CS3591 NETWORKS LABORATORY

LIST OF EXPERIMENTS

1.Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute. Capture ping and traceroute PDUs using a network protocol analyzer and examine.

2.Writea HTTP webclient program to download a webpage using TCP sockets.

3.Applications using TCP sockets like:

- Echo client and echo server
- Chat

4.Simulation of DNS using UDP sockets.

5.Use a tool like Wireshark to capture packets and examine the packets

6.Write a code simulating ARP/RARP protocols.

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

8.Study of TCP/UDP performance using Simulation tool.

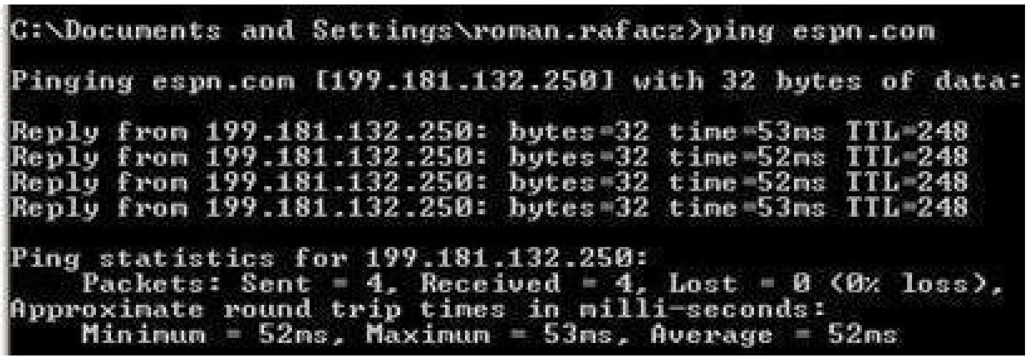
9.Simulation of Distance Vector/Link State Routing algorithm.

10.Simulation of error correction code(likeCRC).

Exp #1 Network UtitilitiesDate:

1. ping

Verifies IP-level connectivity to another TCP/IP computer by sending Internet Control Message Protocol (ICMP) Echo Request messages. The receipt of corresponding Echo Reply messages are displayed, along with round-trip times. Ping is the primary TCP/IP command used to troubleshoot connectivity, reachability, and name resolution.

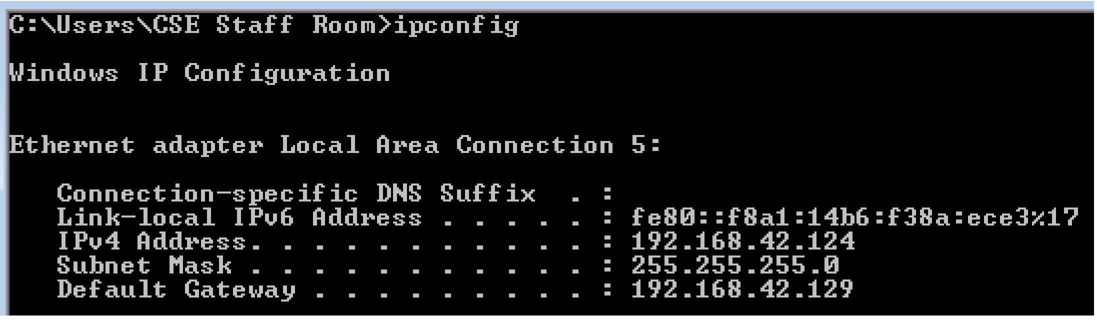


```
C:\Documents and Settings\roman.rafacz>ping espn.com

Pinging espn.com [199.181.132.250] with 32 bytes of data:

Reply from 199.181.132.250: bytes=32 time=53ms TTL=248
Reply from 199.181.132.250: bytes=32 time=52ms TTL=248
Reply from 199.181.132.250: bytes=32 time=52ms TTL=248
Reply from 199.181.132.250: bytes=32 time=53ms TTL=248

Ping statistics for 199.181.132.250:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 52ms, Maximum = 53ms, Average = 52ms
```

To test a TCP/IP configuration, ping the loopback address by typing ping 127.0.0.1
Theresultsshouldtelliftheconnectionwassuccessfulorifthereisanylostpacketsdueto poor network connection or congestion.

2. ifconfig/ipconfig

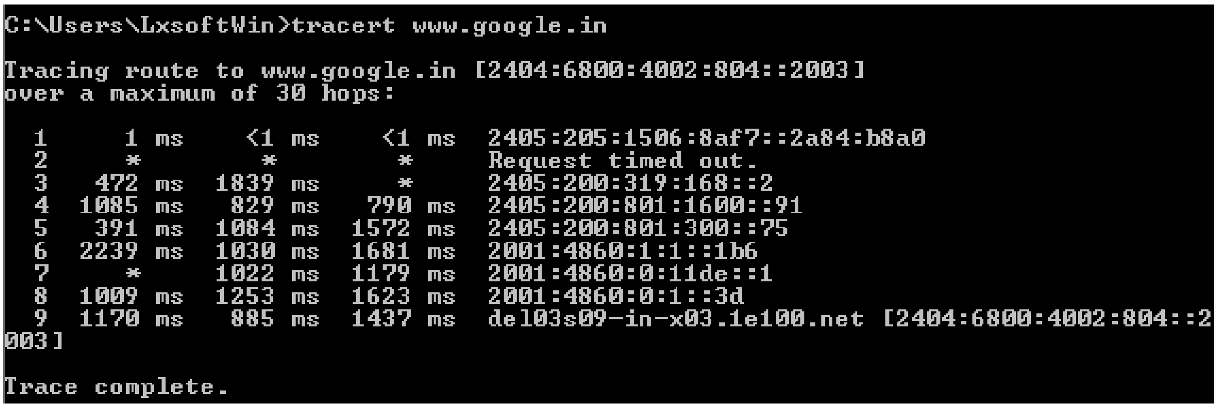
Displays basic current TCP/IP network configuration. It is very useful to troubleshoot networking problems. ipconfig/all is usedtoprovidedetailedinformationsuchasIP address, subnet mask, MAC address, DNS server, DHCP server, default gateway etc. ipconfig/renew is used to renew a DHCP assigned IP address whereas ipconfig/release is used to discard the assigned DHCP IP address.



```
C:\Users\CSE Staff Room>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection 5:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::f8a1:14b6:f38a:ece3%17
   IPv4 Address. . . . . . . . . . . : 192.168.42.124
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.42.129
```

3. traceroutet/tracert

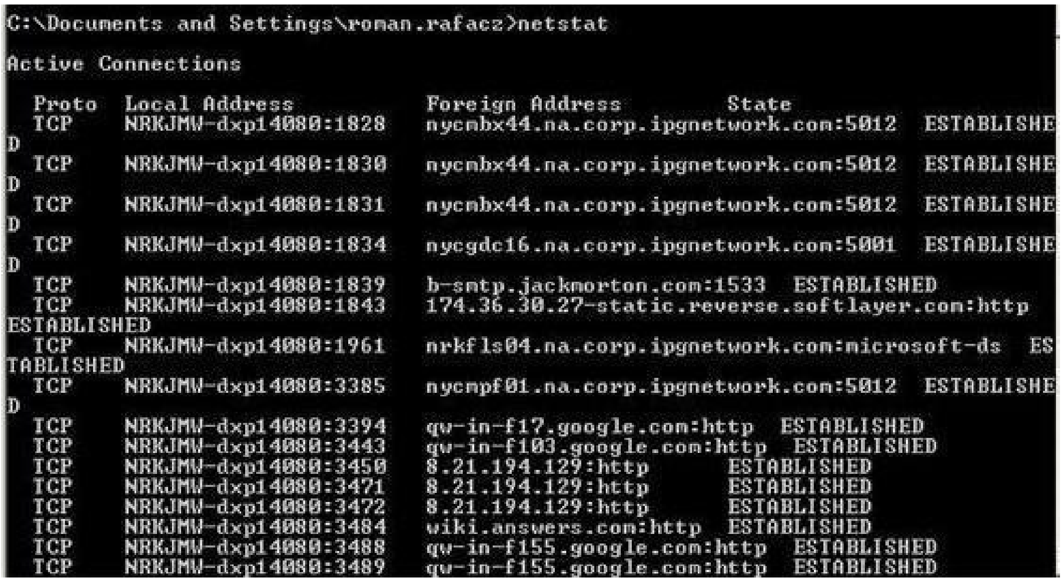
Displays the path taken to a destination by sending ICMP Echo Request messages to the destination with TTL field values. The path displayed is the list of nearestrouterinterfaces taken along each hop in the path between source host and destination.



```
C:\Users\LxsoftWin>tracert www.google.in

Tracing route to www.google.in [2404:6800:4002:804::2003]
over a maximum of 30 hops:

  1     1 ms    <1 ms    <1 ms  2405:205:1506:8af7::2a84:b8a0
  2     *        *        *     Request timed out.
  3   472 ms  1839 ms     *     2405:200:319:168::2
  4  1085 ms   829 ms   790 ms  2405:200:801:1600::91
  5   391 ms  1084 ms  1572 ms  2405:200:801:300::75
  6  2239 ms  1030 ms  1681 ms  2001:4860:1:1::1b6
  7     *     1022 ms  1179 ms  2001:4860:0:11de::1
  8  1009 ms  1253 ms  1623 ms  2001:4860:0:1::3d
  9  1170 ms   885 ms  1437 ms  del03s09-in-x03.1e100.net [2404:6800:4002:804::2
003]

Trace complete.
```

