**ADHIYAMAAN COLLEGE OF ENGINEERING (AUTONOMOUS)**

**DEPARTMENT OF IT**

**COMPUTER NETWORKS LABORATORY MANUAL**

Year : III Year IT

Semester : V

Sub Code : 518ITP08

Regulation : 2018

Prepared by

Mrs.A.SAJITHABEGAM

AP/IT

**LIST OF EXPERIMENTS**

|  |  |
| --- | --- |
| **EXPT**  **NO.** | **TITLE** |
| **1** | **BASIC NETWORK COMMAND LINE UTILITIES SUCH AS PING, NETSTAT, TRACERT, NSLOOKUP, PORT SCAN, ARP, IPCONFIG.** |
| **2** | **GENERATE HAMMING CODE FOR ERROR DETECTION AND CORRECTION** |
| **3** | **IMPLEMENT ERROR DETECTION CODE USING CRC** |
| **4** | **IMPLEMENTATION OF STOP AND WAIT PROTOCOL** |
| **5** | **IMPLEMENTATION OF SLIDING WINDOW PROTOCOL** |
| **6** | **IMPLEMENTATION OF UDP** |
| **7** | **IMPLEMENTATION OF TCP** |
| **8** | **STUDY OF BASIC CONCEPTS OF NETWORK SIMULATOR (NS2), ITS INSTALLATION AND WORKING ENVIRONMENT.** |
| **9** | **SIMULATION OF NS2 OBJECT,NETWORK TOPOLOGY,NETWORK TRAFFIC, NAM AND TRACE** |
| **10** | **SIMULATION OF DUPLEX NETWORK** |
| **11** | **SIMULATION OF WIRED NETWORK** |
| **12** | **SIMULATION OF LINK STATE ROUTING AND DISTANCE VECTOR ROUTING** |
| **13** | **PACKET CAPTURE AND NETWORK TRAFFIC USING WIRE SHARK TOOL.** |

**EX NO: 1 BASIC NETWORK COMMAND: PING, NETSTAT, TRACERT, NSLOOKUP, PORT SCAN, ARP, IPCONFIG.**

**AIM**:

To execute network basic commands in command prompt.

**Ping:**

The ping command helps to verify IP-level connectivity. When **troubleshooting**, you can use ping to send an **ICMP echo request** to a target host name or IP address. Use ping whenever you need to verify that a host computer can connect to the TCP/IP network and network resources. Ping is the primary tool for troubleshooting IP-level connectivity. Type **ping -?** at a command prompt to see a complete list of available command-line options. Ping allows you to specify the size of packets to use (the default is 32 bytes), how many to send, whether to record the route used, what Time To Live (TTL) value to use, and whether to set the "don't fragment" flag.

**Netstat:**

The netstat [command](https://www.lifewire.com/what-is-a-command-2625828) is a [Command Prompt command](https://www.lifewire.com/list-of-command-prompt-commands-4092302) used to display very detailed information about how your computer is communicating with other computers or network devices. Specifically, the netstat command can show details about individual network connections, overall and protocol-specific networking statistics, and much more, all of which could help troubleshoot certain kinds of networking issues.

**Tracert:**

The tracert [command](https://www.lifewire.com/what-is-a-command-2625828) is a [Command Prompt command](https://www.lifewire.com/list-of-command-prompt-commands-4092302), used to show several details about the path that a packet takes from the computer or device to destination you specify.A traceroute is a function which traces the path from one network to another. It allows us to diagnose the source of many problems.

**Nslookup:**

nslookup (which stands for name server lookup) is a network utility program used to obtain information about internet servers. As its name suggests, it finds name server information for domains by querying the [Domain Name System (DNS)](https://www.lifewire.com/what-is-dns-domain-name-system-2625855). Nslookup is a useful tool for troubleshooting DNS problems, such as host name resolution. When you start Nslookup, it shows the host name and IP address of the DNS server that is configured for the local system, and then display a command prompt for further queries. If you type a question mark ( **?** ), Nslookup shows all available commands. You can exit the program by typing **exit** .

**port scan**

The most common host-based tool for checking for open ports on Windows or Unix systems is the **netstat** command. But running this command means actually walking or remotely accessing each and every server and you miss other host systems that might be listening on improper ports. This is where port scanners come in with this tool, a system, network, or security administrator can check a group of hosts all at once.

**ARP:**

**Address Resolution Protocol** (**ARP**) is a protocol for mapping an Internet Protocol address (IP address) to a physical machine address that is recognized in the local network.When you try to ping an IP address on your local network, say 192.168.1.1, your system has to turn the IP address 192.168.1.1 into a MAC address. This involves using ARP to resolve the address, hence its name.Systems keep an ARP look-up table where they store information about what IP addresses are associated with what MAC addresses. When trying to send a packet to an IP address, the system will first consult this table to see if it already knows the MAC address. If there is a value cached, ARP is not used.

If the IP address is not found in the ARP table, the system will then send a broadcast packet to the network using the ARP protocol to ask "who has 192.168.1.1". Because it is a broadcast packet, it is sent to a special MAC address that causes all machines on the network to receive it. Any machine with the requested IP address will reply with an ARP packet that says "I am 192.168.1.1", and this includes the MAC address which can receive packets for that IP.

**Ipconfig:**

**Ipconfig** (sometimes written as **IPCONFIG**) is a command line tool used to control the network connections on Windows NT/2000/XP machines. **Ipconfig** displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings.

**OUTPUT:**

**PING :**

C:\Users\admin>ping gmail.com

Pinging gmail.com [142.250.67.229] with 32 bytes of data:

Reply from 142.250.67.229: bytes=32 time=52ms TTL=115

Reply from 142.250.67.229: bytes=32 time=52ms TTL=115

Reply from 142.250.67.229: bytes=32 time=53ms TTL=115

Reply from 142.250.67.229: bytes=32 time=53ms TTL=115

Ping statistics for 142.250.67.229:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 52ms, Maximum = 53ms, Average = 52m

**NETSTAT**:

C:\Users\admin>netstat

Active Connections

Proto Local Address Foreign Address State

TCP 127.0.0.1:49774 ITPC2:49775 ESTABLISHED

TCP 127.0.0.1:49775 ITPC2:49774 ESTABLISHED

TCP 127.0.0.1:52105 ITPC2:52106 ESTABLISHED

TCP 127.0.0.1:52106 ITPC2:52105 ESTABLISHED

TCP 127.0.0.1:52118 ITPC2:52119 ESTABLISHED

TCP 127.0.0.1:52119 ITPC2:52118 ESTABLISHED

**TRACERT:**

C:\Users\admin>tracert gmail.com

Tracing route to gmail.com [216.58.200.133]

over a maximum of 30 hops:

1 <1 ms 3 ms <1 ms 192.168.49.254

2 3 ms 3 ms 3 ms 210.212.246.130

**NSLOOKUP**

C:\Users\admin>nslookup gmail.com

Server: dns.google

Address: 8.8.8.8

Name: gmail.com

Addresses: 2404:6800:4007:80b::2005

142.250.183.10

**ARP**

C:\Users\admin>ARP –A

Interface: 192.168.85.1 --- 0x7

Internet Address Physical Address Type

192.168.85.254 00-50-56-e1-80-22 dynamic

192.168.85.255 ff-ff-ff-ff-ff-ff static

224.0.0.22 01-00-5e-00-00-16 static

224.0.0.251 01-00-5e-00-00-fb static

224.0.0.252 01-00-5e-00-00-fc static

239.255.255.250 01-00-5e-7f-ff-fa static

255.255.255.255 ff-ff-ff-ff-ff-ff static

**IPCONFIG**

C:\Users\admin>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . :

Link-local IPv6 Address . . . . . : fe80::7888:b0c6:ecc6:3ce0%12

IPv4 Address. . . . . . . . . . . : 192.168.49.234

Subnet Mask . . . . . . . . . . . : 255.255.255.0

Default Gateway . . . . . . . . . : 192.168.49.254

**RESULT:**

Thus the basic network command are implemented.

