K.RAMAKRISHNAN
COLLEGE OF ENGINEERING
Permanently Affiliated to Anna University Chennai and Approved by AICTE, New Delhi
ISO 9001:2015 Certified Institution, Accredited with 'A' grade by NAAC
Samayapuram,Trichy,Tamilnadu

UCS1512 Computer Networks Laboratory

LAB MANUAL

Department of Computer Science And Engineering

V SEMESTER

Ex.no 1 Study of network components, basic network commands and network configuration commands.

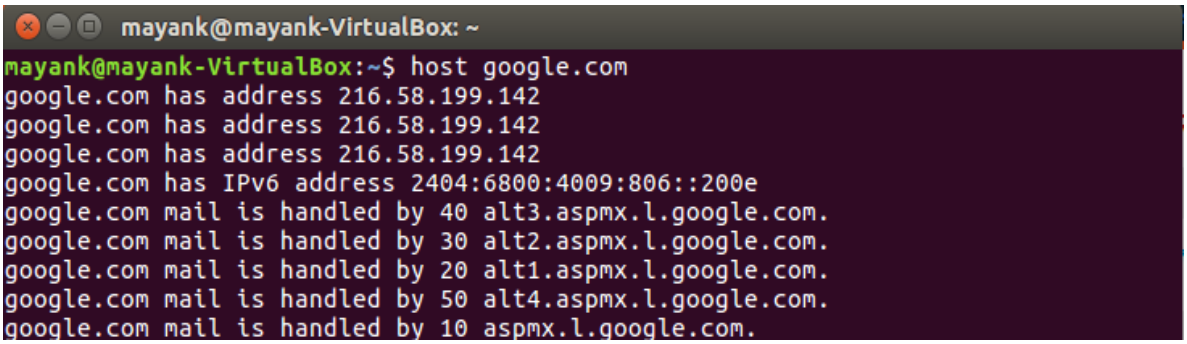
1. **Ping(Packet Internet Groper)**

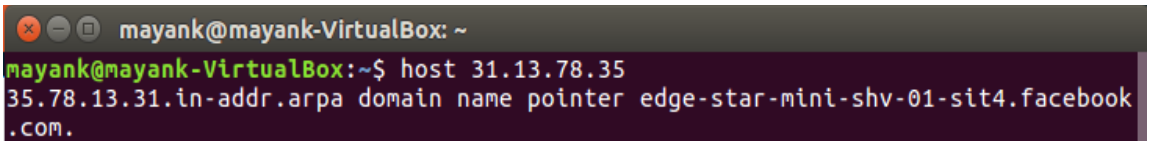
   The ping command is used to ensure that a computer can communicate to a specified device over the network.

```
mayank@mayank-VirtualBox:~$ ping google.com
PING google.com (216.58.203.142) 56(84) bytes of data.
64 bytes from bom05s10-in-f14.1e100.net (216.58.203.142): icmp_seq=1 ttl=54 time
=67.1 ms
64 bytes from bom05s10-in-f14.1e100.net (216.58.203.142): icmp_seq=2 ttl=54 time
=66.6 ms
64 bytes from bom05s10-in-f14.1e100.net (216.58.203.142): icmp_seq=3 ttl=54 time
=66.7 ms
64 bytes from bom05s10-in-f14.1e100.net (216.58.203.142): icmp_seq=4 ttl=54 time
=76.1 ms
64 bytes from bom05s10-in-f14.1e100.net (216.58.203.142): icmp_seq=5 ttl=54 time
=65.9 ms
64 bytes from bom05s10-in-f14.1e100.net (216.58.203.142): icmp_seq=6 ttl=54 time
=66.3 ms
64 bytes from bom05s10-in-f14.1e100.net (216.58.203.142): icmp_seq=7 ttl=54 time
=65.8 ms
^C
--- google.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6008ms
rtt min/avg/max/mdev = 65.801/67.816/76.196/3.464 ms
```

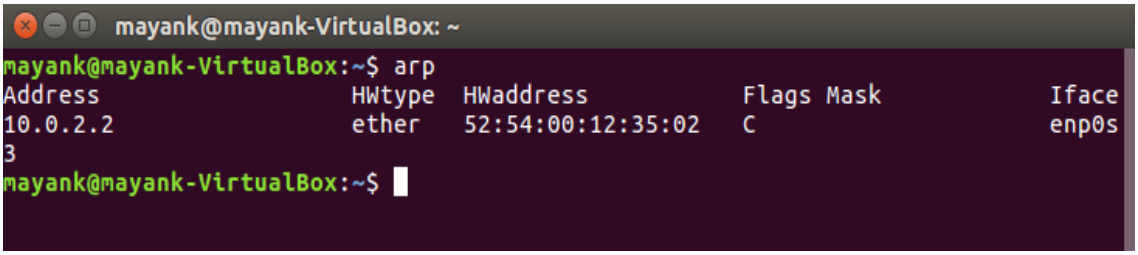
```
mayank@mayank-VirtualBox: ~
mayank@mayank-VirtualBox:~$ ping 216.58.203.142
PING 216.58.203.142 (216.58.203.142) 56(84) bytes of data.
64 bytes from 216.58.203.142: icmp_seq=1 ttl=54 time=65.8 ms
64 bytes from 216.58.203.142: icmp_seq=2 ttl=54 time=66.6 ms
64 bytes from 216.58.203.142: icmp_seq=3 ttl=54 time=68.5 ms
64 bytes from 216.58.203.142: icmp_seq=4 ttl=54 time=66.5 ms
64 bytes from 216.58.203.142: icmp_seq=5 ttl=54 time=78.7 ms
64 bytes from 216.58.203.142: icmp_seq=6 ttl=54 time=66.4 ms
64 bytes from 216.58.203.142: icmp_seq=7 ttl=54 time=66.3 ms
64 bytes from 216.58.203.142: icmp_seq=8 ttl=54 time=66.3 ms
64 bytes from 216.58.203.142: icmp_seq=9 ttl=54 time=66.0 ms
64 bytes from 216.58.203.142: icmp_seq=10 ttl=54 time=69.2 ms
64 bytes from 216.58.203.142: icmp_seq=11 ttl=54 time=68.9 ms
^C
--- 216.58.203.142 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10017ms
rtt min/avg/max/mdev = 65.850/68.151/78.702/3.533 ms
```

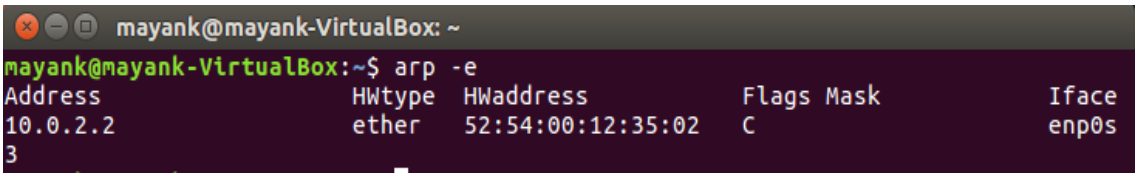
2. **host**
   host command is used to find a domain name associated with the IP address or find an IP address associated with the domain name. The returned IP address is either IPv4 or IPv6.





3. **Arp**
   ARP(Address Resolution Protocol) command is used to display and modify ARP cache, which contains the mapping of IP address to MAC address. The system's TCP/IP stack uses ARP in order to determine the MAC address associated with an IP address.





4. **telnet**
   the **telnet** command is used to create a remote connection with a system over a TCP/IP network. It allows us to administrate other systems by the terminal. We can run a program to conduct administration.

The syntax for the telnet is as Follows:

1. telnet hostname/IP address

Update the Linux system by executing the below command:

1. sudo apt update

To install the telnet, execute the below command:

1. sudo apt install telnetd -y

```
javatpoint@javatpoint-Inspiron-3542:~$  sudo apt install telnetd -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  openbsd-inetd tcpd
The following NEW packages will be installed:
  openbsd-inetd tcpd telnetd
0 upgraded, 3 newly installed, 0 to remove and 301 not upgraded.
Need to get 89.8 kB of archives.
After this operation, 294 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 tcpd amd64 7.6.q
-27 [24.2 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 openbsd-inetd am
d64 0.20160825-3 [26.3 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 telnetd amd64 0.
17-41 [39.3 kB]
Fetched 89.8 kB in 1s (112 kB/s)
Selecting previously unselected package tcpd.
(Reading database ... 175777 files and directories currently installed.)
Preparing to unpack .../tcpd_7.6.q-27_amd64.deb ...
Unpacking tcpd (7.6.q-27) ...
Selecting previously unselected package openbsd-inetd.
Preparing to unpack .../openbsd-inetd_0.20160825-3_amd64.deb ...
Unpacking openbsd-inetd (0.20160825-3) ...
Selecting previously unselected package telnetd.
Preparing to unpack .../telnetd_0.17-41_amd64.deb ...
Unpacking telnetd (0.17-41) ...
Processing triggers for ureadahead (0.100.0-20) ...
ureadahead will be reprofiled on next reboot
Setting up tcpd (7.6.q-27) ...
```

To verify the installation and whether the service is running or not, execute the below command:

1. systemctl status inetd

```
javatpoint@javatpoint-Inspiron-3542:~$ systemctl status inetd
● inetd.service - Internet superserver
   Loaded: loaded (/lib/systemd/system/inetd.service; enabled; vendor preset: en
   Active: active (running) since Sun 2020-03-29 20:38:31 IST; 1h 29min ago
     Docs: man:inetd(8)
 Main PID: 4160 (inetd)
    Tasks: 1 (limit: 4527)
   CGroup: /system.slice/inetd.service
           └─4160 /usr/sbin/inetd

Mar 29 20:38:31 javatpoint-Inspiron-3542 systemd[1]: Starting Internet superserv
Mar 29 20:38:31 javatpoint-Inspiron-3542 systemd[1]: Started Internet superserve
Mar 29 20:40:46 javatpoint-Inspiron-3542 in.telnetd[4639]: connect from 127.0.0.
lines 1-12/12 (END)
```

Now, we have to open port23 in the ufw firewall. Execute the below command:

1. ufw allow 23/tcp

```
javatpoint@javatpoint-Inspiron-3542:~$ ufw allow 23/tcp
ERROR: You need to be root to run this script
javatpoint@javatpoint-Inspiron-3542:~$ sudo ufw allow 23/tcp
[sudo] password for javatpoint:
Rules updated
Rules updated (v6)
```

The next step is to reload the firewall to apply the changes. To reload the firewall, execute the below command:

1. ufw reload

The interactive shell can be started by executing the telnet command as follows:

1. telnet

```
javatpoint@javatpoint-Inspiron-3542:~$ telnet
telnet>
```

```
telnet> h
Commands may be abbreviated.  Commands are:

close           close current connection
logout          forcibly logout remote user and close the connection
display         display operating parameters
mode            try to enter line or character mode ('mode ?' for more)
open            connect to a site
quit            exit telnet
send            transmit special characters ('send ?' for more)
set             set operating parameters ('set ?' for more)
unset           unset operating parameters ('unset ?' for more)
status          print status information
toggle          toggle operating parameters ('toggle ?' for more)
slc             set treatment of special characters

z               suspend telnet
environ         change environment variables ('environ ?' for more)
telnet>
```

For example, we are connecting our system with the localhost. Execute the command as follows:

1. telnet localhost

```
javatpoint@javatpoint-Inspiron-3542:~$ telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Ubuntu 18.04.2 LTS
javatpoint-Inspiron-3542 login: javatpoint
Password:
Last login: Sun Mar 29 22:39:18 IST 2020 from localhost on pts/1
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 5.3.0-40-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

304 packages can be updated.
29 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
```

To exit the telnet command, execute the logout command

1. logout

```
javatpoint@javatpoint-Inspiron-3542:~$ logout
Connection closed by foreign host.
javatpoint@javatpoint-Inspiron-3542:~$
```

Result:

Learn to use network components, basic network commands and network configuration commands.

