully

**DATE:**      **4. Write a program for congestion control using Leaky Bucket Algorithm.**

**AIM:**

To write a program for congestion control using Leaky Bucket Algorithm.

**ALGORITHM:**

**Initialization:**

1. The LeakyBucket class is initialized with a bucketSize and an outputRate.
2. Bucket is a queue that simulates the bucket holding packets.
3. LastTime keeps track of the last time the bucket leaked.

**Adding Packets:**

1. The addPacket method takes the packetSize as input.
2. It calculates the elapsed time since the last operation and calls the leak method to simulate the leaking process.
3. It checks if the bucket has enough space for the new packet. If not, the packet is dropped.

**Leaking Process:**

1. The leak method calculates how much data to leak based on the elapsed time and the outputRate.
2. It removes the appropriate amount of data from the bucket.

**PROGRAM:**

```
import java.util.LinkedList;
import java.util.Queue;
public class LeakyBucket {
    private int bucketSize;
    private int outputRate;
    private Queue<Integer> bucket;
    private long lastTime;
    public LeakyBucket(int bucketSize, int outputRate) {
        this.bucketSize = bucketSize;
        this.outputRate = outputRate;
        this.bucket = new LinkedList<>();
        this.lastTime = System.currentTimeMillis();
    }
    public void addPacket(int packetSize)
{
```

```

    long currentTime = System.currentTimeMillis();
    long elapsedTime = currentTime - lastTime;
    lastTime = currentTime;
    // Leak the bucket based on the elapsed time
    leak(elapsedTime);
    // Check if there is enough space to add the packet
    if (bucket.size() + packetSize > bucketSize)
    {
        System.out.println("Packet of size " + packetSize + " dropped (Bucket overflow).");
    } else
    {
        for (int i = 0; i < packetSize; i++)
        {
            bucket.add(1);
        }
        System.out.println("Packet of size " + packetSize + " added.");
    }
}

private void leak(long elapsedTime)
{
    // Calculate how much to leak
    int leakAmount = (int) (elapsedTime * outputRate / 1000);
    for (int i = 0; i < leakAmount && !bucket.isEmpty(); i++)
    {
        bucket.poll();
    }
    System.out.println("Leaked " + leakAmount + " units. Current bucket size: " +
bucket.size());
}

public static void main(String[] args) {
    int bucketSize = 10;
    int outputRate = 1; // units per second
    LeakyBucket leakyBucket = new LeakyBucket(bucketSize, outputRate);
    // Simulating the arrival of packets
    leakyBucket.addPacket(4);
    try { Thread.sleep(1000); } catch (InterruptedException e) { e.printStackTrace(); }
}

```

```

        leakyBucket.addPacket(3);
        try { Thread.sleep(2000); } catch (InterruptedException e) { e.printStackTrace(); }
        leakyBucket.addPacket(6);
        try { Thread.sleep(1000); } catch (InterruptedException e) { e.printStackTrace(); }
        leakyBucket.addPacket(2);
        try { Thread.sleep(1000); } catch (InterruptedException e) { e.printStackTrace(); }
        leakyBucket.addPacket(1);
    }
}

```

### **OUTPUT:**

```

java -cp /tmp/M3lQefDXSr/LeakyBucket
Leaked 0 units. Current bucket size: 0
Packet of size 4 added.
Leaked 1 units. Current bucket size: 3
Packet of size 3 added.
Leaked 2 units. Current bucket size: 4
Packet of size 6 added.
Leaked 1 units. Current bucket size: 9
Packet of size 2 dropped (Bucket overflow).
Leaked 1 units. Current bucket size: 8
Packet of size 1 added.

```

### **Result:**

Congestion control using Leaky Bucket Algorithm executed and verified successfully.

**DATE:**

## **5. Simulation of DNS using UDP sockets.**

**AIM:**

To write a program to Simulation of DNS using UDP sockets..

**ALGORITHM:**

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

**PROGRAM:**

**/ UDP DNS ServerUdpdnsserver**

```
java import
java.io.*;import
java.net.*;
public class udpdnsserver
{
private static int indexOf(String[] array, String str)
{
str = str.trim();
for (int i=0; i < array.length; i++)
{
if (array[i].equals(str)) return i;
}
return -1;
}
public static void main(String arg[])throws IOException
{
String[] hosts = {"yahoo.com", "gmail.com","cricinfo.com", "facebook.com"};
String[] ip = {"68.180.206.184", "209.85.148.19","80.168.92.140",
"69.63.189.16"};
System.out.println("Press Ctrl + C to
Quit");while (true){
DatagramSocket serversocket=new
DatagramSocket(1362);byte[] senddata = new
```

```

byte[1021];
byte[] receivedata = new byte[1021];
DatagramPacket recvpack = new DatagramPacket(receivedata,
receivedata.length);serversocket.receive(recvpack);
String sen = new String(recvpack.getData()); InetAddress ipaddress =
recvpack.getAddress(); int port = recvpack.getPort();
String capsent;
System.out.println("Request for host " + sen);
if(indexOf (hosts, sen) != -1) capsent =
ip[indexOf(hosts, sen)]; else capsent = "Host
Not Found";
senddata = capsent.getBytes();
DatagramPacket pack = new DatagramPacket (senddata,
senddata.length,ipaddress,port);serversocket.send(pack);
serversocket.close();
}
}
}

