B.Ajay
21011102020

SHIV NADAR
UNIVERSITY
CHENNAI

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF ENGINEERING

LABORATORY RECORD

B.TECH
(YEAR : 20 - 20)

NAME : .....

REG. NO. : .....

DEPT. : ..... **SEM. :** ..... **CLASS & SEC :** .....

B.Ajay
21011102020

SHIV NADAR
UNIVERSITY
CHENNAI

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of the practical work done in the

..... **Laboratory by**

Name .....

Register Number .....

Semester .....Class & Sec. .....

Branch .....

SHIV NADAR UNIVERSITY Chennai

During the Academic year .....

Faculty **Head of the Department**

Submitted for the.....Practical Examination held at SNU CHENNAI on.....

Internal Examiner **External Examiner**

B.Ajay
21011102020

INDEX

Name : ..... Reg. No. .....

Sem : ..... Class & Sec : .....

Ex. No.	Date of Expt.	Title of the Experiment	Page No.	Signature of the Faculty	Remarks

B.Ajay
21011102020

Ex. No: 1	Network Commands
22/12/2022	

Aim :

To use commands like

- tcpdump
- netstat
- ifconfig
- nslookup
- traceroute

Commands:

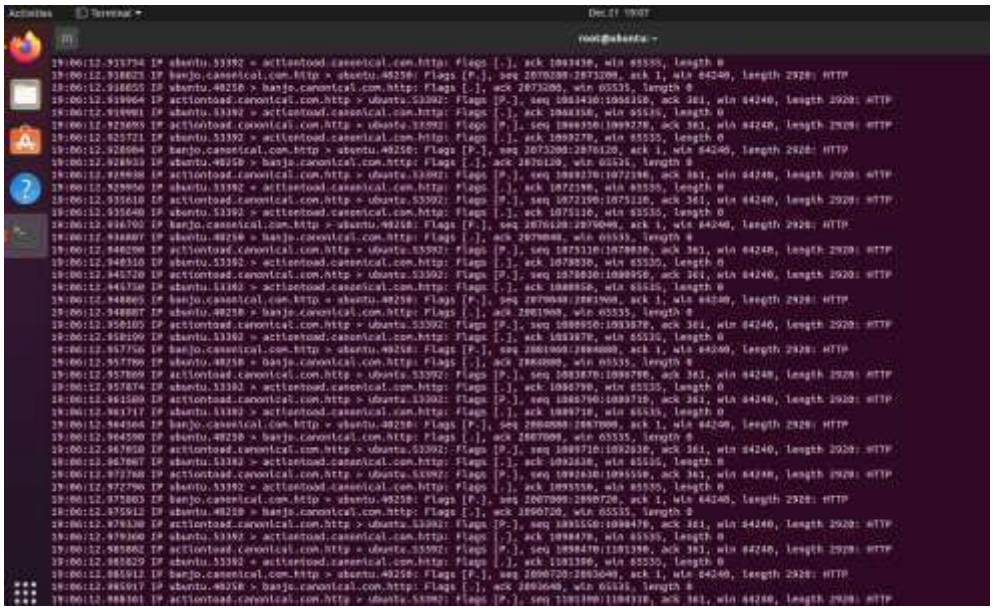
1.tcpdump:

Flags :-

Tcpdump -D

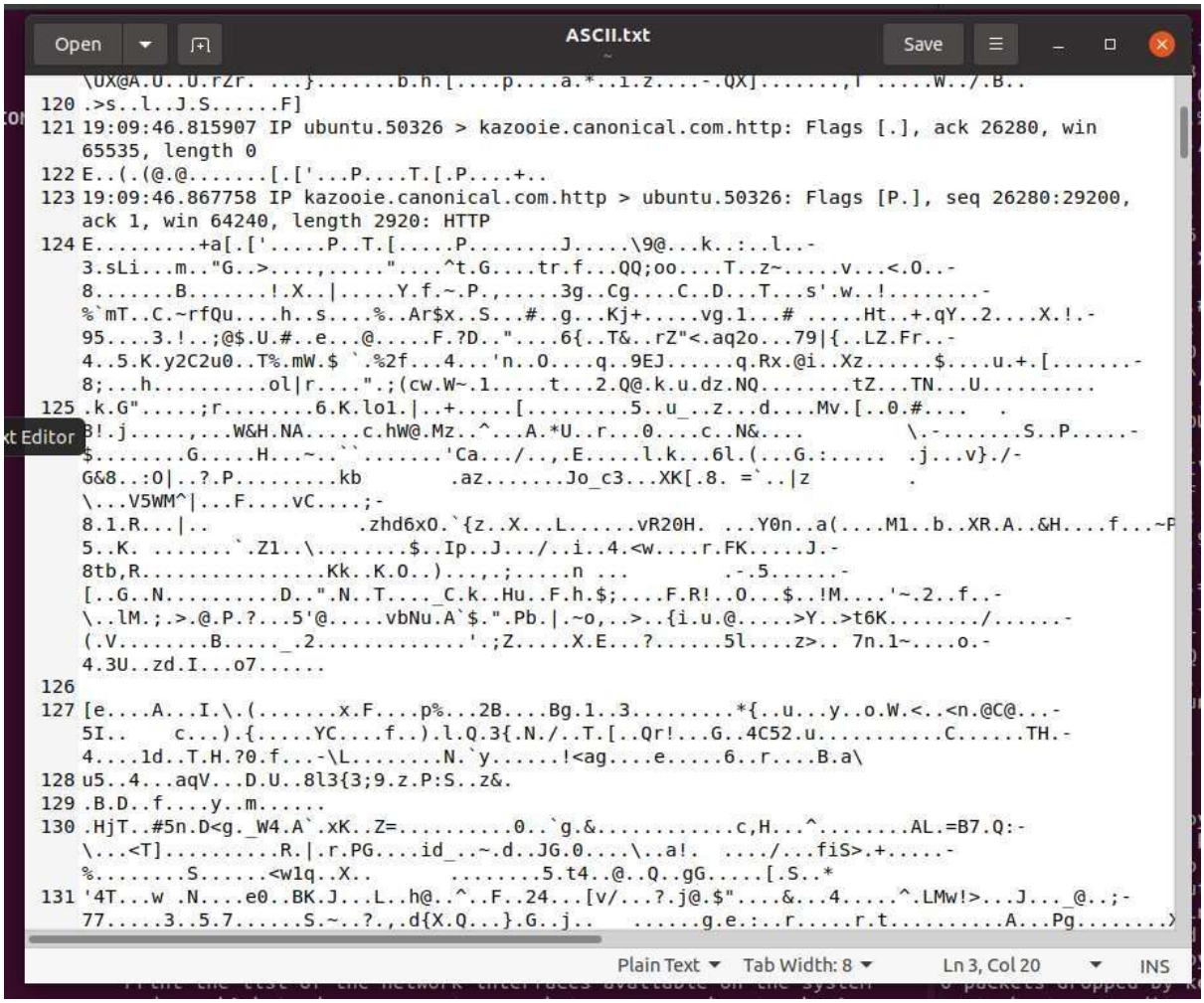
```
root@ubuntu:~# which tcpdump
/usr/sbin/tcpdump
root@ubuntu:~# tcpdump -D
1.ens33 [Up, Running]
2.lo [Up, Running, Loopback]
3.any (Pseudo-device that captures on all interfaces) [Up, Running]
4.bluetooth-monitor (Bluetooth Linux Monitor) [none]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
```

Tcpdump -I any;



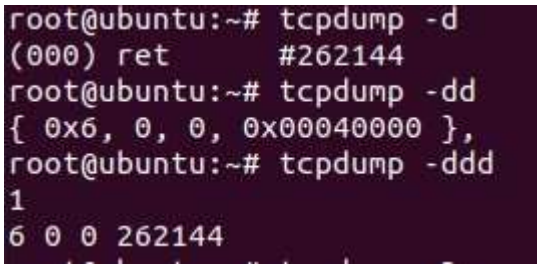
B.Ajay
21011102020

Tcpdump -A—ASCII dump of all packets



Tcpdump- d Dumps packet contents in human readable

form tcpdump-dd Dumps in to C Format tcpdump-ddd



Tcpdump – I ens33 listening to a particular wireless interface

B.Ajay
21011102020

2.Netstat

Netstat -I – list all Interface available ;

```
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
ens33     1500   626184      0      0 0        104903      0      0      0 BMRU
lo       65536     1218      0      0 0          1218      0      0      0 LRU
```

Netstat :