4. netstat

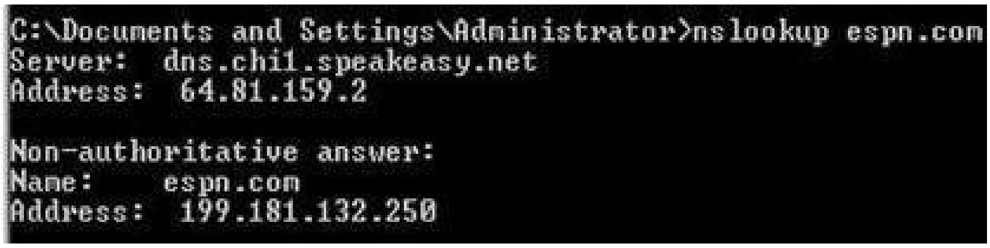
Displays active TCP connections, ports on which the computer is listening, Ethernet statistics, IP routing table, IPv4 statistics and IPv6 statistics. It indicates state of a TCP connection. it's a helpful tool in finding problems and determining the amount oftrafficon the network as a performance measurement.



```
C:\Documents and Settings\roman.rafacz>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    NRKJMW-dxp14080:1828   nycmbx44.na.corp.ipgnetwork.com:5012  ESTABLISHE
D
  TCP    NRKJMW-dxp14080:1830   nycmbx44.na.corp.ipgnetwork.com:5012  ESTABLISHE
D
  TCP    NRKJMW-dxp14080:1831   nycmbx44.na.corp.ipgnetwork.com:5012  ESTABLISHE
D
  TCP    NRKJMW-dxp14080:1834   nycgdc16.na.corp.ipgnetwork.com:5001  ESTABLISHE
D
  TCP    NRKJMW-dxp14080:1839   b-smtp.jackmorton.com:1533  ESTABLISHED
  TCP    NRKJMW-dxp14080:1843   174.36.30.27-static.reverse.softlayer.com:http
ESTABLISHED
  TCP    NRKJMW-dxp14080:1961   nrkfls04.na.corp.ipgnetwork.com:microsoft-ds  ES
TABLISHED
  TCP    NRKJMW-dxp14080:3385   nycmpf01.na.corp.ipgnetwork.com:5012  ESTABLISHE
D
  TCP    NRKJMW-dxp14080:3394   qw-in-f17.google.com:http  ESTABLISHED
  TCP    NRKJMW-dxp14080:3443   qw-in-f103.google.com:http  ESTABLISHED
  TCP    NRKJMW-dxp14080:3450   8.21.194.129:http      ESTABLISHED
  TCP    NRKJMW-dxp14080:3471   8.21.194.129:http      ESTABLISHED
  TCP    NRKJMW-dxp14080:3472   8.21.194.129:http      ESTABLISHED
  TCP    NRKJMW-dxp14080:3484   wiki.answers.com:http  ESTABLISHED
  TCP    NRKJMW-dxp14080:3488   qw-in-f155.google.com:http  ESTABLISHED
  TCP    NRKJMW-dxp14080:3489   qw-in-f155.google.com:http  ESTABLISHED
```

5. nslookup

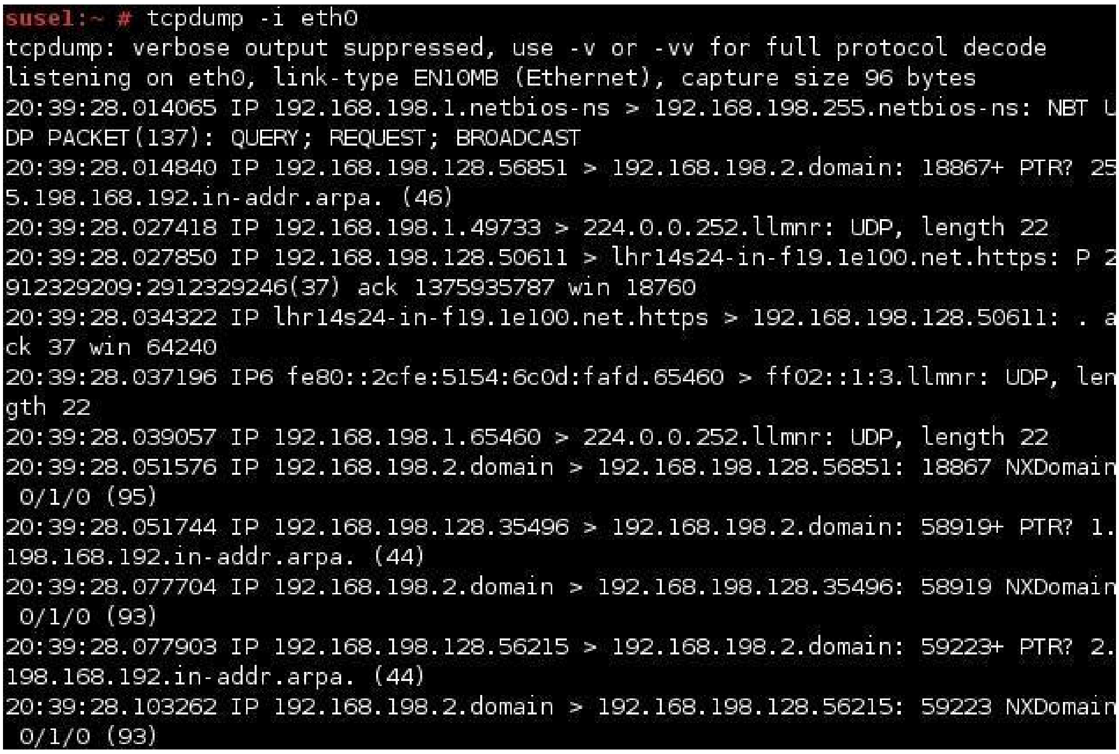
It provides a command-line utility for querying DNS table of a DNS Server. It returns IP address for the given host name.



```
C:\Documents and Settings\Administrator>nslookup espn.com
Server:  dns.chi1.speakeasy.net
Address:  64.81.159.2

Non-authoritative answer:
Name:    espn.com
Address:  199.181.132.250
```

6. tcpdump

tcpdump is a most powerful and widely used command-line packets sniffer or package analyzer tool which is used to capture or filter TCP/IP packets that received or transferred over a network on a specific interface for analysis.



```
susel:~ # tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
20:39:28.014065 IP 192.168.198.1.netbios-ns > 192.168.198.255.netbios-ns: NBT U
DP PACKET(137): QUERY; REQUEST; BROADCAST
20:39:28.014840 IP 192.168.198.128.56851 > 192.168.198.2.domain: 18867+ PTR? 25
5.198.168.192.in-addr.arpa. (46)
20:39:28.027418 IP 192.168.198.1.49733 > 224.0.0.252.llmnr: UDP, length 22
20:39:28.027850 IP 192.168.198.128.50611 > lhr14s24-in-f19.1e100.net.https: P 2
912329209:2912329246(37) ack 1375935787 win 18760
20:39:28.034322 IP lhr14s24-in-f19.1e100.net.https > 192.168.198.128.50611: . a
ck 37 win 64240
20:39:28.037196 IP6 fe80::2cfe:5154:6c0d:fafd.65460 > ff02::1:3.llmnr: UDP, len
gth 22
20:39:28.039057 IP 192.168.198.1.65460 > 224.0.0.252.llmnr: UDP, length 22
20:39:28.051576 IP 192.168.198.2.domain > 192.168.198.128.56851: 18867 NXDomain
 0/1/0 (95)
20:39:28.051744 IP 192.168.198.128.35496 > 192.168.198.2.domain: 58919+ PTR? 1.
198.168.192.in-addr.arpa. (44)
20:39:28.077704 IP 192.168.198.2.domain > 192.168.198.128.35496: 58919 NXDomain
 0/1/0 (93)
20:39:28.077903 IP 192.168.198.128.56215 > 192.168.198.2.domain: 59223+ PTR? 2.
198.168.192.in-addr.arpa. (44)
20:39:28.103262 IP 192.168.198.2.domain > 192.168.198.128.56215: 59223 NXDomain
 0/1/0 (93)
```

Result

Thus TCP/IP network command utilities were executed.

Exp#1a **Ping Command**

Date:

Aim

TotestthecommunicationbetweenhostsatIPlevelusingPingcommand.