```

#### **//UDP DNS Client –**

```

Udpdnsclient
.java import java.io.*; import
java.net.*;public class udpdnsclient
{
public static void main(String args[])throws IOException
{
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));DatagramSocket clientsocket = new
DatagramSocket();
InetAddress ipaddress; if (args.length ==
0) ipaddress =
InetAddress.getLocalHost(); else
ipaddress = InetAddress.getByName(args[0]); byte[]
senddata =new byte[1024];
byte[] receivedata = new
byte[1024];int portaddr = 1362;

```



```

System.out.print("Enter the hostname
: ");String sentence = br.readLine();
Senddata = sentence.getBytes();
DatagramPacket pack = new DatagramPacket(senddata,senddata.length,
ipaddress,portaddr);clientsocket.send(pack);
DatagramPacket recvpack =new
DatagramPacket(receivedata,receivedata.length);
clientsocket.receive(recvpack);
String modified = new String(recvpack.getData());
System.out.println("IP Address: " + modified);
clientsocket.close();
}
}

```

## **OUTPUT :**

### **Server**

```

javac
udpdnsserver.java
java udpdnsserver
Press Ctrl + C to Quit Request for host yahoo.com
Request for host cricinfo.com
Request for host youtube.com

```

### **Client**

```

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

```

Enter the hostname :

youtube.com IP Address: Host

Not Found

### **RESULT:**

Thus the above program a client-server application for chat using UDP was executed and successfully

**DATE:**

**6.a. Write a code simulating ARP protocols.**

## **AIM**

To implement Address Resolution Protocol.

## **ALGORITHM**

### **CLIENT SIDE**

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

### **SERVER SIDE**

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

## **PROGRAM**

### **ARP CLIENT**

```
import java.io.*;
import
java.net.*;
class ArpClient
{
public static void main(String args[])throws IOException
{
try
{
Socket ss=new
```

```

Socket(InetAddress.getLocalHost(),1100);
PrintStream ps=new
PrintStream(ss.getOutputStream());
BufferedReader br=new
    BufferedReader(newInputStreamReader(System.in));String ip;
    System.out.println("Enter the IPADDRESS:");
    ip=br.readLine();
ps.println(ip);
String str,data;
BufferedReader
br2=new
BufferedReader(newInputStreamReader(ss.getInputStream()));
System.out.println("ARP From Server::"); do
{
str=br2.readLine();
System.out.println(st
r);
}
while(!(str.equalsIgnoreCase("end")));
}
catch(IOException e)
{
System.out.println("Error"+e);
}}

```

### **ARP SERVER**

```

import java.io.*;
import
java.net.*;
class ArpServer
{
public static void main(String args[])throws IOException
{
try
{
ServerSocket ss=new

```

```

ServerSocket(1100);Socket
s=ss.accept();
PrintStream ps=new PrintStream(s.getOutputStream());
BufferedReader br1=new
BufferedReader(newInputStreamReader(s.getInputStream()));String ip;
ip=br1.readLine();
Runtime
r=Runtime.getRuntime();
Process p=r.exec("arp -a "+ip);
BufferedReader br2=new
BufferedReader(newInputStreamReader(p.getInputStream()));String str;
while((str=br2.readLine())!=null)
{
ps.println(str);
}}
catch(IOException e)
{
System.out.println("Error"+e); }}}

```

## OUTPUT

C:\Networking Programs>java ArpServer C:\Networking Programs>java ArpClient Enter the IPADDRESS:

192.168.11.58

ARP From Server::

Interface:	192.168.11.57 on Interface 0x1000003	Internet Address	Physical Address	Type
	192.168.11.58	00-14-85-67-11-84	dynamic	

## RESULT

Thus the implementation of ARP is done & executed successfully.

**DATE:** 6.b. Write a code simulating RARP protocols.

**AIM:**

To write a java program for simulating RARP protocols.

**ALGORITHM:**

**CLIENT**

1. Start the program
2. using datagram sockets UDP function is established. 2. Get the MAC address to be converted into IP address. 3. Send this MAC address to server.
4. Server returns the IP address to client.

**SERVER**

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

**CLIENT:**

```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientarp12
{
    public static void main(String args[])
    {
        try
        {
            DatagramSocket client=new DatagramSocket();
            InetAddress
            addr=InetAddress.getByName("127.0.0.1"); byte[]
            sendbyte=new byte[1024];
            byte[] receivebyte=new byte[1024];
            BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Enter the Physical address (MAC):");
            String
            str=in.readLine();
            sendbyte=str.getBytes();
```

```

        DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,1309);
        client.send(sender);
        DatagramPacket-receiver=new
DatagramPacket(receivebyte,receivebyte.length);client.receive(receiver);
        String s=new String(receiver.getData());
        System.out.println("TheLogical 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 Serverarp12
{
    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());
                Strings=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=newDatagramPacket(sendbyte,sendbyte.length,addr,port);

                server.send(send
                er);break;
            }
        }
        break;
    }
}
catch(Exception e)
{
    System.out.println(e);
}
}
}

```

### OUTPUT:

```

I:\ex>java
Serrarp12
I:\ex>java
Clientarp12
Enter the Physical address
(MAC):6A:08:AA:C2
The Logical Address is(IP): 165.165.80.80

```

### RESULT:

Thus the implementation of RARP is done & executed successfully.



