EX.NO: 1 **Learn to use commands like tcpdump, netstat, ifconfig, nslookup and**
DATE: 17/8/2024 **traceroute. Capture ping and trace route PDUs using a network protocol analyzer and examine.**

AIM:

To Learn to use commands like tcpdump, netstat, ifconfig, nslookup ,traceroute and ping.

tcpdump:

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
^C
0 packets captured
0 packets received by filter
0 packets dropped by kernel
```

>Which tcpdump

```
/usr/sbin/tcpdump
```

>tcpdump –D

```
1.enp0s3 [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

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 by
tes
17:11:54.004839 IP rdbms53-VirtualBox.60935 > ads1.licet.edu.domain: 63313+ [1a
u] A? connectivity-check.ubuntu.com. (58)
17:11:54.005645 IP localhost.54778 > localhost.domain: 49806+ [1au] PTR? 16.5.1
68.192.in-addr.arpa. (54)
17:11:54.005974 IP rdbms53-VirtualBox.42686 > ads1.licet.edu.domain: 15475+ [1a
u] PTR? 16.5.168.192.in-addr.arpa. (54)
17:11:54.006335 IP ads1.licet.edu.domain > rdbms53-VirtualBox.60935: 63313 12/0
/1 A 185.125.190.17, A 185.125.190.98, A 185.125.190.97, A 185.125.190.96, A 18
5.125.190.18, A 91.189.91.97, A 91.189.91.96, A 91.189.91.48, A 185.125.190.48,
 A 91.189.91.98, A 185.125.190.49, A 91.189.91.49 (250)
17:11:54.018185 IP rdbms53-VirtualBox.60692 > blackcat.canonical.com.http: Flag
s [S], seq 1875144196, win 64240, options [mss 1460,sackOK,TS val 1723597366 ec
r 0,nop,wscale 7], length 0

18 packets captured
36 packets received by filter
14 packets dropped by kernel
```

>netstat –t

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
```

>netstat –r

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         _gateway        0.0.0.0         UG        0 0          0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
link-local      0.0.0.0         255.255.0.0     U         0 0          0 enp0s3
```

netstat:

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 rdbms53-VirtualB:bootpc _gateway:bootps         ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  3      [ ]         DGRAM      CONNECTED     14920    /run/systemd/notify
unix  2      [ ]         DGRAM                    34600    /run/user/1000/syste
md/notify
unix  2      [ ]         DGRAM                    14935    /run/systemd/journal
/syslog
unix  17     [ ]         DGRAM      CONNECTED     14945    /run/systemd/journal
/dev-log
unix  8      [ ]         DGRAM      CONNECTED     14949    /run/systemd/journal
/socket
unix  3      [ ]         STREAM     CONNECTED     42263
unix  3      [ ]         STREAM     CONNECTED     37711    /run/systemd/journal
/stdout
unix  3      [ ]         DGRAM      CONNECTED     14922
unix  3      [ ]         STREAM     CONNECTED     38353
unix  3      [ ]         STREAM     CONNECTED     36134    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     40053
unix  3      [ ]         STREAM     CONNECTED     20568    /run/systemd/journal
/stdout
unix  3      [ ]         STREAM     CONNECTED     21322
unix  3      [ ]         STREAM     CONNECTED     32397    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     27473
unix  3      [ ]         STREAM     CONNECTED     30462    /run/systemd/journal
/stdout
```

>netstat –s

```
Ip:
    Forwarding: 2
    2428 total packets received
    2 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    2424 incoming packets delivered
    2013 requests sent out
    20 outgoing packets dropped
Icmp:
    40 ICMP messages received
    0 input ICMP message failed
    ICMP input histogram:
        destination unreachable: 40
    41 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 41
IcmpMsg:
        InType3: 40
        OutType3: 41
Tcp:
    28 active connection openings
    0 passive connection openings
    4 failed connection attempts
    4 connection resets received
    0 connections established
```

Ifconfig:

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::e14e:9693:b1f1:73e7  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:1a:8e:21  txqueuelen 1000  (Ethernet)
        RX packets 4456  bytes 6149743 (6.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1775  bytes 140851 (140.8 KB)
        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 328  bytes 29350 (29.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 328  bytes 29350 (29.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Nslookup:

```
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   www.licet.ac.in
Address: 218.248.16.177
```

>host –t ns google.com

```
google.com name server ns2.google.com.
google.com name server ns3.google.com.
google.com name server ns4.google.com.
google.com name server ns1.google.com.
```

>nslookup www.google.com ns1.google.com

```
Server:         ns1.google.com
Address:        216.239.32.10#53

Name:   www.google.com
Address: 142.250.183.228
Name:   www.google.com
Address: 2404:6800:4007:81e::2004
```

Traceroute:

```
Usage:
  traceroute [ -46dFITnreAUDV ] [ -f first_ttl ] [ -g gate,... ] [ -i device ]
[ -m max_ttl ] [ -N squeries ] [ -p port ] [ -t tos ] [ -l flow_label ] [ -w MA
X,HERE,NEAR ] [ -q nqueries ] [ -s src_addr ] [ -z sendwait ] [ --fwmark=num ]
host [ packetlen ]
Options:
  -4                          Use IPv4
  -6                          Use IPv6
  -d  --debug                 Enable socket level debugging
  -F  --dont-fragment         Do not fragment packets
  -f first_ttl  --first=first_ttl
                              Start from the first_ttl hop (instead from 1)
  -g gate,...  --gateway=gate,...
                              Route packets through the specified gateway
                              (maximum 8 for IPv4 and 127 for IPv6)
  -I  --icmp                  Use ICMP ECHO for tracerouting
  -T  --tcp                   Use TCP SYN for tracerouting (default port is 80)
  -i device  --interface=device
                              Specify a network interface to operate with
  -m max_ttl  --max-hops=max_ttl
                              Set the max number of hops (max TTL to be
                              reached). Default is 30
```

>traceroute 192.168.2.17

```
traceroute to 192.168.2.17 (192.168.2.17), 30 hops max, 60 byte packets
 1  _gateway (10.0.2.2)  5.650 ms  5.204 ms  4.848 ms^C
```

Ping:

```
PING 192.168.2.17 (192.168.2.17) 56(84) bytes of data.
^C
--- 192.168.2.17 ping statistics ---
13 packets transmitted, 0 received, 100% packet loss, time 18058ms
```

RESULT:

Thus the various network commands like tepdump, netstat, ifconfig, nslookup, traceroute and ping are executed successfully.

IPROGRAM:

```
import java.io.*;
import java.net.*;
public class SocketPro{
        public static void main(String args[]){
                String hostname = "www.licet.ac.in";
                int portnumber=80;
                try{
                        Socket s = new Socket(hostname,portnumber);
                        PrintWriter out = new PrintWriter(s.getOutputStream(),true);
                        BufferedReader in = new BufferedReader( new
                        InputStreamReader(s.getInputStream()));
                        out.println("GET/HTTP/1.1");
                        out.println("Host: " + hostname);
                        out.println("Connection: Close");
                        String str;
                        while((str=in.readLine())!=null){
                                System.out.println(str);
                        }
                }catch(UnknownHostException e) {
                    System.out.println("Unknown host: "+hostname);
                    System.exit(1);
                } catch(IOException e) {
                    System.out.println("No I/O");
                    System.exit(1);
                }
        }
}
```

OUTPUT:

```
C:\Users\RDBMS-52>java SocketPro
HTTP/1.1 400 Bad Request
Date: Thu, 22 Aug 2024 10:23:20 GMT
Server: Apache/2.4.52 (Ubuntu)
Content-Length: 303
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.52 (Ubuntu) Server at licet.ac.in Port 80</address>
</body></html>
```

RESULT:

PROGRAM:

EchoServer.java

```
import java.io.*;
import java.net.*;
public class EchoServer
{
public EchoServer(int portnum)
{
try
{
server = new ServerSocket(portnum);
}
catch (Exception err)
{
System.out.println(err);
}
}
public void serve()
{
try
{
while (true)
{
Socket client = server.accept();
BufferedReader r = new BufferedReader(new InputStreamReader(client.getInputStream()));
PrintWriter out = new PrintWriter(client.getOutputStream(),  true);
out.println("Welcome to the Java EchoServer. Type 'bye' to close.");
String line;
do
{
line = r.readLine();
if ( line != null )
out.println("Got: "+ line);
System.out.println (line);
}
while ( !line.trim().equals("bye") );
client.close();
}
}
catch (Exception err)
{
System.err.println(err);
}
}
public static void main(String[] args)
{
EchoServer s = new EchoServer(9999);
s.serve();
```

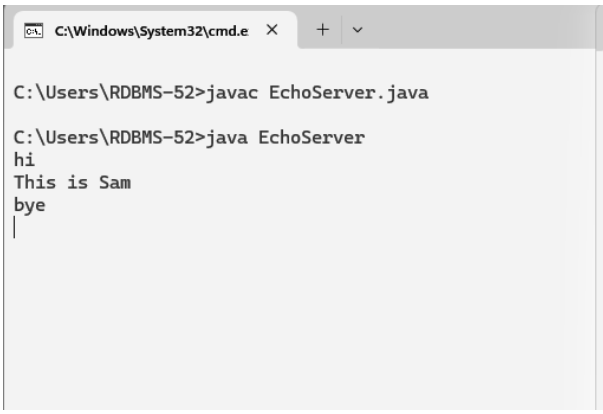
```
}
private ServerSocket server;
}
```

EchoClient.java

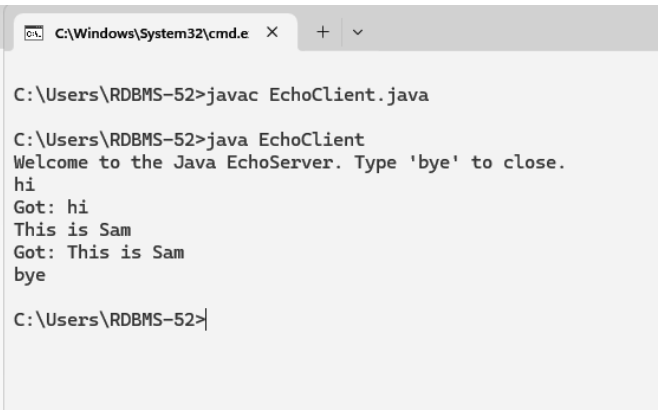
```
import java.io.*;
import java.net.*;
public class EchoClient
{
public static void main(String[] args)
{
try
{
Socket s = new Socket("127.0.0.1", 9999);
BufferedReader r = new BufferedReader(new  InputStreamReader(s.getInputStream()));
PrintWriter w = new PrintWriter(s.getOutputStream(), true);
BufferedReader con = new BufferedReader(new InputStreamReader(System.in));
String line;
do
{
line = r.readLine();
if ( line != null )
System.out.println(line);
line = con.readLine();
w.println(line);
}
while ( !line.trim().equals("bye") );
}
catch (Exception err)
{
System.err.println(err);
}}}
```

OUTPUT:

Server



Client



RESULT:

PROGRAM:

ChatServer.java

```
import java.net.*;
import java.io.*;
public class ChatServer{
 public static void main(String args[]) throws Exception{
 ServerSocket ss = new ServerSocket(9999);
 Socket socket = ss.accept();
 BufferedReader in=new BufferedReader(new InputStreamReader(socket.getInputStream()));
 PrintStream out=new PrintStream(socket.getOutputStream());
 BufferedReader stdin=new BufferedReader(new InputStreamReader(System.in));
 String s;
 while (true){
 s=in.readLine();
 if (s.equalsIgnoreCase("END")){
 out.println("BYE");
 break;
 }
 System. out.print("Client : "+s+"\n");
 System.out.print("Server : ");
 s=stdin.readLine();
 out.println(s);
 }
 ss.close();
 socket.close();
 in.close();
 out.close();
 stdin.close();
 }
}
```

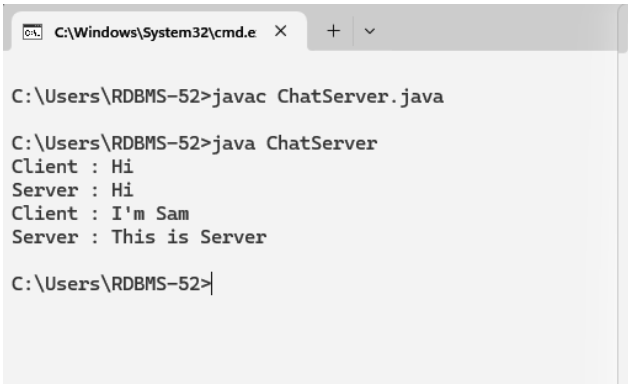
ChatClient.java

```
import java.net.*;
import java.io.*;
public class ChatClient{
 public static void main(String args[]) throws Exception{
 Socket socket=new Socket("127.0.0.1",9999);
 BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
 PrintStream out = new PrintStream(socket.getOutputStream());
 BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
 String s;
 while (true){
 System.out.print("Client : ");
 s=stdin.readLine();
 out.println(s);
 s=in.readLine();
 System.out.print("Server : "+s+"\n");
 if ( s.equalsIgnoreCase("BYE") )
 break;
```

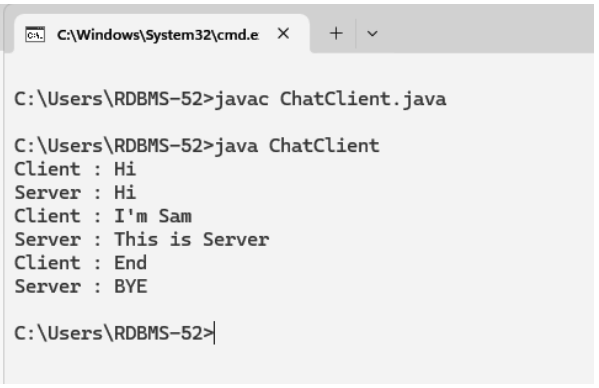
```
}
socket.close();
in.close();
out.close();
stdin.close();
}
}
```

OUTPUT:

Server



Client



PROGRAM:

UDPDNSServer.java

```
import java.io.*;
import java.net.*;
public class UDPDNSServer{
private static int indexOf(String[] array, String str){
str = str.trim();
for (int i=0; i < array.length; i++){
if (array[i].equals(str)) return i;
}
return -1;
}
public static void main(String arg[])throws IOException{
String[] hosts = {"zoho.com", "gmail.com","google.com", "facebook.com"};
String[] ip = {"172.28.251.59", "172.217.11.5","172.217.11.14", "31.13.71.36"};
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();
}
}
}
```

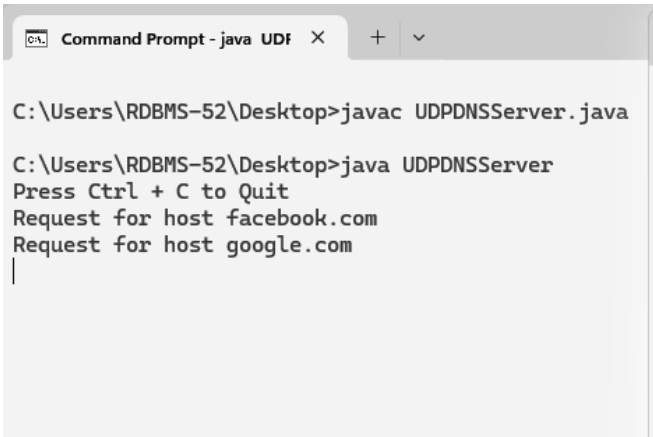
UDPDNSClient,java

```
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();
```