Algorithm

1. GetIPaddress/domainnamefromtheuser.
2. Createaruntimeenvironment.
3. Executepingcommandwithgiveninputasparameter.
4. Analysetheoutput
5. Stop

Program

//PingServer.java:SimplePingProgram

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

class PingServer
{
public static void main(String args[])
{

try
{


String str;
System.out.println("EnterIPaddress/domainname:");
BufferedReader buf1=new BufferedReader(new
InputStreamReader(System.in));
 String ip = buf1.readLine();
Runtime rt=Runtime.getRuntime();
 Process p=rt.exec("ping"+ip);
InputStream in = p.getInputStream();
BufferedReader buf2=new BufferedReader(new
InputStreamReader(in));
while((str=buf2.readLine()) != null)
{

System.out.println(""+str);
}
}
catch(Exception e)
{
System.out.println(e.getMessage());
}
}
}
```

Output

Result

Thus using Ping command,connective and communicative status is determined.

Exp#1b **Traceroute Command**

Date:

Aim

TotracethepathtraversedbyapacketfromhosttodestinationusingTraceroute command.

Algorithm

1. Getdomainnamefromtheuser.
2. Createaruntimeenvironment.
3. Executetraceroutecommandwithgiveninputasparameter.
4. Analysetheoutput
5. Stop

Program

//TraceServer.java:TracerouteProgram

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

class TraceServer
{
public static void main(String args[])
{

try
{
String str;
System.out.println("Enter domain name : ");
 BufferedReader buf1=new BufferedReader(new
InputStreamReader(System.in));
 String ip = buf1.readLine();
 Runtime rt=Runtime.getRuntime();
Process p = rt.exec("tracert " + ip);
 InputStream in = p.getInputStream();
BufferedReader buf2=new BufferedReader(new
InputStreamReader(in)); while((str=buf2.readLine()) !=
null)
{

System.out.println(""+str);
}
}
catch(Exception e)
{
System.out.println(e.getMessage());
}
}
}
```

Output

Result

Thus using traceroute command,path traversed by the packet is determined.

Exp#2 **WebPage Download**

Date:

Aim

TodownloadawebpageusingjavaURLmethod.

Algorithm

1. GetURLfromtheuser.
2. Createafileinstancetostorethedownloadedpage.
3. DownloadthepageusingjavaURLmethods.
4. Viewthedownloadpage
5. Stop

Program

```
//JavafiletodownloadaWebpage–DownloadPage.java import
import java.io.*;
import java.net.*;

class MyDownload
{
public void Download() throws Exception
{

try
{


String WebPage,MyPage;
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

 System.out.print("EntertheURL:");
WebPage=br.readLine();
URL url = new URL(WebPage);
System.out.println("Enterfilenametostore:");
MyPage = br.readLine();
File Out=new File(MyPage);
InputStream in=url.openStream();
FileOutputStream FOS=new FileOutputStream(Out);
//Dowloadthe page
byte buf[] = new byte[1024];
int i,len;
while((len=in.read(buf))>0)
{

for(i=0;i<len;i++)
{
FOS.write((char)buf[i]);
}
}

//Closethestreams in.close();
FOS.close();

}
catch(MalformedURLException M)
{
System.out.println(M);
}

catch(Exception E)
{
System.out.println(E);
}
}
}

class DownloadPage
{
public static void main(String args[]) throws Exception
{
String Choice;
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
MyDownload MDP = new MyDownload();
MDP.Download();
System.out.println("Downloadcomplete.Viewthefile");
}
}
```

Output

Result

Thus using java URL methods, a webpage is downloaded.

TCPSockets

Asocketisanendpoint of a two-way communication link between two programs running on the network. Socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent. User-level process/services generally use port number value > 1024. TCP provides a reliable, point-to-point communication channel that client-server application on the Internet use to communicate with each other. Examples are FTP and Telnet.

To communicate over TCP, a client program and aserverprogramestablishaconnectionto one another. Each program binds a socket to its end of the connection. A server runs on a specific computer and hasasocketthatisboundtoaspecificportnumber.Theserverwaits, listening to the socket for a connection request from the client.

On the client-side, the client knows the hostname of the machine on which the server is running and the port numberonwhichtheserverislistening.Tomakeaconnectionrequest, the client tries to make contact with the server on the server's machine and port.Theclient also needs to identify itself to the server so it binds to a local port number that it will use during this connection.

If everything goes well, the server accepts theconnection.Uponacceptance,theservergetsa newsocketboundtothesamelocalportandalsohasitsremoteendpointsettotheaddress and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.

On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server. The client and server can now communicate by writing toorreadingthroughI/Ostreamsfromtheirsocketsandeventually close it.

Thetwokeyclassesfromthejava.netpackageusedincreationofserverandclient programs are:

- ServerSocket
- Socket

Exp#3a **TCPEchoServer/Client**

Date:

Aim

ToimplementechoserverandclientinjavausingTCPsockets.

Algorithm

<u>Server</u>

1. Createaserversocket.
2. Waitforclienttobeconnected.
3. Readtextfromtheclient
4. Echothetextbacktotheclient.
5. Repeatsteps4-5until‘bye’or‘null’is read.
6. ClosetheI/Ostreams
7. Closetheserversocket
8. Stop

<u>Client</u>

1. Createasocketandestablishconnection with the server
2. Getinputfrom user.
3. Ifequaltobyeornull,thengotostep7.
4. Sendtexttotheserver.
5. Displaythetextechoedbytheserver
6. Repeatsteps2-4
7. ClosetheI/Ostreams
8. Closetheclientsocket
9. Stop

Program

```
//TCPEchoServer--tcpechoserver.java
import java.net.*;
import java.io.*;


public class tcpechoserver
{
public static void main(String[] arg) throws IOException
{
ServerSocket sock = null;
 BufferedReader fromClient=null;
OutputStreamWriter toClient = null;
 Socket client = null;
try
{
sock = new ServerSocket(4000);
System.out.println("ServerReady");
client = sock.accept();
 System.out.println("Client Connected");
fromClient = new BufferedReader(new
InputStreamReader(client.getInputStream()));
toClient=new
OutputStreamWriter(client.getOutputStream());
String line;
while (true)
{
```

```
line=fromClient.readLine();
if ( (line == null) || line.equals("bye")) break;
System.out.println("Client["+line+"]");
 toClient.write("Server [ "+ line +" ]\n"); toClient.flush();
}
fromClient.close();
toClient.close();
client.close();
sock.close();
System.out.println("ClientDisconnected");
}
catch(IOException ioe)
{
System.err.println(ioe);
}
}
}
```

```
//TCPEchoClient--tcpechoclient.java
import java.net.*;
import java.io.*;

public class tcpechoclient
{
public static void main(String[]args) throws IOException
{
BufferedReader fromServer=null,fromUser=null; PrintWriter toServer = null;
Socket sock=null;
 try
{
if(args.length== 0)
sock=new Socket(InetAddress.getLocalHost(),4000);
else
sock=new Socket(InetAddress.getByName(args[0]),4000);
fromServer=new BufferedReader(new InputStreamReader(sock.getInputStream()));
 fromUser = new BufferedReader(new InputStreamReader(System.in));
toServer = new PrintWriter(sock.getOutputStream(),true);
String Usrmsg,Srvmsg;
 System.out.println("Type\"bye\"to quit");
 while (true)
{

System.out.println("Enter msg to server:"); Usrmsg = fromUser.readLine();
if(Usrmsg == null || Usrmsg.equals("bye"))
{
toServer.println("bye");
 break;
}
else
toServer.println(Usrmsg);
 Srvmsg=fromServer.readLine();
System.out.println(Srvmsg);
}
fromUser.close();
fromServer.close();
toServer.close();
sock.close();
}
catch(IOException ioe)
{
System.err.println(ioe);
}

}
}
```

Output:

Result

Thus data from client to server is echoed back to the client to check reliability/noise level of the channel.

Exp#3b **TCPChatServer/Client**

Date:

Aim

ToimplementachatserverandclientinjavausingTCPsockets.