**DATE:**

## **7. Study of Network simulator (NS) and Simulation of Congestion Control Algorithms using NS.**

### **OBJECTIVE:**

To study of network simulator (ns) and simulation of congestion control algorithms using ns.

### **SYSTEM REQUIREMENTS:**

PC: Pentium or higher

One LAN card onboard or on PCI slot with 10/100Mbps speed. 128MB RAM

500MB free space on

Hard drive CD ROM drive

Serial port, LPT port & USB port installed on system

Operating System: Windows 2000 or higher

### **THEORY:**

#### **LTS-01 Local area network / wireless local area network trainer system:**

It is designed to help students understand the basic concepts, modes of operation and protocols involved in networking. The trainer has integrated hardware flow control on panel board for better understanding of different types of LAN topologies involved in networking. The trainer system is provided with windows-based user friendly software with analysis of protocols, different layers, network and measurement of error rate and throughput.

Students can easily do connections in different topologies and can learn actual data transfer either through hardware or through simulated network concept. Facility is provided into system software to introduce errors into packets being sent and analyze the effect of error on different protocols and hence find the effect on throughput graph as well.

Trainer and its various types of experimentation using this system. This system works into server-client base. For any topology user has to select one server and select the network type whether it is LAN or WLAN. To understand the topology concept user can connect two or more clients to hardware. Depending on the topology selected user will have option to select protocols for the selected topology. Upon selection of protocol user can then create network of connected computers.

In any network which is created by user server can send or can communicate with any of the clients however clients can communicate only with server, no client to client communication is possible. Transmitter port protocol & network analysis can be done after communication is over between server and clients. Throughput v/s Packet size graph can

be plotted for which at least two file transfers should be carried out. This plot can be printed to attach in the lab exercise sheet.

For the LAN network LAN cards must be installed prior to start work on this trainer. For wire less LAN USB ports should be available on the computers which are to be used for experimentation. In WLAN wireless access cards gets connected to computer USB ports and access point gets connected to hardware device.

### **L-SIM LAN Protocol Simulator & Analyzer Software:**

It is designed to teach the basic concepts, topologies & various protocols involved in networking. The software is provided with analysis of protocols, different layers, network and measurement of error rate and throughput. Facility is provided to introduce errors into packets beingsent and analyze the effect of error on different protocols and hence find the effect on throughput graph as well. Software is supported with neat operating instruction manual and online help.

### **MODEL WINDOW DIAGRAM FOR L-SIM**



### **N-SIM Network simulation software:**

It is developed to provide basic understanding and implementation of various advanced concepts in networking. The software provides an opportunity to understand network fundamentals through animations & simulations. The simulation provides for network experimentation with various LAN and WAN protocols, network devices, routers, encryption, decryption, file transfer, error insertion and analysis of error rate and throughput etc. This software covers Ethernet LAN, wireless LAN and router. All networking theory is explained using simulation and animation.

## MODEL WINDOW DIAGRAM FOR N-SIM



Rapid advances in computer & communication technologies have resulted in the increasing merger of these two fields. The lines have blurred among computing, switching & digital transmission equipment; and the same digital techniques are being used for data, audio & video transmission. Merging & evolving technologies, coupled with increasing demands for efficient & timely collection, processing & dissemination of information, have led to the development of integrated systems that transmit & process all types of data.

These integrated systems are broadly divided as follows

- **DATA COMMUNICATION** dealing with transmission, transmission media, signal decoding, interfacing, data link control & multiplexing
- **NETWORKING** deals with the technology & architecture of communication network
- **COMMUNICATION PROTOCOLS** which covers the architecture as well

as analysis of individual protocols at various layers depending on the hardware & software

Network laboratory is designed & developed considering the curriculum offered by Anna University. Trainers offered under network laboratory are designed for students at all level to study and understand all the concepts of data communication, data transfer using serial and parallel ports, Ethernet and wireless LAN with complete protocol understanding and actual hands on with hardware & software with ease.

Network laboratory consists of DCT-03 Data communication trainer kit, LTS- 01 LAN / Wireless LAN training system, L-SIM LAN / WLAN protocol simulator and analyzer software & N-SIM Network simulation software.

**The DCT-03:** Data communication trainer is a unique trainer kit for the development of exercises and theoretical-experimental courses to understand the basic concept and working of modes and protocols in serial and parallel communication. The trainer kit consists of functional blocks for serial and parallel communications system.

The trainer kit is highly innovative from a technological as well as an educational point of view. The trainer kit is used as “basic unit” to examine all the peculiar operating standards of serial and parallel communication system. The only external equipments required are two Computers with serial and parallel communication ports and an

Oscilloscope. Utmost care has been laid in the design and quality control of all circuits, to ensure the repeatability of the results of the experiments.

Data communication is a term referred when the sender and receiver are digital devices, which communicate with each other by means of binary information. The objective of this trainer kit is to clear the various aspects of the data communications which comprise of

- The information source or sender.
- The medium for carrying information.
- The information receiver.
- The communication protocols, which ensure proper transfer of data.