**EX NO:2 GENERATING HAMMING CODE FOR ERROR DETECTION AND CORRECTION**

**AIM:**

To Generating Hamming code for error detection and correction

**ALGORITHM:**

**STEP 1**: Using the same formula as in encoding, the number of redundant bits are ascertained.

2r ≥ m + r + 1 where *m* is the number of data bits and *r* is the number of redundant bits.

**STEP2**: The r redundant bits placed at bit positions of powers of 2, i.e. 1, 2, 4, 8, 16 etc.

**STEP 3:** Parity bits are calculated based upon the data bits and the redundant bits using the same rule as during generation of c1,c2 ,c3 ,c

**STEP 4:** The decimal equivalent of the parity bits binary values is calculated. If it is 0, there is no error. Otherwise, the decimal value gives the bit position which has error. For example, if c1c2c3c4 = 1001, it implies that the data bit at position 9, decimal equivalent of 1001, has error. The bit is flipped to get the correct message.

**PROGRAM:**

import java.util.\*;

class DanHamming

{

public static void main(String arg[])

{

Scanner sc=new Scanner(System.in);

System.out.println(“Enter the 7-bit data code”);

int d[]=new int[7];

for(int i=0;i<7;i++)

{

d[i]=sc.nextInt();

}

int p[]=new int[4];

p[0]=d[0]^d[1]^d[3]^d[4]^d[6];

p[1]=d[0]^d[2]^d[3]^d[5]^d[6];

p[2]=d[1]^d[2]^d[3];

p[3]=d[4]^d[5]^d[6];

int c[]=new int[11];

System.out.println(“Complete Code Word is “);

c[0]=p[0];

c[1]=p[1];

c[2]=d[0];

c[3]=p[2];

c[4]=d[1];

c[5]=d[2];

c[6]=d[3];

c[7]=p[3];

c[8]=d[4];

c[9]=d[5];

c[10]=d[6];

for(int i=0; i<11;i++)

{

System.out.print(c[i]+ ” “);

}

System.out.println();

System.out.println(“Enter the Received codeword”);

int r[]=new int[11];

for(int i=0;i<11;i++)

{

r[i]=sc.nextInt();

}

int pr[]=new int[4];

int rd[]=new int[7];

pr[0]=r[0];

pr[1]=r[1];

rd[0]=r[2];

pr[2]=r[3];

rd[1]=r[4];

rd[2]=r[5];

rd[3]=r[6];

pr[3]=r[7];

rd[4]=r[8];

rd[5]=r[9];

rd[6]=r[10];

int s[]=new int[4];

s[0]=pr[0]^rd[0]^rd[1]^rd[3]^rd[4]^rd[6];

s[1]=pr[1]^rd[0]^rd[2]^rd[3]^rd[5]^rd[6];

s[2]=pr[2]^rd[1]^rd[2]^rd[3];

s[3]=pr[3]^rd[4]^rd[5]^rd[6];

int dec=(s[0]\*1)+(s[1]\*2)+(s[2]\*4)+(s[3]\*8);

if(dec==0)

System.out.println(“No error”);

else

{

System.out.println(“Error is at “+dec);

if(r[dec-1]==0)

r[dec-1]=1;

else

r[dec-1]=0;

}

System.out.println(“Corrected code word is : “);

for(int i=0;i<11;i++)

System.out.print(r[i]+” “);

System.out.println();

}

}

**SAMPLE OUTPUT**:

Enter the 7-bit data code

1 1 1 0 1 0 0

Complete Code Word is

1 0 1 0 1 1 0 1 1 0 0

Enter the Received codeword

1 0 0 0 1 1 0 1 1 0 0

Error is at 3

Corrected code word is :

1 0 1 0 1 1 0 1 1 0 0

**RESULT:**

Thus the program has been implemented to generating hamming code for error detection and correction using java

**EX NO:3 IMPLEMENTATION OF CRC**

**AIM:**

To implement the CRC-12, CRC-16, CRC-CCITT in data link layer

**ALGORITHM:**

**STEP 1**: let r be the degree of G(x).Append r zero bits to the low-order end of the frame. So it now contains m+r bits and corresponds to the polynomial x2 m(x).

**STEP 2**: divide the bit string corresponding to G(x) into the bit string corresponding to x2 m(x) using modulo-2 division.

**STEP 3**: subtract the remainder from the bit string corresponding to x2 m(x) using modulo-2 sub. The result is the check summed frame to be transmitted. We call it as a polynomial.

**STEP 4**: Take the frame as input from the user.

**STEP 5**: Take the user‟s choice of use of the CRC polynomial that he wants to use:

CRC-12 = x12+x11+x 3+x 2+x+1 CRC-16 = x16+x15+x 2+1 CRC-CCITT = x16+x 12+x 5+1

**STEP 6**: Depending upon the user‟s choice, append the zeros to the end of the frame and display.

**STEP 7**: Calculate the CRC by performing modulo 2 division.

**STEP 8**: Replace the appended zeros in step 3 by the remainder of the above performed division.

**STEP 9**: This is the transmitted data.

**STEP 10**: At the receiver side, the input string is subjected to modulo 2 division by the same polynomial that was used at the sender side.