Display network statistics

```
root@ubuntu:~# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 ubuntu:48738            123.208.120.34.bc:https TIME_WAIT
tcp        0      0 ubuntu:49838            ec2-35-161-26-194:https ESTABLISHED
tcp        0      0 ubuntu:46194            stackoverflow.com:https ESTABLISHED
udp        0      0 ubuntu:bootpc           192.168.152.254:bootps  ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    45154    /run/user/1000/systemd/notify
unix  3      [ ]         DGRAM      CONNECTED     27031    /run/systemd/notify
unix  2      [ ]         DGRAM                    27045    /run/systemd/journal/syslog
unix  17     [ ]         DGRAM      CONNECTED     27055    /run/systemd/journal/dev-log
unix  8      [ ]         DGRAM      CONNECTED     27059    /run/systemd/journal/socket
unix  3      [ ]         STREAM     CONNECTED     46840
unix  3      [ ]         SEQPACKET  CONNECTED     82558
unix  3      [ ]         STREAM     CONNECTED     47506
unix  3      [ ]         STREAM     CONNECTED     42621    /run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     36204    /run/dbus/system_bus_socket
unix  2      [ ]         STREAM     CONNECTED     80093
unix  3      [ ]         STREAM     CONNECTED     58571
unix  3      [ ]         STREAM     CONNECTED     78379
unix  3      [ ]         STREAM     CONNECTED     49712
unix  3      [ ]         STREAM     CONNECTED     50279    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     47521    @/home/ajay/.cache/ibus/dbus-7h509Es7
unix  3      [ ]         STREAM     CONNECTED     45366    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     36500
unix  3      [ ]         STREAM     CONNECTED     85938    @/tmp/dbus-OFwog0MGn7
unix  3      [ ]         STREAM     CONNECTED     85934
unix  3      [ ]         STREAM     CONNECTED     47644    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     85979
unix  3      [ ]         STREAM     CONNECTED     46949    @/tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTED     85993
unix  3      [ ]         STREAM     CONNECTED     57293    /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     50734
unix  3      [ ]         STREAM     CONNECTED     50476    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     46999    /run/user/1000/bus
unix  2      [ ]         DGRAM                    46952
```

Netstat -s ->display network statistics.

```
Ip:
    Forwarding: 2
    114128 total packets received
    1 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    114112 incoming packets delivered
    106043 requests sent out
    20 outgoing packets dropped
Icmp:
    40 ICMP messages received
    0 input ICMP message failed
    ICMP input histogram:
        destination unreachable: 40
    40 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 40
IcmpMsg:
        InType3: 40
        OutType3: 40
Tcp:
    176 active connection openings
    0 passive connection openings
    4 failed connection attempts
    5 connection resets received
    3 connections established
    111503 segments received
    103959 segments sent out
    2 segments retransmitted
    0 bad segments received
    41 resets sent
Udp:
    2529 packets received
    40 packets to unknown port received
    0 packet receive errors
    2055 packets sent
    0 receive buffer errors
```

3.Ifconfig

Ifconfig

```
root@ubuntu: ~
root@ubuntu:~# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.152.131  netmask 255.255.255.0  broadcast 192.168.152.255
        inet6 fe80::5971:eaa2:506c:c626  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:f0:88:2e  txqueuelen 1000  (Ethernet)
        RX packets 626258  bytes 936451127 (936.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 104936  bytes 6907830 (6.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1226  bytes 127813 (127.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1226  bytes 127813 (127.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

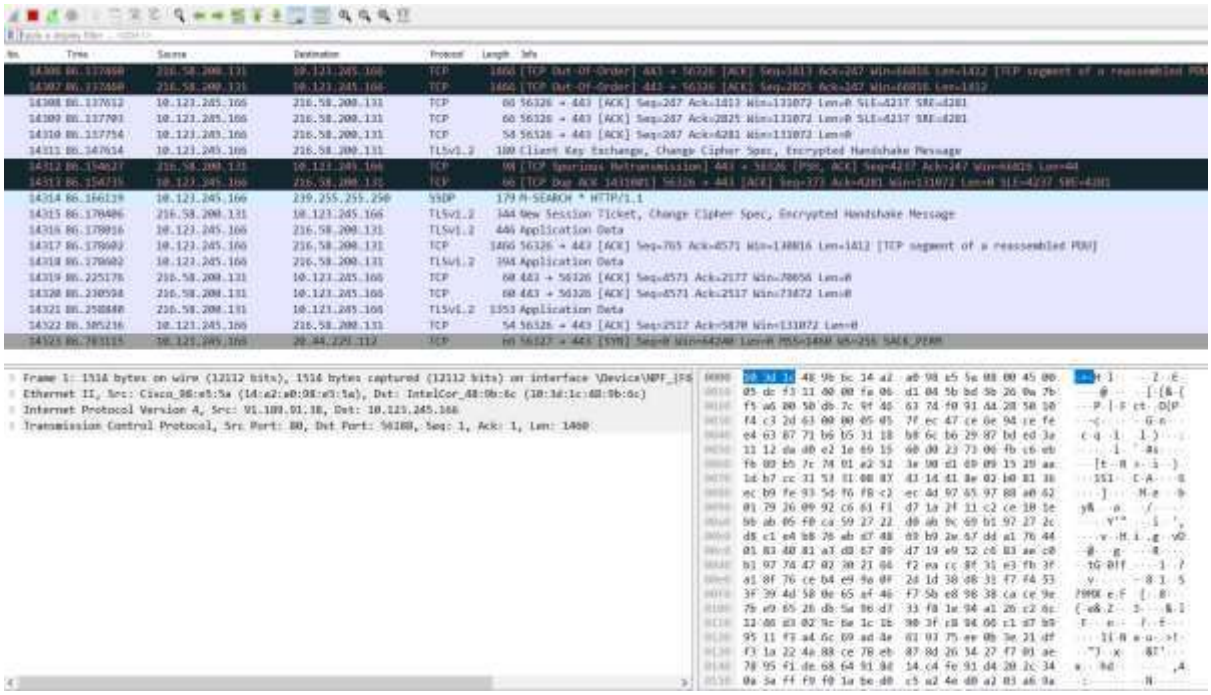
4.Nslookup

Nslookup -host

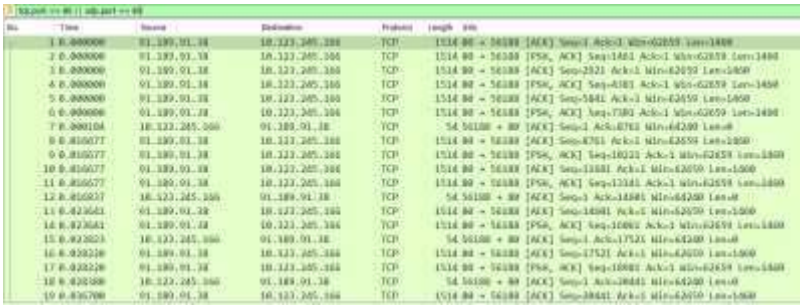
B.Ajay
21011102020

```
ajay@ubuntu:~$ nslookup www.google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   www.google.com
Address: 142.250.193.100
Name:   www.google.com
Address: 2404:6800:4007:81b::2004

ajay@ubuntu:~$ nslookup lms.snuchennai.edu.in
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   lms.snuchennai.edu.in
Address: 115.243.215.8
```

7.Wireshark



Listening on port 80



5.Ping

```
ajay@ubuntu:~$ ping www.google.com
PING www.google.com (142.250.182.132) 56(84) bytes of data.
64 bytes from maa05s22-in-f4.1e100.net (142.250.182.132): icmp_seq=1 ttl=128 time=9.30 ms
64 bytes from maa05s22-in-f4.1e100.net (142.250.182.132): icmp_seq=2 ttl=128 time=37.2 ms
64 bytes from maa05s22-in-f4.1e100.net (142.250.182.132): icmp_seq=3 ttl=128 time=14.7 ms
64 bytes from maa05s22-in-f4.1e100.net (142.250.182.132): icmp_seq=4 ttl=128 time=41.8 ms
64 bytes from maa05s22-in-f4.1e100.net (142.250.182.132): icmp_seq=5 ttl=128 time=18.5 ms
64 bytes from maa05s22-in-f4.1e100.net (142.250.182.132): icmp_seq=6 ttl=128 time=12.1 ms
^C
--- www.google.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5010ms
rtt min/avg/max/mdev = 9.302/22.274/41.774/12.552 ms
```

6. traceroute

```
ajay@ubuntu:~$ traceroute www.snuchennai.edu.in
traceroute to www.snuchennai.edu.in (182.75.25.245), 64 hops max
  1   192.168.152.2  0.258ms  0.203ms  0.216ms
  2   *  *  *
  3   *  *  *
  4   *  *
```

Result :

Thus the given commands were executed and their output Observed

B.Ajay
21011102020

Ex. No: 2	Download Webpage Using TCP Sockets
29/12/2022	

Aim:

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

Algorithm:

1.Create URL Object and pass URL as string to constructor .

2.Use .openStream() and use .readAllBytes() and to the byte array.

3.Using FileOutputStream create another object and write the byte array.

4.Close the stream.

5.Catch all Exceptions If any may occour.

Program:

```
package Exercise2;

import java.net.URL;

import java.io.*;
public class downloadpage {

	public static void main(String []args) {
		try {
			URL page=new URL("https://www.snuchennai.edu.in/");
			System.out.println("Port : "+page.getDefaultPort());
			System.out.println("Host : "  +page.getHost());
			System.out.println("Protocol: "+page.getProtocol());
			byte []arr=page.openStream().readAllBytes();
			String out=new String(arr);
			System.out.println(out);
			FileOutputStream out1=new
FileOutputStream("webdownload.html");
			out1.write(arr);
			out1.close();
		}catch(Exception e) {
			e.printStackTrace();
		}

	}
}
```

Sample Input:

Webdownload.html and snuchennai.edu.in

Sample Output:

B.Ajay
21011102020

```
var ekit_config = {"ajaxurl":"https:\/\/www.snuchennai.edu.in\/wp-admin\/admin-ajax.php","nonce":"964133699c"};
</script>
<script src="https://www.snuchennai.edu.in/wp-content/plugins/elementskit-lite/widgets/init/assets/js/elementor.js?ver=2.7.0" id="elements
<script src="https://www.snuchennai.edu.in/wp-content/plugins/elementor/assets/js/preloaded-modules.min.js?ver=3.7.7" id="preloaded-module
<script id="wp-util-js-extra">
var _wpUtilSettings = {"ajax":{"url":"\/wp-admin\/admin-ajax.php"}};
</script>
<script src="https://www.snuchennai.edu.in/wp-includes/js/wp-util.min.js?ver=5.8.4" id="wp-util-js"></script>
<script id="wpforms-elementor-js-extra">
var wpformsElementorVars = {"captcha_provider":"recaptcha","recaptcha_type":"v2"};
</script>
<script src="https://www.snuchennai.edu.in/wp-content/plugins/wpforms-lite/assets/js/integrations/elementor/frontend.min.js?ver=1.7.7.2" i
                        <script>
                        /(trident|msie)/i.test(navigator.userAgent)&&document.getElementById&&window.addEventListener&&window.addEventList
                        </script>
                                </body>
</html>
```

Result:

Thus Using the script in java the required webpage was downloaded.

B.Ajay
21011102020

Ex. No: 3.1	Sockets Programming Using Java
05/01/2023	

AIM:

To write a socket program for implementation of echo

Algorithm:

CLIENT :

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 a data to server.
4. Receive and print the same data from server.
5. Close the socket.
6. End the program

SERVER

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.

Code:

EchoClient.java

```
package Exercise3;
import java.net.*;
import java.io.*;
public class EchoClient {
public static void main(String [] args) {
	Socket s1;
	BufferedReader r1,r2;
	PrintStream p1;
	String op="";
	try {
		s1=new Socket(InetAddress.getLocalHost(),8080);
```

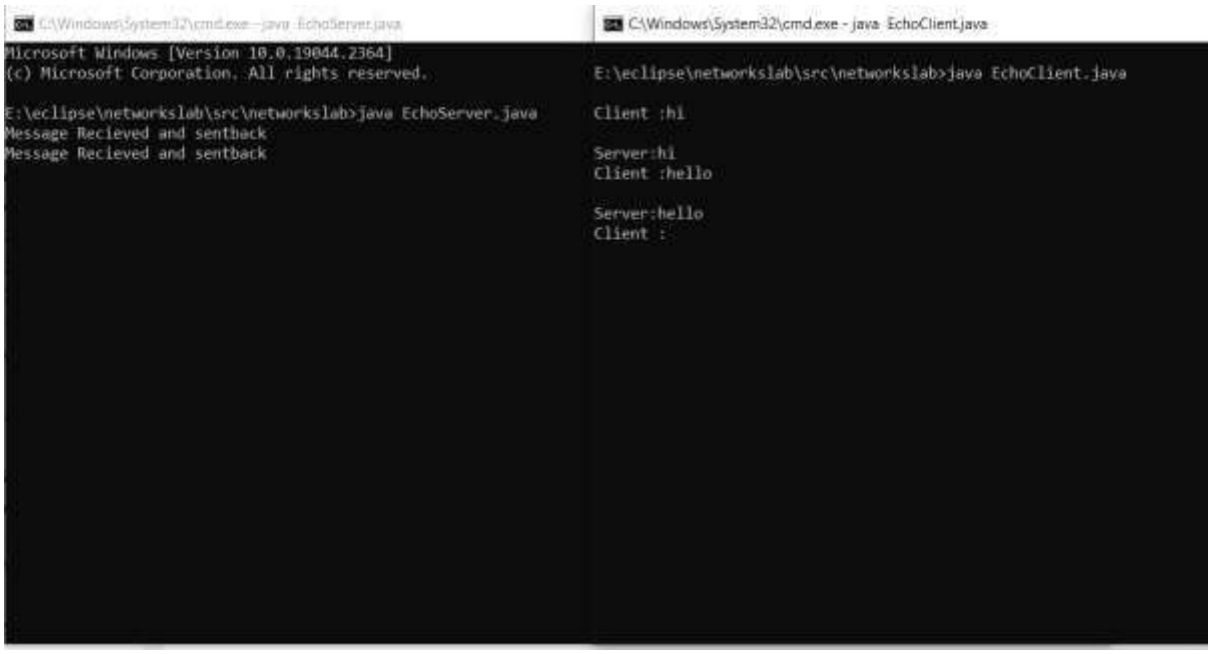
```
            r1=new BufferedReader(new InputStreamReader(System.in));

            r2=new BufferedReader(new InputStreamReader(s1.getInputStream()));

            p1=new PrintStream(s1.getOutputStream());
            while(!op.equals("end")) {
                    System.out.print("\nClient :");
                    op=r1.readLine();
                    p1.println(op);
                    System.out.print("\nServer:"+r2.readLine());

            }
        } catch (Exception e) {

            e.printStackTrace();
        }

}
}
```

EchoServer.java

```
package Exercise3;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;

public class EchoServer {
public static void main(String[] args) {
        String op1="";
        Socket s1;
        ServerSocket serv;
        BufferedReader r2;
        PrintStream p1;
        try {
                serv=new ServerSocket(8080);
                s1=serv.accept();
                r2=new BufferedReader(new InputStreamReader(s1.getInputStream()));
                p1=new PrintStream(s1.getOutputStream());
                while(!op1.equals("end")) {
                        op1=r2.readLine();
                        System.out.println("Message Recieved and sentback");
                        p1.println(op1);
                }
        }catch(Exception e) {
                e.printStackTrace();
        }
}
}
```

B.Ajay
21011102020

OUTPUT:



Result: Thus the echo server and client was implemented using java sockets library.

B.Ajay
21011102020

Ex. No: 3.2	Client Server Chat Application TCP Sockets
05/01/2023	

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.

Code:

TCPClient.java

```
package Exercise3;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.InetAddress;
import java.net.Socket;

public class TCPClient {
```

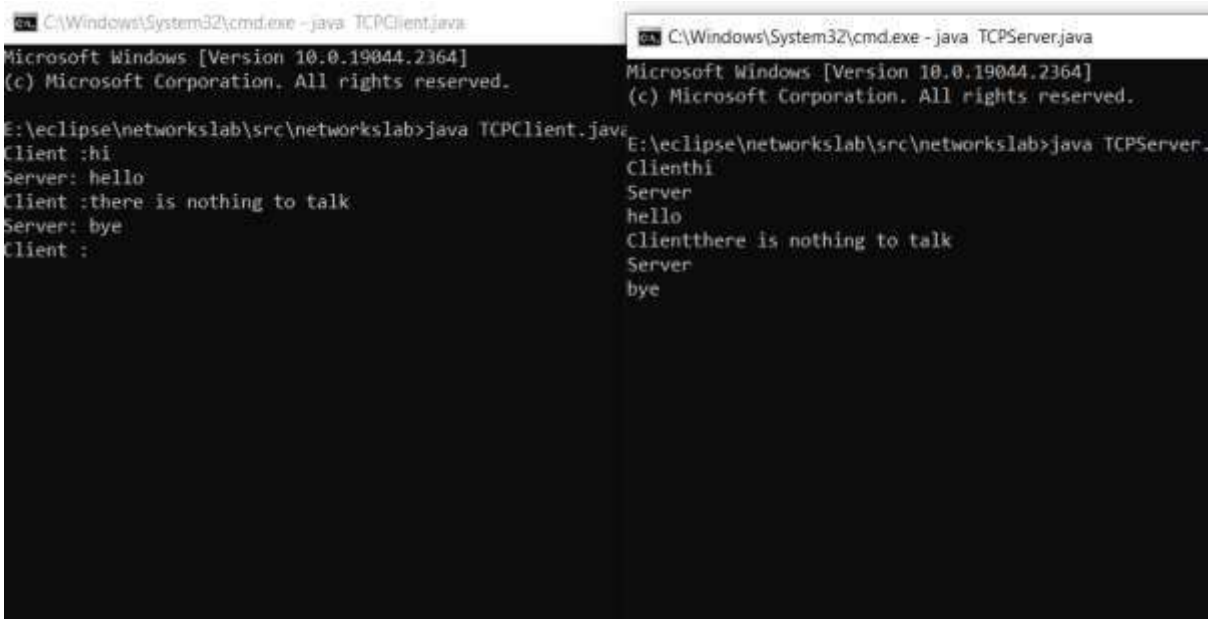
```
	public static void main(String [] args) {
		Socket s1;
		BufferedReader r1,r2;
		PrintStream p1;
		String op="";
		try {
			s1=new Socket(InetAddress.getLocalHost(),8080);
			r1=new BufferedReader(new InputStreamReader(System.in));
			r2=new BufferedReader(new
InputStreamReader(s1.getInputStream()));
			p1=new PrintStream(s1.getOutputStream());
			while(!op.equals("end")) {
				System.out.print("Client :");
				op=r1.readLine();
				p1.println(op);
				System.out.println("Server: "+r2.readLine());