With an increasing demand in information exchange the field of data communication technique is emerging as the only solution, to satisfy the various needs of today's communication sector and to achieve very high bandwidth along with highest accuracy. The communication media is shifting from analog signal transfer towards digital communication.

With PC becoming the biggest storage devices in digital form, it becomes the main source and destination for information exchange. With rapid growth in both the communication technologies as well as computer hardware and software technologies, these two fields are merged to form a data communication network. Now the digital data is used for data, voice and image transmission.

Depending upon the application the communication link can be of point to point communication between two devices or a multipoint communication between at least 3 devices and data transfer can be serial or in parallel form.

## **RESULT:**

The study of network simulator (ns) and simulation of congestion control algorithms using ns is executed and verified.

**DATE:**                      **8. Study of TCP/UDP performance using Simulation tool.**

**AIM:**

To Study of TCP/UDP performance using Simulation tool.

**TOOLS USED:**

Opnet Simulator

**INTRODUCTION:**

The transport layer protocols provide connection- oriented sessions and reliable data delivery services. This paper seeks to reflect a comparative analysis between the two transport layer protocols, which are TCP/IP and UDP/IP, as well to observe the effect of using these two protocols in a client server network. The similarities and differences between TCP and UDP over the Internet are also presented in our work. We implement a network structure using Opnet Modeler and finally, based on the practical results obtained we present the conclusions-showing the difference between these two protocols and how they work.

The transport layer is not just another layer. It is the heart of the whole protocol hierarchy. Its task is to provide reliable, cost-effective data transport from the source machine to the destination machine, independently of the physical network or networks currently in use.

TCP and UDP are transport layer components that provide the connection point through which applications access network services. TCP and UDP use IP, which is a lower-layer best effort delivery service. IP encapsulates TCP packets and UDP datagrams and delivers this information across router-connected internet works.

The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective service to its users, normally processes in the application layer. To achieve this goal, the transport layer makes use of the services provided by the network layer. Without the transport layer, the whole concept of layered protocols would make little sense e.g. The Transport Layer prepares applications data for transport over the network and processes network data to be used by applications. It is responsible for the end-to-end transfer of data over the network and is the four of the OSI model. The Transport layer meets a number of functions:

- enabling the applications to communicate over the network at the same time when using a single device;
- ensure that all amount of data is receive by the correct application;
- responsible for fragmentation and reassembly;
- develop mechanism for handling errors.

### Comparison Between TCP And UDP

Service	TCP	UDP
Flow controls	The receiver can signal the sender to slow down.	ACKs, which are used in TCP to control packet flow, are not returned.
Connection setup	It takes time, but with TCP reliability is ensured.	No connection is required.
Guaranteed message delivery	Returns acknowledgments.	UDP does not return ACKs, the receiver can't signal that packets have been successfully delivered.
Congestion controls	Network devices can take advantage of TCP ACK to control the behavior of sender.	If ACK, are missing, the network cannot signal congestion to the sender.

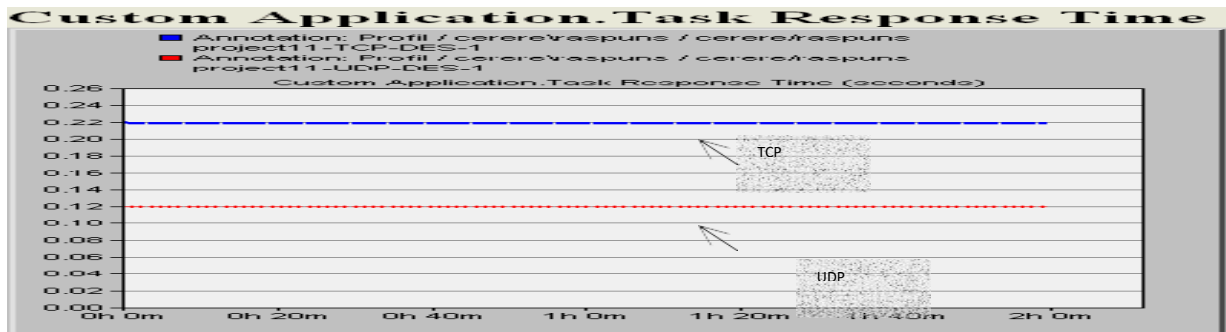
A big difference between TCP and UDP is the congestion control algorithm. For the TCP, congestion algorithm prevents the sender from overrunning the network capacity, while TCP can adapt the sender's rate with the network capacity and attempt to avoid potential congestions problems.

User Datagram Protocol (UDP), another transport protocol in IP networks, is described e.g. The User Datagram Protocol (UDP) provides an unreliable connectionless delivery service using IP to transport messages between machines e.g. [5]. It uses IP to carry messages, but adds the ability to distinguish among multiple destinations within a given host computer. Is a connectionless protocol which doesn't provide flow control, reliability or error recovery and the retransmissions of data in case of errors must be ordered by other protocols. UDP is designed for applications that do not have to recompose the data segment that arrives from the sender. In another way, application-level protocols are directly responsible for the security of data transmitted.