Algorithm

<u>Server</u>

1.Createaserver socket
2. Waitforclienttobeconnected.
3. ReadClient'smessageanddisplay it
4. Getamessagefromuserandsendittoclient
5. Repeatsteps3-4untiltheclientsends"end"
6. Closeallstreams
7. Closetheserverandclientsocket
8. Stop

<u>Client</u>

1.Createaclientsocketandestablish connection with the server
2. Getamessagefromuserandsendittoserver
3. Readserver'sresponseanddisplayit
4. Repeatsteps2-3untilchatisterminatedwith "end" message
5. Closeallinput/outputstreams
6. Closetheclientsocket
7. Stop

Program

//TCPChatServer--tcpchatserver.java

```
import java.io.*;

import java.net.*;


class tcpchatserver {

  public static void main(String[] args) {

    PrintWriter toClient = null;

    BufferedReader fromUser = null, fromClient = null;


    try {

      ServerSocket srv = new ServerSocket(5555);

      System.out.println("\nServer started\n");


      Socket clt = srv.accept();

      System.out.println("Client connected");


      toClient = new PrintWriter(new
BufferedWriter(new
OutputStreamWriter(clt.getOutputStream())), true);

      fromClient = new BufferedReader(new
InputStreamReader(clt.getInputStream()));

      fromUser = new BufferedReader(new
InputStreamReader(System.in));


      String cltMsg, srvMsg;


      while (true) {

        cltMsg = fromClient.readLine();

        if (cltMsg == null ||
cltMsg.equalsIgnoreCase("end")) {

          System.out.println("Client disconnected");

          break;

        }
```

```
                System.out.println("\nClient >>> " + cltMsg);
                System.out.print("Message to Client: ");
                srvMsg = fromUser.readLine();
                toClient.println(srvMsg);
            }

            // Close resources
            fromClient.close();
            toClient.close();
            fromUser.close();
            clt.close();
            srv.close();

        } catch (Exception e) {
 System.out.println("Error: " + e.getMessage());
        }
    }
}
```

```
//TCPChatClient--tcpchatclient.java
import java.io.*;
import java.net.*;

class tcpchatclient {
    public static void main(String args[]) throws Exception {
        Socket clt;
        PrintWriter toServer;
        BufferedReader fromUser, fromServer;

        try {
            // Check for valid arguments
            if (args.length > 1) {
                System.out.println("Usage: java TcpChatClient
[host_ip_address]");
                System.exit(-1);
            }

            // Connect to server
            if (args.length == 0)
                clt = new Socket(InetAddress.getLocalHost(),
5555);
            else
                clt = new
Socket(InetAddress.getByName(args[0]), 5555);

            toServer = new PrintWriter(new
BufferedWriter(new
OutputStreamWriter(clt.getOutputStream())), true);
            fromServer = new BufferedReader(new
InputStreamReader(clt.getInputStream()));
            fromUser = new BufferedReader(new
InputStreamReader(System.in));

            String cltMsg, srvMsg;
            System.out.println("Type \"end\" to quit");

            while (true) {
                System.out.print("\nMessage to Server: ");
                cltMsg = fromUser.readLine();
                toServer.println(cltMsg);

                if (cltMsg.equalsIgnoreCase("end")) {
                    break;
                }

                srvMsg = fromServer.readLine();
                if (srvMsg == null) {
                    System.out.println("Server disconnected.");
                    break;
                }
               System.out.println("Client <<< " + srvMsg);
            }

            // Clean up resources
            fromUser.close();
            fromServer.close();
            toServer.close();
            clt.close();

        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output

Result

Thus both the client and server exchanged at a using TCP socket programming.

UDPSockets

TCP guarantees the delivery of packets andpreservestheirorderondestination.Sometimes these features are not required, since they do not come withoutperformancecosts,itwould be better to use a lighter transport protocol such asUDP(UserDatagramProtocol).UDPis an unreliable protocol, i.e., it does not include software mechanisms for retrying on transmissionfailuresordatacorruption(unlikeTCP),andhasrestrictionsonmessagelength (< 65536 bytes). Examples are NFS, DNS, SNMP, Clock Server, Ping, VoIP, online games etc.

Unlike TCP there is no concept of a connection, UDP is a protocol that sends independent packets of data, called *datagrams*, from one computer to another with no guarantees about arrival and sequencing. No packet has any knowledgeoftheprecedingorfollowingpacket. The recipient does not acknowledge packets, thereby the senderdoesnotknowwhetherthe transmission was successful. The format of datagram packet is

Message	Length	Host	ServerPort

A program can use a single UDP socket to communicate with more than one hostandport number, but it is convenient for most UDPclientprogramstomaintainthefictionthatthere is a connection, by keeping a local record of each server host and port number. A UDP server does not have to listen for and accept client connections, and a UDP client neednot connect to a server.

Javasupportsdatagramcommunicationthroughthefollowingclasses:

- DatagramPacket
- DatagramSocket

TheDatagramPacketobjectisthedatacontainer,whiletheDatagramSocketisthemechanismused to send or receive the DatagramPackets.

Exp#4 **UDPDNSServer/Client**

Date:

Aim

ToimplementaDNSserverandclientinjavausingUDPsockets.

Algorithm

Server

1.Defineaarrayofhostsanditscorresponding IP address in another array
2. Createadatagramsocket
3. Createadatagrampackettoreceiveclient request
4. Readthedomainnamefromclienttobe resolved
5. Lookupthehostarrayforthedomain name
6. Iffoundthenretrievecorrespondingaddress
7. Constructadatagrampackettosendresponse back to the client
8. Repeatsteps3-7toresolvefurtherrequests from clients
9. Closetheserversocket
10. Stop

Client

1.Createadatagramsocket
2. Getdomainnamefrom user
3. Constructadatagrampackettosenddomain name to the server
4. Createadatagrampackettoreceiveserver message
5. IfitcontainsIPaddressthendisplayit,else display "Domain does not exist"
6. Closetheclientsocket
7. Stop

Program

//UDPDNSServer--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].equalsIgnoreCase(str)) {
            return i;
         }
      }
      return -1;
   }

   public static void main(String arg[]) throws IOException
{
      String[] hosts = {"yahoo.com", "gmail.com",
"cricinfo.com", "facebook.com"};
      String[] ip = {"68.180.206.184", "209.85.148.19",
"80.168.92.140", "69.63.189.16"};

      System.out.println("DNS Server running on UDP port
1362");
      System.out.println("Press Ctrl+C to quit");

      DatagramSocket serverSocket = new
DatagramSocket(1362);

      byte[] sendData = new byte[1024];
      byte[] receiveData = new byte[1024];

      while (true) {
         try {
            // Receive request
            DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);
            serverSocket.receive(receivePacket);

            String receivedHost = new
String(receivePacket.getData(), 0,
receivePacket.getLength()).trim();
            InetAddress clientAddress =
receivePacket.getAddress();
            int clientPort = receivePacket.getPort();

            System.out.println("Request for host: " +
receivedHost);

            String response;
            int index = indexOf(hosts, receivedHost);
            if (index != -1) {
               response = ip[index];
            } else {
               response = "Host Not Found";
            }

            // Send response
            sendData = response.getBytes();
            DatagramPacket sendPacket = new
DatagramPacket(sendData, sendData.length, clientAddress,
clientPort);
            serverSocket.send(sendPacket);
         } catch (IOException e) {
            System.err.println("Error: " + e.getMessage());
         }
      }
   }
}
```

//UDPDNSClient--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 port = 1362;

      System.out.print("Enter the hostname: ");
      String sentence = br.readLine();
      sendData = sentence.getBytes();

      DatagramPacket sendPacket = new
DatagramPacket(sendData, sendData.length, ipAddress,
port);
      clientSocket.send(sendPacket);

      DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);
      clientSocket.receive(receivePacket);

      String response = new String(receivePacket.getData(),
0, receivePacket.getLength());
      System.out.println("IP Address: " + response);

      clientSocket.close();
   }
}
```

Output

Result

Thus domain name requests by the client are resolved into their respective logical address using lookup method.

Ex.No.5 USE A TOOL LIKE WIRESHARK TO CAPTURE PACKETS AND EXAMINE THE PACKETS

Aim

To capture, filter, and analyze network packets using the Wireshark application.

Requirements

1. Software: Wireshark (latest version)
2. Hardware: Ethernet-enabled computer with internet access

Theory

Wireshark is a widely-used network protocol analyzer that lets you capture and interactively browse the traffic running on a computer network. It is used for network troubleshooting, analysis, and communication protocol development.

Steps to Perform the Experiment

1. Install Wireshark

- Download Wireshark from https://www.wireshark.org.
- Install it on your system with default options.

2. Launch Wireshark

- Open the Wireshark application.
- You will see a list of available network interfaces on your computer.

3. Start Packet Capture

1. Select the Ethernet Interface:
   - Choose the network interface corresponding to your Ethernet connection.
   - It may appear as Ethernet0, eth0, or similar depending on your system.
2. Start Capturing:
   - Click on the Start button (green shark fin icon) to begin packet capture.

4. Perform the Network Activity

- While capturing, perform a network activity, such as:
  - Running your DNS program or socket programs.
  - Browsing a website or downloading a file.

5. Stop Packet Capture

- After completing the activity, click on the Stop button (red square icon) to stop capturing packets.

Steps to Analyze the Captured Packets

1. Apply Filters

- Use the Filter bar at the top of the Wireshark window to filter specific types of packets.
- Common filters:
  - DNS Traffic: udp.port == 53
  - HTTP Traffic: http
  - TCP Traffic: tcp
  - UDP Traffic: udp

    - Filter by IP Address: ip.addr == <your-IP-address>

2. Examine the Packets

- Click on a packet in the Packet List Pane to view its details.
- The middle pane shows protocol details, and the bottom pane displays the raw packet data (hexadecimal and ASCII formats).

3. Look for Specific Protocol Details

- DNS Analysis:
    - Look for DNS queries and responses with domain names and IP addresses.
- HTTP Analysis:
    - Analyze GET and POST requests and their server responses.
- TCP Analysis:
    - Observe source and destination ports, flags, and handshake details.

Steps to Export Packet Data

- Go to File > Save As to save the captured packets for later analysis.

Sample Observations

DNS Traffic

- Query: Standard query A www.google.com
- Response: 142.250.74.14

HTTP Traffic

- Request: GET /index.html
- Response: HTTP/1.1 200 OK

TCP Traffic

- Source Port: 12345
- Destination Port: 80
- Flags: SYN, ACK

Result

The Wireshark tool was successfully used to capture, filter, and analyze network packets for DNS, HTTP, TCP, and UDP protocols.

AddressResolution

- A host or router to send an IP datagram,needs to know *both* the logical and physical address of the destination.

Address Resolution Protocol(ARP)

- *Address Resolution Protocol*(ARP)enables a source host to know the physical address of another node when the logical address is known.
- ARP relies on *broadcast* support from physical networks such as ethernet,token ring, etc.
- ARP is a request/reply protocol
    - ARP Request packet is broadcasted by the source host
    - ARP Reply packet is sent by destination host to source host
- ARP enables each host on a network build up a *mapping* table between IPaddress and physical address.

Reverse Address Resolution Protocol(RARP)

- A*diskless* workstation booted from its ROM or newly booted workstation doesnot know its IP address as it is assigned by the network administrator.
- *Reverse Address Resolution protocol*(RARP)allows a host to find its IPaddress using RARP *request* (broadcasted) and *RARP reply*.
- RARP is *replaced* by protocols suchas BOOTP and DHCP.

Exp#6a **ARP Client/Server**

Date:

Aim

To know the physical address of a host when its logical address is known using ARP protocol.

Algorithm

Target/Server

1. Create a server socket.
2. Accept client connection.
3. Read IPaddress from the client request
4. Check its configuration file and compare with its logical address.
5. If there is a match,send the host physical address.
6. Stop

Client

1. Create a socket.
2. Send IP address to the target machine
3. Receive target's response
4. If It is a MAC address then display it and goto step 6
5. Display"Host not found"
6. Stop

Program