			}
		}  catch (Exception e) {

			e.printStackTrace();
		}
	}
}
```

TCPServer.java

```
package Exercise3;


import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;

public class TCPServer {
	public static void main(String[]args) {
		String op1="";
		Socket s1;
		ServerSocket serv;
		BufferedReader r2,r1;
		PrintStream p1;
		try {
			serv=new ServerSocket(8080);
			s1=serv.accept();
			r1=new BufferedReader(new InputStreamReader(System.in));
			r2=new BufferedReader(new
InputStreamReader(s1.getInputStream()));
			p1=new PrintStream(s1.getOutputStream());
			while(!op1.equals("end")) {
				System.out.println("Client"+r2.readLine());
				System.out.println("Server");
				op1=r1.readLine();


				p1.println(op1);
			}
		}catch(Exception e) {
			e.printStackTrace();
```

```
            }
        }
}
```

OUTPUT :



Result:

Thus a simple chat server client setup was initiated using TCP Protocol using java sockets..

B.Ajay
21011102020

Ex. No: 3.3	File Transfer Using TCP Sockets
05/01/2023	

AIM:

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

Algorithm :

CLIENT :

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:

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

Code :

FileClient.java

```
package networkslab;
import java.io.*;
import java.net.*;
import java.net.InetAddress;
public class FileClient {
static DataInputStream k;
       static DataOutputStream k1;
public static void main(String [] args) {
              try(Socket s=new Socket("localhost",8080)){
```

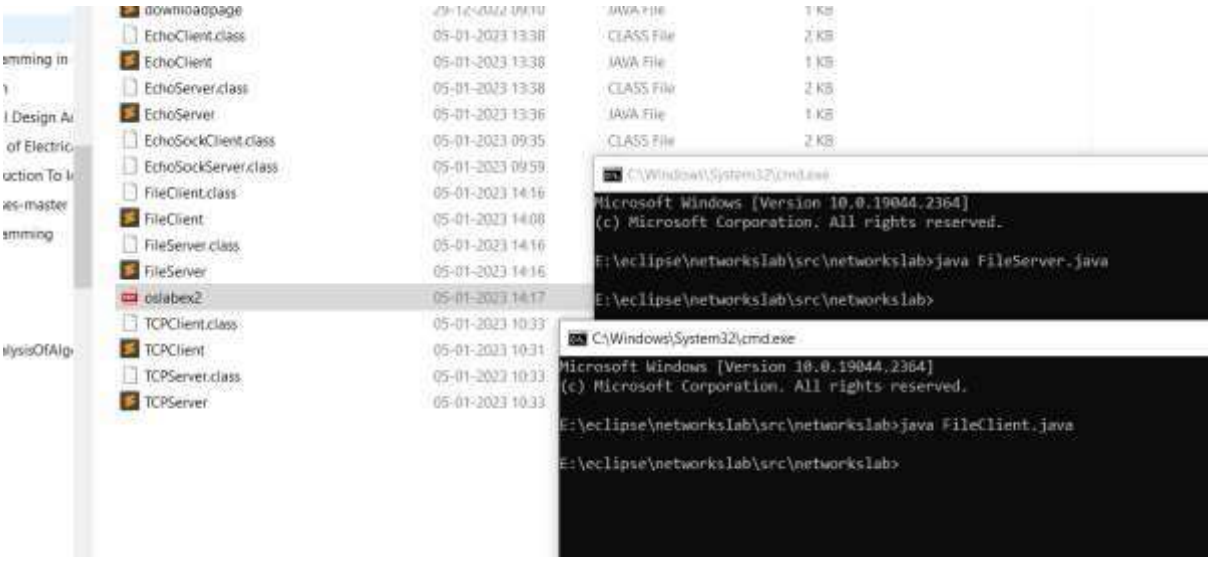
```
                k=new DataInputStream(s.getInputStream());
        k1=new DataOutputStream(s.getOutputStream());
        send("E:\\oslab\\oslabex1.pdf");
                k.close();                k1.close();
            }catch(Exception e) {
                e.printStackTrace();
            }
        }
        public static void send(String path) {
            int bytes=0;
            File f1=new File(path);
            try {
                FileInputStream is=new FileInputStream(f1);
                k1.writeLong(f1.length());
                byte[] bfr=new byte[4*1024];
                while((bytes=is.read(bfr))!=-1) {
                    k1.write(bfr,0,bytes);
                    k1.flush();
                }
                is.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
}
```

FileServer.java

```
package Exercise3;
import java.io.*;
import java.net.*;

public class FileServer {
        private static DataInputStream in=null;
        private static DataOutputStream out=null;


        public static void main(String args[]) {
            try(ServerSocket s=new ServerSocket(8080)){
                Socket client;
                client=s.accept();

                in=new DataInputStream(client.getInputStream());
                out=new DataOutputStream(client.getOutputStream());
                recv("oslabex2.pdf");
                in.close();
                //in.
                out.close();


            }catch(Exception e) {
                e.printStackTrace();;
            }
        }
        public static void recv(String fname) {
            int bits=0;

            try {
                FileOutputStream f=new FileOutputStream(fname);
                long inr=in.readLong();
                byte [] buf=new byte[4*1024];
```

```
                    while(inr>0 && (bits=in.read(buf, 0,
(int)Math.min(inr,buf.length )))!=-1) {
                            f.write(buf,0,bits);
                            inr-=bits;

                    }
                    f.close();
            } catch (IOException e) {

                    e.printStackTrace();
            }



        }
}
```

OUTPUT:



Result:

Thus the File was sent by the client and received by the fileserver in the specified location.

B.Ajay
21011102020

Ex. No: 4	Simulation Of DNS Using UDP Sockets
12/01/2023	

Aim: To Simulate DNS using Datagram Packets.

Algorithm:

Server:

1. Run Process .exec command nslookup to get dns ip address from cmd
2. Create a datagram socket and bind it to a port 3. Create a datagram packet to receive client request
4. Read the domain name from client to be resolved.
5. Lookup the host array for the domain name
6. If found then retrieve corresponding address
7. Create a datagram packet and send ip address to client
8. Repeat steps 3-7 to resolve further requests from clients
9. Close the server socket

Client:

1. Create a datagram socket
2. Get domain name from user
3. Create a datagram packet and send domain name to the server
4. Create a datagram packet to receive server message
5. Read server's response
6. If ip address is found then display it else display "Domain does not exist"
7. Close the client socket

Code:

UdpDnsServer.java:

```
package networkslab;
import java.net.*;
import java.util.Scanner;
 import java.io.*;
public class UdpDnsServer {
```

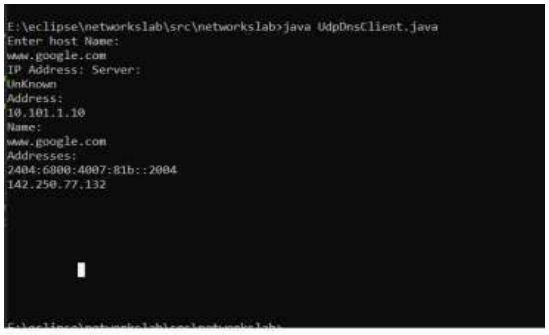
```
@SuppressWarnings("deprecation")
public static void main(String [] args) {
try {
             String command="nslookup ";
             String send1= "";
             String op="";
             while(true) {
                    DatagramSocket s=new DatagramSocket(8210);
      byte[] send=new byte[1024];                   byte [] recv=new byte[1024];
                    DatagramPacket rec=new DatagramPacket(recv,
recv.length,InetAddress.getLocalHost(),8210);
                    s.receive(rec);
                    String s12=new String (rec.getData());
                    command+=s12;
                    System.out.println(command);                  InetAddress a=
rec.getAddress();               int port=rec.getPort();
                    Process p=Runtime.getRuntime().exec(command.trim());
                    Scanner r=new Scanner(p.getInputStream());
                    //DatagramPacket send=new DatagramPacket(recv, port, a, port)
                    while(r.hasNext()) {
                           op+=r.next();
                           op+="\n";
                    }