**STEP 11**: If the remainder is 0 there is no error otherwise there was an error during transmission.

**PROGRAM:**

import java.io.\*;

class crc

{

public static void main(String args[]) throws IOException

{

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

System.out.println("Enter Generator:");

String gen = br.readLine();

System.out.println("Enter Data:");

String data = br.readLine();

String code = data;

while(code.length() < (data.length() + gen.length() - 1))

code = code + "0";

code = data + div(code,gen);

System.out.println("The transmitted Code Word is: " + code);

System.out.println("Please enter the received Code Word: ");

String rec = br.readLine();

if(Integer.parseInt(div(rec,gen)) == 0)

System.out.println("The received code word contains no errors.");

else

System.out.println("The received code word contains errors.");

}

static String div(String num1,String num2)

{

int pointer = num2.length();

String result = num1.substring(0, pointer);

String remainder = "";

for(int i = 0; i< num2.length(); i++)

{

if(result.charAt(i) == num2.charAt(i))

remainder += "0";

else

remainder += "1";

}

while(pointer < num1.length())

{

if(remainder.charAt(0) == '0')

{

remainder = remainder.substring(1, remainder.length());

remainder = remainder + String.valueOf(num1.charAt(pointer));

pointer++;

}

result = remainder;

remainder = "";

for(int i = 0; i< num2.length(); i++)

{

if(result.charAt(i) == num2.charAt(i))

remainder += "0";

else

remainder += "1";

}

}

return remainder.substring(1,remainder.length());

}

}

**OUTPUT:**

Enter Generator:

1010

Enter Data:

10101010

The transmitted Code Word is: 10101010010

Please enter the received Code Word:

10101010010

The received code word contains no errors.

**RESULT:**

Thus the program has been implemented to generating CRC for error detection using java.

**EXNO:4 IMPLEMENTATION OF STOP AND WAIT PROTOCOL**

**AIM:**

To write a program for implementation of stop and wait protocol through client and server.

**ALGORITHM:**

**SENDER:**

**STEP 1**:Start the program.

**STEP 2**:Declare the socket address.

**STEP 3**:Specify the port address to get binded with receiver

**STEP 4**:If the above specified is already in use display an error message.

**STEP 5**:Initialize a variable to store length of data.

**STEP 6**:Display the received data.

**STEP 7**:close the socket.

**RECEIVER**

**STEP 1**:Receiver window is shrinked after storing message.

**STEP 2**:Once the message is received,the window size is expanded.

**STEP 3**:Message received from sender is displayed

**STEP 4**:puts(received) dislay the data.

**STEP 5**:window size is again shrinked.

**STEP 6**:The data in the size of window size is decreased.

**STEP 7**:receive the message.

**STEP 8**:Stop the program.

**PROGRAM**

**SENDER:**

import java.io.\*;

import java.net.\*;

importjava.util.Scanner;

class stopwaitsender

{

public static void main(String args[]) throws Exception

{

stopwaitsendersws = new stopwaitsender();

sws.run();

}

public void run() throws Exception

{

Scanner sc=new Scanner(System.in);

System.out.println(“Enter no of frames to be sent:”);

int n=sc.nextInt();

Socket myskt=new Socket(“localhost”,9999);

PrintStreammyps=new PrintStream(myskt.getOutputStream());

for(inti=0;i<=n;)

{

if(i==n)

{

myps.println(“exit”);

break;

}

System.out.println(“Frame no “+i+” is sent”);

myps.println(i);

BufferedReader bf=new BufferedReader(new InputStreamReader (myskt. getInputStream ()));

String ack=bf.readLine();

if(ack!=null)

{

System.out.println(“Acknowledgement was Received from receiver”);

i++;

Thread.sleep(4000);

}

else

{

myps.println(i);

}

}

}

}

**RECEIVER:**

import java.io.\*;

import java.net.\*;

class stopwaitreceiver

{

public static void main(String args[])throws Exception

{

stopwaitreceiver swr = new stopwaitreceiver();

swr.run();

}

public void run() throws Exception

{

String temp="any message",str="exit";

ServerSocket myss=new ServerSocket(9999);

Socket ss\_accept=myss.accept();

BufferedReaderss\_bf=new BufferedReader(new InputStreamReader(ss\_accept.GetInputStream()));

PrintStream myps=new PrintStream(ss\_accept.getOutputStream());

while(temp.compareTo(str)!=0)

{

Thread.sleep(1000);

temp=ss\_bf.readLine();

if(temp.compareTo(str)==0)

{

break;

}

System.out.println(“Frame “+temp+” was received”);

Thread.sleep(500);

myps.println(“Received”);

}

System.out.println(“ALL FRAMES WERE RECEIVED SUCCESSFULLY”);

}

}

**OUTPUT:**

**SENDER:**

[usera37@redhatnewit gunaseelan]$ javac stopwaitsender.java

[usera37@redhatnewit gunaseelan]$ java stopwaitsender

Enter no of frames to be sent:

5

Frame no0is sent

Acknowledgement was Received from receiver

Frame no1is sent

Acknowledgement was Received from receiver

Frame no2is sent

Acknowledgement was Received from receiver

Frame no3is sent

Acknowledgement was Received from receiver

Frame no4is sent

Acknowledgement was Received from receiver

RECEIVER

[usera37@redhatnewit gunaseelan]$ javac stopwaitreceiver.java

[usera37@redhatnewit gunaseelan]$ java stopwaitreceiver

Frame 0 was received

Frame 1 was received

Frame 2 was received

Frame 3 was received

Frame 4 was received

ALL FRAMES WERE RECEIVED SUCCESSFULLYSERVER:

**RESULT**

Thus the program has been implemented to stop and wait protocol using java.

**EXNO:5 SLIDING WINDOW**

**AIM:**

To write a program for implementation of sliding window protocol through client and server.

**ALGORITHM:**

**SENDER:**

**STEP 1**:Start the program.

**STEP 2**:Declare the socket address.

**STEP 3**:Specify the port address to get binded with receiver

**STEP 4**:If the above specified is already in use display an error message.

**STEP 5**:Initialize a variable to store length of data.

**STEP 6**:Display the received data.

**STEP 7**:close the socket.

**RECEIVER**

**STEP 1**:Receiver window is shrinked after storing message.

**STEP 2**:Once the message is received,the window size is expanded.

**STEP 3**:Message received from sender is displayed

**STEP 4**:puts(received) dislay the data.

**STEP 5**:window size is again shrinked.

**STEP 6**:The data in the size of window size is decreased.

**STEP 7**:receive the message.

**STEP 8**:Stop the program.

**PROGRAM:**

**CLIENT PROGRAM**

import java.io.\*;

import java.net.\*;

class sliclient

{

public static void main(String aafhbasgj[])throws Exception

{

Socket s=new Socket("localhost",6483);

String opmsg;

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

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

PrintStream dos=new PrintStream(s.getOutputStream());

int iterations=Integer.parseInt(in.readLine());

String msg=in.readLine();

int no=Integer.parseInt(msg);

int i=0,intr=0,iframe=0;

int framecounter=0;

while(intr<iterations)

{

msg=in.readLine();

System.out.println(msg);

i++;

framecounter++;

if(framecounter%no==0)

{

System.out.println("sending ack to server");

opmsg="ACK : FRAME "+iframe+" RECIEVED";

dos.println(opmsg);

iframe++;

}

if(i==8)

{

i=0;

intr++;

}

}

s.close();

}

}

**SERVER:**

import java.io.\*;

import java.net.\*;

class slisender

{

public static void main(String aafhbasgj[])throws Exception

{

ServerSocket ss=new ServerSocket(6483);

Socket s=ss.accept();

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

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

String str1;

System.out.println("Enter number of iterations");

int it=Integer.parseInt(br.readLine());

int intr=0;

int msg[]=new int[8];

for(int i=0;i<8;i++)

msg[i]=i;