Difference from the TCP is that there is no mechanism for error detections. If applications that use UDP doesn't have their own mechanism for information retrieval can lose those data and be forced to retransmitted again. On the other side this applications are not slow down by the confirmation process and the memory will be available for work much faster.

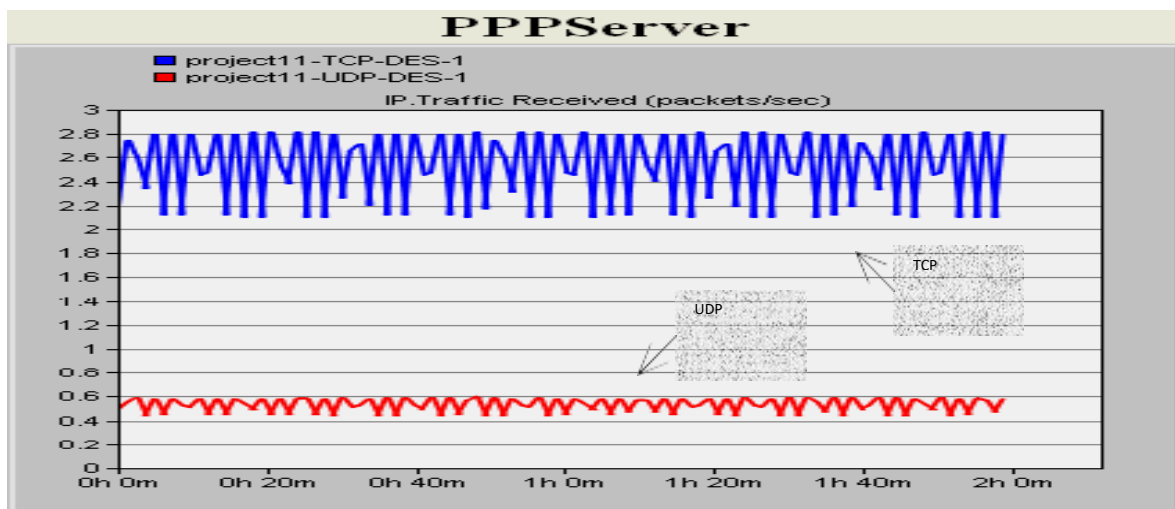
## SIMULATION RESULTS:

The simulation time is set for two hours data transfer between LAN network and the server with no packet latency and packet discard ratio of 0% while packets traverse thru the WAN. The task response time, in seconds, Fig. 1, shows how long the application need to be completed. The time when using TCP to complete the task is greater that the one using UDP. When using TCP, source and destination need to perform a three-way

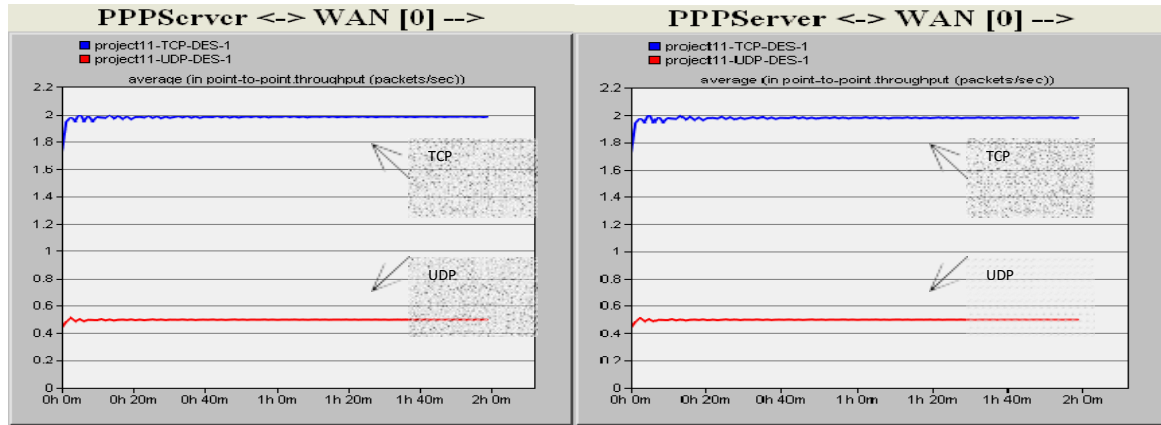


handshake before starting sending data and all amount of data need to be acknowledge by the destination when it is receive, so is taking more time than UDP, which doesn't perform this tasks.