                    System.out.println(op);
                    send1+=op;
                    send=send1.getBytes();
                    send1="";
                    DatagramPacket sendd=new DatagramPacket(send,
send.length,InetAddress.getLocalHost(),rec.getPort());
                    s.send(sendd);
                    s.close();
             }
             /*
             command+="www.google.com";
  Process p=Runtime.getRuntime().exec(command);   Scanner r=new
Scanner(p.getInputStream());
             while(r.hasNext()) {
                    op+=r.next();
                    op+="\n";

             }
             System.out.println(op);
             */
             } catch (IOException e) {              // TODO Auto-generated catch
block
             e.printStackTrace();
      }
}
}
```

UdpDnsClient.java

```
package networkslab;

import java.net.*;

import java.io.*;

public class UdpDnsClient {
```

```
public static void main(String args[]) {
   BufferedReader r=new BufferedReader(new InputStreamReader(System.in));
      try {
             DatagramSocket    s=new    DatagramSocket();
       byte[] send=new byte[1024];
byte [] recv=new byte[1024];
System.out.println("Enter host Name:");
String input=r.readLine();
send=input.getBytes();
DatagramPacket p=new DatagramPacket(send, send.length, InetAddress.getLocalHost(),
8210);


 DatagramPacket                                                          q=new
DatagramPacket(recv,recv.length,InetAddress.getLocalHost(),8210);
  s.send(p);
  s.receive(q);
  String ip=new String(q.getData());
  System.out.println("IP Address: "+ip);
    } catch (Exception e) {
           e.printStackTrace();
      }
}
}
```

Sample Input/Output:



Result:

Thus DNS was simulated using Datagram Packets in java

B.Ajay
21011102020

Ex. No: 5	Simulation Of ARP and RARP Protocols
19/01/2023	

Aim:

To simulate ARP and RARP Protocols in java.

Algorithm:

ARP:

Server:

1.Start the program

2. Accept the socket which is created by the client.

3. Server maintains the table in which IP and corresponding MAC addresses are stored.

4. Read the IP address which is send by the client.

5. Map the IP address with its MAC address and return the MAC address to client.

Client:

1. Start the program

2. Using socket connection is established between client and server.

3. Get the IP address to be converted into MAC address.

4. Send this IP address to server.

5. Server returns the MAC address to client

RARP:

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:

1.Start the program

2. using datagram sockets UDP function is established.

3.Get the MAC address to be converted into IP address.

4.Send this MAC address to server.

5.Server returns the IP address to client

Code:

ArpServer.java:

```
package networkslab;
import java.net.*;
import java.io.IOException;
import java.util.*;
 import java.io.*;

public class ArpServer {
	@SuppressWarnings("deprecation")
	public static void main(String args[]) {

		String ip="";

		String command="arp -a ";

		String out="";
		try {


			while(true) {
				DatagramSocket s=new DatagramSocket(7080);
		byte [] send=new byte[1024];
				byte [] recv=new byte[1024];
DatagramPacket p=new
DatagramPacket(recv,recv.length,InetAddress.getLocalHost(),7080);
				s.receive(p);
				ip=new String(p.getData());
				String c_out=command+ip;
				Process p1=Runtime.getRuntime().exec(c_out.trim());
				System.out.println(c_out);
				Scanner r=new Scanner(p1.getInputStream());

				while(r.hasNext()) {
					out+=r.next();
	out+="\n";
				}
				send=out.getBytes();
				out="";
```

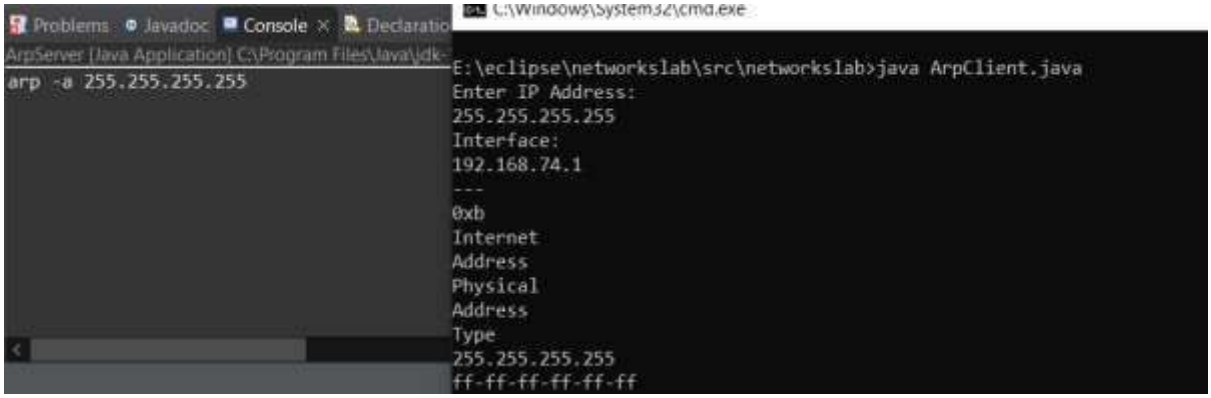
```
DatagramPacket send1=new
DatagramPacket(send,send.length,InetAddress.getLocalHost(),p.getPort());
                          s.send(send1);
                          s.close();
                    }
             } catch (IOException e) {
                    e.printStackTrace();
             }
       }
}
```

ArpClient.java:

```
package Exercise5;


import java.net.*;

import java.net.SocketException;

import java.io.*;

import java.util.*;

public class ArpClient {

public static void main(String[] args) {

       Scanner r=new Scanner(System.in);

       try {

              DatagramSocket s1=new DatagramSocket();

              byte [] send=new byte[1024];

              byte [] recv=new  byte[1024];

              System.out.println("Enter IP Address:");

              String ip=r.next();

              send=ip.getBytes();

              DatagramPacket                                          send1=new
DatagramPacket(send,send.length,InetAddress.getLocalHost(),7080);

              DatagramPacket                                          recv1=new
DatagramPacket(recv,recv.length,InetAddress.getLocalHost(),7080);

              s1.send(send1);

              s1.receive(recv1);

              String s=new String(recv1.getData());

              System.out.println(s);

              s1.close();


       } catch (Exception e) {

              // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        }


}


}
```

OUTPUT:



RarpServer.java:

```
package Exercise5;

import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class RarpServer {
        public static void main(String[]args) {
            String ip[]={"165.165.80.80","165.165.79.1"};
            String mac[]={"6A:08:AA:C2","8A:BC:E3:FA"};

            String send2="";
            int flag=0;
            try {


                while(true) {
                    DatagramSocket s=new DatagramSocket(8090);
                    byte[] send=new byte[1024];
                    byte [] recv=new byte[1024];
                    DatagramPacket                                recv1=new
DatagramPacket(recv,recv.length,InetAddress.getLocalHost(),8090);
                    s.receive(recv1);
                    String uin=new String(recv1.getData());
                    System.out.println(uin);
                    for(int i=0;i<ip.length;i++) {
```

```
                                if(mac[i].equalsIgnoreCase(uin.trim())) {
                                        send2=ip[i];
                                        flag=1;
                                        break;
                                }
                        }
                        if(flag==0) {
                                send2="Not Found";

                        }
                        send=send2.getBytes();
                        DatagramPacket                                  send1=new
DatagramPacket(send,send.length,InetAddress.getLocalHost(),recv1.getPort());
                        System.out.println(send2);
                        s.send(send1);
                        s.close();
                }
            }catch(Exception e) {
                e.printStackTrace();
            }
        }

}
```

RarpClient.java:

```
package Exercise5;
import java.io.*;
import java.util.*;
import java.net.*;

public class RarpClient {
        public static void main(String[]args)
        {
            System.out.println("Enter MAC ADDRESS: ");
            Scanner s=new Scanner(System.in);
            String ip=s.next();
            try {
                DatagramSocket s1=new DatagramSocket();
                byte []send=new byte[1024];
                byte [] recv=new byte[1024];
                send=ip.getBytes();
                DatagramPacket p1=new
DatagramPacket(send,send.length,InetAddress.getLocalHost(),8090);
                s1.send(p1);
                DatagramPacket p2=new DatagramPacket(recv, recv.length,
InetAddress.getLocalHost(), 8090);
                s1.receive(p2);
                String out=new String(p2.getData());
                System.out.println(out);
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
```

```
        }

}
```

Output:

```
E:\eclipse\networkslab\src\networkslab>javac RarpClient.java

E:\eclipse\networkslab\src\networkslab>java RarpClient.java
Enter MAC ADDRESS:
6A:08:AA:C2
165.165.80.80
```

Result:

Thus The ARP and RARP protocols were simulated in java.

B.Ajay
21011102020

Ex. No: 6	Simple Network Topology Creation using NS2
02/02/2023	

Aim:

To create simple topology using Network Simulator

Algorithm:

Step 1: Start network simulator OTCL editor.

Step 2: Create new simulator using set ns [new Simulator] syntax

Step 3: Create Trace route to Network Animator set nf [open out.nam w] $ns namtrace-all $nf

Step 4: Create procedure to trace all path

Step 5: Create full/simplex connection

Step 6: Connect TCP with null command/udp

Step7:visualise the same in nam

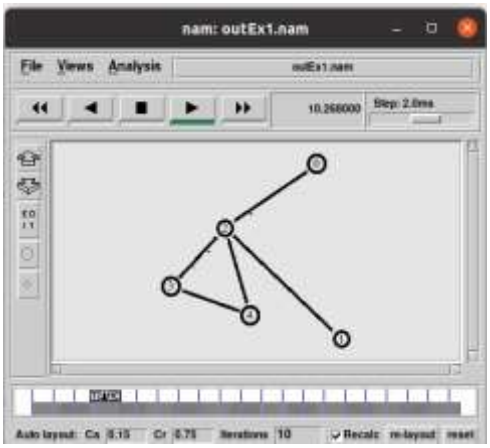
Code:

```
set ns [new Simulator]
set f [open outEx1.tr w]
set nf [open outEx1.nam w]
$ns namtrace-all $nf
$ns trace-all $f

proc finish {} {

     global ns nf f
     $ns flush-trace
     puts " completed"
     close $nf
     close $f
```

```
     exit 0
}



set n0 [$ns node]
      puts "n0: [$n0 id]"
set n1 [$ns node]
      puts "n1: [$n1 id]"
set n2 [$ns node]
      puts "n2: [$n2 id]"
set n3 [$ns node]
      puts "n3: [$n3 id]"
set n4 [$ns node]
      puts "n4: [$n4 id]"

$ns duplex-link $n0 $n2 100Mb 5ms DropTail
$ns duplex-link $n2 $n4 54Mb 10ms DropTail
$ns duplex-link $n1 $n2 100Mb 5ms DropTail
$ns duplex-link $n2 $n3 54Mb 10ms DropTail
$ns queue-limit $n2 $n3 40
$ns simplex-link $n3 $n4 10Mb 15ms DropTail
$ns simplex-link $n4 $n3 10Mb 15ms DropTail

set tcp0 [new Agent/TCP]
$ns attach-agent $n1 $tcp0

set udp1 [new Agent/UDP]
$udp1 set dst_addr_ Unicast
$udp1 set fid_ 1
$ns attach-agent $n0 $udp1

set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
```

```
set sink0 [new Agent/TCPSink]
$ns attach-agent $n4 $sink0


set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set rate_ 2Mb
$cbr0 set packetSize_ 1000
$cbr0 attach-agent $udp1
$ns connect $udp1 $null0
$udp1 set fid_ 0
$ns connect $tcp0 $null0
$tcp0 set fid_ 1


$ns at 0.05 "$ftp0 start"
$ns at 0.1 "$cbr0 start"
$ns at 60.0 "$ftp0 stop"
$ns at 60.5 "$cbr0 stop"
$ns at 61.0 "finish"
$ns run
```

Output:



B.Ajay
21011102020

My Understanding:

We write out the physical node and define the connection by duplex/simplex and we set up udp/tcp connections for the nodes whose types may vary .When we run ns x.tcl this generate a nam and a trace file which can be further processed in anyother programming language,here we use NAM to visualise the connections and packet delivery between nodes.

Result:

Thus a simple network topology was created and visualised using nam and network simulator

B.Ajay
21011102020

Ex. No: 6.1	Simulation Of Congestion Control Algorithms in NS2
16/02/2023	

Aim:

To Simulate Congestion Control using Network Simulator

Algorithm:

The size of the sender window is determined by the following two factors

1. Receiver window size
2. Congestion window size

1.Reciever Window Size:

Sender should not send data greater than receiver window size. • Otherwise, it leads to dropping the TCP segments which causes TCP Retransmission. • So, sender should always send data less than or equal to receiver window size. • Receiver dictates its window size to the sender through TCP Header

2.Congestion Window Size:

Sender should not send data greater than congestion window size. • Otherwise, it leads to dropping the TCP segments which causes TCP retransmission. • So, sender should always send data less than or equal to congestion window size. • Different variants of TCP use different approaches to calculate the size of congestion window. • Congestion window is known only to the sender and is not sent over the links.

Code:

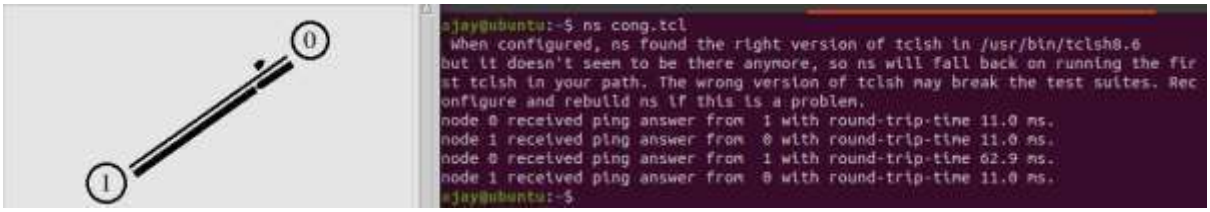
```
    set ns [new Simulator]
set f [open congestion.tr w]
$ns trace-all $f
set nf [open congestion.nam w]
$ns namtrace-all $nf
proc finish {} {
 exec nam congestion.nam &
exit 0
```

```
}
set n0 [$ns node]
set n1 [$ns node]
$ns   duplex-link
$n1  $n0  1Mb  5ms
DropTail


set tcp1 [new Agent/TCP/Reno]
$ns attach-agent $n0 $tcp1
$tcp1 set fid_ 1



set sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $sink1

$ns connect $tcp1 $sink1
set       ftp1       [new
Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP

set p0 [new Agent/Ping]
$ns   attach-agent   $n0
$p0    set    p1    [new
Agent/Ping]
$ns attach-agent $n1 $p1
$ns connect $p0 $p1
Agent/Ping instproc recv {from rtt} { $self
instvar  node_  puts  "node  [$node_  id]
received ping answer from \
$from with round-trip-time $rtt ms."
}
$ns at 0.5 "$p0 send"
$ns at 0.8 "$p1 send"
```

B.Ajay
21011102020

```
$ns at 1.0   "$ftp1 start"
$ns at 70.0 "$ftp1 stop"
$ns at 70.1 "$p0 send"
$ns at 70.2 "$p1 send"
$ns at 80.0 "finish"



$ns run
```

Output:



Result :

Thus The TCP Congestion was simulated using ns2

B.Ajay
21011102020

Ex. No: 7	Simulation Of Error Correction Code
23/02/2023	

Aim:

To write a Java program to simulate Error Correction Code (CRC) in Java

Algorithm:

1.Define a function that does bit to bit Xor operation .Concatenate the bits into one Single String

2.Define another function to Divide the Polynomial with the CRC Polynomial and Obtain the Remainder using repeated Division .Ignore the leading zeros while division.

3.Take a Transmission String and CRC Polynomial. Divide and obtain CRC Remainder

4.Pad the CRC Remainder to the Transmission String and Again Divide Using CRC Polynomial If no Error the is Transmitted properly Function returns True

5.Introduce another Error to Transmission String and Divide Using CRC Polynomial .This time the isTransmittedProperly Function should return false

Code:

CRC.java:

```
package Exercise7;


public class CRC {

public  String Xor(String s1,String s2) {

             String res="";

             for(int i=0;i<s2.length();i++) {

                    if(s1.charAt(i)==s2.charAt(i)) {

                           res+='0';

                    }

                    else {

                           res+='1';

                    }

             }

             return res;

          }

public String  padBit(String s1,String s2) {

             int d2=s2.length();
```

```
            for (int i=0;i<d2-1;i++) {
                  s1+='0';
            }
            return s1;
        }

public String Divide(String s1,String s2) {
            int d1=s1.length();
            String tmp="";
            int track=0;
            String tmp2="";
            tmp=Xor(s1.substring(0, s2.length()),s2);
            track+=tmp.length();
            while(track<d1) {
                        if(tmp.charAt(0)=='0') {
                              tmp=tmp.substring(1);
                              tmp+=String.valueOf(s1.charAt(track++));
                        }

                  tmp2=tmp;
                  tmp="";
                  tmp=Xor(tmp2,s2);
            }

            return tmp2;

        }
public boolean isTransmittedProperly(String originalBits,String CRCRemainder,String
divisor) {
            //String k=new CRC().Divide(padBit(originalBits,divisor), divisor);
            String s2=originalBits+CRCRemainder.substring(1);
            //System.out.println(new CRC().Divide(s2, divisor));
            if(new CRC().Divide(s2, divisor).equals("0000")) {
```

```
                return true;
            }
            return false;
        }
public static void main(String [] args) {
            String s1="10111011";
            System.out.println("No Error While Transmission: ");
            System.out.println(" Original BitArrangement:"+s1);