Ex.No:2: Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute. Capture ping and traceroute PDUs using a network protocol analyzer and examine.

TCPDUMP:

Breaking down the Tcpdump Command Line

:~$ sudotcpdump -i eth0 -nn -s0 -v port 80

-i : Select interface that the capture is to take place on, this will often be an ethernet card or wireless adapter but could also be a vlan or something more unusual. Not always required if there is only one network adapter.

-nn : A single (n) will not resolve hostnames. A double (nn) will not resolve hostnames or ports. This is handy for not only viewing the IP / port numbers but also when

capturing a large amount of data, as the name resolution will slow down the capture.

-s0 : Snap length, is the size of the packet to capture. -s0 will set the size to unlimited - use this if you want to capture all the traffic. Needed if you want to pull binaries / files from network traffic.

-v : Verbose, using (-v) or (-vv) increases the amount of detail shown in the output, often showing more protocol specific information.

port 80 : this is a common port filter to capture only traffic on port 80, that is of course usually HTTP.

Capture only HTTP GET and POST packets:

:~$ sudotcpdump -s 0 -A -vv 'tcp[((tcp[12:1] & 0xf0) >> 2):4] =0x47455420'

:~$ sudotcpdump -s 0 -A -vv 'tcp[((tcp[12:1] & 0xf0) >> 2):4] =0x504f5354'

Extract HTTP Request URL's

:~$ sudotcpdump -s 0 -v -n -l | egrep -i "POST /|GET /|Host:"

tcpdump: listening on enp7s0, link-type EN10MB (Ethernet), capture size 262144 bytes

POST /wp-login.php HTTP/1.1

Host:dev.example.com

GET /wp-login.php HTTP/1.1

Host:dev.example.com

GET /favicon.ico HTTP/1.1
Host: dev.example.com GET
/ HTTP/1.1

Host: dev.example.com

NETSTAT:

Netstat Command Syntax

netstat [-a] [-b] [-e] [-f] [-n] [-o] [-p protocol] [-r] [-s] [-t] [-x] [-y] [time_interval] [/?]

1. **netstat-f**

netstat to show all active TCP connections. However, we do want to see the computersthat we're connected to in FQDN format [-f] instead of a simple IP address.

2. **netstat–o**

netstat will be run normally so it only shows active TCP connections, but we also want to see the corresponding process identifier [-o] for each connection so that we can determine which program on the computer initiated each one.

3. **netstat -0 | findstr28604**

Instead of displaying all connections, we're telling the netstat command to show only the connections that are using a specific PID

4. **netstat -s -p tcp-f**

In this example, we want to see protocol specific statistics [-s] but not all of them, just TCP stats [-p tcp]. We also want the foreign addresses displayed in FQDN format [-f].

5. **netstat -e -t5**

In this final example, netstat command is executed to show some basic network interface statistics [-e] and so that these statistics continually updated in the command window every five seconds [-t 5].

IFCONFIG:

ifconfig stands for "interface configuration." It is used to view and change the configuration of the network interfaces on your system.

Running the ifconfig command with no arguments, like this:

Ifconfig

eth0 Link encap:EthernetHWaddr09:00:12:90:e3:e5

inet addr:192.168.1.29 Bcast:192.168.1.255Mask:255.255.255.0

inet6addr: fe80::a00:27ff:fe70:e3f5/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500Metric:1

RX packets:54071 errors:1 dropped:0 overruns:0 frame:0
TX packets:48515 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000

RX bytes:22009423 (20.9 MiB) TX bytes:25690847 (24.5 MiB)

Interrupt:10 Base
address:0xd020 lo Link
encap:LocalLoopback

inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host

UP LOOPBACK RUNNING MTU:16436 Metric:1

RX packets:83 errors:0 dropped:0 overruns:0 frame:0
TX packets:83 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0

RX bytes:7766 (7.5 KiB) TX bytes:7766 (7.5 KiB)
wlan0 Link encap:EthernetHWaddr58:a2:c2:93:27:36

inet addr:192.168.1.64 Bcast:192.168.2.255Mask:255.255.255.0

inet6addr: fe80::6aa3:c4ff:fe93:4746/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500Metric:1

RX packets:436968 errors:0 dropped:0 overruns:0 frame:0
TX packets:364103 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000

RX bytes:115886055 (110.5 MiB) TX bytes:83286188 (79.4 MiB)

ifconfig –a

This will produce output similar to running ifconfig, but if there are any inactive interfaces on the system, their configuration will also be displayed.

ifconfig eth0

To view the configuration of a specific interface, specify its name as an option.

sudoifconfig eth1 up

ifconfig to change the status of a network interface from inactive to active, or vice-versa.

sudoifconfig wlan0 down

sudoifconfig eth1 netmask 255.255.255.0

To assign a network mask to an interface, use the keyword netmask and the netmask address. For instance, to configure the interface eth1 to use a network mask of 255.255.255.0, the command would be:

NSLOOKUP

How to find the A record of a domain.

Command line:

$ nslookup example.com

to check the NS records of a domain.

Command line:

$nslookup -type=ns example.com

to query the SOA record of a domain.

Command line:

$nslookup -type=soa example.com

to find the MX records responsible for the email exchange.

Command line:

$ nslookup -query=mx example.com

to find all of the available DNS records of a domain.

Command line:

$ nslookup -type=any example.com

to check the using of a specific DNS Server.

Command line:

$ nslookup example.com ns1.nsexample.com

to check the Reverse DNS Lookup.

Command line:

$ nslookup 10.20.30.40

to change the port number for the connection.

Command line:

$ nslookup -port=56 example.com

TRACERT

Tracert Command Syntax

tracert[**-d**] [**-h** *MaxHops*] [**-w** *TimeOut*] [**-4**] [**-6**] *target* [**/?**]

tracert 192.168.1.1

thetracert command is used to show the path from the networked computer on which the tracert command is being executed by a network device, in this case, a router on a local network, that's assigned the 192.168.1.1 IP address.

tracertwww.google.com

With the tracert command shown above, we're asking tracert to show us the path from the local computer all the way to the network device with the hostname www.google.com.

tracert -d www.yahoo.com

this tracert command example, we're again requesting the path to a website, this time www.yahoo.com, but now we're preventing tracert from resolving hostnames by using the -d option.

tracert -h 3 lifewire.com > z:\tracertresults.txt

using -h to limit the hop count to 3, but instead of displaying the results in Command Prompt, we'll use the > redirection operator to send it all to a TXT file located on Z:, an external hard drive.

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

Ex.No:3 Write a HTTP web client program to download a web page using TCPsockets.

AIM:

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

ALGORITHM:

CLIENT SIDE:

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

SERVER SIDE

1) Start the program.
2) Create a server socket to activate the portaddress.
3) Create a socket for the server socket which accepts theconnection.
4) After establishing connection receive url fromclient.
5) Download the content of the url received and send the data toclient.
6) Close thesocket.
7) End theprogram.

PROGRAM:

```
import java.io.*;
import java.net.URL;
importjava.net.MalformedURLException;
public class download {
        public static void DownloadWebPage(String webpage)
        {
                try {
                        // Create URL object
                        URL url = new URL(webpage);
                        BufferedReaderreadr =
                        newBufferedReader(new InputStreamReader(url.openStream()));
```

```
                // Enter filename in which you want to download
                BufferedWriter writer =
                newBufferedWriter(new FileWriter("Download.html"));
                // read each line from stream till end
                String line;
                while ((line = readr.readLine()) != null) {
                        writer.write(line);
                }
                readr.close();
                writer.close();
                System.out.println("Successfully Downloaded.");
        }
        // Exceptions
        catch (MalformedURLExceptionmue) {
                System.out.println("Malformed URL Exception raised");
        }
        catch (IOExceptionie) {
                System.out.println("IOException raised");
        }
    }
    public static void main(String args[])
        throwsIOException
    {
        String url = "https://www.geeksforgeeks.org/";
        DownloadWebPage(url);
    }
}
```

Output:

Successfully Downloaded.

RESULT:

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

EXP: 4A Socket Program for Echo.

AIM

To write a socket program for implementation of echo.

ALGORITHM CLIENT SIDE

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

SERVER SIDE

1. Start theprogram.
2. Create a server socket to activate the portaddress.
3. Create a socket for the server socket which accepts the connection.
4. After establishing connection receive the data fromclient.
5. Print and send the same data toclient.
6. Close thesocket.
7. End the program.

PROGRAM ECHO CLIENT

```
import java.io.*; import java.net.*; public class eclient
{
public static void main(String args[])
{
Socket c=null; String line; DataInputStream
is,is1; PrintStreamos; try
{
c=new Socket("localhost",8080);
}
catch(IOException e)
{
System.out.println(e);
}
try
{
os=new PrintStream(c.getOutputStream()); is=new
DataInputStream(System.in);
is1=new DataInputStream(c.getInputStream()); do
```

```
{
System.out.println("client"); line=is.readLine(); os.println(line); if(!line.equals("exit"))
System.out.println("server:"+is1.readLine());
}while(!line.equals("exit"));
}
catch(IOException e)
{
System.out.println("socket closed");
}}}
```

Echo Server:

```
import java.io.*; import java.net.*; import java.lang.*; public class
eserver
{
public static void main(String args[])throws IOException
{
ServerSocket s=null; String line; DataInputStream is; PrintStreamps;
Socket c=null;
try
{
s=new ServerSocket(8080);
}
catch(IOException e)
{
System.out.println(e);
}
try
{
c=s.accept();
is=new DataInputStream(c.getInputStream()); ps=new
PrintStream(c.getOutputStream()); while(true)
{
line=is.readLine();
System.out.println("msg received and sent back to client");
ps.println(line);
}
}
catch(IOException e)
{
System.out.println(e);
}
}
}
```

OUTPUT

CLIENT

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

SERVER

CONNECTION ACCEPTED
Server received KRCE

RESULT

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

EXP: 4B CLIENT- SERVER APPLICATION FOR CHAT

AIM

To write a client-server application for chat using TCP

ALGORITHM CLIENT

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

SERVER

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

PROGRAM

TCPserver1.java

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

public class TCPserver1

{
 public static void main(String arg[])
{
ServerSocket s=null; String line;
DataInputStream is=null,is1=null; PrintStreamos=null;
Socket c=null; try
{
s=new ServerSocket(9999);
}
```

```
catch(IOException e)
{
System.out.println(e);
 } try
{
c=s.accept();
is=new DataInputStream(c.getInputStream()); is1=new
DataInputStream(System.in); os=new
PrintStream(c.getOutputStream()); do
{
line=is.readLine(); System.out.println("Client:"+line);
System.out.println("Server:"); line=is1.readLine();
os.println(line);
}
while(line.equalsIgnoreCase("quit")==false); is.close(); os.close();
}
catch(IOException e)
{
System.out.println(e);
}
}
}
```

TCPclient1.java

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

 public class TCPclient1
{
 public static void main(String arg[])
{
Socket c=null; String line; DataInputStream
is,is1; PrintStreamos;
try
{
c=new Socket("10.0.200.36",9999);
}

catch(IOException e)
{
System.out.println(e);
}
try
{
 os=new PrintStream(c.getOutputStream());
```

```
 is=new DataInputStream(System.in);
is1=new DataInputStream(c.getInputStream()); do
{
System.out.println("Client:"); line=is.readLine(); os.println(line); System.out.println("Server:"
+ is1.readLine());
}
while(line.equalsIgnoreCase("quit")==false); is1.close();
os.close();
}
catch(IOException e)
{
System.out.println("Socket Closed!Message Passing is over");
}}
```

OUTPUT:

SERVER

```
C:\Program Files\Java\jdk1.5.0\bin>javac TCPserver1.java
Note: TCPserver1.java uses or overrides a deprecated API.
Note: Recompile with -deprecation for details. C:\Program
Files\Java\jdk1.5.0\bin>java TCPserver1

Client: Hai Server
Server:Hai Client
Client: How are you
Server:Fine
Client:quit
Server:quit
```