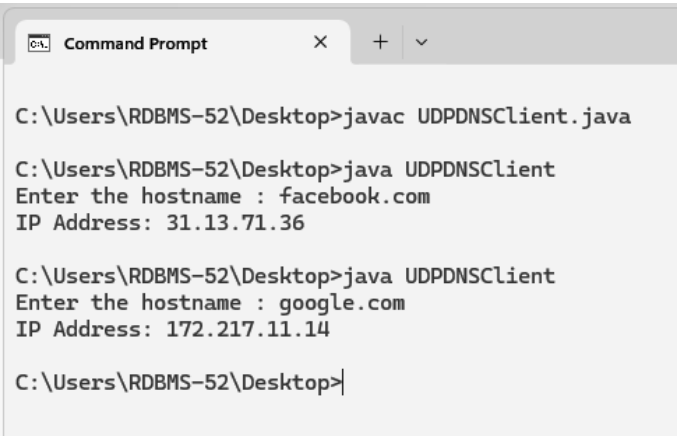
```
InetAddress ipaddress;
if (args.length == 0)
        ipaddress = InetAddress.getLocalHost();
else
        ipaddress = InetAddress.getByName(args[0]);
byte[] senddata = new byte[1024];
byte[] receivedata = new byte[1024];
int portaddr = 1362;
System.out.print("Enter the hostname : ");
String sentence = 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



Client



EX.NO: 5 **Use a tool like Wireshark to capture packets**
DATE:12/09/2024 **and examine the packets**

Aim:

Wireshark:

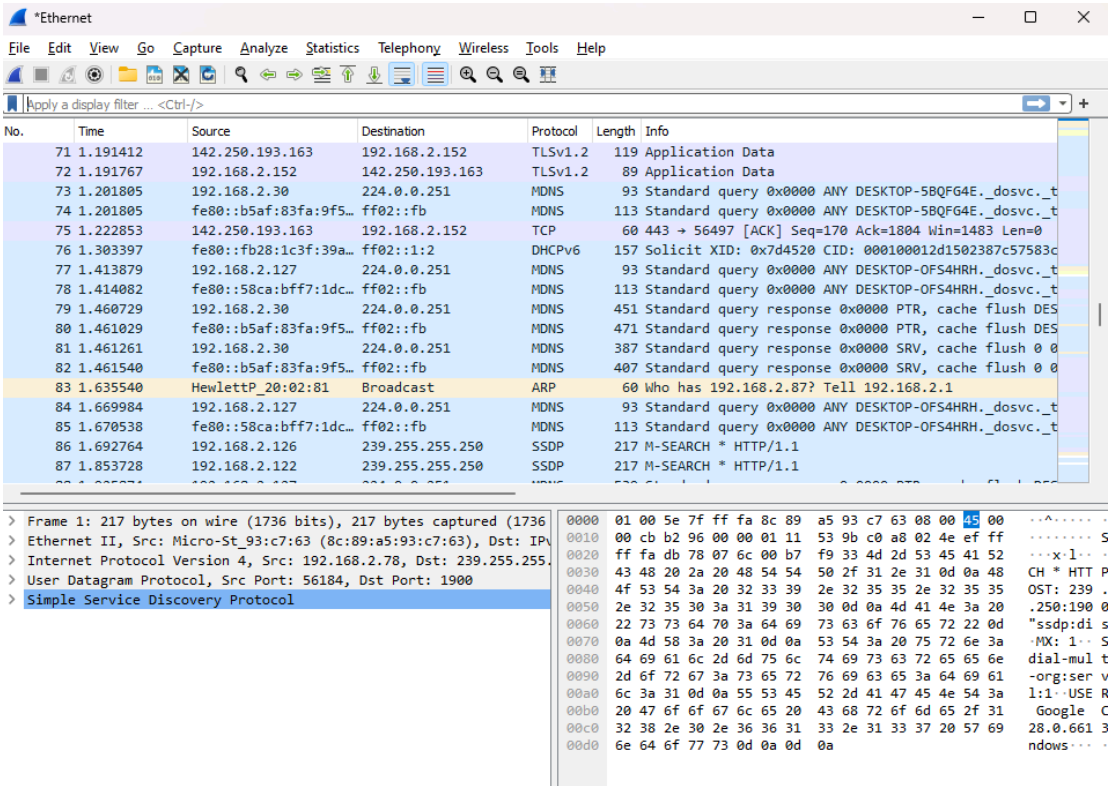
Wireshark is a widely used, open source network analyzer that can capture and display real-time details of network traffic. It is particularly useful for troubleshooting network issues, analyzing network protocols and ensuring network security. Networks must be monitored to ensure smooth operations and security.

Functionality of Wireshark:

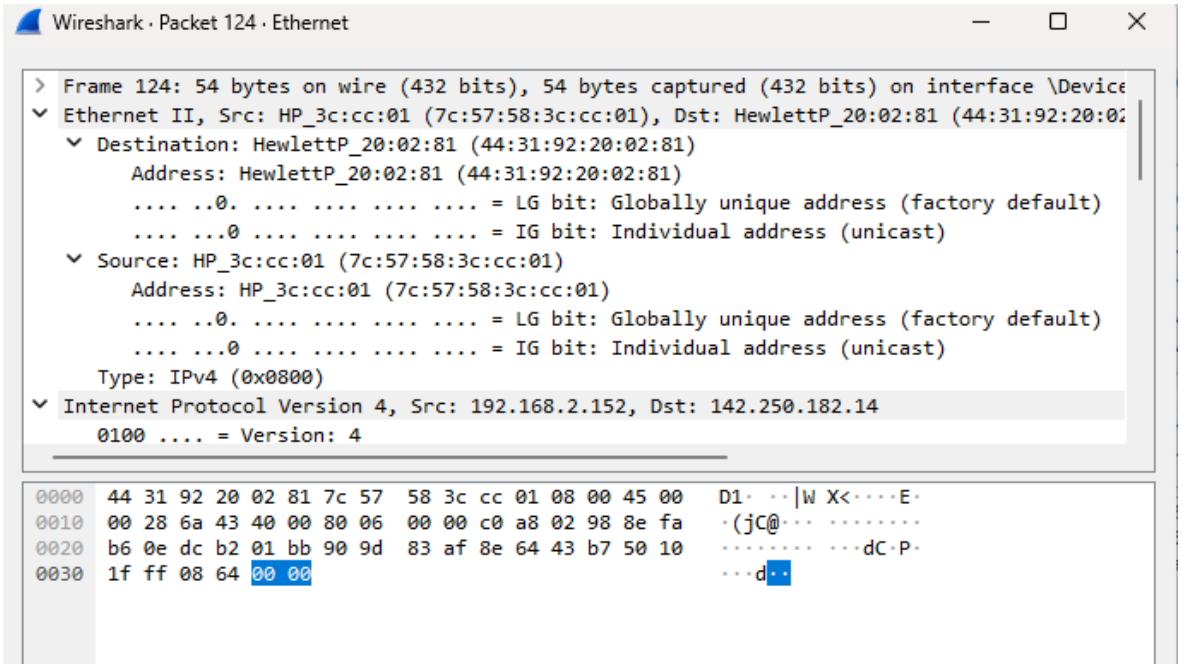
- Packet capture (PCAP). Converts network traffic into a human-readable format, making it easier to understand and diagnose concerns.
- Real-time analysis. Provides a live view of network traffic, offering immediate insights into ongoing network activities.
- Filtering capabilities. Enables users to focus on specific types of network traffic, making analysis more efficient and targeted.
- Graphical user interface (GUI). Designed for ease of use, ensures that both beginners and experts can navigate and analyze data effectively.

Procedure:

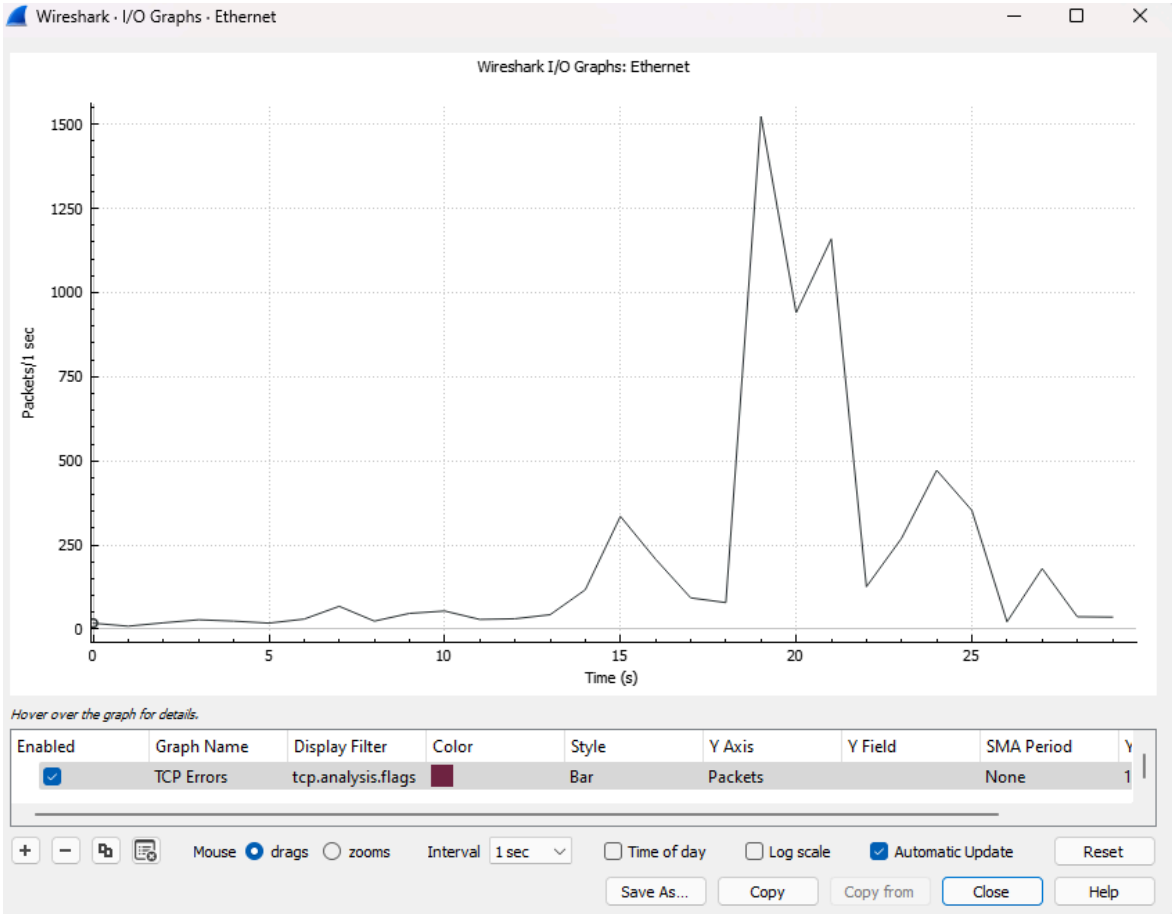
1. Install Wireshark 4.0 and open it.
2. Then, select **Capture** to capture the packets.



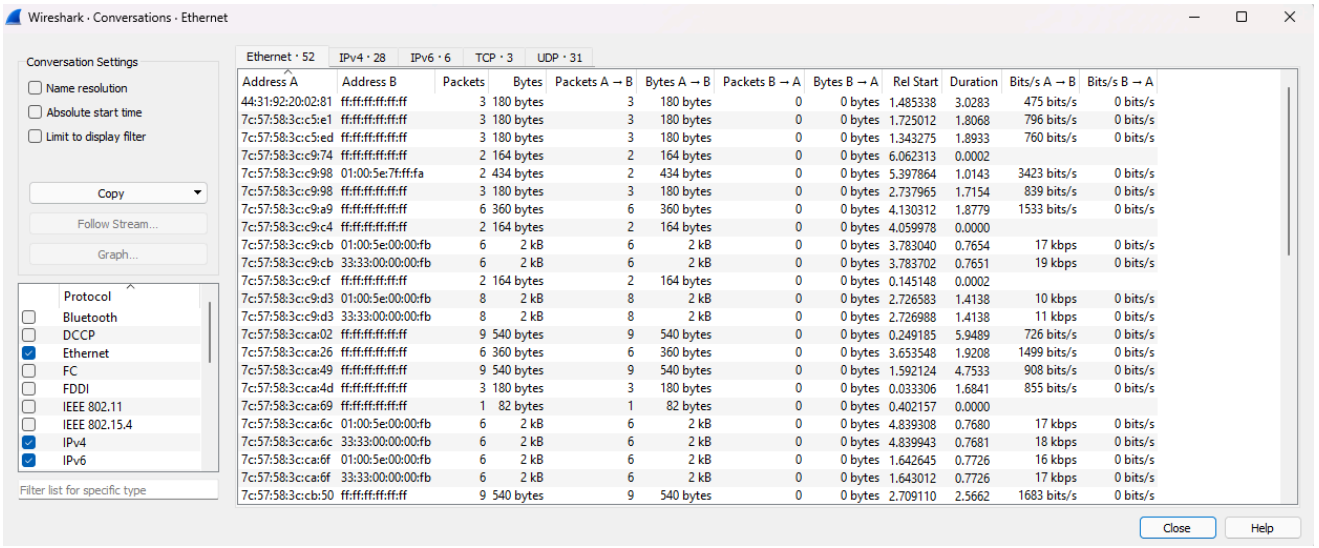
3. Select individual packets to further analyze the frames, internet protocol version, user datagram protocol, multi domain name system. It shows the hardware communication.



4. Select **Statistics** → **I/O graph** and analyze it as packets/sec.



5. Select **Statistics** → **Conversation** it shows the conversations of the various platforms.



Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
44:31:92:20:02:81	ff:ff:ff:ff:ff:ff	3	180 bytes	3	180 bytes	0	0 bytes	1.485338	3.0283	475 bits/s	0 bits/s