            String chk=s1;
            String s2="1001";
            System.out.println(" CRC Polynomial:"+s2);
            s1=new CRC().padBit(s1,s2);
            String s3=new CRC().Divide(s1, s2);
            //System.out.println(s3);
            chk+=s3.substring(1);
            //System.out.println(new CRC().Divide(chk, s2));
            /*Check for proper Transmission*/

            System.out.println("CRC                Match                :->"+new
CRC().isTransmittedProperly("10111011","0110","1001"));
            /*Flip a bit in the original bit to simulate Error*/
            System.out.println("\n\nError  Introduced  into  Bits:\n  ErrenousBit
Arrangement:10111111\n CRC Polynomial:1001\n");

            System.out.println("CRC                Match                :->"+new
CRC().isTransmittedProperly("10111111","0110","1001"));
            //System.out.println(f.substring(4,5));
            //System.out.println(new CRC().Xor("1100","1101"));
      //System.out.println(new CRC().Divide("10010000", "1101"));
            //System.out.println(new CRC().Divide("10111011", "1001"));
            //System.out.println(new CRC().Divide("1110", "1101"));
        }
}
```

B.Ajay
21011102020

Output:

```
No Error While Transmission:
 Original BitArrangement:10111011
 CRC Polynomial:1001
CRC Match :->true


Error Introduced into Bits:
 ErrenousBit Arrangement:10111111
 CRC Polynomial:1001

CRC Match :->false
```

Result:

Thus the CRC Error Correction/Detection Algorithm was implemented in Java.

B.Ajay
21011102020

Ex. No: 8	Study of TCP/UDP Performance using simulation Tool
02/03/2023	

Aim:

To analyze the performance of tcp/udp networks.

Algorithm:

1. Declare one node with udp and cbr
2. Declare another node with tcp and cbr
3. Let node 3 be tcpsink and node 4 be null
4. We then send udp traffic from 1-4
5. We send tcp traffic from 2-3 and analyse the performance and bandwith and packet loss in case of udp

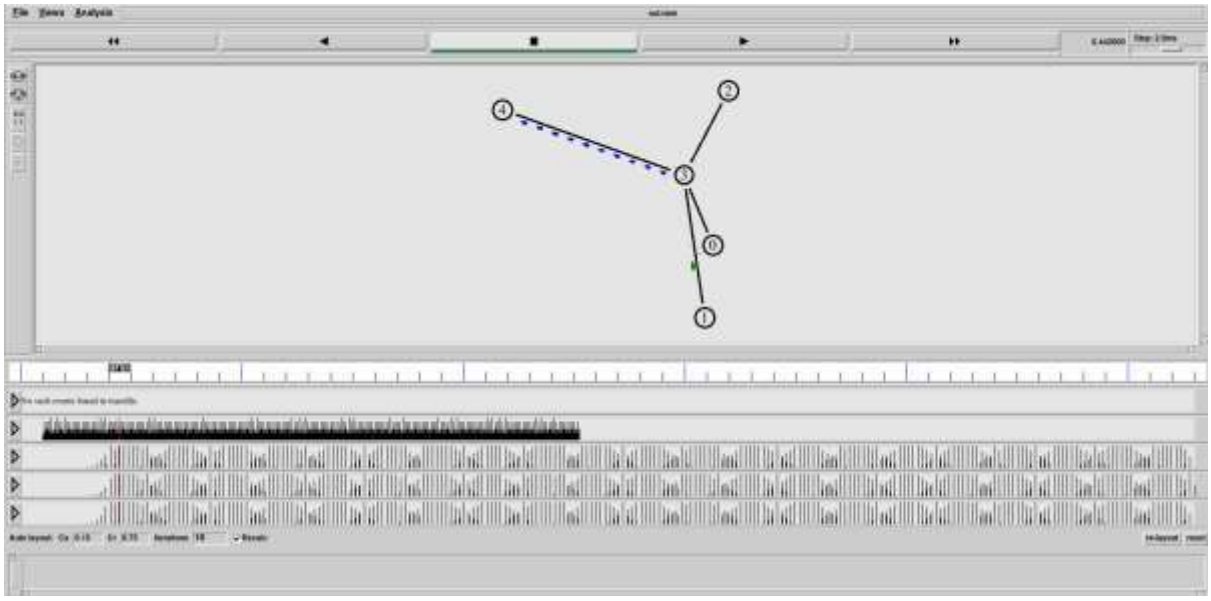
Code:

```
     set ns [new Simulator]
set file_trace [open out.tr w]
$ns trace-all $file_trace

set nf [open out.nam w]
$ns namtrace-all $nf

set n1 [$ns node]
set n2 [$ns node]

set n3 [$ns node]
set mid [$ns node]

set n4 [$ns node]

proc finish {} {
    global ns nf file_trace
        $ns flush-trace
           close $nf
       close $file_trace
             exit 0
```

```
}
$ns  duplex-link $n1 $mid 10Mb 0ms DropTail
$ns  duplex-link $n2 $mid 1000Mb 0.1ms DropTail

$ns  duplex-link $n3 $mid 103Mb 10ms DropTail

$ns  duplex-link $n4 $mid 10Mb 10ms DropTail

set udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.001
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent  $n4 $null0
$ns connect $udp0 $null0

set tcp0 [new Agent/TCP]
$ns attach-agent $n2 $tcp0
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.001
$cbr1 attach-agent $tcp0
set tcpsink0 [new Agent/TCPSink]
$ns attach-agent $n3 $tcpsink0
$ns connect $tcp0 $tcpsink0

$tcp0 set fid_ 1
$udp0 set fid_ 2
 $ns color 1 Green
 $ns color 2 Blue
 $ns at 0.1 "$cbr0 start"
 $ns at 2.5 "$cbr0 stop"
```

```
$ns at 0.3 "$cbr1 start"
$ns at 5.1 "$cbr1 stop"
$ns at 5.3 "finish"
$ns run
```

Output:



Result:

Thus the performance of udp/tcp networks were analyzed using a network simulator.

B.Ajay
21011102020

Ex. No: 9	Simulation Of Distance Vector/Link State Routing Algorithm
16/03/2023	

Aim:

To Implement Distance Vector And Link State Routing Algorithm in Java

Algorithm:

1.***Distance Vector Routing*** :

1. Accept all nodes in the form of adjacency matrix
2. After accepting the nodes we run Bellman Ford Algorithm
3. Here we discover all nodes and update vector according to cost if less than existing
4. Mark unreachable nodes as inf
5. Traverse in lexicographic order
6. Repeat step 3
7. Change source nodes and run through same algorithm and print distance vector
8. Indices are the nodes

2.***Link State Routing Algorithm***:

1. Accept All Inputs as adjacency Matrix
2. After accepting the nodes we run Dijkstra Algorithm

3, Here also we discover the nodes and keep track of previous nodes in the form od ArrayList in Java [Node no,isDiscovered,cost,Node Previous]

4. We run the loop until all IsDiscovered Subarray is 1

5.We cut the array into 4 part chunks as Subarray in java for easier use and manipulation of list entries

6.Inside the loop we check for minimum cost nodes and jump to that node and discover others while we keep track of existing node cost and update the subarray of neighbouring nodes iff the cost + adj[i][j]<existing cost.

7.We write another function to traceback the path to 0 or source node by jumping to the subarray.get[3]->pre node and print the path + cost

B.Ajay
21011102020

Code:

Distance VectorRouting.java

```
package Exercise9;

import java.util.ArrayList;
import java.util.List;

class BellmanFord{
        int [][] adjmat;
        int source,len;
        int inf=45558;

       BellmanFord(int[][]adjmat,int arrlen,int source){
              this.adjmat=adjmat;
              this.source=source;
              len=arrlen;
       }
       List <Integer>vector=new ArrayList <Integer>(len);


        void initialize() {
               for(int i=0;i<len;i++) {

                            vector.add(adjmat[source][i]);
                     }


       }
        void iterate() {
               initialize();
               int cur;
               int j=(source+1)%len;

                int [][] arr1=new int[len-1][len];
                int row=0;
                     while(j!=source) {
                            cur=vector.get(j);
                            for(int k=0;k<len;k++) {
                                   arr1[row][k]=adjmat[j][k];

                            }
                            row++;
                            if(row==len-1) {break;}

                            j++;
                            j=j%4;
                     }

                            //System.out.println();


                     j=(source+1)%len;
```

```
                for(int i=0;i<len-1;i++) {
                        cur=vector.get(j);
                        for(int k=0;k<len;k++) {
                                if(Math.abs(cur+arr1[i][k])<vector.get(k)) {

                                        vector.set(k, (cur+arr1[i][k]));
                                }
                        }