CLIENT

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

Client:Hai Server
Server: Hai Client
Client:How are you
Server: Fine
Client:quit
Server: quit
```

RESULT

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

EXP:4C FILE TRANSFER IN CLIENT &SERVER

AIM

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

ALGORITHM CLIENT SIDE

1. Start.
2. Establish a connection between the Client andServer.
3. Socket ss=new Socket(InetAddress.getLocalHost(),1100);
4. Implement a client that can send tworequests.

i) To get a file from theserver.

ii) To put or send a file to theserver.

5. After getting approval from the server ,the client either get file from the server orsend
6. file to theserver.

SERVER SIDE

1. Start.
2. Implement a server socket that listens to a particularport 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’srequest.
6. Stop.

PROGRAM CLIENT SIDE

```
import java.net.*;

import java.io.*;

public class FileClient{
    public static void main (String [] args ) throws IOException
     {
    intfilesize=6022386;
     // filesize temporary hardcoded
    long start = System.currentTimeMillis();
    intbytesRead; int current = 0;
    // localhost for testing
    Socket sock = new Socket("127.0.0.1",13267);
    System.out.println("Connecting...");
    // receive file
    byte [] mybytearray = new byte [filesize];
```

```
InputStream is = sock.getInputStream();
FileOutputStreamfos = new FileOutputStream("source-
copy.pdf"); BufferedOutputStreambos = new
BufferedOutputStream(fos); bytesRead =
is.read(mybytearray,0,mybytearray.length);
current = bytesRead;
do {
bytesRead =
is.read(mybytearray, current, (mybytearray.length-current));
if(bytesRead>= 0) current += bytesRead;
} while(bytesRead> -1);

bos.write(mybytearray, 0 , current);
bos.flush();
long end = System.currentTimeMillis(); System.out.println(end-
start); bos.close();
sock.close();
}}
```

SERVER SIDE

```
import java.net.*;

import java.io.*;

public class FileServer
{
public static void main (String [] args ) throws IOException {
ServerSocketservsock = new ServerSocket(13267);
while (true)
{
System.out.println("Waiting...");
Socket sock = servsock.accept();
System.out.println("Accepted connection : " + sock);
 File myFile= new File ("source.pdf");
byte [] mybytearray = new byte [(int)myFile.length()];
FileInputStreamfis = new FileInputStream(myFile);
BufferedInputStreambis = new BufferedInputStream(fis);
bis.read(mybytearray,0,mybytearray.length);
OutputStreamos = sock.getOutputStream();
System.out.println("Sending...");
os.write(mybytearray,0,mybytearray.length);
os.flush();
sock.close();
}}}
```

OUTPUT SERVER OUTPUT

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

Waiting for clients...
Connection Established Client wants file:network.txt

CLIENT OUTPUT

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

Connection request Connected Enter

the filename: network.txt

Computer networks: A computer network, often simply referredto as a network, is acollection of computers and devices connected by communications channels thatfacilitates communications among users and allows users to shareresources with otheruser

RESULT

Thus the File transfer Operation is done & executed successfully.

Ex.No: 5 Write a java program for socket programming.

AIM

To write a java program for socket programming.

ALGORITHM CLIENT SIDE

- To connect to another machine we need a socket connection.
- A socket connection means the two machines have information about each other's network location (IP Address) and TCP port.
- The java.net.Socket class represents a Socket.
- To open a socket:

  Socket socket = new Socket("127.0.0.1", 5000)

- To communicate over a socket connection, streams are used to both input and output the data.
- The socket connection is closed explicitly once the message to the server is sent.

SERVER SIDE

- A ServerSocket which waits for the client requests (when a client makes a new Socket())
- A plain old Socket socket to use for communication with the client.
- getOutputStream() method is used to send the output through the socket.
- After finishing, it is important to close the connection by closing the socket as well as input/output streams.

PROGRAM CLIENT SIDE

```
import java.net.*;
import java.io.*;
public class Client
{
   // initialize socket and input output streams
   private Socket socket           = null;
   privateDataInputStream  input   = null;
   privateDataOutputStream out     = null;

   // constructor to put ip address and port
   public Client(String address, int port)
   {
      // establish a connection
      try
      {
         socket = new Socket(address, port);
         System.out.println("Connected");

         // takes input from terminal
         input  = new DataInputStream(System.in);

         // sends output to the socket
         out    = new DataOutputStream(socket.getOutputStream());
      }
      catch(UnknownHostException u)
      {
         System.out.println(u);
      }
```

```
        catch(IOExceptioni)
        {
            System.out.println(i);
        }

        // string to read message from input
        String line = "";

        // keep reading until "Over" is input
        while (!line.equals("Over"))
        {
            try
            {
                line = input.readLine();
                out.writeUTF(line);
            }
            catch(IOExceptioni)
            {
                System.out.println(i);
            }
        }

        // close the connection
        try
        {
            input.close();
            out.close();
            socket.close();
        }
        catch(IOExceptioni)
        {
            System.out.println(i);
        }
    }

    public static void main(String args[])
    {
        Client client = new Client("127.0.0.1", 5000);
    }
}

Server program
import java.net.*;
import java.io.*;

public class Server
{
    //initialize socket and input stream
    private Socket          socket   = null;
    privateServerSocket    server   = null;
    privateDataInputStream in       =  null;

    // constructor with port
    public Server(int port)
```

```
    {
        // starts server and waits for a connection
        try
        {
            server = new ServerSocket(port);
            System.out.println("Server started");

            System.out.println("Waiting for a client ...");

            socket = server.accept();
            System.out.println("Client accepted");

            // takes input from the client socket
            in = new DataInputStream(
                newBufferedInputStream(socket.getInputStream()));

            String line = "";

            // reads message from client until "Over" is sent
            while (!line.equals("Over"))
            {
                try
                {
                    line = in.readUTF();
                    System.out.println(line);

                }
                catch(IOExceptioni)
                {
                    System.out.println(i);
                }
            }
            System.out.println("Closing connection");

            // close connection
            socket.close();
            in.close();
        }
        catch(IOExceptioni)
        {
            System.out.println(i);
        }
    }

    public static void main(String args[])
    {
        Server server = new Server(5000);
    }
}
```

RESULT

Thus the java program for socket programming has been executed successfully and verified.

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

NET WORK SIMULATOR (NS2)

Ns overview

- Ns programming: A Quickstart
- Case study I: A simple Wirelessnetwork
- Case study II: Create a new agent inNs

Ns overview

- NsStatus
- Periodical release (ns-2.26, Feb2003)
- Platformsupport
- FreeBSD, Linux, Solaris, Windows andMac

Ns Functionalities

Routing, Transportation, Traffic sources,Queuing disciplines, QoS

Wireless

Ad hoc routing, mobile IP, sensor-MAC
Tracing, visualization and various utilitie

NS(Network Simulators)

Most of the commercial simulators are GUI driven, while some network simulators are CLI driven. The network model / configuration describe the state of the network (nodes, routers, switches, and links) and the events (data transmissions, packet error etc.). An important output of simulations is the trace files. Trace files log every packet, every event that occurred in the simulation and are used for analysis. Network simulators can also provide other tools to facilitate visual analysis of trends and potential trouble spots.

Most network simulators use discrete event simulation, in which a list of pending "events" is stored, and those events are processed in order, with some events triggering future events—such as the event of the arrival of a packet at one node triggering the event of the arrival of that packet at a downstream node.

Simulation of networks is a very complex task. For example, if congestion is high, then estimation of the average occupancy is challenging because of high variance. To estimate the likelihood of a buffer overflow in a network, the time required for an accurate answer can be extremely large. Specialized techniques such as "control variates" and "importance sampling" havebeen developed to speedsimulation.

Examples of network simulators

There are many both free/open-source and proprietary network simulators. Examples of notable network simulation software are, ordered after how often they are mentioned in research papers:

1. ns (opensource)
2. OPNET (proprietarysoftware)
3. NetSim (proprietarysoftware)

Uses of network simulators

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive. They allow engineers, researchers to test scenarios that might be particularly difficult or expensive to emulate using real hardware - for instance, simulating a scenario with several nodes or experimenting with a new protocol in the network. Network simulators are particularly useful in allowing researchers to test new networking protocols or changes to existing protocols in a controlled and reproducible environment. A typical network simulator encompasses a wide range of networking technologies and can help the users to build complex networks from basic building blocks such as a variety of nodes and links. With the help of simulators, one can design hierarchical networks using various types of nodes like computers, hubs, bridges, routers, switches, links, mobile units etc.

Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP etc. and Local Area Network (LAN) technologies like Ethernet, token rings etc., can all be simulated with a typical simulator and the user can test, analyze various standard results apart from devising some novel protocol or strategy for routing etc. Network simulators are also widely used to simulate battlefield networks in Network-centric warfare

There are a wide variety of network simulators, ranging from the very simple to the very complex. Minimally, a network simulator must enable a user to represent a network topology, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to handle traffic in a network. Graphical applications allow users to easily visualize the workings of their simulated environment. Text-based applications may provide a less intuitive interface, but may permit more advanced forms of customization.

Packet loss

occurs when one or morepacketsof data travelling across a computer networkfail to reachtheir destination. Packet loss is distinguished as one of the three main error types encountered in digital communications; the

other two being bit error and spurious packets caused due to noise.

Packets can be lost in a network because they may be dropped when a queue in the network node overflows. The amount of packet loss during the steady state is another important property of a congestion control scheme. The larger the value of packet loss, the more difficult it is for transportlayer protocols to maintainhigh bandwidths, the sensitivity to loss of individual packets, as well as to frequency and patterns of loss among longer packet sequences is strongly dependent on the applicationitself.

Throughput

This is the main performance measure characteristic, and most widely used. Incommunicationnetworks, such asEthernetorpacket radio, throughputor network throughputis the average rate of successfulmessage delivery over a communication channel. The throughput is usually measured inbitsper second (bit/s orbps), andsometimes indata packetsper second or data packets pertime slotThis measure how soon the receiver is able to get a certain amount of data send by the sender. It is determined as the ratio of the total data received to the end to end delay. Throughput is an important factor which directly impacts the network performance

Delay

Delay is the time elapsed while a packet travels from one point e.g., source premise or network ingress to destination premise or network degrees. The larger the valueof delay, the more difficult it is for transport layer protocols to maintain highbandwidths. We will calculate end to end delay

Queue Length

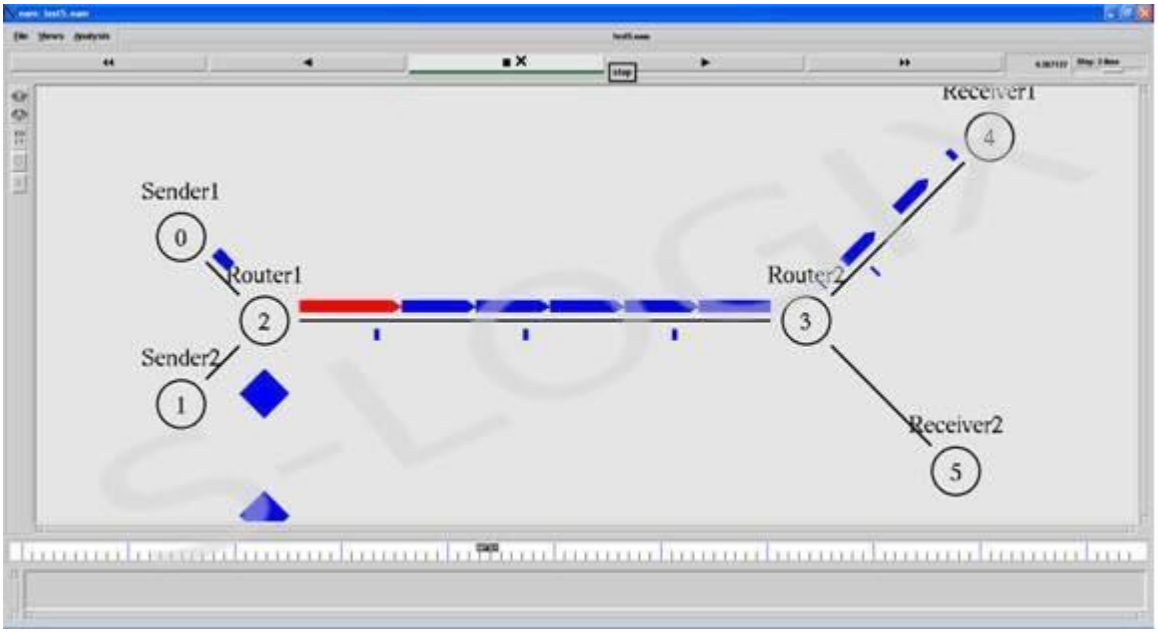
A queuing system in networks can be described as packets arriving for service, waiting for service if it is not immediate, and if having waited for service, leaving thesystem after being served. Thus queue length is very important characteristic to determine that how well the active queue management of the congestion control algorithm has been working.

 # Filename: test5.tcl

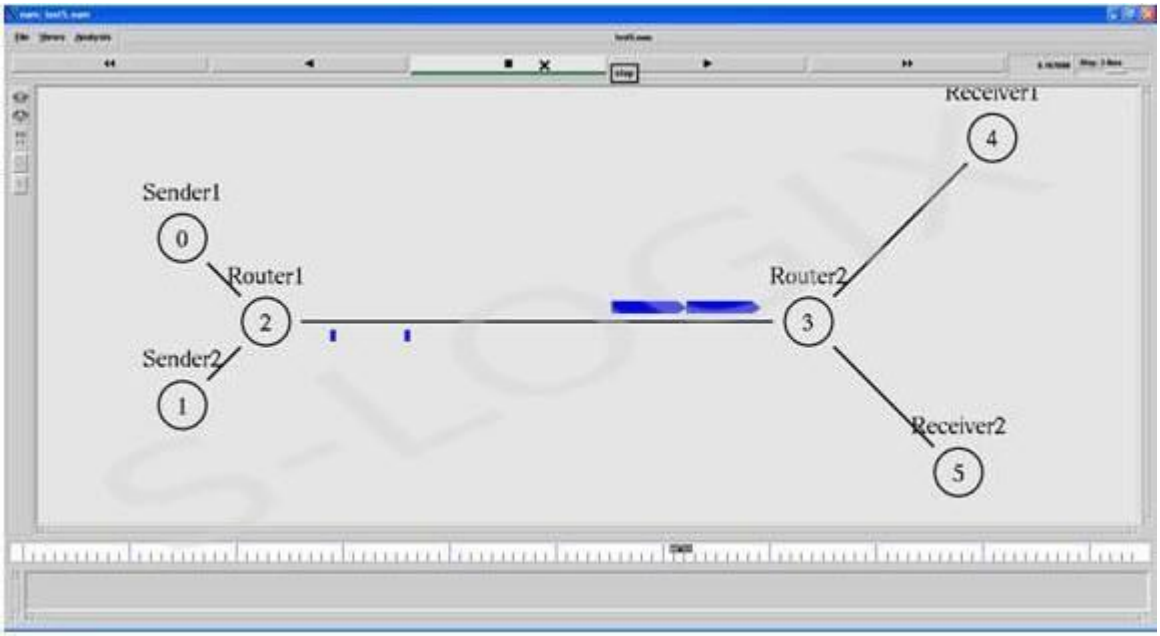
#create links between the nodes