PrintStream p;

int no;

p=new PrintStream(s.getOutputStream());

p.println(it);

System.out.println("Enter window size");

no=Integer.parseInt(br.readLine());

p.println(no);

int i=0;

while(intr<it)

{

p.println(msg[i]);

i++;

if(i==no)

{

str1=in.readLine();

System.out.println(str1);

}

if(i==8)

{

i=0;

intr++;

}

}

ss.close();

s.close();

}

}

**OUTPUT:**

**SERVER:**

Enter number of iterations

4

Enter window size

2

ACK : FRAME 0 RECIEVED

ACK : FRAME 1 RECIEVED

ACK : FRAME 2 RECIEVED

ACK : FRAME 3 RECIEVED

**CLIENT:**

project@ubuntu:~/lab/sliding$ java sliclient

0

1

sending ack to server

2

3

sending ack to server

4

5

sending ack to server

0

1

sending ack to server

2

3

sending ack to server

4

5

sending ack to server

0

1

sending ack to server

2

3

sending ack to server

4

5

sending ack to server

0

1

sending ack to server

2

3

sending ack to server

4

5

sending ack to server

0

1

sending ack to server

2

3

sending ack to server

4

5

sending ack to server

6

7

sending ack to server

**RESULT:**

Thus the program has been implemented to sliding window using java.

**EX NO: 6 IMPLEMENTATION OF UDP**

**AIM:**

To write a program for implementation of udp through client and server.

**ALGORITHM:**

**SERVER:**

**STEP1**:Create a server socket and bind it to port.

**STEP 2**:Listen for new connection and when a connection arrives, accept it.

**STEP 3:**Send message to the server.

**STEP 4**:Read client’s IP address sent by the client.

**STEP 5**:Display the client details.

**STEP 6**:Repeat steps 2-5 until the server is terminated.

**STEP 7:**Close the server socket.

**CLIENT**

**STEP1:**Create a client socket and connect it to the server’s port number.

**STEP2**:Retrieve its own IP address using built-in function.

**STEP3**:Send its address to the server.

**STEP4**:Display message send by server.

**STEP5**:Close the client socket

**PROGRAM:**

**SERVER:**

import java.io.\*;

import java.net.\*;

class UDPServer

{

public static DatagramSocket serversocket;

public static DatagramPacket dp;

public static BufferedReader dis;

public static InetAddress ia;

public static byte buf[] = new byte[1024];

public static int cport = 789,sport=790;

public static void main(String[] a) throws IOException

{

serversocket = new DatagramSocket(sport);

dp = new DatagramPacket(buf,buf.length);

dis = new BufferedReader

(new InputStreamReader(System.in));

ia = InetAddress.getLocalHost();

System.out.println("Server is Running...");

while(true)

{

serversocket.receive(dp);

String str = new String(dp.getData(), 0,

dp.getLength());

if(str.equals("STOP"))

{

System.out.println("Terminated...");

break;

}

System.out.println("Client: " + str);

String str1 = new String(dis.readLine());

buf = str1.getBytes();

serversocket.send(new

DatagramPacket(buf,str1.length(), ia, cport));

}

}

}

**CLIENT:**

import java.io.\*;

import java.net.\*;

class UDPClient

{

public static DatagramSocket clientsocket;

public static DatagramPacket dp;

public static BufferedReader dis;

public static InetAddress ia;

public static byte buf[] = new byte[1024];

public static int cport = 789, sport = 790;

public static void main(String[] a) throws IOException

{

clientsocket = new DatagramSocket(cport);

dp = new DatagramPacket(buf, buf.length);

dis = new BufferedReader(new

InputStreamReader(System.in));

ia = InetAddress.getLocalHost();

System.out.println("Client is Running... Type 'STOP'to Quit");

while(true)

{

String str = new String(dis.readLine());

buf = str.getBytes();

if(str.equals("STOP"))

{

System.out.println("Terminated...");

clientsocket.send(new

DatagramPacket(buf,str.length(), ia,

sport));

break;

}

clientsocket.send(new DatagramPacket(buf,

str.length(), ia, sport));

clientsocket.receive(dp);

String str2 = new String(dp.getData(), 0,

dp.getLength());

System.out.println("Server: " + str2);

}

}

}

**OUTPUT:**

**CLIENT**

project@ubuntu:~/lab/udp$ javac udpclient.java

project@ubuntu:~/lab/udp$ java udpclient

Hello message sent.

Hello

**SERVER**

project@ubuntu:~/lab/udp$ javac udpserver.java

project@ubuntu:~/lab/udp$ java udpserver

hello

**RESULT:**

Thus the program has been implemented for udp communication between sever and client using java

**EXNO:7 IMPLEMENTATION OF TCP**

**AIM:**

To write a program for implementation of tcp.

**ALGORITHM:**

**SERVER:**

**STEP1**:Create a server socket and bind it to port.

**STEP 2**:Listen for new connection and when a connection arrives, accept it.

**STEP 3:**Send message to the server.

**STEP 4**:Read client’s IP address sent by the client.

**STEP 5**:Display the client details.

**STEP 6**:Repeat steps 2-5 until the server is terminated.

**STEP 7:**Close the server socket.

**CLIENT**

**STEP1**:Create a client socket and connect it to the server’s port number.

**STEP2**:Retrieve its own IP address using built-in function.

**STEP3**:Send its address to the server.

**STEP4**:Display message send by server.

**STEP5**:Close the client socket

**CLIENT**

import java.net.\*;

import java.io.\*;

public class ReverseClient {

public static void main(String[] args) {

if (args.length < 2) return;

String hostname = args[0];

int port = Integer.parseInt(args[1]);

try (Socket socket = new Socket(hostname, port)) {

OutputStream output = socket.getOutputStream();

PrintWriter writer = new PrintWriter(output, true);

Console console = System.console();

String text;

do {

text = console.readLine("Enter text: ");

writer.println(text);

InputStream input = socket.getInputStream();

BufferedReader reader = new BufferedReader(new InputStreamReader(input));

String time = reader.readLine();

System.out.println(time);

} while (!text.equals("bye"));

socket.close();

} catch (UnknownHostException ex) {

System.out.println("Server not found: " + ex.getMessage());

} catch (IOException ex) {

System.out.println("I/O error: " + ex.getMessage());

}

}

}

**SERVER**

import java.io.\*;

import java.net.\*;

public class ReverseServer {

public static void main(String[] args) {

if (args.length < 1) return;

int port = Integer.parseInt(args[0]);

try (ServerSocket serverSocket = new ServerSocket(port)) {

System.out.println("Server is listening on port " + port);

while (true) {

Socket socket = serverSocket.accept();

System.out.println("New client connected");

InputStream input = socket.getInputStream();

BufferedReader reader = new BufferedReader(new InputStreamReader(input));

OutputStream output = socket.getOutputStream();

PrintWriter writer = new PrintWriter(output, true);

String text;

do {

text = reader.readLine();

writer.println("Server: " +text);

} while (!text.equals("bye"));

socket.close();

}

} catch (IOException ex) {

System.out.println("Server exception: " + ex.getMessage());

ex.printStackTrace();

}

}

}

CLIENT

project@ubuntu:~/lab/tcp$ java ReverseClient localhost 8056

Enter text: hello

Server: hello

Enter text: hi

Server: hi

SERVER

project@ubuntu:~/lab/tcp$ java ReverseServer 8056

Server is listening on port 8056

New client connected

Hello

Hi

**RESULT**

Thus the program has been implemented for tcp communication between sever and client using java.