7c:57:58:3c:c5:e1	ff:ff:ff:ff:ff:ff	3	180 bytes	3	180 bytes	0	0 bytes	1.725012	1.8068	796 bits/s	0 bits/s
7c:57:58:3c:c5:ed	ff:ff:ff:ff:ff:ff	3	180 bytes	3	180 bytes	0	0 bytes	1.343275	1.8933	760 bits/s	0 bits/s
7c:57:58:3c:c9:74	ff:ff:ff:ff:ff:ff	2	164 bytes	2	164 bytes	0	0 bytes	6.062313	0.0002		
7c:57:58:3c:c9:98	01:00:5e:7f:ff:fa	2	434 bytes	2	434 bytes	0	0 bytes	5.397864	1.0143	3423 bits/s	0 bits/s
7c:57:58:3c:c9:98	ff:ff:ff:ff:ff:ff	3	180 bytes	3	180 bytes	0	0 bytes	2.737965	1.7154	839 bits/s	0 bits/s
7c:57:58:3c:c9:a9	ff:ff:ff:ff:ff:ff	6	360 bytes	6	360 bytes	0	0 bytes	4.130312	1.8779	1533 bits/s	0 bits/s
7c:57:58:3c:c9:c4	ff:ff:ff:ff:ff:ff	2	164 bytes	2	164 bytes	0	0 bytes	4.059978	0.0000		
7c:57:58:3c:c9:cb	01:00:5e:00:00:fb	6	2 kB	6	2 kB	0	0 bytes	3.783040	0.7654	17 kbps	0 bits/s
7c:57:58:3c:c9:cb	33:33:00:00:00:fb	6	2 kB	6	2 kB	0	0 bytes	3.783702	0.7651	19 kbps	0 bits/s
7c:57:58:3c:c9:cf	ff:ff:ff:ff:ff:ff	2	164 bytes	2	164 bytes	0	0 bytes	0.145148	0.0002		
7c:57:58:3c:c9:d3	01:00:5e:00:00:fb	8	2 kB	8	2 kB	0	0 bytes	2.726583	1.4138	10 kbps	0 bits/s
7c:57:58:3c:c9:d3	33:33:00:00:00:fb	8	2 kB	8	2 kB	0	0 bytes	2.726988	1.4138	11 kbps	0 bits/s
7c:57:58:3c:ca:02	ff:ff:ff:ff:ff:ff	9	540 bytes	9	540 bytes	0	0 bytes	0.249185	5.9489	726 bits/s	0 bits/s
7c:57:58:3c:ca:26	ff:ff:ff:ff:ff:ff	6	360 bytes	6	360 bytes	0	0 bytes	3.653548	1.9208	1499 bits/s	0 bits/s
7c:57:58:3c:ca:49	ff:ff:ff:ff:ff:ff	9	540 bytes	9	540 bytes	0	0 bytes	1.592124	4.7533	908 bits/s	0 bits/s
7c:57:58:3c:ca:4d	ff:ff:ff:ff:ff:ff	3	180 bytes	3	180 bytes	0	0 bytes	0.033306	1.6841	855 bits/s	0 bits/s
7c:57:58:3c:ca:69	ff:ff:ff:ff:ff:ff	1	82 bytes	1	82 bytes	0	0 bytes	0.402157	0.0000		
7c:57:58:3c:ca:6c	01:00:5e:00:00:fb	6	2 kB	6	2 kB	0	0 bytes	4.839308	0.7680	17 kbps	0 bits/s
7c:57:58:3c:ca:6c	33:33:00:00:00:fb	6	2 kB	6	2 kB	0	0 bytes	4.839943	0.7681	18 kbps	0 bits/s
7c:57:58:3c:ca:6f	01:00:5e:00:00:fb	6	2 kB	6	2 kB	0	0 bytes	1.642645	0.7726	16 kbps	0 bits/s
7c:57:58:3c:ca:6f	33:33:00:00:00:fb	6	2 kB	6	2 kB	0	0 bytes	1.643012	0.7726	17 kbps	0 bits/s
7c:57:58:3c:cb:50	ff:ff:ff:ff:ff:ff	9	540 bytes	9	540 bytes	0	0 bytes	2.709110	2.5662	1683 bits/s	0 bits/s

Result:

PROGRAM:

Serverarp.java