```
$ns duplex-link $n0 $n2 2Mb 10msDropTail
$ns duplex-link $n1 $n2 2Mb 10msDropTail
$ns simplex-link $n2 $n3 0.3Mb 100msDropTail
$ns simplex-link $n3 $n2 0.3Mb 100msDropTail
$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail #Set Queue Size of
link (n2-n3) to 10
$ns queue-limit $n2 $n3 20 #Setup a TCP
connection
settcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552
#Setup a FTP over TCP connection set ftp [new
Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP #Setup a UDP connection
set udp [newAgent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2
#Setup a CBR over UDP connection set cbr [new
Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 10.0 "$ftp stop"
$ns at 10.5 "$cbr stop"
```

OUTPUT:



Congestion control mechanism of TCP



RESULT

Thus the Study of Network simulator (NS) and Simulation of Congestion Control Algorithms using NS is done successfully.

Ex.No:7 Simulate the basic network components.

AIM

To create the basic network components.

ALGORITHM

- set *ns* [new Simulator]: generates an NS simulator object instance, and assigns it to variable *ns* (italics is used for variables and values in this section). What this line does is the following:
    - Initialize the packet format (ignore this for now)
    - Create a scheduler (default is calendar scheduler)
    - Select the default address format (ignore this for now)

The "Simulator" object has member functions that do the following:

- Create compound objects such as nodes and links (described later)
- Connect network component objects created (ex. attach-agent)
- Set network component parameters (mostly for compound objects)
- Create connections between agents (ex. make connection between a "tcp" and "sink")
- Specify NAM display options

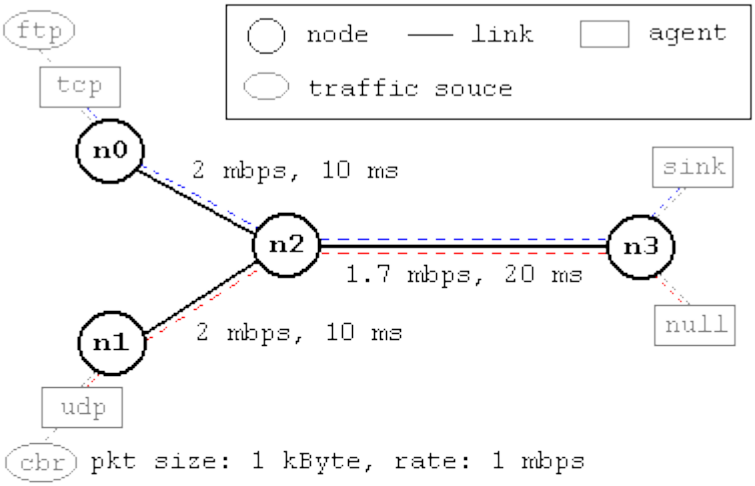
Program

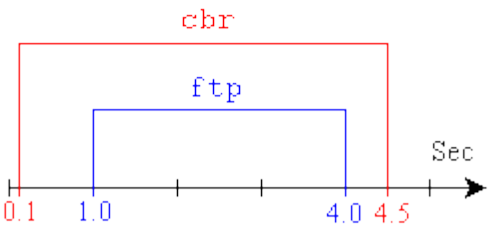
```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the NAM trace file
setnf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
global ns nf
      $ns flush-trace
      #Close the NAM trace file
close $nf
      #Execute NAM on the trace file
execnamout.nam&
exit 0
}

#Create four nodes
set n0 [$ns node]
```

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

#Create links between the nodes
$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

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

#Give node position (for NAM)
$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

#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5


#Setup a TCP connection
settcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP


#Setup a UDP connection
setudp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
setcbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false
```

```
#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Detachtcp and sink agents (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

#Run the simulation
$ns run
```

Output :





Result:

Thus the basic network components has been simulate successfully.

Ex.No:8 **Simulate the performance of TCP.**

AIM:

To simulate the performance of TCP using NS tool simulator.

Alogrithm:

- set *ns* [new Simulator]: generates an NS simulator object instance, and assigns it to variable *ns* (italics is used for variables and values in this section). What this line does is the following:
    - Initialize the packet format (ignore this for now)
    - Create a scheduler (default is calendar scheduler)
    - Select the default address format (ignore this for now)

The "Simulator" object has member functions that do the following:

1.Create a server node and bind it to a specific port number

2. Listen for a connection from the client node and accept it. This results in a client is created for the connection.

3. Read data from the client node.

4. Send data to the client via the connection.

5. Close the connection with the client.

Program:

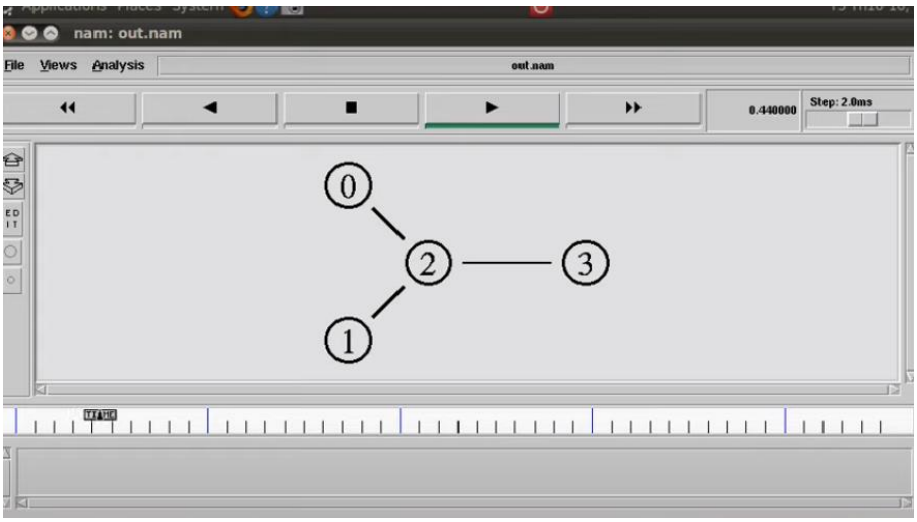
```
set ns [new Simulator]
set f [ open congestion.tr w ]
$ns trace-all $f
set nf [ open congestion.nam w ]
$ns namtrace-all $nf
$ns color 1 Red
$ns color 2 Blue
$ns color 3 White
$ns color 4 Green
#to create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

# to create the link between the nodes with bandwidth, delay and queue
$ns duplex-link $n0 $n2   2Mb  10ms DropTail
$ns duplex-link $n1 $n2   2Mb  10ms DropTail
$ns duplex-link $n2 $n3 0.3Mb 200ms DropTail
$ns duplex-link $n3 $n4 0.5Mb  40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb  30ms DropTail
```