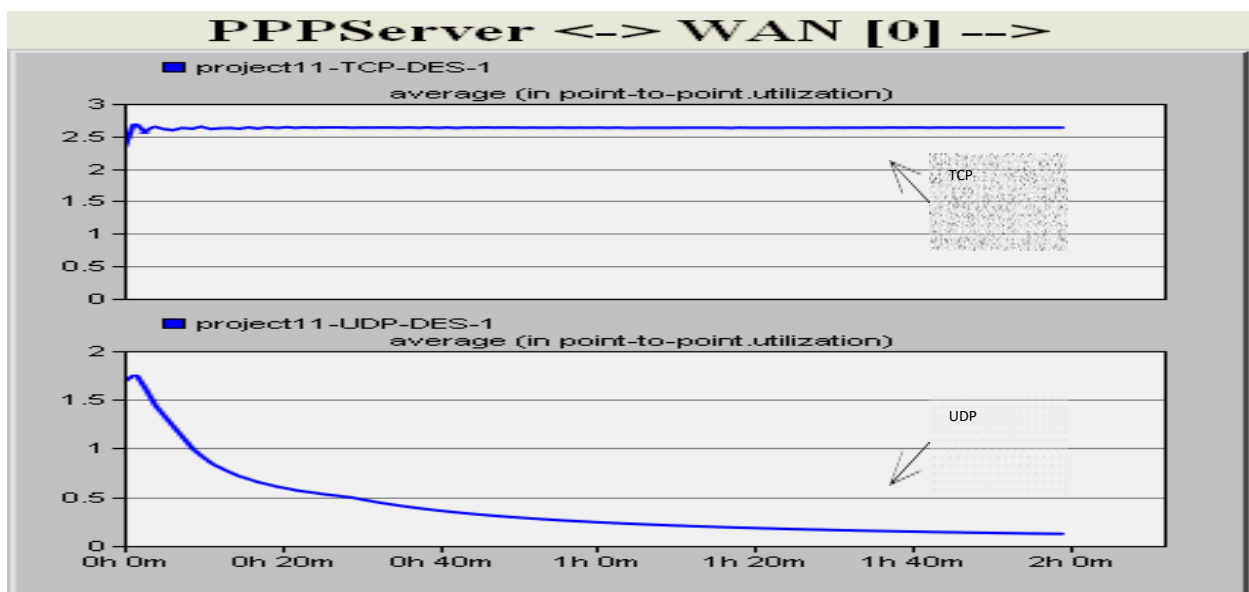
Response time for TCP and UDP



1. Traffic received (packets/sec) for the server



## 2. Traffic/Link utilization from the WAN to the server



Link utilization with a 0.5% packets discard ratio

The main difference between these two protocols is that TCP provides reliability and congestion control services, while UDP is orientated to improve performance.

The most important and common thing that TCP and UDP are using is the ability to set a host-to-host communication channel, so the packets will be delivered between processes running on two different computers. UDP is the right choice for application where reliability is not a must but the speed and performance is. Instead, TCP, even if it takes more time for the processes, has additional functions like same order delivery, reliability and flow control. As future work, we plan to conduct several studies regarding packets routing in computer networks to improve the fairness of data transmissions using different network protocols.



**RESULT:**

Thus the TCP/UDP performance has been simulated successfully using OPNET.

**DATE:**

## **9.a Simulation of Distance Vector Routing algorithm.**

**AIM:**

To implement the Distance – Vector Routing Algorithm

### **APPARATUS REQUIRED:**

1. VI-RTSIM software.
2. Personal computer.

### **THEORY:**

#### **Distance Vector Algorithm:**

- ❖ A Distance vector routing, each router periodically share its knowledge about the entire network with its neighbors.
- ❖ The three keys to understand this algorithm are
  1. Knowledge about the whole network.
  2. Routing only to neighbor.
  3. Information sharing at regular intervals.

#### **Knowledge about the whole network:**

- ❖ Each router shares its knowledge about entire network. It sends all of its collected knowledge about the network to its neighbors.

#### **Routing only to neighbor:**

- ❖ Each router periodically sends its knowledge about the network only to those routers to which it has direct links. It sends whatever knowledge it has.

#### **Information sharing at regular intervals:**

- ❖ The every 30 seconds, each router sends its information about the whole network to its neighbors.

#### **Sharing Information:**

- ❖ LAN's are connected by router, represented by the assuming A, B, C, D, E and F.
- ❖ Distance vector routing simplifies the routing process by assuming a cost of one unit for every link.
- ❖ The efficiency of transmission is a function only of the number of links required to reach a destination. In this, the cost is hop count.