**EX NO:8 STUDY OF BASIC CONCEPTS OF NETWORK SIMULATOR (NS2), ITS INSTALLATION AND WORKING ENVIRONMENT.**

**AIM:**

To Study about Basic concepts of Network Simulator (NS2), its installation and working environment.

**I. Basic concepts of Network Simulator (NS2):**

* NS2 stands for Network Simulator Version 2. It is an open-source event-driven simulator designed specifically for research in computer communication networks.
* Network simulator is a discrete event network simulator.
* A network simulator is software that predicts the behavior of a computer network. Network simulator is used to understand system behavior accurately.
* In simulators, the computer network is typically modeled with devices, links, and applications to analyze the performance.
* Simulators provide support for the most popular technologies and networks.

**Use of Network Simulator**

* Network simulators provide a cost effective method for network design and validation for sensor networks
* Facility for addition or modification to exiting network simulator must enable user to model the network topology to specify the nodes and the links between
* Those nodes Model the application flow (traffic) between the nodes.
* Provides network performance metrics as output
* Visualization and animation of packet flow.

### Features of NS2

1. It is a discrete event simulator for networking research.

2. It provides substantial support to simulate bunch of protocols like TCP, FTP, UDP, HTTP and DSR.

3. It simulates wired and wireless network.

4. It is primarily Unix based.

5. Uses TCL as its scripting language.

6. Otcl: Object oriented support

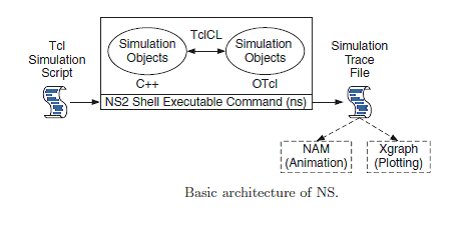
7. Tclcl: C++ and otcl linkage

8. Discrete event scheduler

### 

### Basic Architecture

NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events. The C++ and the OTcl are linked together using TclCL. Ns2 is basically an Object Oriented TCL (OTcl) script interpreter with network simulation event scheduler, network component object libraries and network setup (plumbing) module libraries. Network simulator is used for setting up and running a network simulation and user writes a simulation program in OTcl script language. The OTcl script is used to initiate the event scheduler, setup the network topology and tell traffic source when to start and stop sending packets through event scheduler



**II. INSTALLATION**

### Installing NS2 on windows 7

* NS2 builds and runs under windows using Cygwin.
* Cygwin provides Linux like environment under windows.
* System Requirements: A computer with C++ compiler. Building full NS2 package requires large memory space approximately 250MB

#### A. Steps to install NS 2 on windows 7

1. Download Cygwin from following link http://www.cygwin.com/setup.exe  
2. Run the downloaded setup.exe and you will see screen shown below click next.

3. Select option “Install From Internet”. If you have already downloaded the package select “Install from local directory” and click next.

4. Keep the default installation directory as “C:\cygwin” and click next.

5. Keep default local package directory as your download folder and click next.

6.  Next screen will ask for your Internet connection type keep it as “Direct connection” and click next and in next screen choose one site to download the packages and click next.

7. In next screen Cygwin will allow to select the packages you want to install .

8. Uncheck the option “Hide obsolete packages” then click on “view” button till the word “category” changes to “Full”.

To install NS2 you need to select and install following packages:  
gcc  
gcc-g++  
gnuplot  
make  
patch  
perl  
tar  
X-startup-scripts  
xorg-x11-base  
xorg-x11-bin  
xorg-x11-devel  
xorg-x11-bin-dlls  
xorg-x11-bin-lndir  
xorg-x11-etc  
xorg-x11-fenc  
xorg-x11-fnts  
xorg-x11-libs-data  
xorg-x11-xwin  
libxt-devel  
libXmu-devel  
  
To select a package search the package name and click on word “skip” this will change it to version number of the package. Do this for all above packages and click next to start download and installation .

9. Once installation is complete create desktop icons if you need.

NOTE: If you missed any package while installing Cygwin first time you can install it by running the setup.exe again and selecting the package in step 8.

10. Cygwin installation is complete now you can run Cygwin from desktop and see its interface.

**B. Steps is to install NS2**

1. Download NS2 from following link:  http://www.isi.edu/nsnam/dist/ns-allinone-2.28.tar.gz

2. Decompress the file use winrar. Copy the decompressed folder the Cygwin installation directory under the subdirectory home. It will be C:\cygwin\home\system\_name : where system\_name is name of your system in above Cygwin installation this path will be C:\Cygwin\home\sys27

3. Run Cygwin from desktop and change the directory to folder you copied just now in step 2 command to change directory:  
    cd /home/sys27/ns-allinone-2.28

    NOTE: please change sys27 to name of your system

4. To start installation type following command:    "./install"     (WITHOUT qoutes)

This will began the installation process if any Cygwin package is missing it will be reported to you if so the run Cygwin setu.exe and install the missing    package and start again from step 2.

5. Add following lines to the .bashrc  
  
export NS\_HOME=/home/sys27/ns-allinone-2.28  
export PATH=$NS\_HOME/nam-1.11:$NS\_HOME/tcl8.4.5/unix:$NS\_HOME/tk8.4.5/unix:$NS\_HOME/bin:$PATH  
export LD\_LIBRARY\_PATH=$NS\_HOME/tcl8.4.5/unix:$NS\_HOME/tk8.4.5/unix:$NS\_HOME/otcl-1.9:$NS\_HOME/lib:$LD\_LIBRARY\_PATH  
export TCL\_LIBRARY=$NS\_HOME/tcl8.4.5/library  
  
NOTE: replace sys27 with your system name.

**Working environment.**

### Why NS2 uses two languages? (TCL and C++)

NS2 uses OTcl to create and configure a network, and uses C++ to run simulation. All C++ codes need to be compiled and linked to create an executable file.  
**Use OTcl**

 - For configuration, setup, or one time simulation, or

 - To run simulation with existing NS2 modules.

This option is preferable for most beginners, since it does not involve complicated internal mechanism of NS2. Unfortunately, existing NS2 modules are fairly limited. This option is perhaps not sufficient for most researchers.

**Use C++**

- When you are dealing with a packet, or – when you need to modify existing NS2 modules.

This option perhaps discourages most of the beginners from using NS2. This book particularly aims at helping the readers understand the structure of NS2 and feel more comfortable in modifying NS2 modules.