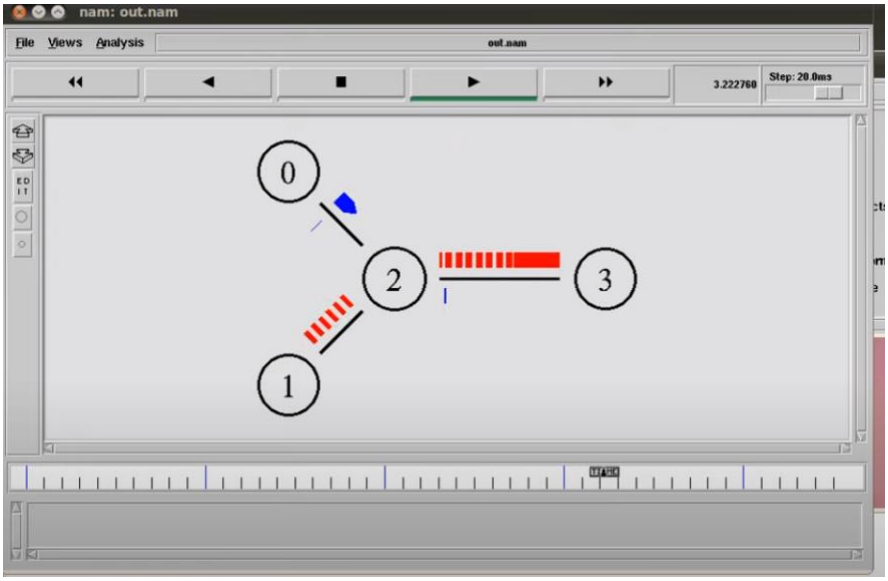
```
# Sending node with agent as Reno Agent
set tcp1 [new Agent/TCP/Reno]
$ns attach-agent $n0 $tcp1
set tcp2 [new Agent/TCP/Reno]
$ns attach-agent $n1 $tcp2
set tcp3 [new Agent/TCP/Reno]
$ns attach-agent $n2 $tcp3
set tcp4 [new Agent/TCP/Reno]
$ns attach-agent $n1 $tcp4

$tcp1 set fid_ 1
$tcp2 set fid_ 2
$tcp3 set fid_ 3
$tcp4 set fid_ 4

# receiving (sink) node
set sink1 [new Agent/TCPSink]
$ns attach-agent $n4 $sink1
set sink2 [new Agent/TCPSink]
$ns attach-agent $n5 $sink2
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
set sink4 [new Agent/TCPSink]
$ns attach-agent $n4 $sink4

# establish the traffic between the source and sink
$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2
$ns connect $tcp3 $sink3
$ns connect $tcp4 $sink4

# Setup a FTP traffic generator on "tcp"
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set type_ FTP

set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
$ftp3 set type_ FTP

set ftp4 [new Application/FTP]
$ftp4 attach-agent $tcp4
$ftp4 set type_ FTP

# RTT Calculation Using Ping ----------------------------------------------
```

```
set p0 [new Agent/Ping]
$ns attach-agent $n0 $p0
set p1 [new Agent/Ping]
$ns attach-agent $n4 $p1

#Connect the two agents
$ns connect $p0 $p1

# Method call from ping.cc file
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id] received ping answer from \
$from with round-trip-time $rtt ms."
}
# ---------------------------------------------------------------------

# start/stop the traffic
$ns at 0.2 "$p0 send"
$ns at 0.3 "$p1 send"
$ns at 0.5   "$ftp1 start"
$ns at 0.6   "$ftp2 start"
$ns at 0.7 "$ftp3 start"
$ns at 0.8   "$ftp4 start"
$ns at 66.0 "$ftp4 stop"
$ns at 67.0  "$ftp3 stop"
$ns at 68.0 "$ftp2 stop"
$ns at 70.0 "$ftp1 stop"
$ns at 70.1 "$p0 send"
$ns at 70.2 "$p1 send"

# Set simulation end time
$ns at 80.0 "finish"

# procedure to plot the congestion window
# cwnd_ used from tcp-reno.cc file
proc plotWindow {tcpSource outfile} {
global ns
set now [$ns now]
set cwnd_ [$tcpSource set cwnd_]

# the data is recorded in a file called congestion.xg.
puts  $outfile  "$now $cwnd_"
$ns at [expr $now+0.1] "plotWindow $tcpSource  $outfile"
}

set outfile [open  "congestion.xg"  w]
$ns  at  0.0  "plotWindow $tcp1  $outfile"
proc finish {} {
exec nam congestion.nam &
exec xgraph congestion.xg -geometry 300x300 &
exit 0
```

}
# Run simulation
$ns run

Output:





Result:

Thus the simulation of performance of TCP has been implemented successfully using NS tool.

Ex.No:9 **Simulate the performance of UDP.**

AIM:

To simulate the performance of TCP using NS tool simulator.

Alogrithm:

- set *ns* [new Simulator]: generates an NS simulator object instance, and assigns it to variable *ns* (italics is used for variables and values in this section). What this line does is the following:
    - Initialize the packet format (ignore this for now)
    - Create a scheduler (default is calendar scheduler)
    - Select the default address format (ignore this for now)

The "Simulator" object has member functions that do the following:

1. Create a server node and bind it to a specific port number
2. Implementation of UDP Server node in NS tool
3. Implementation of UDP Client node in NS tool
4. Running the Java UDP Server and Client node
5. Monitoring UDP Network Traffic with Wires hark

Program:

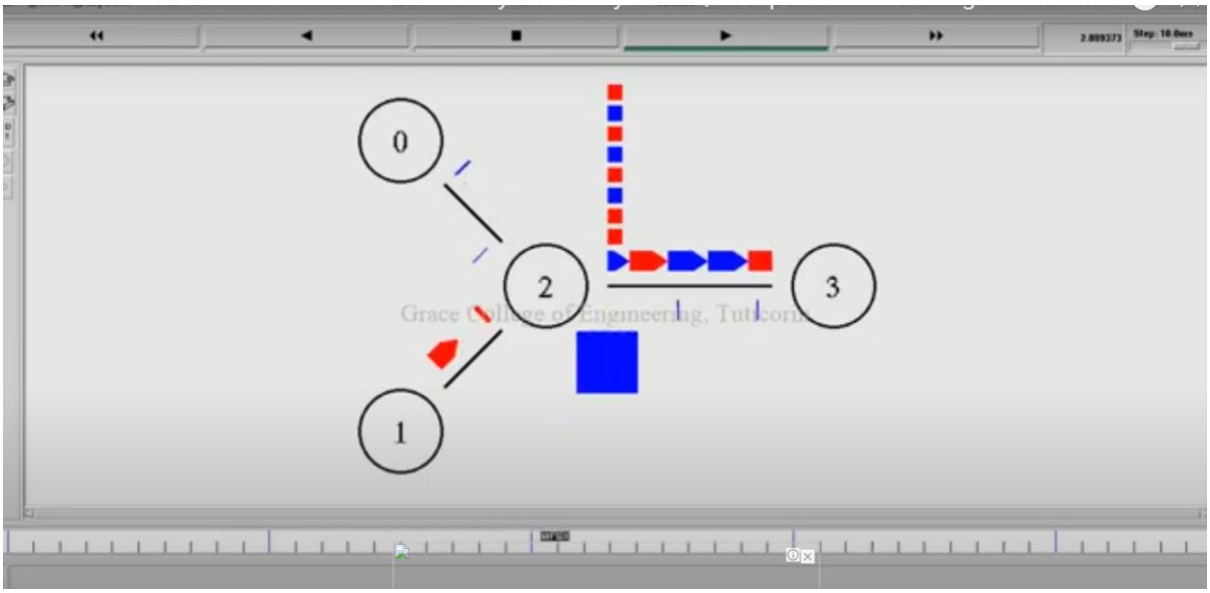
```
set ns [new Simulator]
set f [ open congestion.tr w ]
$ns trace-all $f
set nf [ open congestion.nam w ]
$ns namtrace-all $nf
$ns color 1 Red
$ns color 2 Blue
$ns color 3 White
$ns color 4 Green
#to create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

# to create the link between the nodes with bandwidth, delay and queue
$ns duplex-link $n0 $n2   2Mb  10ms DropTail
$ns duplex-link $n1 $n2   2Mb  10ms DropTail
$ns duplex-link $n2 $n3 0.3Mb 200ms DropTail
$ns duplex-link $n3 $n4 0.5Mb  40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb  30ms DropTail
```

```
# Sending node with agent as Reno Agent
set tcp1 [new Agent/TCP/Reno]
$ns attach-agent $n0 $tcp1
set tcp2 [new Agent/TCP/Reno]
$ns attach-agent $n1 $tcp2
set tcp3 [new Agent/TCP/Reno]
$ns attach-agent $n2 $tcp3
set tcp4 [new Agent/TCP/Reno]
$ns attach-agent $n1 $tcp4

$tcp1 set fid_ 1
$tcp2 set fid_ 2
$tcp3 set fid_ 3
$tcp4 set fid_ 4

# receiving (sink) node
set sink1 [new Agent/TCPSink]
$ns attach-agent $n4 $sink1
set sink2 [new Agent/TCPSink]
$ns attach-agent $n5 $sink2
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
set sink4 [new Agent/TCPSink]
$ns attach-agent $n4 $sink4

# establish the traffic between the source and sink
$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2
$ns connect $tcp3 $sink3
$ns connect $tcp4 $sink4

# Setup a FTP traffic generator on "tcp"
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set type_ FTP

set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
$ftp3 set type_ FTP

set ftp4 [new Application/FTP]
$ftp4 attach-agent $tcp4
$ftp4 set type_ FTP

# RTT Calculation Using Ping ----------------------------------------------

set p0 [new Agent/Ping]
```

```
$ns attach-agent $n0 $p0
set p1 [new Agent/Ping]
$ns attach-agent $n4 $p1

#Connect the two agents
$ns connect $p0 $p1

# Method call from ping.cc file
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id] received ping answer from \
$from with round-trip-time $rtt ms."
}
# ---------------------------------------------------------------------

# start/stop the traffic
$ns at 0.2 "$p0 send"
$ns at 0.3 "$p1 send"
$ns at 0.5   "$ftp1 start"
$ns at 0.6   "$ftp2 start"
$ns at 0.7 "$ftp3 start"
$ns at 0.8   "$ftp4 start"
$ns at 66.0 "$ftp4 stop"
$ns at 67.0  "$ftp3 stop"
$ns at 68.0 "$ftp2 stop"
$ns at 70.0 "$ftp1 stop"
$ns at 70.1 "$p0 send"
$ns at 70.2 "$p1 send"

# Set simulation end time
$ns at 80.0 "finish"

# procedure to plot the congestion window
# cwnd_ used from tcp-reno.cc file
proc plotWindow {tcpSource outfile} {
global ns
set now [$ns now]
set cwnd_ [$tcpSource set cwnd_]

# the data is recorded in a file called congestion.xg.
puts  $outfile  "$now $cwnd_"
$ns at [expr $now+0.1] "plotWindow $tcpSource  $outfile"
}

set outfile [open  "congestion.xg"  w]
$ns  at  0.0  "plotWindow $tcp1  $outfile"
proc finish {} {
exec nam congestion.nam &
exec xgraph congestion.xg -geometry 300x300 &
exit 0
}
```

Run simulation
$ns run

Output:



Result:

Thus the simulation of performance of UDP has been implemented successfully using NS tool.

Ex.No:10 **Performance Evaluation of Routing protocols using Simulation tool.**

(a) UNICASTROUTINGPROTOCOL

AIM:

Towriteans2programforimplementingunicastroutingprotocol.

PRELAB DISCUSSION:

- When a device has multiple paths to reach a destination, it always selects one pathby preferringitover others. This selection process is termed as Routing. Routingis done by special network devices called routers or it can be done by means ofsoftware processes.
- The software based routers havelimitedfunctionality andlimited scope.A routeris always configured with some defaultroute. A default route tells the routerwhereto forward apacket ifthereisnoroutefound forspecificdestination.
- In case there are multiple path existing to reach the same destination, router canmakedecisionbasedonthefollowinginformation.Routescanbestaticallyconfigured or dynamically learnt. One route can be configured to be preferred overothers.Most of the traffic on the internet and intranets known as unicast data orunicast traffic is sent with specified destination. Routing unicast data over theinternetis calledunicastrouting.
- It is the simplest form of routing because the destination is already known. Hencetherouterjusthastolookuptherouting tableandforward thepackettonexthop.
- Multicasting in computer network is a group communication, where a sender(s)send data to multiple receivers simultaneously. It supports one – to – many andmany – to – many data transmission across LANs or WANs. Through the processof multicasting, the communication and processing overhead of sending the samedata packetordataframeinminimized.
- Multicast IPRoutingprotocolsare usedto distributedata(forexample,audio/video streaming broadcasts) to multiple recipients. Using multicast, a sourcecansendasinglecopyofdatatoasingle multicast address,whichisthendistributedtoanentiregroupofrecipients.
- Thekey differencebetweenbroadcast and multicast isthat inthebroadcast thepacketis delivered to all the host connected to the network whereas,in multicastpacketisdelivered tointendedrecipientsonly.
- MulticastMessage. Multicasting identifieslogicalgroupsofcomputers.Asingle message can then be sent to the group. MulticastMessage. Multicasting uses the Internet Group Management Protocol (IGMP) toidentifygroups andgroupmembers.