**Routing Table:**

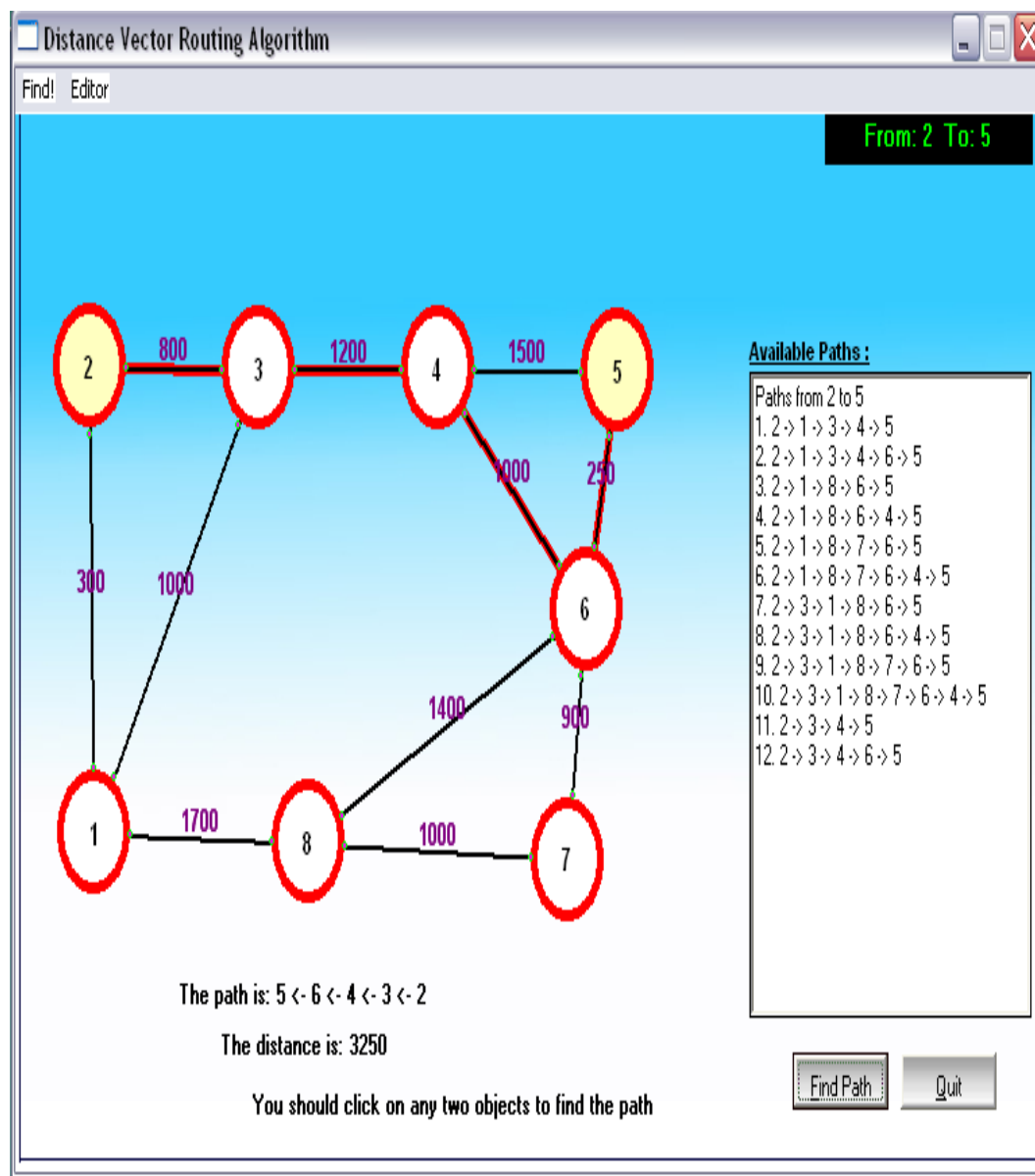
- ❖ Each router gets its initial knowledge about the internet work and how it uses shared information to update that knowledge.
- ❖ The routing table has 3 columns network address, router ID.
- ❖ The first block is final destination of packet.
- ❖ The second block is no of hop count.
- ❖ The third block is that to which a packet delivers must.

**Updating algorithm:**

- ❖ Updating algorithm requires that the router first has one hop to the hop count field for each advertised router.
- ❖ The router should apply the below rules to each router, if the advertised destination is not in routing table
- ❖ If next hop field is same, router should replace the entry in the table with advertised one.
- ❖ If next hop field is same, router should replace the entry in the table with advertised one.
- ❖ . If next hop field is not the same, advertised hop count is smaller than the one in the table, the router should replace the entry in the table with new one.
- ❖ IF advertised hop count is not smaller, the router should do no routing.

**PROCEDURE**

1. Open VI-RTSIM software from desktop
2. Click the Simulation menu bar
3. Select the "Distance – Vector Routing Algorithm" option from Routing algorithm menu bar.
4. Network with routers connected through link is drawn by using option in editor (add router, join link, delete router, delete link, Add caption to link, add caption to router)
5. Select any two nodes to find the shortest distance between them.
6. Click the Find path Button to run the program.
7. Now the shortest paths between the two nodes are calculated.



Find Shortest Path

Distance vector table:

To	1	2	3	4	5	6	7	8
1	0	300	1000	0	0	0	0	1700
2	300	0	800	0	0	0	0	0
3	1000	800	0	1200	0	0	0	0
4	0	0	1200	0	1500	1000	0	0
5	0	0	0	1500	0	250	0	0
6	0	0	0	1000	250	0	900	1400
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Calculate

Distance: From: 2 To:

Node	1	2	3	4	5	6	7	8
Distance	300	0	800	2000	3250	3000	3000	2000

Path:

Node	1	2	3	4	5	6	7	8
Path	2	0	2	3	6	4	8	1

The path is: 5 <- 6 <- 4 <- 3 <- 2  
The distance is: 3250

## RESULT:

Thus Distance Vector routing algorithm has been implemented and shortest-path has been circulated.

**DATE:**

## **9.b Simulation of Link State Routing Algorithm.**

**AIM:**

To implement the Link State Routing Algorithm

**APPARATUS REQUIRED:**

1. VI-RTSIM software.
2. Personal computer.

**THEORY:**

**Link State Vector Algorithm:**

- ❖ In Link state routing, each router share its information of its neighbors with every other router in the inter-network.

**Knowledge about the neighborhood:**

- ❖ Instead of sending its entire routing table, a router sends information about its neighborhood only.

**To all router:**

- ❖ Each router send this information to every other router on the internetworking, not just to its neighbors.
- ❖ If s does so by a process called “flooding” it means that a router sends its information.