```
// ARP Server –ArpServer.java
import java.io.*;
import java.net.*;

class ArpServer {
    public static void main(String args[]) throws IOException {
        try {
            ServerSocket soc = new ServerSocket(2500);
            System.out.println("Server started...");

            Socket client = soc.accept(); // Wait for client
            System.out.println("Client connected...");

            // Read the IP address sent by the client
            BufferedReader br = new BufferedReader(new InputStreamReader(client.getInputStream()));
            String ipaddr = br.readLine();
            System.out.println("Requested IP: " + ipaddr);

            // Prepare to write response to client
            PrintStream ps = new PrintStream(client.getOutputStream());

            // Run ifconfig to get network details
            Runtime r = Runtime.getRuntime();
            Process p = r.exec("ifconfig eth0");

            BufferedReader pin = new BufferedReader(new InputStreamReader(p.getInputStream()));
            String str;
            String haddr = "";
            int flag = 0;

            while ((str = pin.readLine()) != null) {
                System.out.println(str); // Debug: print each line

                // Look for HWaddr (MAC address)
                if (str.contains("HWaddr")) {
                    int index = str.indexOf("HWaddr");
                    haddr = str.substring(index + 7).trim(); // Extract MAC after "HWaddr"
                }

                // Check if IP address is in this line
                if (str.contains(ipaddr)) {
                    flag = 1;
                }
            }

            // Send result back to client
            if (flag == 1) {
                ps.println("MAC Address: " + haddr);
            } else {
                ps.println("IP address not found.");
            }

            // Close resources
            ps.close();
            br.close();
            pin.close();
            client.close();
            soc.close();
        } catch (IOException io) {
            System.err.println("Exception: " + io.toString());
        }
    }
}

// ARP Client -- ArpClient.java
import java.io.*;
import java.net.*;

class ArpClient {
    public static void main(String args[]) {
        try {
            // Connect to the server on localhost and port 2500
            Socket client = new Socket("localhost", 2500);

            // Input reader from keyboard (user input)
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

            // Output stream to send data to the server
            PrintStream ps = new PrintStream(client.getOutputStream());

            // Input stream to receive response from the server
            BufferedReader sin = new BufferedReader(new InputStreamReader(client.getInputStream()));

            // Prompt user to enter IP address
            System.out.print("Enter the IP address: ");
            String ipaddr = br.readLine();

            // Send IP address to server
            ps.println(ipaddr);

            // Read MAC address from server
            String haddr = sin.readLine();

            // Check and display result
            if (haddr == null || haddr.isEmpty()) {
                System.out.println("Host does not exist or MAC address not found.");
            } else {
                System.out.println("Physical Address: " + haddr);
            }

            // Close connections
            ps.close();
            br.close();
            client.close();
        } catch (IOException io) {
            System.err.println("Exception: " + io.toString());
        }
    }
}
```

Output:

Result

Thus using ARP protocol,server’sMAC address is obtained

Exp#6b **RARP Client/Server**

Date:

Aim

To know the logical address of a host when its physical address is known using RARP protocol.

Algorithm

Target/Server

1. Create a server socket.
2. Accept client connection.
3. Read MAC address from the client request
4. Check its configuration file and compare with its physical address.
5. If there is a match,send the host logical address.
6. Stop

Client

1. Create a socket.
2. Send physical address to the target machine
3. Receive target's response
4. If it is a Ip address then display it and goto step6
5. Display"Host not found"
6. Stop

Program

```
//RARPServer--RarpServer.java
import java.io.*;
import java.net.*;

class RarpServer {
    public static void main(String args[]) throws IOException {
        try {
            ServerSocket soc = new ServerSocket(2500);
            System.out.println("Server started, waiting for client...");

            Socket client = soc.accept();
            System.out.println("Client connected.");

            // Stream to receive MAC address from client
            BufferedReader br = new BufferedReader(new InputStreamReader(client.getInputStream()));

            // Stream to send IP address back to client
            PrintStream ps = new PrintStream(client.getOutputStream());

            String haddr = br.readLine(); // MAC address received from client
            String str;
            String ipaddr = "";
            int flag = 0;

            // Execute ifconfig to get network details
            Runtime r = Runtime.getRuntime();
            Process p = r.exec("ifconfig eth0"); // or use "ip addr show eth0" for modern systems

            BufferedReader pin = new BufferedReader(new InputStreamReader(p.getInputStream()));

            while ((str = pin.readLine()) != null) {
                System.out.println(str); // Debugging

                // Check if MAC address is found
                if (str.contains(haddr)) {
                    flag = 1;
                }

                // Extract IP address if line contains "inet"
                if (str.toLowerCase().contains("inet addr") || str.contains("inet ")) {
                    // Try extracting IP address
                    String[] parts = str.trim().split("\\s+");
                    for (String part : parts) {
                        if (part.contains("addr:")) {
                            ipaddr = part.substring(part.indexOf("addr:") + 5);
                            break;
                        } else if (part.matches("\\d+\\.\\d+\\.\\d+\\.\\d+")) {
                            ipaddr = part;
                            break;
                        }
                    }
                }
            }
            if (flag == 1 && !ipaddr.isEmpty()) {
                ps.println("IP Address: " + ipaddr);
            } else {
                ps.println("IP address not found for given MAC.");
            }

            // Cleanup
            ps.close();
            br.close();
            pin.close();
            client.close();
            soc.close();

        } catch (IOException io) {
            System.err.println("Exception: " + io.toString());
        }
    }
}

//RARPClient--RarpClient.java
import java.io.*;
import java.net.*;

class RarpClient {
    public static void main(String args[]) {
        try {
            // Connect to the RARP server on localhost at port 2500
            Socket client = new Socket("localhost", 2500);

            // Reader to get MAC address from the user
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

            // Writer to send MAC address to the server
            PrintStream ps = new PrintStream(client.getOutputStream());

            // Reader to receive IP address from the server
            BufferedReader sin = new BufferedReader(new InputStreamReader(client.getInputStream()));

            // Prompt user for MAC address
            System.out.print("Enter the physical (MAC) address: ");
            String haddr = br.readLine();

            // Send MAC address to the server
            ps.println(haddr);

            // Receive response from the server
            String ipaddr = sin.readLine();