ALGORITHM

1. Starttheprogram.

2. Declaretheglobalvariables ns for creatinganewsimulator.
3. Setthecolorfor packets.
4. Openthenetwork animator fileinthenameoffile2inthewritemode.
5. Openthetracefileinthenameoffile1inthewritemode.
6. Setthe unicastroutingprotocoltotransferthe packetsinnetwork.
7. Createtherequirednoofnodes.
8. Createtheduplex-linkbetweenthenodesincludingthedelaytime,bandwidthanddroppingqueuemechanism.
9. Givethepositionforthelinks betweenthenodes.
10. Setatcprenoconnectionfor sourcenode.
11. Setthedestinationnodeusingtcpsink.
12. Setupaftp connectionoverthetcpconnection.
13. Downtheconnectionbetweenany nodesataparticulartime.
14. Reconnectthedownedconnectionataparticulartime.
15. Definethefinishprocedure.
16. Inthedefinitionofthefinishproceduredeclaretheglobalvariables ns,file1, andfile2.
17. Closethetracefileandnamefileandexecutethenetworkanimationfile.
18. Attheparticular timecallthefinishprocedure.
19. Stopthe program.

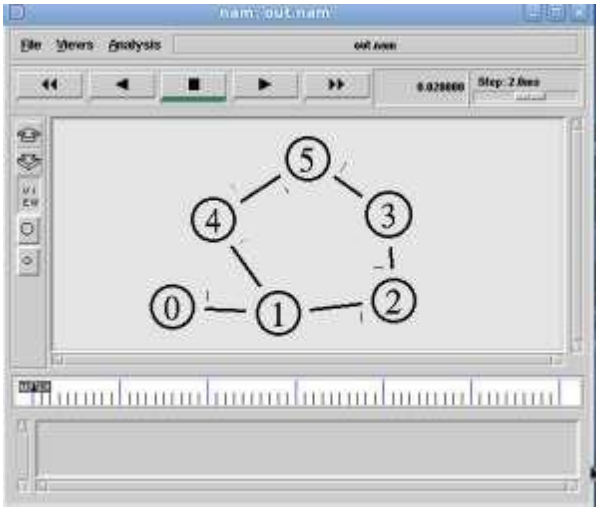
PROGRAM:

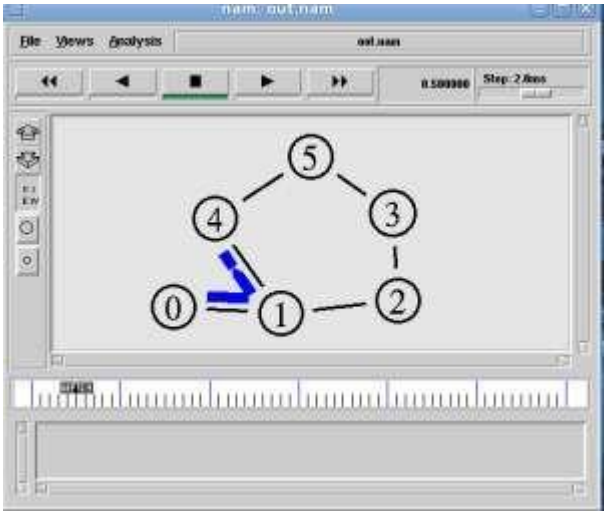
```
setns[newSimulator]
#Definedifferentcolorsfordataflows(forNAM)
$nscolor 1Blue
$nscolor2Red#Op
entheTracefile
setfile1[openout.trw]
$nstrace-all$file1
#Open the NAM trace
filesetfile2
[openout.namw]
$ns namtrace-all
$file2#Define a 'finish'
procedureprocfinish{}
{
    globalnsfile1file2
    $nsflush-
    traceclose
    $file1close$f
    ile2
```

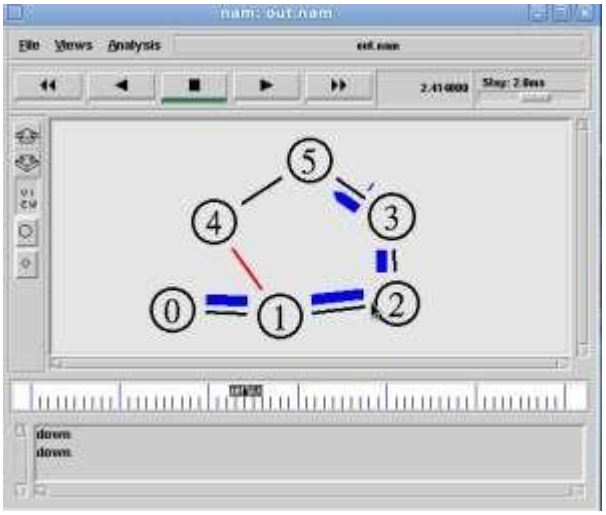
```
    execnamout.nam&
    exit3
}
#Nextlineshouldbecommentedouttohavethestaticrouting
$nsrtprotoDV#
Createsixnodess
et n0 [$ns
node]set n1
[$ns node]set
n2 [$ns
node]set n4
[$ns node]set
n4 [$ns
node]setn5[$ns
node]
#Createlinksbetweenthenodes
$nsduplex-link$n0$n1 0.3Mb10msDropTail
$nsduplex-link$n1$n2 0.3Mb10msDropTail
$nsduplex-link$n2$n3 0.3Mb10msDropTail
$nsduplex-link$n1$n4 0.3Mb10msDropTail
$nsduplex-link$n3$n5 0.5Mb10msDropTail
$nsduplex-link$n4$n5 0.5Mb10msDropTail

#Givenodeposition(forNAM)
$nsduplex-link-op$n0$n1orientright
$nsduplex-link-op$n1$n2orientright
$nsduplex-link-op$n2$n3orientup
$nsduplex-link-op$n1$n4orientup-left
$nsduplex-link-op$n3$n5orientleft-up
$nsduplex-link-op$n4$n5orientright-up

#Setupa TCPconnection
settcp[newAgent/TCP/Newreno]
$nsattach-agent$n0 $tcp
setsink[newAgent/TCPSink/DelAck]
$nsattach-agent$n5$sink
$nsconnect$tcp$sink
$tcpsetfid_1

#Setup a FTP over TCP
connectionsetftp
[newApplication/FTP]
```

$ftpattach-agent$tcp
$ftpsettype_FTP

$nsrtmodel-at1.0down$n1$n4
$nsrtmodel-at4.5up$n1$n4
$nsat0.1 "$ftp start"
$nsat6.0"finish"
$nsrun







B) MULTICASTINGROUTINGPROTOCOL

AIM:

Towriteans2programforimplementing multicastingroutingprotocol.

ALGORITHM:

1. Starttheprogram.
2. Declaretheglobalvariables ns for creatinganewsimulator.
3. Setthecolorfor packets.
4. Openthenetworkanimatorfileinthenameoffile2inthewritemode.
5. Openthetracefileinthenameoffile1inthewritemode.
6. Setthemulticastroutingprotocoltotransfer thepacketsinnetwork.
7. Createthemulticastcapablenoofnodes.
8. Createtheduplex-link betweenthenodesincludingthedelaytime,bandwidthanddroppingqueue mechanism.
9. Givethepositionforthelinks betweenthenodes.
10. Setaudpconnectionforsourcenode.
11. Setthedestinationnode,portandrandomfalseforthesourceanddestinationfiles.
12. Setupatrafficgenerator CBRforthesourceanddestinationfiles.
13. Downtheconnectionbetweenany nodesataparticulartime.
14. Createthereceiveagentforjoiningandleavingifthenodesinthegroup.
15. Definethefinishprocedure.
16. Inthedefinitionofthefinishproceduredeclaretheglobalvariables.
17. Closethetracefileandnamefileandexecutethenetworkanimationfile.
18. Attheparticular timecallthefinishprocedure.
19. Stopthe program.

PROGRAM:

```
#Createscheduler
#Create an event scheduler wit multicast
turned onsetns [new Simulator -multicaston]
#$nsmulticast#
TurnonTracing
settf[openoutput.trw]
$nstrace-all$tf
#TurnonnamTracingset
fd[openmcast.namw]
$nsnamtrace-
all$fd#Createnode
s
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 [$ns
node]setn7[$ns
node]

# Createlinks
$nsduplex-link$n0$n2 1.5Mb10msDropTail
$nsduplex-link$n1$n2 1.5Mb10msDropTail
$nsduplex-link$n2$n3 1.5Mb10msDropTail
$nsduplex-link$n3$n4 1.5Mb10msDropTail
$nsduplex-link$n3$n7 1.5Mb10msDropTail
$nsduplex-link$n4$n5 1.5Mb10msDropTail
$nsduplex-link$n4$n6 1.5Mb10msDropTail

#Routingprotocol:saydistancevector
#Protocols: CtrMcast, DM, ST,
BSTsetmprotoDM
setmrthandle[$ns mrtproto$mproto{}]

# Allocate group
addressesset
```

```
group1[Nodeallocaddr]se
t group2[Nodeallocaddr]

#UDPTransportagentforthetrafficsources
etudp0[new Agent/UDP]

$nsattach-agent$n0$udp0
$udp0 setdst_addr_$group1
$udp0setdst_port_0
setcbr1[newApplication/Traffic/CBR]
$cbr1attach-agent$udp0

#Transportagentforthetrafficsource
setudp1[newAgent/UDP]
$nsattach-agent$n1$udp1
$udp1 setdst_addr_$group2
$udp1setdst_port_0
setcbr2[newApplication/Traffic/CBR]
$cbr2attach-agent$udp1

#Createreceiver
set rcvr1[newAgent/Null]
$nsattach-agent$n5$rcvr1
$ns at 1.0 "$n5 join-group $rcvr1
$group1"setrcvr2[new Agent/Null]
$nsattach-agent$n6$rcvr2
$ns at 1.5 "$n6 join-group $rcvr2
$group1"setrcvr3[new Agent/Null]
$nsattach-agent$n7$rcvr3
$ns at 2.0 "$n7 join-group $rcvr3
$group1"setrcvr4[new Agent/Null]
$nsattach-agent$n5$rcvr1
$ns at 2.5 "$n5 join-group $rcvr4
$group2"setrcvr5[new Agent/Null]
$nsattach-agent$n6$rcvr2
$ns at 3.0 "$n6 join-group $rcvr5
$group2"setrcvr6[new Agent/Null]
$nsattach-agent$n7$rcvr3
$nsat3.5 "$n7 join-group$rcvr6 $group2"
$nsat4.0"$n5leave-group$rcvr1$group1"
$nsat4.5"$n6leave-group$rcvr2$group1"
$nsat5.0"$n7leave-group$rcvr3$group1"
```