**Information sharing when there is a Change:**

- ❖ Each router sends out information about the neighbors when there is a change.

**Information sharing:**

- ❖ Link state routing process use the same internet work as distance vector algorithm.
- ❖ Here each other sends its knowledge about its neighbors to every other router in the internet work.
- ❖ Cost is applied only by routers and not by any other station on a network, if cost was added by every station, instead of by routers alone, it would accumulate unpredictably.
- ❖ Cost is applied as a packet leaves the router rather than as it enters. Most networks are broadcast networks. When a packet is in network every station, including the router, can pick it up, we cannot assign any cost to a packet.

**Link state packet:**

- ❖ When a router floods the network with information about its neighborhood, it is said to be advertising. The basis of this advertising is a short packet called a link state packet (LSP).

Advertiser	Network	Cost	Neighbor
------------	---------	------	----------

**Getting information about neighbors:**

- ❖ A router gets its information about its neighbors by periodically sending them a short greeting packet.
- ❖ If the neighbor responds to the greeting as expected, it is assumed to be alive and functioning.

**Initialization:**

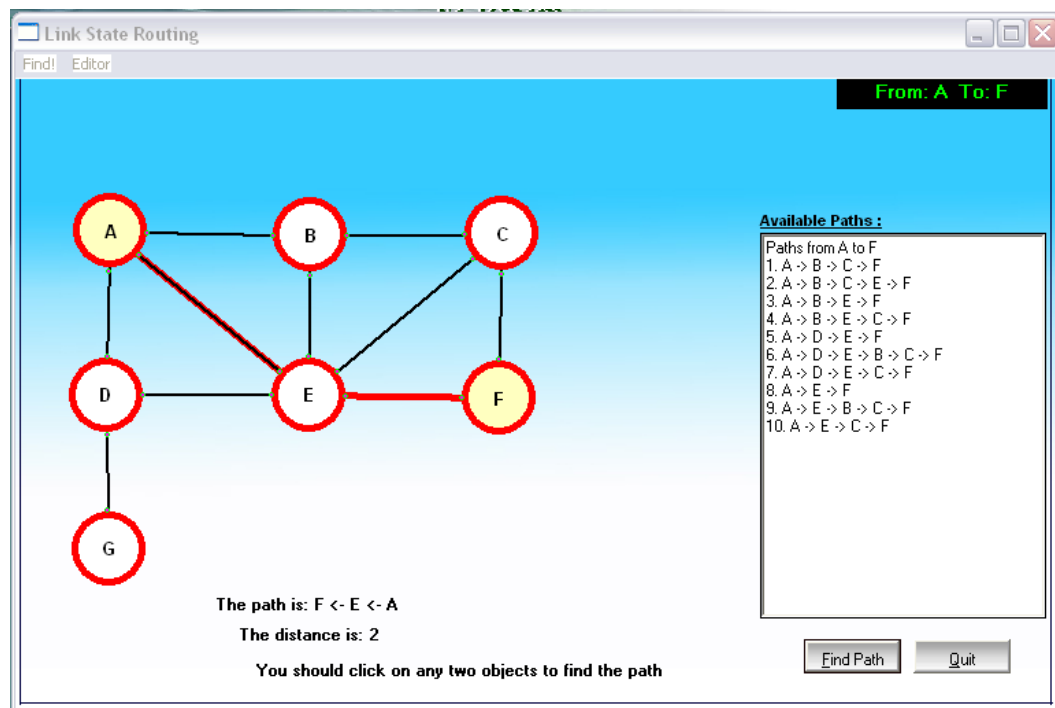
- ❖ Imagine that all routers in our sample internet work come up at the same time.
- ❖ Each router sends a greeting packet to its neighbors to find out the state of each link.

**Link – State Database:**

- ❖ Every router every LSP and puts the information into a link-state database.
- ❖ Because every router receives the same LSPs every router builds the same database.
- ❖ It stores this database on its disk and uses it to calculate its routing table. If a router is added to be deleted from the system, the whole database must be shared for fast updating.

**PROCEDURE**

1. Open VI-RTSIM software from desktop
2. Click the Simulation menu bar
3. Select the “Link State Routing Algorithm” option from Routing algorithm menu bar.
4. Network with routers connected through link is drawn by using option in editor(add router, joinlink, delete router, delete link, Add caption to link, add caption to router)
5. Select any two nodes to find the shortest distance between them.
6. Click the Find path Button to run the program.
7. Now the shortest paths between the two nodes using link state routing algorithm was calculated.





Find Shortest Path

Link State table:

To	A	B	C	D	E	F	G
A	0	1	0	1	1	0	0
B	1	0	1	0	1	0	0
C	0	1	0	0	1	1	0
D	1	0	0	0	1	0	1
E	1	1	1	1	0	1	0
F	0	0	1	0	1	0	0
G	0	0	0	1	0	0	0

Calculate

Distance: From: A To:

Node	A	B	C	D	E	F	G
Distance	0	1	2	1	1	2	2

Path:

Node	A	B	C	D	E	F	G
Path	0	1	2	1	1	5	4

The path is: F <- E <- A

The distance is: 2

## RESULT:

Thus Link-State routing algorithm has been implemented and shortest-path has been circulated.