```
import java.io.*;import java.net.*;import java.util.*;
class Serverarp{
public static void main(String args[]){
try{
ServerSocket obj = new ServerSocket(5604);
Socket obj1 = obj.accept();
while(true){
DataInputStream din=new DataInputStream(obj1.getInputStream());
DataOutputStream dout=new DataOutputStream(obj1.getOutputStream());
String str=din.readLine();
String ip[]={"192.168.1.1","255.56.1.2"};
String mac[]={"5A:08:AA:C2","9A:BA:ED:F5"};
for(int i=0;i<ip.length;i++){
if(str.equals(ip[i])){
  dout.writeBytes(mac[i]+'\n');break;}}
obj.close();}}
catch(Exception e){System.out.println(e);}}}
```

Clientarp.java

```
import java.io.*;import java.net.*;import java.util.*;
class Clientarp{
public static void main(String args[]){
try{
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
Socket sk = new Socket("127.0.0.1",5604);
DataInputStream din = new DataInputStream(sk.getInputStream());
DataOutputStream dout = new DataOutputStream(sk.getOutputStream());
System.out.println("Enter the Logical address(IP):");
String str1 = in.readLine();
dout.writeBytes(str1+'\n');
String str = din.readLine();
System.out.println("The Physical Address is: "+str);
sk.close();}
catch (Exception e){
System.out.println(e);}}}
```

OUTPUT:

```
C:\Users\RDBMS-52\Desktop>java Clientarp
Enter the Logical address(IP):
192.168.1.1
The Physical Address is: 5A:08:AA:C2
```

PROGRAM:

Serverrarp.java

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverrarp{
public static void main(String args[]){
try{
DatagramSocket sk = new DatagramSocket(1309);
while(true){
byte[] sendbyte = new byte[1024];
byte[] receivebyte = new byte[1024];
DatagramPacket receiver = new DatagramPacket(receivebyte,receivebyte.length);
sk.receive(receiver);
String str = new String(receiver.getData());
String s = str.trim();
InetAddress addr = receiver.getAddress();
int port = receiver.getPort();
String ip[] = {"192.168.1.1","255.56.1.2"};
String mac[] = {"5A:08:AA:C2","9A:BA:ED:F5"};
for(int i=0;i<ip.length;i++){
if(s.equals(mac[i])){
sendbyte=ip[i].getBytes();
DatagramPacket sender = new DatagramPacket(sendbyte,sendbyte.length,addr,port);
sk.send(sender);
break;}}
break;}}
catch(Exception e){
System.out.println(e);
}}}
```

Clientrarp.java

```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientrarp{
public static void main(String args[]){
try{
DatagramSocket sk = 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);
sk.send(sender);
DatagramPacket receiver = new DatagramPacket(receivebyte,receivebyte.length);
sk.receive(receiver);
String s = new String(receiver.getData());
System.out.println("The Logical Address is(IP): "+s.trim());
sk.close();}
catch(Exception e){
System.out.println(e);}}}
```

OUTPUT:

```
C:\Users\RDBMS-52\Desktop>javac Clientrarp.java

C:\Users\RDBMS-52\Desktop>java Clientrarp
Enter the Physical address (MAC):
5A:08:AA:C2
The Logical Address is(IP): 192.168.1.1
```

EX: 7A **STUDY OF NETWORK SIMULATOR (NS)**

DATE:26.09.2024

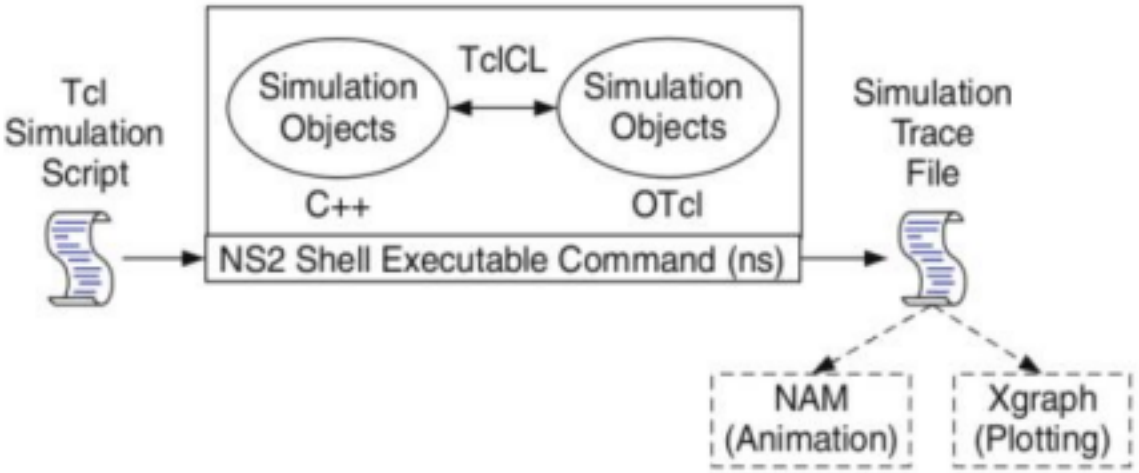
AIM:

THEORY:

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2.

In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989.

Basic Architecture of NS2:



The above figure shows the basic architecture of NS2. NS2 provides users with an executable command ns which takes on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns.

NAM:

The NAM is one of the Tcl/TK based animation tool. The tool used to view network simulation traces and real world packet traces. By using this nam tool we can perform the process like topology layout, packet level animation, and various data inspection tools.