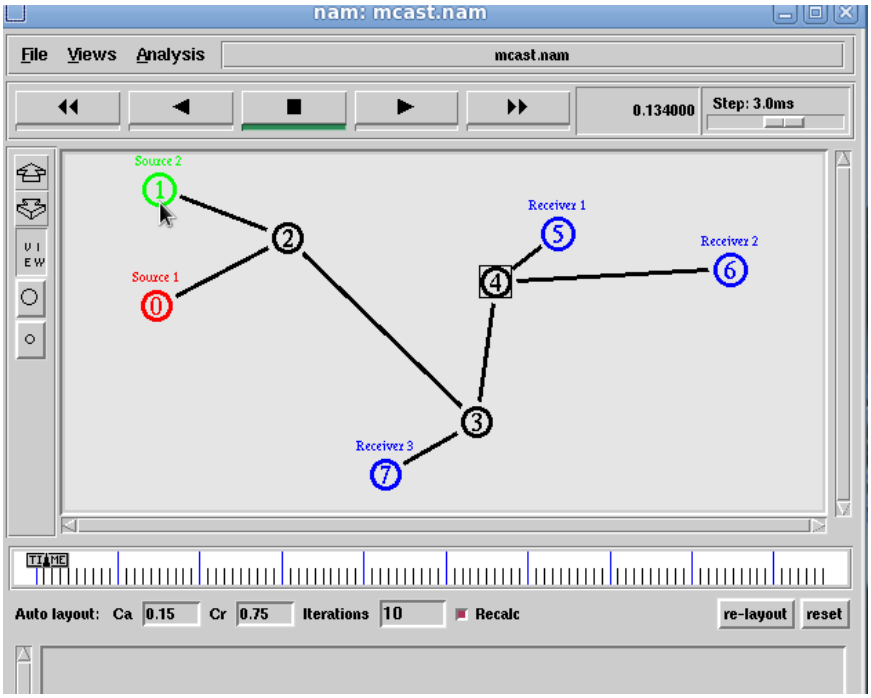
```
$nsat5.5"$n5leave-group$rcvr4$group2"
$nsat6.0"$n6leave-group$rcvr5$group2"
$ns at 6.5 "$n7 leave-group $rcvr6

$group2"#Scheduleevents

$nsat0.5"$cbr1start"
$nsat9.5 "$cbr1stop"
$nsat0.5"$cbr2start"
$nsat9.5 "$cbr2stop"

#post-processing
$nsat10.0"finish"
procfinish{}
{
        globalns tf
        $ns flush-
        traceclose $tf
        execnammcast.nam&
        exit0
}

#Fornam
#Colorsforpacketsfromtwomcastgroups
$nscolor10red
$nscolor11 green
$nscolor30purple
$nscolor31 green

#Manual
layout:orderofthelinkissignificant!#$nsdupl
ex-link-op $n0 $n1 orientright
#$nsduplex-link-op $n0 $n2orientright-
up#$ns duplex-link-op $n0 $n3 orient
right-down#Show queueonsimplexlinkn0-
>n1
#$nsduplex-link-op $n2 $n3 queuePos0.5

# Group0source
$udp0set fid_10
$n0colorred
```

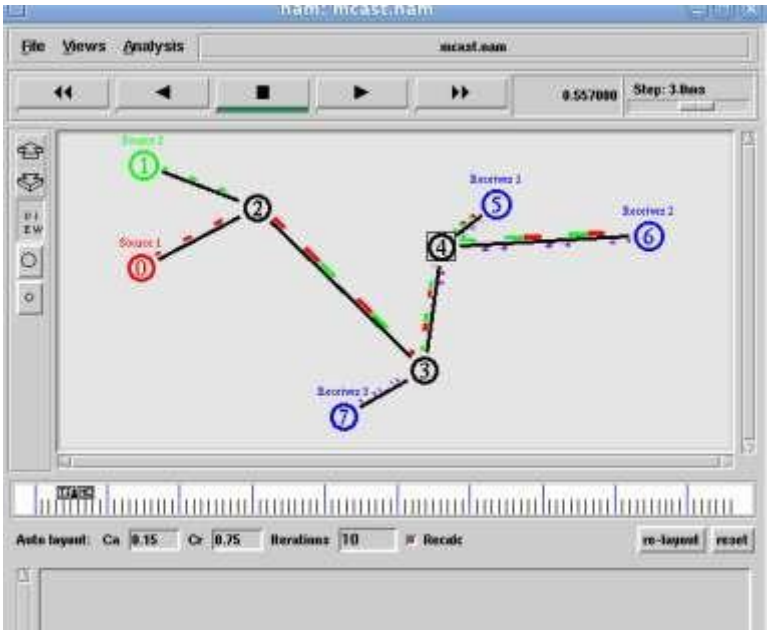
```
$n0label"Source 1"

# Group1source
$udp1set fid_11
$n1color green
$n1label"Source 2"
$n5label"Receiver1"
```

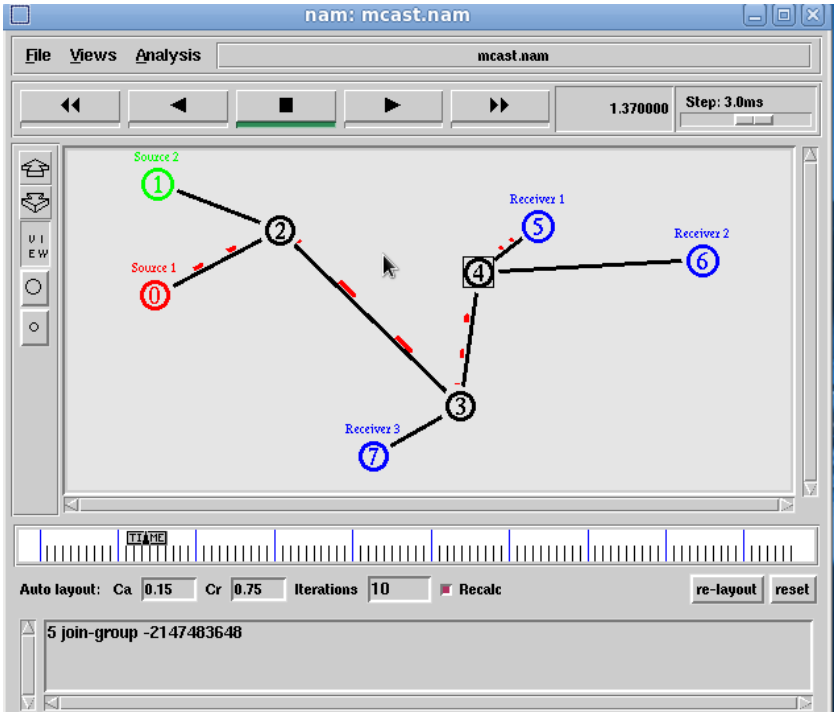
$n5colorblue
$n6label"Receiver2"
$n6colorblue
$n7label"Receiver3"
$n7colorblue

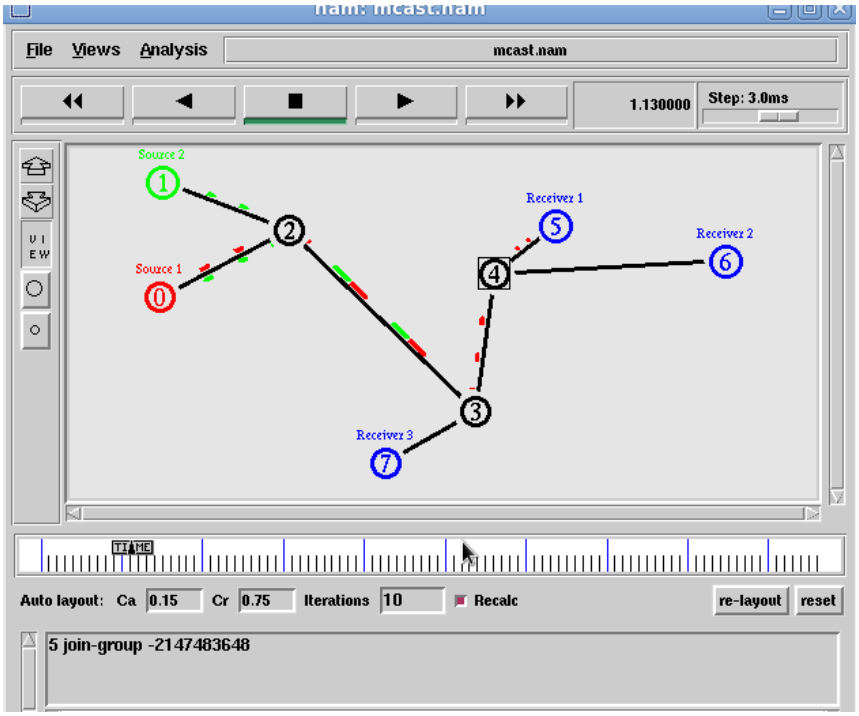
#$n2add-
markm0red#$n2dele
te-markm0"

#Animationrate
$nsset-animation-rate3.0ms
$nsrun




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


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



RESULT:

Thusthecasestudyaboutthedifferentrouting algorithmstoselectthenetworkpathwithitsoptimumandeconomicalduringdatatransferisdone .

Ex.no:11 Install OPNET and build a simple network using OPNET.

Aim:

To install OPNET and build a simple network using OPNET

Procedure:

Create a New Project

Inthisfirstphaseyouwillopenandnameyourprojectandnamethefirstsimulationscenario. Thefirstsimulationscenariowillconsistof16networkedstations(PCs)andonehub.Inthis first phase you will specify the geographic size of thenetwork.

1. Startthe**OPNETITGuruAcademicEdition**→Choose**New**fromthe**File**menu
2. Select**Project**andclick**OK**→;Nametheproject**<youremailusername>_Lab1** (e.g. xy123_Lab1) → Name the scenario **OnlyHub**; Click **OK**
3. In the *Startup Wizard:Initial Topology* dialog box, make sure that **Create Empty Scenario**isselected→Click**Next**→Choose**Office**fromthe*NetworkScale*List→ Click **Next** three times → Click**Finish**.
4. **Close** the *Object Pallete*dialogbox.

Create the Network

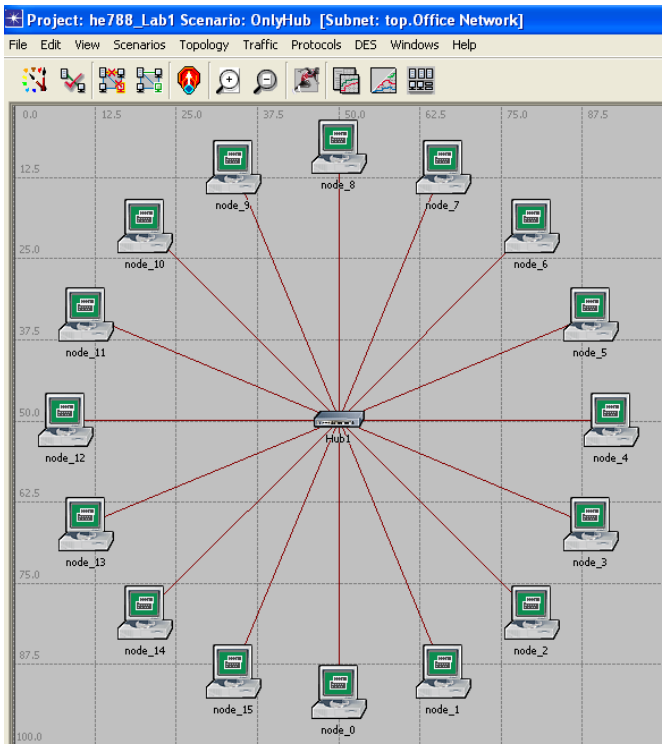
To create our switched LAN:

1. Select**Topology→RapidConfiguration**.Fromthedrop-downmenuchoose**Star** and click **OK**.
2. Clickthe**SelectModels**buttoninthe*RapidConfiguration*dialogbox.Fromthe*Model List* drop-down menu choose **ethernet**and click**OK**.
3. IntheRapidConfigurationdialogbox,setthefollowingsixvalues:**CenterNodeModel**

   =ethernet16_hub,PeripheryNodeModel=ethernet_station,LinkModel= 10BaseT,Number=16,Y=50,andRadius=42→Click**OK**.

4. Right-clickon**node_16**,whichisthehub→**EditAttributes**→Changethe**name** attribute to **Hub1** and click **OK**.

5. NowthatyouhavecreatedthenetworkitshouldlookliketheNetworkonFigure below.

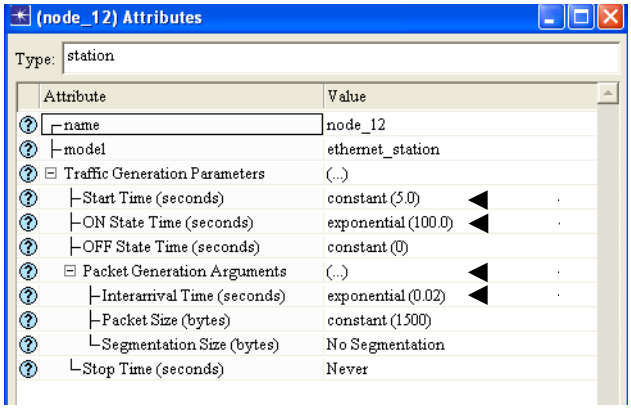


6. Make sure to save yourproject.

Configure the network nodes

Here you will configure the traffic generated by the stations.