            // Output result
            if (ipaddr == null || ipaddr.isEmpty()) {
                System.out.println("Host does not exist or IP address not found.");
            } else {
                System.out.println("Logical Address (IP): " + ipaddr);
            }
            // Clean up
            ps.close();
            br.close();
            sin.close();
            client.close();
        } catch (IOException io) {
            System.err.println("Exception: " + io.toString());
        }
    }
}
```

Output:

Result

Thus using RARP protocol, IP address of the server is obtained.

Exp#7 **NS2 SIMULATION**

Date:

A simulator is a device,software or system which behaves or operates like a given system when provided with a set of controlled inputs. The need for simulators is:

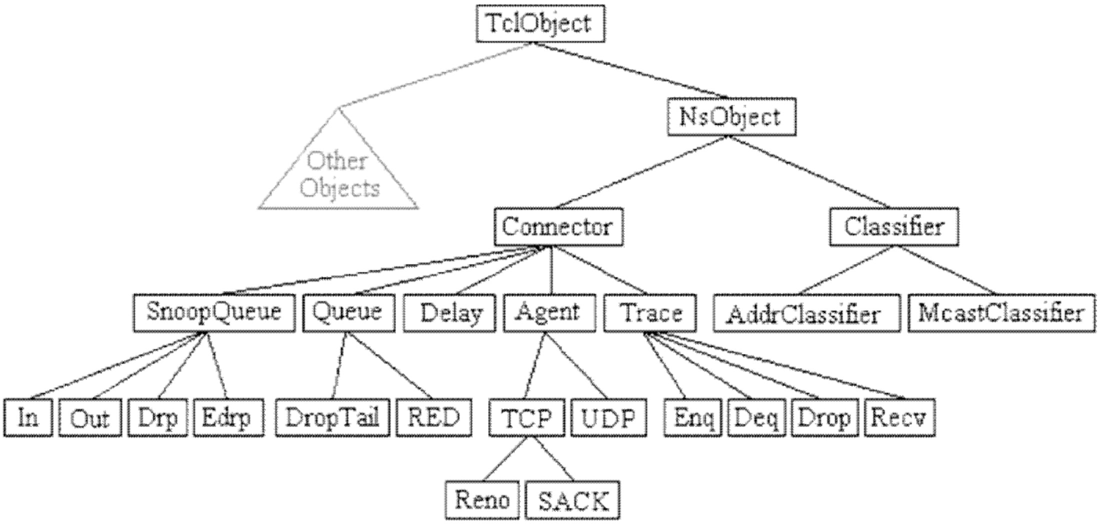
- Provide users with practical feedback such as accuracy,efficiency,cost,etc.,when designing real world systems.
- Permit system designers to study at several different levels of abstraction
- Simulation can give results that are not experimentally measurable with our current level of technology.
- Simulations take the building/rebuilding phase out of the loop by using the model already created in the design phase.
- Effective means for teaching or demonstrating concepts to students.
- A few popular network simulators areNS-2,OPNET,GLOMOSIM,etc.

Network SimulatorNS2

NS2 is anobject-oriented,discrete event driven network simulator developed at UCBerkley written in C++ and OTcl (Object-oriented Tool Command Language). NS is useful for simulating local and wide area networks. NS2 is an open-source simulation tool that primarily runs on Linux (cygwin for Windows). The features of NS2 are:

- Is a discrete event simulator for networking research
- Works at packet level.
- Provide support to simulate bunch of protocols likeTCP,UDP,FTP,etc.
- Simulate wired and wireless network.
- Is a standard experiment environment in research community.

Class Hierarchy



Network Animator(NAM)

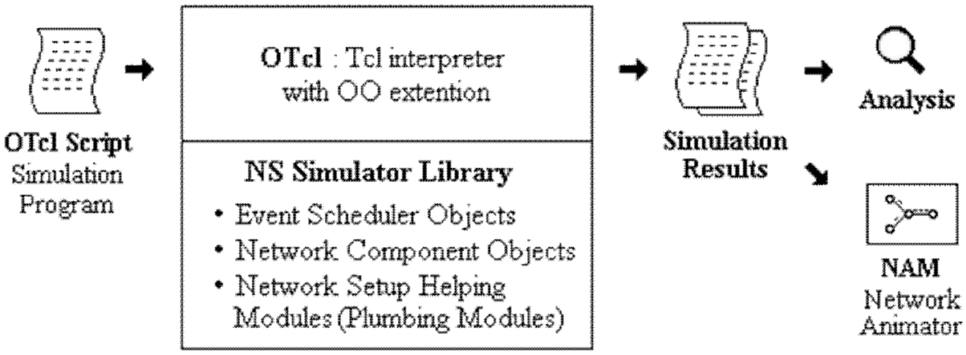
NS together with NAM forms a very powerful set of tools for teaching networking concepts.

With NAM protocols can be visualized as animations. The NAM graphical editor is the latest addition to NAM.With this *editor*,one can create their network topology and simulate various protocols and traffic sources by dragging the mouse.

Create	Visualize
▪ Terrestrial ,satellite eand wireless network with various routing algorithm (DV, LS, PIM, DSR). ▪ Traffic sources like web,ftp,telnet,cbr, and stochastic traffic. ▪ Failures, including deterministic, probabilistic loss, link failure, etc. ▪ Various queuing disciplines(drop-tail, RED, FQ, SFQ, etc.) and QoS	▪ Packet flow, queue build-up and packet drops. ▪ Protocol behavior: TCP slow start, self-clocking, congestion control, fast retransmit and recovery. ▪ Node movement in wirelesss networks. ▪ Annotations to highlight important events. ▪ Protocol state(e.g.,TCPcwnd).

NS2 Execution

The overall simulation procedure in NS is shown below.NS is composed of OTcl Script and Interpreter. NS simulation results can be observed through graphs by analyzing the trace file or viewing animations with NAM.


OTcl Script
Simulation Program
OTcl : Tcl interpreter with OO extention
NS Simulator Library
• Event Scheduler Objects
• Network Component Objects
• Network Setup Helping Modules (Plumbing Modules)
Simulation Results
Analysis
NAM
Network Animator

$ns *filename*.tcl

NS2 Program Elements

Event Scheduler

a Creating event scheduler

Set ns[new Simulator]

b Schedule events

$ns at *time* "*event*"

c Start scheduler

$ns run

Creating Network

a Create set of
Nodes set n0 [$ns
node] set n1 [$ns
node]

b Create links and queuing
$ns duplex-link $n0 $n1 *bandwidth delay*
queue_type Bandwidth is generally in MB
Delay is generally in ms
Queue type is either Drop Tail,RED,CBQ,FQ,SFQ,*etc*
$nsduplex-link$n0$n21Mb10msDropTail

c Layout
$nsduplex-link-op$n0$n2orient*position*
where position is either right, right-up, right-down, left, left- up, left-down, up, down

d Marking flows
$ns color1Blue
$ns color2Red
$udp0 setclass_1
$udp1 setclass_2

Tracing

a NAM Trace all links(must succeed scheduler creation)
Setnf [open out.namw]
$ns nam trace-all $nf

b Trace all links(must succeed scheduler creation)
Set tf[open out.trw]
$ns trace-all $tf
Trace file ouput format
event,time,from_node,to_node,pkttype,pktsize,flags,fid,src_addr,dst_addr, seq_num, pkt_id
where events are r*received,+enqueued,-dequeued,*d*dropped*

c Tracing specific links
$nstrace-queue$n0$n1
$nsnamtrace-queue$n0$n1

Connection

a UDP
set udp [new Agent/UDP] set
null [new Agent/Null]
$ns attach-agent$n0$udp0
$ns attach-agent$n1$null
$ns connect$udp0$null

b TCP
settcp0[new Agent/TCP/Full Tcp]
$tcp0 setwindow_30

```
$tcp0 setsegsize_536
$nsattach-agent$n0$tcp0
setsink0[new Agent/TCP/Full Tcp]
$ns attach-agent$n5$sink0
$sink0 listen
$ns connect$tcp0$sink0
```

Traffic Generation

a UDP

```
Set src[new Application/Traffic/type]
$src attach-agent$udp0
```

Where type is either CBR,Exponential,Pareto

b TCP

```
Set ftp[new Application/FTP]
$ftp attach-agent $tcp
Set telnet[new Application/Telnet]
$telnet attach-agent $tcp
```

Finish procedure

a Flush NS tracing,Close tracing files and execute any post-analysis programs (display results, run NAM, etc)

```
proc finish {} { global
    ns nf
    $ns
    flush-trace close
    $nf
    exec nam out.nam & exit
    0
}
```

Result

Thus simulator NS2 and its basic commands was studied.

Exp#8a **Study of UDP Performance**

Date:

Aim

To study the performance of UDP by simulating as implementation of network

Algorithm

1. Create a simulator object
2. Define different color for data flows
3. Trace all events in a nam file.
4. Create four nodes *n0*,*n1*,*n2* and *n3*
5. Describe their layout topology
6. Specify the link capacity between nodes
7. Monitor queue on the link *n2* to *n3* vertically 90°
8. Create a UDP agent s*udp0*,*udp1*and attach it to nodes *n0* and *n1* respectively
9. Create a CBR traffic *cbr0*,*cbr1* and attach it to *udp0* and *udp1* respectively
10. Create a traffic sink and attach it to node *n3*
11. Connect sources to the sink
12. Label the nodes
13. Schedule *cbr0* to start at0.5 and stop at4.5seconds
14. Schedule *cbr1*to start at1.0 and stop at4.0seconds
15. Call finish procedure at 5.0seconds
16. Run the simulation
17. Execute NAM on the trace file
18. Observe simulated events on theNAM and packet flow on link n2 to n3
19. Stop

Program