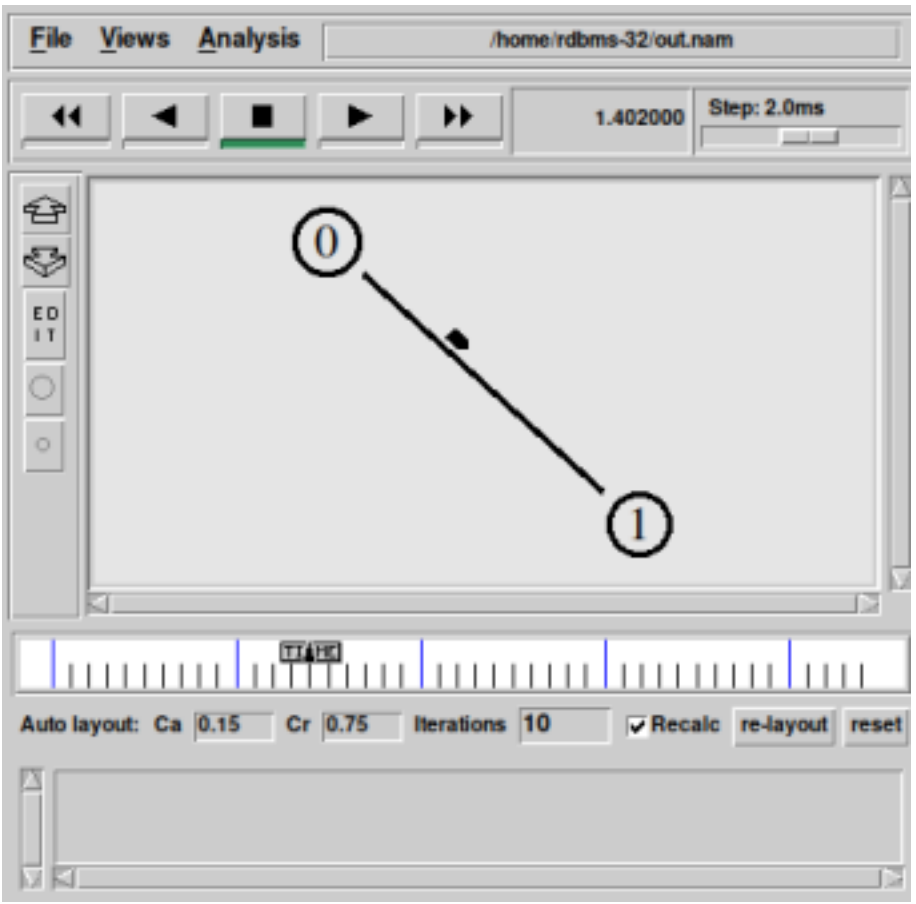
Constant Bit Rate(CBR):

A transmission method that transmits data at a fixed rate over a network. CBR is a popular choice for streaming because it's compatible with most devices and ensures a consistent viewing experience.

CODE:

```
set ns [new Simulator]
set namfile [open out.nam w]
$ns namtrace-all $namfile
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 10Mb 5ms DropTail
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set rate_ 200kb
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
$ns connect $udp0 $null0
$ns at 0.5 "$cbr0 start"
$ns at 4.4 "$cbr0 stop"
$ns run
```

OUTPUT:



RESULT:

EX 7B **Simulation of congestion control algorithms using ns**
DATE:26.09.2024

AIM:

Algorithm:

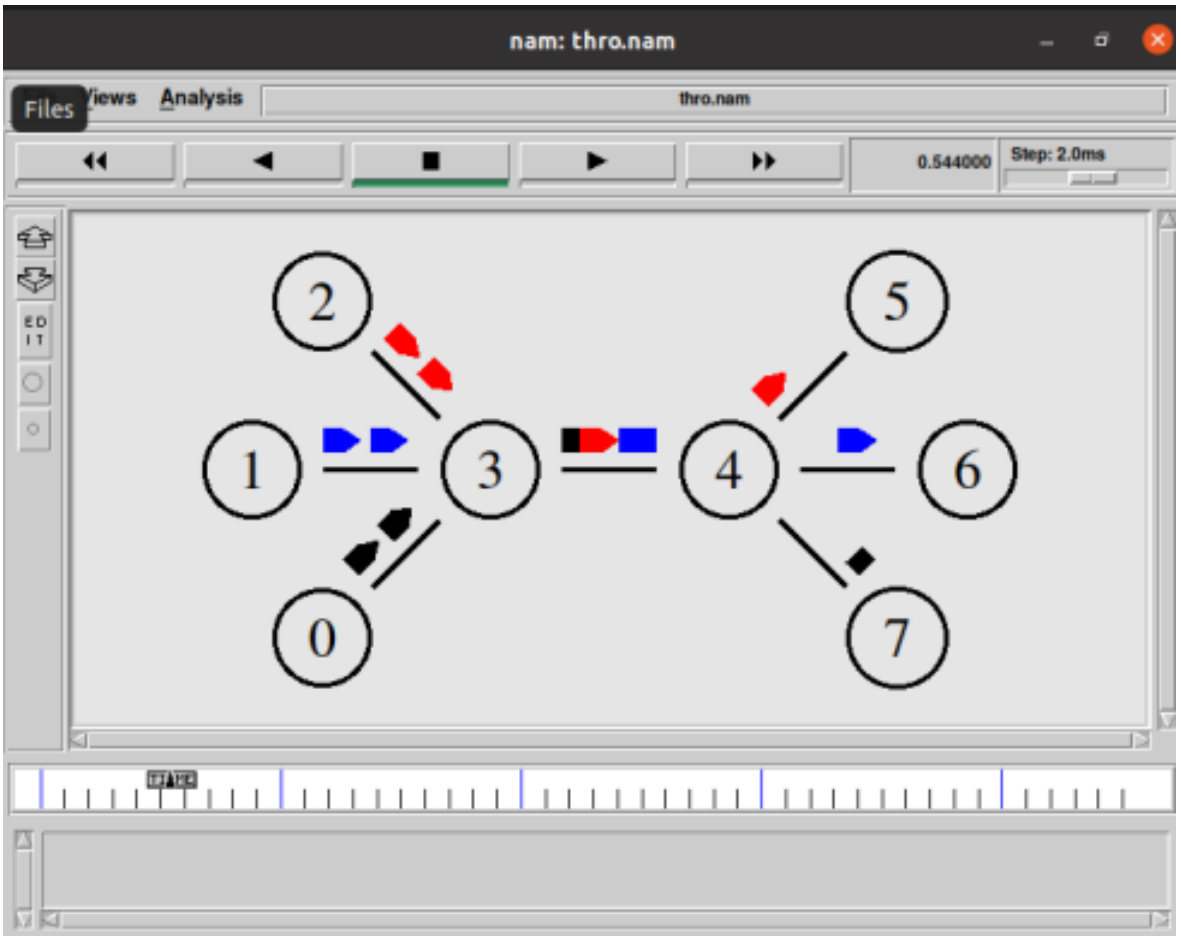
CODE:

```
set ns [new Simulator]
set nr [open thro_red.tr w]
$ns trace-all $nr
set nf [open thro.nam w]
$ns namtrace-all $nf
proc finish { } {
global ns nr nf
$ns flush-trace
close $nf
close $nr
exec nam thro.nam &
exit 0 }
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
$ns duplex-link $n0 $n3 1Mb 10ms RED
$ns duplex-link $n1 $n3 1Mb 10ms RED
$ns duplex-link $n2 $n3 1Mb 10ms RED
$ns duplex-link $n3 $n4 1Mb 10ms RED
$ns duplex-link $n4 $n5 1Mb 10ms RED
```

```
$ns duplex-link $n4 $n6 1Mb 10ms RED
$ns duplex-link $n4 $n7 1Mb 10ms RED
$ns duplex-link-op $n0 $n3 orient right-up
$ns duplex-link-op $n3 $n4 orient middle
$ns duplex-link-op $n2 $n3 orient right-down
$ns duplex-link-op $n4 $n5 orient right-up
$ns duplex-link-op $n4 $n7 orient right-down
$ns duplex-link-op $n1 $n3 orient right
$ns duplex-link-op $n6 $n4 orient left
set udp0 [new Agent/UDP]
$ns attach-agent $n2 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n5 $null0
$ns connect $udp0 $null0
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n6 $null0
$ns connect $udp1 $null0
set udp2 [new Agent/UDP]
$ns attach-agent $n0 $udp2
set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 500
$cbr2 set interval_ 0.005
$cbr2 attach-agent $udp2
set null0 [new Agent/Null]
$ns attach-agent $n7 $null0
$ns connect $udp2 $null0
$udp0 set fid_ 1
$udp1 set fid_ 2
$udp2 set fid_ 3
```

```
65 $ns color 1 Red
66 $ns color 2 Green
67 $ns color 2 Blue
68 $ns at 0.1 "$cbr0 start"
69 $ns at 0.2 "$cbr1 start"
70 $ns at 0.3 "$cbr2 start"
71 $ns at 4.0 "$cbr2 stop"
72 $ns at 4.2 "$cbr1 stop"
73 $ns at 4.5 "$cbr0 stop"
74 $ns at 5.0 "finish"
75 $ns run
```