**NETWORK ANIMATOR [NAM]**

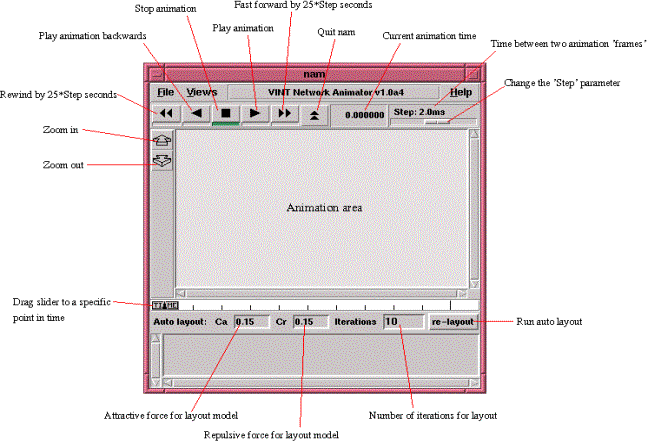
NAM is a **Tcl/TK based animation tool** for viewing network simulation traces and real world packet traces. It supports topology layout, packet level animation, and various data inspection tools.It has a graphical interface, which can provide information such as number of packets drops at each link. The network animator "NAM'' began in 1990 as a simple tool for animating packet trace data. Nam began at LBL. It has evolved substantially over the past few years.

We can either start NAM with the command

**'nam<nam-file>'**

where '<nam-file>' is the name of a NAM trace file that was generated by NS or one can execute it directly out of the Tcl simulation script for visualization of node movement.

NAM window is showed on the following figure;



**Structure of NS2**● NS is an object oriented discrete event simulator

– Simulator maintains list of events and executes one event after another

– Single thread of control: no locking or race conditions

● Back end is C++ event scheduler

– Protocols mostly

– Fast to run, more control

* Front end is OTCL

Creating scenarios, extensions to C++ protocols

**Platforms:** It can be employed in most unix systems (FreeBSD, Linux, Solaris) and Windows.

**Source code:** Most of NS2 code is in C++

**Scripting language:** It uses TCL as its scripting language OTcl adds object orientation to TCL.

**Protocols implemented in NS2**

Transport layer(Traffic Agent) – TCP, UDP

Network layer(Routing agent)

Interface queue – FIFO queue, Drop Tail queue, Priority queue

Logic link contol layer – IEEE 802.2, AR

**NS programming Structure**

● Create the event scheduler

● Turn on tracing

● Create network topology

● Create transport connections

● Generate traffic

● Insert errors

**Creating Event Scheduler**

● Create event scheduler: set ns [new simulator]

● Schedule an event: $ns at <time><event>

– event is any legitimate ns/tcl function

$ns at 5.0 “finish”

proc finish {} {

global ns nf

close $nf

exec nam out.nam &

exit 0

}

● Start Scheduler

$ns run

**Tracing**

● All packet trace

$ns traceall[open out.tr w]

<event><time><from><to><pkt><size>

<flowid><src><dst><seqno><aseqno>

+  0.51  0  1 cbr  500 —–  0  0.0  1.0   0   2

\_   0.51  0  1 cbr  500 —– 0   0.0  1.0  0   2

R  0.514 0  1 cbr  500 —–0   0.0   1.0  0   0

● Variable trace

set par [open output/param.tr w]

$tcp attach $par

$tcp trace cwnd\_

$tcp trace maxseq\_

$tcp trace rtt\_

**Tracing and Animation**

● Network Animator

set nf [open out.nam w]

$ns namtraceall

$nf

proc finish {} {

global ns nf

close $nf

exec nam out.nam &

exit 0 }

**Creating topology**

● Two nodes connected by a link

● Creating nodes

set n0 [$ns node]

set n1 [$ns node]

● Creating link between nodes

$ns <link\_type> $n0 $n1 <bandwidth><delay><queue-type>

$ns duplex-link$n0 $n1 1Mb 10ms DropTail

**Data Sending**

**●**Create UDP agent

set udp0 [new Agent/UDP]

$ns attach-agent $n0 $udp0

**●**Create CBR traffic source for feeding into UDP agent

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent$udp0

**●**Create traffic sink

set null0 [new Agent/Null]

$ns attach-agent$n1 $null0

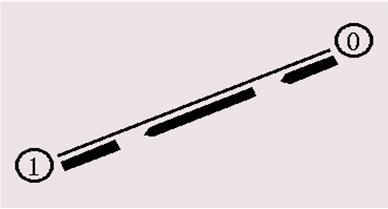
● Connect two agents

$ns connect $udp0 $null0

● Start and stop of data

$ns at 0.5 “$cbr0 start”

$ns at 4.5 “$cbr0 stop”

[](https://i0.wp.com/www.eexploria.com/wp-content/uploads/2012/03/NS2-programs.jpg)

**RESULT:**

Thus the basic concepts, its installation procedure and working environment of NS2 was studied successfully.

**EX NO:9 CREATION OF SIMPLE TOPOLOGY**

**AIM :**

To Write a simple program for creation of topology using ns2

**ALGORITHM**

**STEP 1**: define the procedure as finish

**STEP 2**: close all the tracefile,trifle and nam file also

**STEP 3**: create a new simulator and assign the variable simulator

**STEP 4**: create a new file outer and keep it as x.

**STEP 5:** Store all trifle in traceall

**STEP 6**: Store all nam file in nam for the result.

**STEP 7**:connect UDP and null

**STEP 8**: stop the program.

**PROGRAM**

set ns [new Simulator]

#define color for data flows

$ns color 1 Blue

$ns color 2 Red

#open tracefiles

set tracefile1 [open out.tr w]

set winfile [open winfile w]

$ns trace-all $tracefile1

#open nam file

set namfile [open out.nam w]

$ns namtrace-all $namfile

#define the finish procedure

proc finish {} {

global ns tracefile1 namfile

$ns flush-trace

close $tracefile1

close $namfile

exec nam out.nam &

exit 0

}

#create six 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]

$n1 color Red

$n1 shape box

#create links between the nodes

$ns duplex-link $n0 $n2 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 2Mb 10ms DropTail

$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail

$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail

set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]

#Give node position

$ns duplex-link-op $n0 $n2 orient right-down

$ns duplex-link-op $n1 $n2 orient right-up

$ns simplex-link-op $n2 $n3 orient right

$ns simplex-link-op $n3 $n2 orient left

#set queue size of link(n2-n3) to 20

$ns queue-limit $n2 $n3 20

#setup TCP connection

set tcp [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 packet\_size\_ 552

#set ftp over tcp connection

set ftp [new Application/FTP]

$ftp attach-agent $tcp

#setup a UDP connection

set udp [new Agent/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

#scheduling the events

$ns at 0.1 "$cbr start"

$ns at 1.0 "$ftp start"

$ns at 124.0 "$ftp stop"

$ns at 125.5 "$cbr stop"

proc plotWindow {tcpSource file} {

global ns

set time 0.1

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

puts $file "$now $cwnd"

$ns at [expr $now+$time] "plotWindow $tcpSource $file"