1. Right-clickonanyofthe16stations(node_0tonode_15)→**SelectSimilarNodes**. Now all stations in the network areselected.
2. Right-click on any of the 16 stations →**Edit Attributes**.
   a. Checkthe**ApplyChangestoSelectedObjects**checkbox.Thisisimportant to avoid reconfiguring each nodeindividually.
3. Expandthehierarchiesofthe**TrafficGenerationParameters**attributeand**the PacketGenerationArguments**attribute→Setthefollowingfourvalues:



4. Click**OK**toclosetheattributeeditingwindow.Saveyourproject

Noteherethatwehaveintroducedatrafficgenerationateachnode.Thetrafficmodelfollowsa well-knownON-OFFmodel,inwhicheachnodeswitchesbetweenOnstateinwhichthetraffic isgenerated,andOFFstateinwhichthereisnotraffic.ThedurationofONandOFFstatesis random, and in this example follows exponential distribution. In this example, the duration of OFF state is0.

Result:

Thus the OPNET simulator has been installed successfully and the simple network has been build successfully

RARP protocol implementation in java

Posted by SHOBANADEVIIVARAJA on OCTOBER 12, 2017

RARP means Reverse Address Resolution Protocol. It maps the physical address to logical address.

Cn.java – //Server program

1.Create a socket.Bind with port number
2.Store mac address in a array and ip address in a array
3.Read the data from user and check with mac address array if match found send corresponding ip address value .Else send not found message to client
4.Close the connection

/*

```
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package cn;
import java.net.*;
import java.io.*;

public class Cn {
 public static void main(String[] args) {
 try{
 ServerSocket s=new ServerSocket(6666);
 Socket ss=s.accept();
 DataOutputStream out=new DataOutputStream(ss.getOutputStream());
 DataInputStream in=new DataInputStream(ss.getInputStream());
 String msg=(String)in.readUTF();
 String ip[]={"10.2.4.1"};
 String mac[]={"ada:1f:2f:sd:2f:ff"};
 for(int i=0;i<ip.length;i++){
 if(msg.equals(mac[i])){
 String ans=ip[i];
 out.writeUTF(ans);
 }else{
 System.out.println("Not Found");
 }
 }
 // System.out.println("message"+msg);
 s.close();
```

```
 }

 catch(Exception e){

 System.out.println(e);

 }

 }

}
```

Client.java -//Client file
1.Create a connection with server using ip and port number
2.Get mac address from the user and send to the server.
3.Read the data sent by the server and display it
4.Close the connection

```
/*

 * To change this license header, choose License Headers in Project Properties.

 * To change this template file, choose Tools | Templates

 * and open the template in the editor.

 */

package cn;

import java.net.*;

import java.io.*;

import java.util.Scanner;

public class Client {

 public static void main(String[] args) {

 try{

 Socket s=new Socket("localhost",6666);

 DataOutputStream in=new DataOutputStream(s.getOutputStream());

 DataInputStream dis=new DataInputStream(s.getInputStream());

 Scanner sc=new Scanner(System.in);

 String msg=sc.next();

 in.writeUTF(msg);

 String ip=(String)dis.readUTF();
```

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

OUTPUT:

First run the server file and then run client file

Server:

Conneted

Client:

Connected to server

Enter the physical address

DA12:1R:2E:sd:2f:ff

The logical Address is 10.2.4.1

Result:

Thus the java program to implement RARP protocol has been executed successfully.

Simulation of Go Back N protocol

AIM:

To Simulate and to study of Go Back N protocol

PRE LAB DISCUSSION:

- Go Back N is a connection oriented transmission. The sender transmits the frames continuously. Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size.
- The sender has a window i.e. a buffer to store the frames. This buffer size is the number of frames to be transmitted continuously.
- A station may send multiple frames as allowed by the window size.
- Receiver sends an ACK i if frame i has an error. After that, the receiver discards all incoming frames until the frame with error is correctly retransmitted.
- Go-Back-N Automatic Repeat reQuest (Go-Back-N ARQ), is a data link layer protocol that uses a sliding window method for reliable and sequential delivery of data frames. It is a case of sliding window protocol having to send window size of N and receiving window size of 1.
- Go – Back – N ARQ uses the concept of protocol pipelining, i.e. sending multiple frames before receiving the acknowledgment for the first frame. The frames are sequentially numbered and a finite number of frames. The maximum number of frames that can be sent depends upon the size of the sending window. If the acknowledgment of a frame is not received within an agreed upon time period, all frames starting from that frame are retransmitted.
- The size of the sending window determines the sequence number of the outbound frames. If the sequence number of the frames is an n-bit field, then the range of sequence numbers that can be assigned is 0 to 2n−1. Consequently, the size of the sending window is 2n−1. Thus in order to accommodate a sending window size of 2n−1, an n-bit sequencenumber is chosen.
- The sequence numbers are numbered as modulo-n. For example, if the sending window size is 4, then the sequence numbers will be 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, and so on. The number of bits in the sequence number is 2 to generate the binary sequence 00, 01, 10, 11.
- The size of the receiving window is 1

ALGORITHM :

1. The source node transmits the frames continuously.
2. Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size.
3. The source node has a window i.e. a buffer to store the frames. This buffer size is the number of frames to be transmitted continuously.

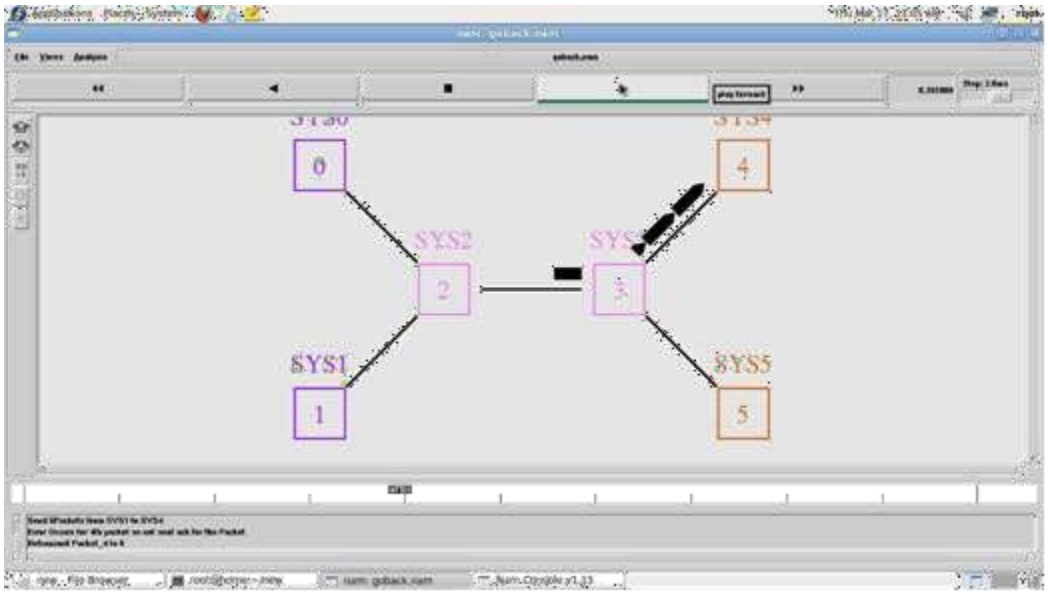
4. The size of the window depends on the protocol designer.
5. For the first frame, the receiving node forms a positive acknowledgement if the frame is received without error.
6. If subsequent frames are received without error (up to window size) cumulative positive acknowledgement is formed.
7. If the subsequent frame is received with error, the cumulative acknowledgment error-free frames are transmitted. If in the same window two frames or more frames are received with error, the second and the subsequent error frames are neglected. Similarly even the frames received without error after the receipt of a frame with error are neglected.
8. The source node retransmits all frames of window from the first error frame.
9. If the frames are errorless in the next transmission and if the acknowledgment is error free, the window slides by the number of error-free frames being transmitted.
10. If the acknowledgment is transmitted with error, all the frames of window at source are retransmitted, and window doesn't slide.
11. This concept of repeating the transmission from the first error frame in the window is called as GOBACKN transmission flow control protocol.

PROGRAM :

```
#send packets one by one
set ns [new Simulator] set n0 [$ns node]
set n1 [$ns node] set n2 [$ns node] set n3 [$ns node] set n4 [$ns node] set n5 [$ns node] $n0
color "purple" $n1 color "purple" $n2 color "violet" $n3 color "violet" $n4 color "chocolate" $n5
color "chocolate" $n0 shape box ;
$n1 shape box ; $n2 shape box ; $n3 shape box ; $n4 shape box ; $n5 shape box ;
$ns at 0.0 "$n0 label SYS0"
$ns at 0.0 "$n1 label SYS1"
$ns at 0.0 "$n2 label SYS2"
$ns at 0.0 "$n3 label SYS3"
$ns at 0.0 "$n4 label SYS4"
 $ns at 0.0 "$n5 label SYS5"
set nf [open goback.nam w] $ns namtrace-all $nf
set f [open goback.tr w] $ns trace-all $f
$ns duplex-link $n0 $n2 1Mb 20ms DropTail $ns duplex-link-op $n0 $n2 orient right-down $ns
queue-limit $n0 $n2 5
$ns duplex-link $n1 $n2 1Mb 20ms DropTail $ns duplex-link-op $n1 $n2 orient right-up $ns
duplex-link $n2 $n3 1Mb 20ms DropTail $ns duplex-link-op $n2 $n3 orient right
$ns duplex-link $n3 $n4 1Mb 20ms DropTail $ns duplex-link-op $n3 $n4 orient right-up $ns
duplex-link $n3 $n5 1Mb 20ms DropTail $ns duplex-link-op $n3 $n5 orient right-down
Agent/TCP set_nam_tracevar_true
set tcp [new Agent/TCP] $tcp set fid 1
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink] $ns attach-agent $n4 $sink $ns connect $tcp $sink
set ftp [new Application/FTP] $ftp attach-agent $tcp
$ns at 0.05 "$ftp start"
$ns at 0.06 "$tcp set windowlnit 6"
$ns at 0.06 "$tcp set maxcwnd 6"
```

```
$ns at 0.25 "$ns queue-limit $n3 $n4 0"
$ns at 0.26 "$ns queue-limit $n3 $n4 10"
$ns at 0.305 "$tcp set windowlnit 4"
$ns at 0.305 "$tcp set maxcwnd 4"

$ns at 0.368 "$ns detach-agent $n1 $tcp ; $ns detach-agent $n4 $sink"
$ns at 1.5 "finish"
$ns at 0.0 "$ns trace-annotate \"Goback N end\""
$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.01\""
$ns at 0.06 "$ns trace-annotate \"Send 6Packets from SYS1 to SYS4\""
$ns at 0.26 "$ns trace-annotate \"Error Occurs for 4th packet so not sent ack for the
Packet\""
$ns at 0.30 "$ns trace-annotate \"Retransmit Packet_4 to 6\""
$ns at 1.0 "$ns
trace-annotate \"FTP
stops\""
proc finish {} {
global ns nf
$ns flush-trace close
$nf
puts "filtering..."
#exec
tclsh../bin/namfilter.
tcl goback.nam
#puts "running
nam..."
exec nam
goback.nam & exit
0
}
$ns run.
```

OUTPUT



RESULT:

Thus the Go back N and Selective Repeat protocols were Simulated and studied