```
#Study of UDP performance-UDP.tcl #Create a

simulator object

setns[newSimulator]

#Define different colors for dataflows
$nscolor1Blue
$nscolor2Red

#Open the namtrace file
set nf [open out.nam w]
$nsnamtrace-all$nf

#Create four nodes
set n0 [$nsnode]set
n1 [$ns node] set n2
[$ns node] set n3 [$ns
node]

#Create links between the nodes
$nsduplex-link$n0$n21Mb10msDropTail
$nsduplex-link$n1$n21Mb10msDropTail
$nsduplex-link$n3$n21Mb10msSFQ

#Specify layout to fnodes
$nsduplex-link-op$n0$n2orientright-down
$nsduplex-link-op$n1$n2orientright-up
$nsduplex-link-op$n2$n3orientright

#Monitor the  queue for the link2—3 vertically
$nsduplex-link-op$n2$n3queuePos0.5

#Createa UDP agent and attach it to node n0 set udp0 [new
Agent/UDP]
$udp0setclass_1
$nsattach-agent$n0$udp0

#Createa CBR traffic source and attach it to udp0 set cbr0 [new
Application/Traffic/CBR]
$cbr0setpacketSize_500
$cbr0setinterval_0.005
$cbr0attach-agent$udp0

#Createa UDP agent and attach it to node n1 set udp1 [new
Agent/UDP]
$udp1setclass_2
$nsattach-agent$n1$udp1
```

```
#Createa CBR traffic source and attach it to udp1 set cbr1 [new
Application/Traffic/CBR]
$cbr1setpacketSize_500
$cbr1setinterval_0.005
$cbr1attach-agent$udp1

#Createa Null agent(a traffic sink)and attach it to node n3 set null0 [new Agent/Null]
$nsattach-agent$n3$null0

#Connect traffic sources with the traffic sink
$nsconnect$udp0$null0
$nsconnect$udp1$null0

#Define finish
procedureprocfinish
{}{
     globalnsnf
     $nsflush-trace

     #Closethetracefile close $nf

     #Executenamonthetracefile exec nam -a
     out.nam &
     exit0
}

#Define label for nodes
$nsat0.0"$n0labelSender1"
$nsat0.0"$n1labelSender2"
$nsat0.0"$n2labelRouter"
$nsat0.0"$n3labelReceiver"

#Schedule events for the CBR agents
$nsat0.5"$cbr0start"
$nsat1.0"$cbr1start"
$nsat4.0"$cbr1stop"
$nsat4.5"$cbr0stop"

#Call finish procedure after 5seconds of simulation time
$nsat5.0"finish"

#Run the simulation
$ns run
```

Output

Result:

Thus the performance of UDP and basic network terminologies were studied using NS2

Exp#8b Study of TCP Performance

Date:

Aim

To study the performance of a TCP network with drop tail queue mechanismon the gateway

Algorithm

1. Create a simulator object
2. Define different flows for dataflows
3. Trace all events in a nam file and text file
4. Create source nodes(*s1*,*s2*,*s3*),gateway(*G*)and receiver(*r*)
5. Describe their layout topology
6. Specify the link between nodes
7. Definethequeuesizebetweennodes*G*and*r*as5
8. Monitor queue on all links vertically 90°
9. Create TCP agents*tcp1*,*tcp2*,*tcp3*and attach it to nodes*s1*,*s2*and*s3* respectively
10. Create three TCP sinks and attach it to node *r*
11. Connect traffic sources to the sink
12. Create FTP agents *ftp1,ftp2,ftp3* and attach it to *tcp1*,*tcp2*and *tcp3* respectively
13. Label the nodes at start time
14. Schedule *ftp1,ftp2,ftp3* to start at 0.1and stop at 5.0seconds
15. Call *finish* procedure at 5.25 seconds
16. Run the simulation
17. Execute NAM on the trace file
18. Observe the simulated events on the NAM editor and packet flow on link G to r
19. View the trace file and analyse the events
20. Stop

Program

```
#Study  of TCP performance-TCP.tcl #Create a
simulator object
setns[newSimulator]

#Open trace files
setf[opendroptail-queue-out.trw]
$nstrace-all$f

#Open the nam trace file
setnf[opendroptail-queue-out.namw]
$nsnamtrace-all$nf

#s1,s2ands3actassources. set s1 [$ns node]
set s2 [$ns node]sets3
[$ns node]

#G acts as a
gatewaysetG[$ns node]
#r acts as a
receiversetr[$ns node]

#Define different colors for dataflows
$nscolor1red
$nscolor2SeaGreen
$nscolor3blue

#Create links between the nodes
$nsduplex-link$s1$G6Mb10msDropTail
$nsduplex-link$s2$G6Mb10msDropTail
$nsduplex-link$s3$G6Mb10msDropTail
$nsduplex-link$G$r3Mb10msDropTail

#Define the layout of the nodes
$nsduplex-link-op$s1$Gorientright-up
$nsduplex-link-op$s2$Gorientright
$nsduplex-link-op$s3$Gorientright-down
$nsduplex-link-op$G$rorientright

#Define  the queue size for the link between node  G and r
$nsqueue-limit$G$r5

#Monitorthequeuesforlinksvertically
$nsduplex-link-op$s1$GqueuePos0.5
$nsduplex-link-op$s2$GqueuePos0.5
$nsduplex-link-op$s3$GqueuePos0.5
$nsduplex-link-op$G$rqueuePos 0.5
```

```
#Createa TCP agent  and attach it to  node s1 settcp1 [new
Agent/TCP/Reno]
$ns        attach-agent$s1$tcp1
$tcp1setwindow_8
$tcp1set fid_                    1

#Createa TCP agent and attach it to node s2 set tcp2 [new
Agent/TCP/Reno]
$ns        attach-agent$s2$tcp2
$tcp2setwindow_8
$tcp2set fid_                    2

#Createa TCP agent and attach it to node s3 set tcp3 [new
Agent/TCP/Reno]
$nsattach-agent$s3$tcp3
$tcp3setwindow_4
$tcp3set fid_                    3

#Create TCP sink agents and attach them to node r set sink1 [new
Agent/TCPSink]
setsink2[newAgent/TCPSink]
setsink3[newAgent/TCPSink]
$nsattach-agent$r$sink1
$nsattach-agent$r$sink2
$nsattach-agent$r$sink3

#Connect the traffic sources with the traffic sinks
$nsconnect$tcp1$sink1
$nsconnect$tcp2$sink2
$nsconnect$tcp3$sink3

#Create FTP applications and attach them to agents set ftp1
[new Application/FTP]
$ftp1attach-agent$tcp1
setftp2[newApplication/FTP]
$ftp2attach-agent$tcp2
setftp3[newApplication/FTP]
$ftp3attach-agent$tcp3

#Definea'finish'procedure proc finish {} {
    globalns
    $nsflush-trace
    puts"runningnam."
    execnam-adroptail-queue-out.nam& exit 0
}
```

```
#Define label for nodes
$nsat0.0"$s1labelSender1"
$nsat0.0"$s2labelSender2"
$nsat0.0"$s3labelSender3"
$nsat0.0"$GlabelGateway"
$nsat0.0"$rlabelReceiver"

#Schedule ftp events
$nsat0.1"$ftp1start"
$nsat0.1"$ftp2start"
$nsat0.1"$ftp3start"
$nsat5.0"$ftp1stop"
$nsat5.0"$ftp2stop"
$nsat5.0"$ftp3stop"

#Call finish procedure after 5seconds of simulation time
$nsat5.25"finish"

#Run the simulation
$ns run
```

Output

Result

Thus the behaviour of TCP was observed and the basic terminologies of TCP transmission were understood.

Exp#9a **Distance Vector Routing Protocol**

Date:

Aim

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

Algorithm

1. Create a simulator object
2. Set routing protocol l to Distance Vector routing
3. Trace packets on all links onto NAM trace and text trace file
4. Define finish procedure to close files,flush tracing and run NAM
5. Create eight nodes
6. Specify the link characteristics between nodes
7. Describe their layout topology as aoctagon
8. Add UDP agent for node n1
9. Create CBR traffic on top of UDP and set traffic parameters.
10. Add a sink agent to node n4
11. Connect source and the sink
12. Schedule events as follows:
    a. Start traffic flow at 0.5
    b. Down the linkn3-n4at1.0
    c. Up the linkn3-n4at2.0
    d. Stop traffic at 3.0
    e. Call finish procedure at 5.0
13. Start the scheduler
14. Observe the traffic route when link is up and down
15. View the simulated events and trace file analyze it
16. Stop

Program

```
#Distance vector routing protocol – distvect.tcl#Create

a simulator object set ns [new Simulator]

#Use distance vector  routing
$nsrtproto DV

#Open the  nam trace file
set nf [open out.nam w]
$nsnamtrace-all$nf

#Open tracefile
setnt[opentrace.trw]
$nstrace-all$nt

#Define 'finish'
procedureprocfinish{}
{
      globalnsnf
      $ns flush-trace
      #Closethetrace file close
      $nf
      #Execute nam on the trace
      fileexecnam-aout.nam& exit 0
}