                        j++;
                        j=j%len;
                }


        }
        void printVector() {
                for(int i=0;i<len;i++) {
                        System.out.print(vector.get(i)+"\t");
                }
                System.out.println();
        }
        void PrintRouterVectorTable(int[][]mat,int arrlen,int src) {
                for(int i=0;i<len;i++) {
                        System.out.println(i+" Routing Table: ");
                        this.source=i;
                        vector=new ArrayList <Integer>(len);
                        iterate();
                        printVector();
                        System.out.println("------------------");
                }
        }

}
public class distanceVectorRouting {


        public static void main(String []args) {
                int inf=45556;
                int [][] adjacencyMatrix= {{0,2,inf,1},
                                                        {2,0,3,7},
                                                        {inf,3,0,11},
                                                        {1,7,11,0}};
                int [][]adjmat=     { { inf, inf, 1, 2, inf, inf, inf },
                                { inf, inf, 2, inf, inf, 3, inf },
                                { 1, 2, inf, 1, 3, inf, inf },
                        { 2, inf, 1, inf, inf, inf, 1 },
                        { inf, inf, 3, inf, inf, 2, inf },
                        { inf, 3, inf, inf, 2, inf, 1 },
                        { inf, inf, inf, 1, inf, 1, inf } };


                BellmanFord f= new BellmanFord(adjacencyMatrix,4,0);
                f.PrintRouterVectorTable(adjacencyMatrix,4,0);
                //f.iterate();
                //f.printVector();

        }
```

```
}
```

Output:

```
0 Routing Table:
0       2       5       1
------------------
1 Routing Table:
2       0       3       3
------------------
2 Routing Table:
5       3       0       10
------------------
3 Routing Table:
1       3       6       0
------------------
```

LinkStateRouting.java:

```
package Exercise9;
import java.util.*;
class dijkstra{
	int mat[][];
	int nodes,node_no=0,n_nodes;
		// Node no , isVisited, Distance , prev
	dijkstra(int n_nodes,int[][] mat){
		this.mat=mat;
		this.n_nodes=n_nodes;
	}
	ArrayList <Integer> sl=new ArrayList<Integer>(n_nodes*n_nodes);
	int k=0;
	void set() {
		sl.add(k++,node_no++ );
		sl.add(k++, 0);
		sl.add(k++, 0);
		sl.add(k++, null);
		for(int i=1;i<n_nodes;i++) {
			sl.add(k++,node_no++ );
			sl.add(k++, 0);
			sl.add(k++, Integer.MAX_VALUE);
			sl.add(k++, null);
		}

	}
	void print() {
		for(int i=0;i<sl.size();i+=4) {
			//System.out.println(i);
			System.out.println(sl.subList(i, i+4));
		}
	}

	int findmin(ArrayList <Integer> k) {
		int min=k.subList(4, 8).get(2),idx=k.subList(4, 8).get(0);

		for(int i=4;i<mat.length*4;i+=4) {
			//System.out.println(k.subList(i,i+4).get(2)+"====");

			if(k.subList(i,i+4).get(2)<=min &&
(k.subList(i,i+4).get(1))==0) {
```

```
                        min=k.subList(i,i+4).get(2);
                        idx=k.subList(i, i+4).get(0);
                }
        }
        //System.out.println(min+"  "+idx);
        return idx;
    }
    void run() {
        int cur;
        int iter=0;

        int k=0,m,i,cost=0;
        for(int j=0;j<mat.length;j++) {
            i=j*4;
            if(mat[k][j]>0) {
                if(sl.subList(i,i+4).get(2)==Integer.MAX_VALUE) {
                    sl.subList(i, i+4).set(2, mat[k][j]);
                    sl.subList(i, i+4).set(3, 0);
                }
            }
        }
        sl.subList(0, 4).set(1, 1);
        int listidx,precost,pre,collst,tmp;
        while(!isVisited()) {
            i=findmin(sl);
            //print();
            listidx=i*4;
            precost=sl.subList(listidx,listidx+4).get(3);
            cost=sl.subList(precost*4,(precost*4)+4).get(2);
            //cost=sl.subList(listidx,listidx+4).get();
            for(int c=0;c<mat.length;c++) {
                if(mat[i][c]>0) {
                    collst=c*4;
                     if
(cost+mat[i][c]<sl.subList(collst,collst+4).get(2)) {
                         tmp=sl.subList(collst,collst+4).get(2);
                        sl.subList(collst, collst+4).set(2,
cost+mat[i][c]);
                        sl.subList(collst,collst+4).set(3, i);
                    }
                }
            }
            sl.subList(listidx, listidx+4).set(1, 1);
        }


    }
    boolean isVisited() {

        for(int i=0;i<mat.length;i+=4) {
            if(sl.subList(i, i+4).get(1)==0) {
                return false;
            }
        }
        return true;
    }
    void RouteConfigPrint(int targetNode) {
        int listidx=targetNode;
```

```
            listidx*=4;
            int cost=0;
            System.out.print("0");
            while(sl.subList(listidx, listidx+4).get(0)!=0) {
                  cost+=sl.subList(listidx, listidx+4).get(2);
                  System.out.print("-->"+sl.subList(listidx, listidx+4).get(0));
                  listidx=sl.subList(listidx, listidx+4).get(3);
                  listidx*=4;
            }
            System.out.print("=" +cost);
            cost=0;
      }
}
public class linkStateRouting {
      public static void main(String []  args) {


            int [][]adjmat=    { { 0, 0, 1, 2, 0, 0, 0 },
                        { 0, 0, 2, 0, 0, 3, 0 },
                        { 1, 2, 0, 1, 3, 0, 0 },
                  { 2, 0, 1, 0, 0, 0, 1 },
                  { 0, 0, 3, 0, 0, 2, 0 },
                  { 0, 3, 0, 0, 2, 0, 1 },
                  { 0, 0, 0, 1, 0, 1, 0 } };

            dijkstra s=new dijkstra(7,adjmat);
            s.set();
            //s.findmin(s.sl);
            s.run();

            s.print();
            for(int i=1;i<7;i++) {
                  System.out.println();
                  s.RouteConfigPrint(i);
      }

      }
}
```

Output:

```
[0, 1, 0, null]
[1, 1, 2, 2]
[2, 1, 1, 0]
[3, 1, 1, 2]
[4, 0, 3, 2]
[5, 1, 2, 6]
[6, 1, 2, 3]

0-->1-->2=3
0-->2=1
0-->3-->2=2
0-->4-->2=4
0-->5-->6-->3-->2=6
0-->6-->3-->2=4
```

Result:

Thus Distance Vector And Link State Routing Algorithms were studied and implemented in Java

B.Ajay
21011102020

Ex. No: 10	Study of Performance of Routing Protocols Using ns2
23/03/2023	

Aim:

To Simulate Routing Protocols like Distance Vector and Link State using ns2

Algorithm :

1.Create 4 nodes and make the necessary out.nam,out.tr files

2. After creating 4 nodes set node 2 as tcp node and node 3 as sink node

3. Set FTP to tcp agent using attach agent

4. set rtproto to Ls for link state / DV for Distance Vector

5.Analyse the performance by using the tools in the network animator

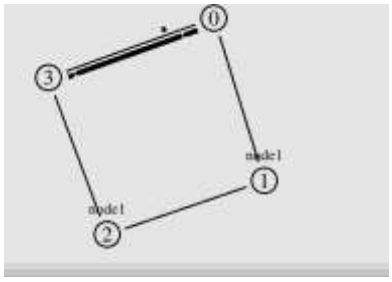
Code:

```
set ns [new Simulator]
$ns rtproto LS

set node1 [$ns node]
set node2 [$ns node]
set node3 [$ns node]
set node4 [$ns node]
set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {

global ns nf
$ns flush-trace
close $nf
exec nam out.nam &
exit 0

}
$node1 label "node1"
$node2 label "node1"
$node3 label "node1"
$ns duplex-link $node1 $node2 1.0Mb 10ms DropTail
$ns duplex-link $node2 $node3 1.0Mb 10ms DropTail
$ns duplex-link $node3 $node4 1.0Mb 10ms DropTail
$ns duplex-link $node4 $node1 1.0Mb 10ms DropTail
set tcp0 [new Agent/TCP]
$ns attach-agent $node1 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $node4 $sink0
$ns connect $tcp0 $sink0

set traffic [new Application/FTP]
$traffic attach-agent $tcp0
$ns at 0.5 "$traffic start"
$ns rtmodel-at 1.0 down $node2 $node3
$ns rtmodel-at 2.0 up $node2 $node3
ns at 3.0 "$traffic start"
```

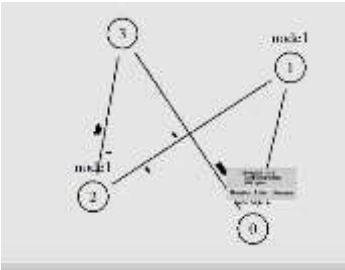
```
ns at 4.0 "$traffic stop"
ns at 5.0 "finish"
$ns run
```

Output:

Distance Vector



Link State Routing



Result:

Thus the Two routing protocols Link State and Distance Vector Routing were simulated and studied using ns2 and network animator