}

$ns at 0.1 "plotWindow $tcp $winfile"

$ns at 125.0 "finish"

$ns run

**DELAY**

BEGIN

{

seqno = 0;

droppedPackets = 0;

receivedPackets = 0;

count = 0;

}

#packet delivery ratio

if($1 == "s" && seqno < $6)

{

seqno = $6;

}

else if($1 == "r")

{

receivedPackets++;

}

else if ($1 == "D" && $8 > 512)

{

droppedPackets++;

}

#end-to-end delayif($1 == "s")

{

Start\_time[$6] = $2;

}

else if(($1 == "r"))

{

end\_time[$6] = $2;

}

else if($1 == "D")

{

end\_time[$6] = -1;

}

}

END

{

for(i=0; i<=seqno; i++)

{

if(end\_time[i] > 0)

{

delay[i] = end\_time[i] - start\_time[i]; count++;

}

Else

{

delay[i] = -1;

}

}

for(i=0; i<count; i++)

{

if(delay[i] > 0)

{

n\_to\_n\_delay = n\_to\_n\_delay + delay[i];

}

n\_to\_n\_delay = n\_to\_n\_delay/count;

}

printf( "\n")

printf("GeneratedPackets =%d\n", seqno+1)

printf("ReceivedPackets = %d\n", receivedPackets)

printf("Packet Delivery Ratio = %.2f\n", receivedPackets/(seqno+1)\*100"%")

printf("Packet Loss = %d\n", droppedPackets)

printf("Average End-to-End Delay= %.2f\n", n\_to\_n\_delay \* 1000 " ms")

printf("\n") }

**THROUGUPUT:**

BEGIN

{

recvdSize = 0

txsize=0

drpSize=0

startTime = 400

stopTime = 0

thru=0

}

event = $1time = $2node\_id = $4pkt\_size = $6level = $5#

Store start timeif (event == "s" && pkt\_size >=512 )

{

if (time < startTime)

{

startTime = time

}

}

# Update total received packets size and store packets arrival time

if (event == "r" && pkt\_size >=512 )

{

if (time > stopTime)

{

stopTime = time

}

# Rip off the header

# hdr\_size = pkt\_size % 512

# pkt\_size -= hdr\_size

# Store received packets sizerecvdSize += pkt\_size

}

}

END {

printf("Average Throughput[kbps] = %.2f\t\t StartTime=%.2f\tStopTime=%.2f\n",(recvdSize/(stopTimestartTime)\*3/1000),startTime,stopTime)

}

OUTPUT:

|  |  |
| --- | --- |
|  |  |
|  |  |

**RESULT:**

Thus the ns2 simulation for creation of topology is excueted.

**EXNO:10 SIMULATION OF DUPLEX NETWORK**

**AIM:**

To Simulate a four Duplex network and apply TCP agent between two nodes and UDP agents between other two nodes and by changing the parameters, determine the number of packets sent and dropped by TCP/UDP.

**ALGORITHM:**

**STEP 1**:Create a simulator object

**STEP 2**: open namfile,tracefile

**STEP 3**:Define Finish procedure

**STEP 4**:Create four node n0,n1,n2,n3,n4

**STEP 5**:create link between node and set queue size of link

**STEP 6:** Set up the TCP connection and set up a FTP over TCP connection

**STEP 7**:schedules for cbr agent

**STEP 8**: Detach the Tcp and link agent

**STEP 9**:print the packet size

**ALGORITHM FOR CALCULATE NUMBER OF PACKET SEND AND DROPPED PACKET**

**STEP 1**: start the program

**STEP 2**:Initialize the UDP of received packet .dropped packet,TCP received and dropped packet

**STEP 3**:Assign column 1 as action and column 2 as type

**STEP 4**:Check if action ==”r” and type =”cbr” then increment the received packet

**STEP 5:** check is action=”d” and tye=”tcp” increment yhe tcp recived packet

**PROGRAM:**

#===================================

# Simulation parameters setup

#===================================

set val(stop) 5.0 ;# time of simulation end

#===================================

# Initialization

#===================================

#Create a ns simulator

set ns [new Simulator]

#Open the NS trace file

set tracefile [open out.tr w]

$ns trace-all $tracefile

#Open the NAM trace file

set namfile [open out.nam w]

$ns namtrace-all $namfile

#===================================

# Nodes Definition

#===================================

#Create 6 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]

$n1 color red

$n1 shape box

$n0 color blue

$n0 shape hexagon

#===================================

# Links Definition

#===================================

#Createlinks between nodes

$ns duplex-link $n0 $n2 100.0Mb 10ms DropTail

$ns queue-limit $n0 $n2 30

$ns duplex-link $n1 $n2 100.0Mb 10ms DropTail

$ns queue-limit $n1 $n2 30

$ns duplex-link $n3 $n4 100.0Mb 10ms DropTail

$ns queue-limit $n3 $n4 30

$ns duplex-link $n3 $n5 100.0Mb 10ms DropTail

$ns queue-limit $n3 $n5 30

$ns simplex-link $n2 $n3 100.0Mb 10ms DropTail

$ns queue-limit $n2 $n3 20

$ns simplex-link $n3 $n2 100.0Mb 10ms DropTail

$ns queue-limit $n3 $n2 20

#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 $n3 $n4 orient right-up

$ns duplex-link-op $n3 $n5 orient right-down

$ns simplex-link-op $n2 $n3 orient right

$ns simplex-link-op $n3 $n2 orient left

#===================================

# Agents Definition

#===================================

#Setup a TCP connection

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set sink1 [new Agent/TCPSink]

$ns attach-agent $n4 $sink1

$ns connect $tcp0 $sink1

$tcp0 set packetSize\_ 1500

#Setup a UDP connection

set udp2 [new Agent/UDP]

$ns attach-agent $n1 $udp2

set null3 [new Agent/Null]

$ns attach-agent $n5 $null3

$ns connect $udp2 $null3

$udp2 set packetSize\_ 1500

#===================================

# Applications Definition

#===================================

#Setup a FTP Application over TCP connection

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

$ns at 1.0 "$ftp0 start"

$ns at 2.0 "$ftp0 stop"

#Setup a CBR Application over UDP connection

set cbr1 [new Application/Traffic/CBR]

$cbr1 attach-agent $udp2

$cbr1 set packetSize\_ 1000

$cbr1 set rate\_ 1.0Mb

$cbr1 set random\_ null

$ns at 2.0 "$cbr1 start"

$ns at 4.0 "$cbr1 stop"

#===================================

# Termination

#===================================

#Define a 'finish' procedure

proc finish {} {

global ns tracefile namfile

$ns flush-trace

close $tracefile

close $namfile

exec nam out.nam &

exit 0

}

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "finish"

$ns at $val(stop) "puts \"done\" ; $ns halt"

$ns run

**OUTPUT**

|  |  |
| --- | --- |
|  |  |
|  |  |

**RESULT**

Thus the simulation for duplex network with four node is implemented.