#Create8nodes set n1
[$ns node] set n2 [$ns
node] set n3 [$ns
node] set n4 [$ns
node] set n5 [$ns
node] set n6 [$ns
node] set n7 [$ns

 node] set n8 [$ns
node]

#Specify link characterestics
$nsduplex-link$n1$n21Mb10msDropTail
$nsduplex-link$n2$n31Mb10msDropTail
$nsduplex-link$n3$n41Mb10msDropTail
$nsduplex-link$n4$n51Mb10msDropTail
$nsduplex-link$n5$n61Mb10msDropTail
$nsduplex-link$n6$n71Mb10msDropTail
$nsduplex-link$n7$n81Mb10msDropTail
$nsduplex-link$n8$n11Mb10msDropTail
```

```
#specify layout as aoctagon
$nsduplex-link-op$n1$n2orientleft-up
$nsduplex-link-op$n2$n3orient up
$nsduplex-link-op$n3$n4orientright-up
$nsduplex-link-op$n4$n5orientright
$nsduplex-link-op$n5$n6orientright-down
$nsduplex-link-op$n6$n7orientdown
$nsduplex-link-op$n7$n8orientleft-down
$nsduplex-link-op$n8$n1orientleft

#Createa UDP agent  and attach it to node n1 set udp0
[new Agent/UDP]
$nsattach-agent$n1$udp0

#Createa CBR traffic source and attach it to udp0 set cbr0 [new
Application/Traffic/CBR]
$cbr0setpacketSize_500
$cbr0setinterval_0.005
$cbr0attach-agent$udp0

#Createa Null agent(a traffic sink)and attach it to node n4 set null 0 [new Agent/Null]
$nsattach-agent$n4$null0

#Connect the traffic source with  the traffic sink
$nsconnect$udp0$null0

#Schedule events for the CBR agent and the network dynamics
$nsat0.0                "$n1labelSource"
$nsat0.0                "$n4labelDestination"
$nsat0.5"$cbr0start"
$nsrtmodel-at1.0down$n3$n4
$nsrtmodel-at2.0up$n3$n4
$nsat4.5"$cbr0stop"

#Call the  finish procedure after  5seconds of simulation time
$nsat5.0"finish"

#Run the simulation
$ns run
```

Result

Thus, performance of distance vector protocol and routing path was studied using NS2.

Exp.No.9b **Link State Routing Protocol**

Date:

Aim

To simulate a link failure and to observe link state routing protocol inaction.

Algorithm

1. Create a simulator object
2. Set routing protocol to Link State routing
3. Trace packets on all links on to NAM trace and text trace file
4. Define finish procedure to close files,flush tracing and run NAM
5. Create twelve nodes
6. Specify the link characteristics between nodes
7. Describe their layout topology in an adhoc manner.
8. Create CBR traffic on top of UDP and set traffic parameters.
9. Create source and sink and connect them
10. Schedule events as follows:
    a. Start traffic flows at1.0and2.0
    b. Down the link n5-n11at10.0and restore it at 30.0
    c. Down the lin kn7-n6 at15.0and restore it at20.0
    d. Call finish procedure at 45.0
11. Start the scheduler
12. Observe the traffic route when link is upand down
13. View the simulated events and trace file analyze it
14. Stop

Program

```
setns[newSimulator] set nr
[open thro.tr w]
$nstrace-all$nr
setnf[openthro.namw]

$nsnamtrace-all
$nfprocfinish{}
{
      globalnsnrnf
      $nsflush-trace
      close$nf close $nr
      exec nam
      thro.nam&exit 0
}

for{seti0}{$i<12}{incri1}{ set n($i) [$ns node]}

for{seti0}{$i<8}{incri}{
      $nsduplex-link$n($i)$n([expr$i+1])1Mb10msDropTail}

$nsduplex-link$n(0)$n(8)1Mb10msDropTail
$nsduplex-link$n(1)$n(10)1Mb10msDropTail
$nsduplex-link$n(0)$n(9)1Mb10msDropTail
$nsduplex-link$n(9)$n(11)1Mb10msDropTail
$nsduplex-link$n(10)$n(11)1Mb10msDropTail
$nsduplex-link$n(11)$n(5)1Mb10msDropTail

setudp0[newAgent/UDP]
$nsattach-agent$n(0)$udp0
setcbr0[newApplication/Traffic/CBR]
$cbr0setpacketSize_500
$cbr0setinterval_0.005
$cbr0attach-agent$udp0 set null0
[new Agent/Null]
$nsattach-agent$n(5)$null0
$nsconnect$udp0$null0

setudp1[newAgent/UDP]
$nsattach-agent$n(1)$udp1
setcbr1[newApplication/Traffic/CBR]
$cbr1setpacketSize_500
$cbr1setinterval_0.005
$cbr1attach-agent$udp1 set null0
[new Agent/Null]
$nsattach-agent$n(5)$null0
$nsconnect$udp1$null0
```

```
$nsrtproto LS

$nsrtmodel-at10.0down$n(11)$n(5)
$nsrtmodel-at15.0down$n(7)$n(6)
$nsrtmodel-at30.0up$n(11)$n(5)
$nsrtmodel-at20.0up$n(7) $n(6)

$udp0setfid_1
$udp1setfid_2
$nscolor1Red
$nscolor2Green

$nsat1.0"$cbr0start"
$nsat2.0"$cbr1start"

$nsat45"finish"
$ns run
```

Result:

Thus performance of link state protocol and its routing path was simulated using NS2.

Exp#10 **CRC Error Detection**

Date:

Aim

To detect whether the given data is corrupted or not using CRC method.

Algorithm

1. Read number of data bits.
2. Read the data bit-by-bit
3. Read number of divisor bits
4. Enter the divisor bit-by-bit
5. Append zeroes to the message
6. Generate remainder by using XOR division
7. Subtract remainder from message using XOR
8. Display the CRC code word
9. Accept transmitted message as receiver side data
10. Perform polynomial division using XOR
11. If remainder is zero then display“No error”else display“Error”
12. Stop

Program

```
import java.io.*;

class crcgen {
  public static void main(String args[]) throws IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    int[] data;
    int[] div;
    int[] divisor;
    int[] rem;
    int[] crc;

    int data_bits, divisor_bits, tot_length;

    // Input data bits
    System.out.println("Enter number of data bits:");
    data_bits = Integer.parseInt(br.readLine());
    data = new int[data_bits];

    System.out.println("Enter data bits:");
    for (int i = 0; i < data_bits; i++)
      data[i] = Integer.parseInt(br.readLine());

    // Input divisor (generator polynomial)
    System.out.println("Enter number of bits in divisor:");
    divisor_bits = Integer.parseInt(br.readLine());
    divisor = new int[divisor_bits];

    System.out.println("Enter Divisor bits:");
    for (int i = 0; i < divisor_bits; i++)
      divisor[i] = Integer.parseInt(br.readLine());

    // Prepare for division: data bits + (divisor_bits - 1) zeros
    tot_length = data_bits + divisor_bits - 1;
    div = new int[tot_length];
    rem = new int[tot_length];
    crc = new int[tot_length];

    // Copy data bits into div array
    for (int i = 0; i < data.length; i++)
      div[i] = data[i];

    System.out.print("Dividend (after appending 0's): ");
    for (int i = 0; i < div.length; i++)
      System.out.print(div[i]);
    System.out.println();

    // Copy div into rem for division
    for (int j = 0; j < div.length; j++)
      rem[j] = div[j];

    rem = divide(div, divisor, rem);

    // Append remainder to data
    for (int i = 0; i < div.length; i++) {
      crc[i] = (div[i] ^ rem[i]);
    }
```

```
        System.out.println("CRC code:");
        for (int i = 0; i < crc.length; i++)
            System.out.print(crc[i]);

        // Error Detection
        System.out.println("\nEnter CRC code of " + tot_length + " bits:");
        for (int i = 0; i < crc.length; i++)
            crc[i] = Integer.parseInt(br.readLine());

        for (int j = 0; j < crc.length; j++)
            rem[j] = crc[j];

        rem = divide(crc, divisor, rem);

        boolean error = false;
        for (int i = 0; i < rem.length; i++) {
            if (rem[i] != 0) {
                System.out.println("Error");
                error = true;
                break;
            }
        }

        if (!error)
            System.out.println("No Error");
    }

    static int[] divide(int div[], int divisor[], int rem[]) {
        int cur = 0;

        while (true) {
            for (int i = 0; i < divisor.length; i++)
                rem[cur + i] = (rem[cur + i] ^ divisor[i]);

            while (cur < rem.length && rem[cur] == 0)
                cur++;

            if ((rem.length - cur) < divisor.length)
                break;
        }

        return rem;
    }
}
```

Result

Thus error detection is done using cyclic redundancy check method.