OUTPUT:



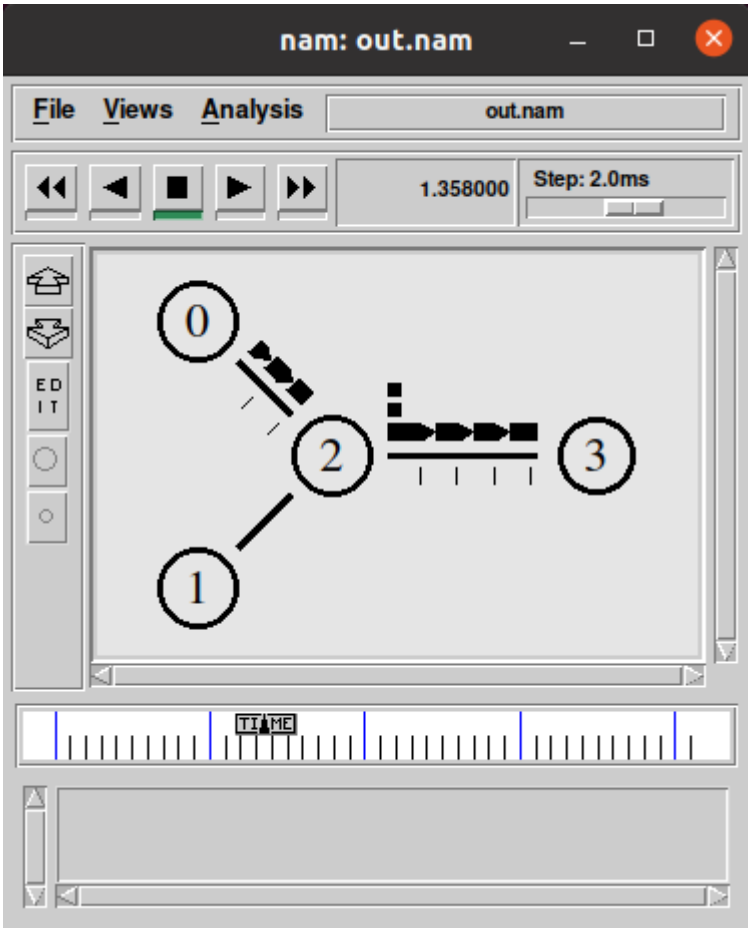
Result:

8A STUDY OF TCP

PROGRAM:

```
set ns [new Simulator]
$ns color 1 Blue
$ns color 2 Red
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
   global ns nf
   $ns flush-trace
   close $nf
   exec nam out.nam &
   exit 0}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail
$ns queue-limit $n2 $n3 10
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n2 $n3 queuePos 0.5
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$ns detach-agent $n0 $tcp;
$ns detach-agent $n3 $sink"
$ns at 5.0 "finish"
puts "TCP configuration complete."
$ns run
```

OUTPUT:


nam: out.nam
File
Views
Analysis
out.nam
1.358000
Step: 2.0ms
0
1
2
3
TIME

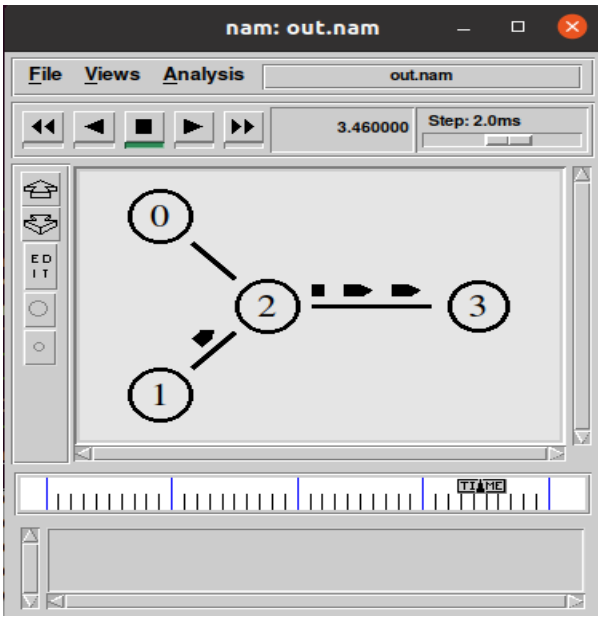
8B STUDY OF TCP

PROGRAM:

```
set ns [new Simulator]
$ns color 1 Blue
$ns color 2 Red
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
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail
$ns queue-limit $n2 $n3 10
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n2 $n3 queuePos 0.5

set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packet_size_ 1000
$cbr set rate_ 1Mb
$cbr set random_ false
```

```
$ns at 0.1 "$cbr start"
$ns at 4.0 "$cbr stop"
$ns at 4.5 "$ns detach-agent $n1 $udp; $ns detach-agent $n3 $null"
$ns at 5.0 "finish"

puts "CBR packet size = [$cbr set packet_size_]"

$ns run
```

OUTPUT:



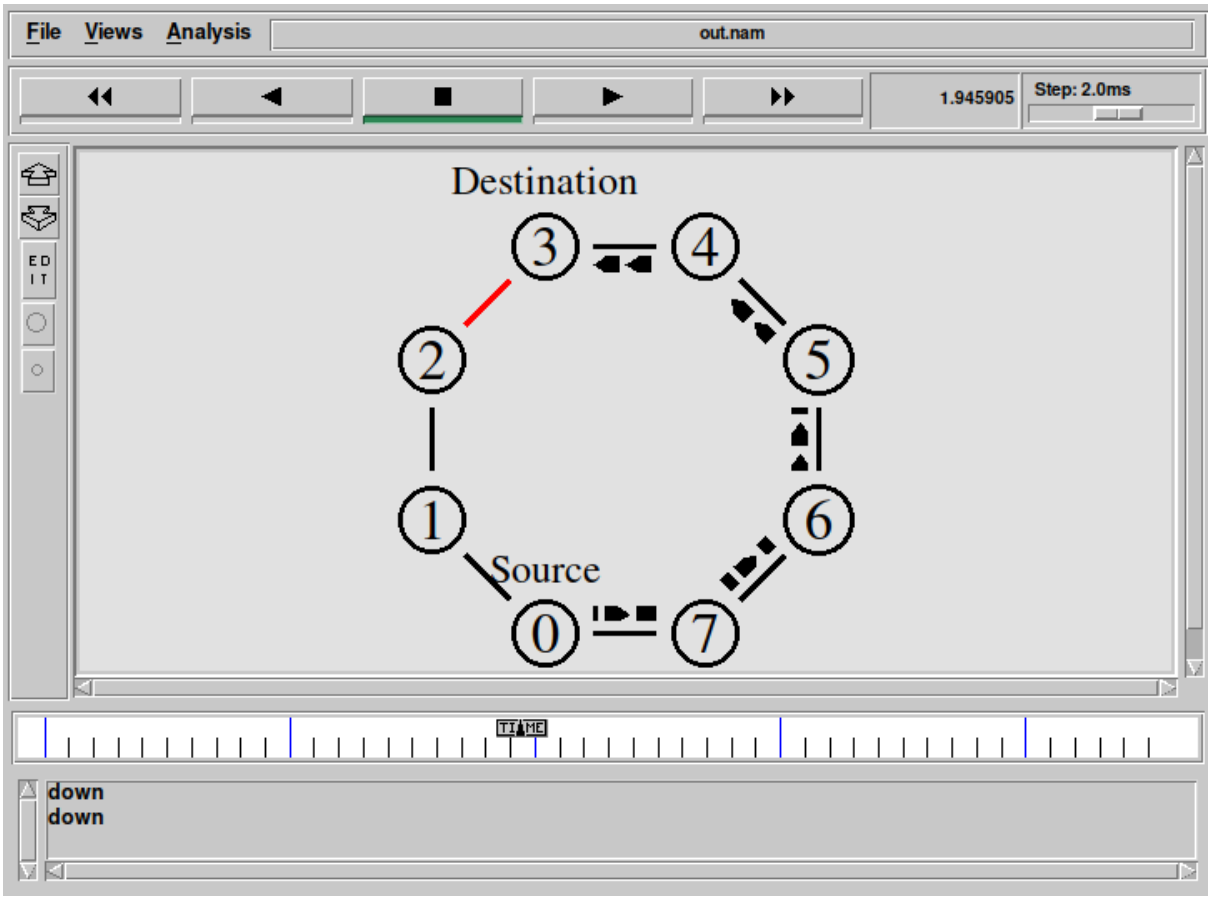
9A DISTANCE VECTOR

PROGRAM:

```
set ns [new Simulator]
$ns rtproto DV
set nf [open out.nam w]
$ns namtrace-all $nf
set nt [open trace.tr w]
$ns trace-all $nt
proc finish {} {
   global ns nf nt
   $ns flush-trace
   close $nf
   close $nt
   exec nam -a out.nam &
   exit 0
}
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 1Mb 10ms DropTail
$ns duplex-link $n5 $n6 1Mb 10ms DropTail
$ns duplex-link $n6 $n7 1Mb 10ms DropTail
$ns duplex-link $n7 $n8 1Mb 10ms DropTail
$ns duplex-link $n8 $n1 1Mb 10ms DropTail
$ns duplex-link-op $n1 $n2 orient left-up
$ns duplex-link-op $n2 $n3 orient up
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n5 $n6 orient right-down
$ns duplex-link-op $n6 $n7 orient down
$ns duplex-link-op $n7 $n8 orient left-down
$ns duplex-link-op $n8 $n1 orient left
set udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

```
set null0 [new Agent/Null]
$ns attach-agent $n4 $null0
$ns connect $udp0 $null0
$ns at 0.0 "$n1 label Source"
$ns at 0.0 "$n4 label Destination"
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n3 $n4
$ns rtmodel-at 2.0 up $n3 $n4
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

OUTPUT:



RESULT:

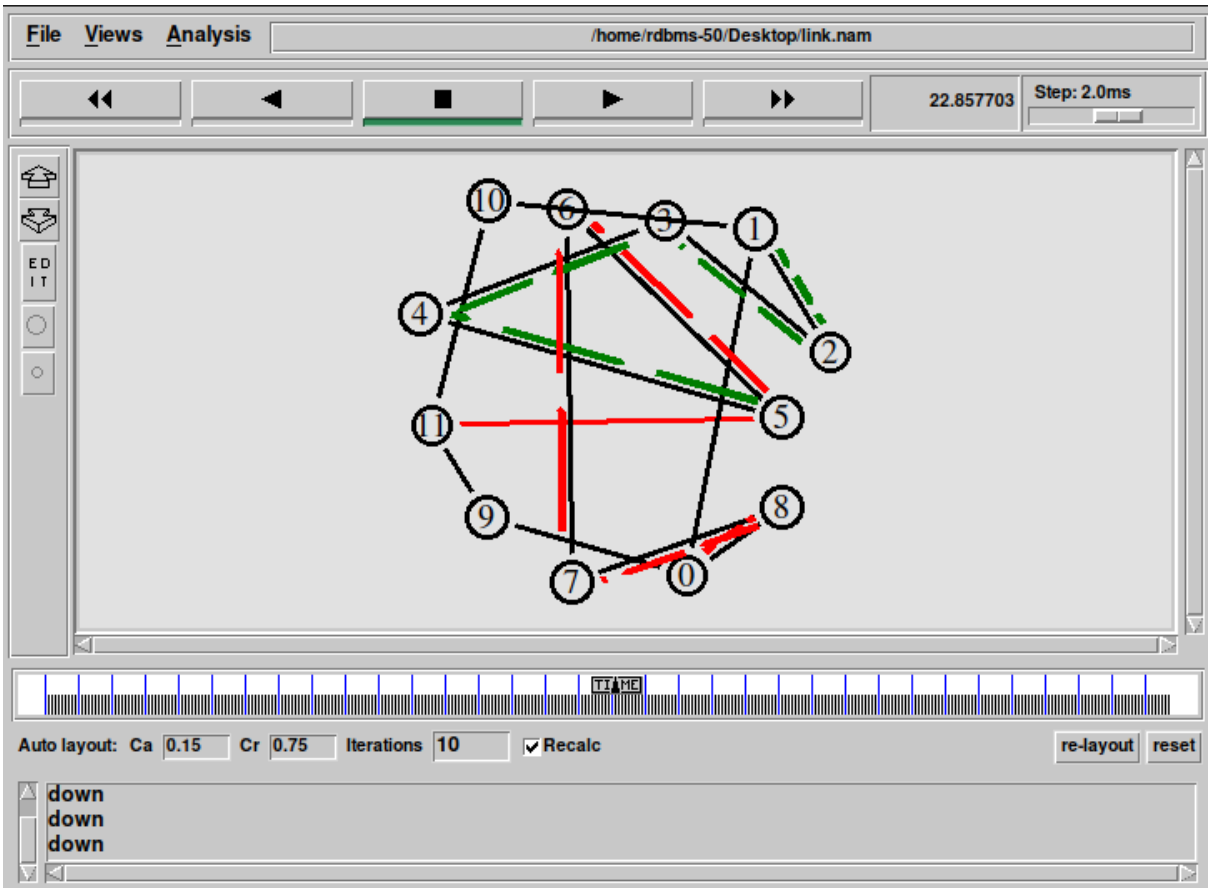
9B LINK STATE

PROGRAM:

```
set ns [new Simulator]
set nr [open link.tr w]
$ns trace-all $nr
set nf [open link.nam w]
$ns namtrace-all $nf
proc finish { } {
global ns nr nf
$ns flush-trace
close $nf
close $nr
exec nam link.nam &
exit 0
}
for { set i 0 } { $i < 12} { incr i 1 } {
set n($i) [$ns node]}
for {set i 0} {$i < 8} {incr i} {
$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }
$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp0 $null0
set udp1 [new Agent/UDP]
$ns attach-agent $n(1) $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp1 $null0
$ns rtproto LS
$ns rtmodel-at 10.0 down $n(11) $n(5)
$ns rtmodel-at 15.0 down $n(7) $n(6)
```

```
$ns rtmodel-at 30.0 up $n(11) $n(5)
$ns rtmodel-at 20.0 up $n(7) $n(6)
$udp0 set fid_ 1
$udp1 set fid_ 2
$ns color 1 Red
$ns color 2 Green
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr1 start"
$ns at 45 "finish"
$ns run
```

OUTPUT:



RESULT:

CODE:

```
import java.util.Scanner;
class CRC {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of data bits: ");
        int dataSize = sc.nextInt();
        int[] data = new int[dataSize];
        System.out.println("Enter data bits:");
        for (int i = 0; i < dataSize; i++) data[i] = sc.nextInt();
        System.out.print("Enter number of divisor bits: ");
        int divisorSize = sc.nextInt();
        int[] divisor = new int[divisorSize];
        System.out.println("Enter divisor bits:");
        for (int i = 0; i < divisorSize; i++) divisor[i] = sc.nextInt();
        int[] remainder = divide(data, divisor);
        System.out.print("Generated CRC Code: ");
        for (int i : remainder) System.out.print(i);
        System.out.println("\nData received error-free: " + checkError(data, divisor)); }
    static int[] divide(int[] data, int[] divisor) {
        int[] remainder = new int[divisor.length];
        int[] temp = new int[data.length + divisor.length - 1];
        System.arraycopy(data, 0, temp, 0, data.length);
        for (int i = 0; i < data.length; i++) {
            if (temp[i] == 1)
                for (int j = 0; j < divisor.length; j++)
                    temp[i + j] ^= divisor[j];
        }
        System.arraycopy(temp, data.length, remainder, 0, divisor.length - 1);
        return remainder;
    }

    static boolean checkError(int[] data, int[] divisor) {
        int[] remainder = divide(data, divisor);
        for (int bit : remainder) if (bit != 0) return false;
        return true;
    }
}
```

OUTPUT:

Enter number of data bits: 5
Enter data bits:
4
1
2
3
5
Enter number of divisor bits: 6
Enter divisor bits:
2
1
4
3
7
8
Generated CRC Code: 780000
Data received error-free: false