**EXNO:11 SIMULATE A WIRED NETWORK AND MEASURE THE THROUGHPUT,DELAY,PACKE LOSS**

**AIM**

To Simulate a wired network and measure the Throughput,DELAY,PACKE LOSS

**ALGORITHM**

**STEP 1**: Create a simulator object instance

**STEP 2:**open the nam file and tracefile

**STEP 3**: define the finsh procedure

**STEP 4:**create four node

**STEP 5**:create the lnk between the node

**STEP 6:**set queue size of link (n2-n3) as 10

**STEP 7:**give the node position to be displayed in nam

**STEP 8**: setup th tcp connection between ther node

**STEP 9:**Setup the udp connection between othe node

**STEP 10**:setup agent ftp over tcp and cbr over udp connection

**STEP 11**:run the simulator

**CALCULATE NO OF PACKET SENT,RECEIVED AND DROPPED**

**STEP 1**: start the program

**STEP 2:**Initialize the UDP of received packet .dropped packet,TCP received and dropped packet

**STEP 3**:Assign column 1 as action and column 2 as type

**STEP 4**:Check if action ==”r” and type =”cbr” then increment the received packet

**STEP 5**: check is action=”d” and tye=”tcp” increment the tcp recived packet

**STEP 6**:stop the program

**CALCULATE THROUGHTPUT**

**STEP 1**: start the program

**STEP 2**: if action $”r” and $”s and if s==0 assign s=1and start

**STEP 3:**Assign the time

**STEP 4**:display throughtput of flow

**PROGRAM**

#===================================

# Simulation parameters setup

#===================================

set val(stop) 5.0 ;# time of simulation end

#===================================

# Initialization

#===================================

#Create a ns simulator

set ns [new Simulator]

#Open the NS trace file

set tracefile [open out.tr w]

$ns trace-all $tracefile

#Open the NAM trace file

set namfile [open out.nam w]

$ns namtrace-all $namfile

#===================================

# Nodes Definition

#===================================

#Create 10 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]

set n6 [$ns node]

set n7 [$ns node]

set n8 [$ns node]

set n9 [$ns node]

#===================================

# Links Definition

#===================================

#Createlinks between nodes

$ns duplex-link $n0 $n1 100.0Mb 10ms DropTail

$ns queue-limit $n0 $n1 50

$ns duplex-link $n1 $n2 100.0Mb 10ms DropTail

$ns queue-limit $n1 $n2 50

$ns duplex-link $n2 $n3 100.0Mb 10ms DropTail

$ns queue-limit $n2 $n3 50

$ns duplex-link $n3 $n9 100.0Mb 10ms DropTail

$ns queue-limit $n3 $n9 50

$ns duplex-link $n9 $n8 100.0Mb 10ms DropTail

$ns queue-limit $n9 $n8 50

$ns duplex-link $n8 $n7 100.0Mb 10ms DropTail

$ns queue-limit $n8 $n7 50

$ns duplex-link $n0 $n6 100.0Mb 10ms DropTail

$ns queue-limit $n0 $n6 50

$ns duplex-link $n6 $n5 100.0Mb 10ms DropTail

$ns queue-limit $n6 $n5 50

$ns duplex-link $n5 $n4 100.0Mb 10ms DropTail

$ns queue-limit $n5 $n4 50

$ns duplex-link $n4 $n3 100.0Mb 10ms DropTail

$ns queue-limit $n4 $n3 50

$ns duplex-link $n5 $n0 100.0Mb 10ms DropTail

$ns queue-limit $n5 $n0 50

$ns duplex-link $n7 $n5 100.0Mb 10ms DropTail

$ns queue-limit $n7 $n5 50

$ns duplex-link $n5 $n1 100.0Mb 10ms DropTail

$ns queue-limit $n5 $n1 50

$ns duplex-link $n5 $n8 100.0Mb 10ms DropTail

$ns queue-limit $n5 $n8 50

$ns duplex-link $n4 $n2 100.0Mb 10ms DropTail

$ns queue-limit $n4 $n2 50

$ns duplex-link $n4 $n9 100.0Mb 10ms DropTail

$ns queue-limit $n4 $n9 50

$ns duplex-link $n6 $n7 100.0Mb 10ms DropTail

$ns queue-limit $n6 $n7 50

#Give node position (for NAM)

$ns duplex-link-op $n0 $n1 orient right

$ns duplex-link-op $n1 $n2 orient right

$ns duplex-link-op $n2 $n3 orient right-down

$ns duplex-link-op $n3 $n9 orient left-down

$ns duplex-link-op $n9 $n8 orient left

$ns duplex-link-op $n8 $n7 orient left

$ns duplex-link-op $n0 $n6 orient left-down

$ns duplex-link-op $n6 $n5 orient right

$ns duplex-link-op $n5 $n4 orient right

$ns duplex-link-op $n4 $n3 orient right

$ns duplex-link-op $n5 $n0 orient left-up

$ns duplex-link-op $n7 $n5 orient right-up

$ns duplex-link-op $n5 $n1 orient right-up

$ns duplex-link-op $n5 $n8 orient right-down

$ns duplex-link-op $n4 $n2 orient right-up

$ns duplex-link-op $n4 $n9 orient right-down

$ns duplex-link-op $n6 $n7 orient right-down

#===================================

# Agents Definition

#===================================

#Setup a TCP connection

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set sink1 [new Agent/TCPSink]

$ns attach-agent $n3 $sink1

$ns connect $tcp0 $sink1

$tcp0 set packetSize\_ 1500

#Setup a TCP connection

set tcp2 [new Agent/TCP]

$ns attach-agent $n2 $tcp2

set sink3 [new Agent/TCPSink]

$ns attach-agent $n8 $sink3

$ns connect $tcp2 $sink3

$tcp2 set packetSize\_ 1500

#Setup a TCP connection

set tcp4 [new Agent/TCP]

$ns attach-agent $n8 $tcp4

set sink5 [new Agent/TCPSink]

$ns attach-agent $n1 $sink5

$ns connect $tcp4 $sink5

$tcp4 set packetSize\_ 1500

#Setup a TCP connection

set tcp6 [new Agent/TCP]

$ns attach-agent $n5 $tcp6

set sink7 [new Agent/TCPSink]

$ns attach-agent $n0 $sink7

$ns connect $tcp6 $sink7

$tcp6 set packetSize\_ 1500

#Setup a UDP connection

set udp8 [new Agent/UDP]

$ns attach-agent $n6 $udp8

set null9 [new Agent/Null]

$ns attach-agent $n3 $null9

$ns connect $udp8 $null9

$udp8 set packetSize\_ 1500

#Setup a UDP connection

set udp10 [new Agent/UDP]

$ns attach-agent $n7 $udp10

set null11 [new Agent/Null]

$ns attach-agent $n7 $null11

$ns connect $udp10 $null11

$udp10 set packetSize\_ 1500

#Setup a UDP connection

set udp12 [new Agent/UDP]

$ns attach-agent $n4 $udp12

set null13 [new Agent/Null]

$ns attach-agent $n6 $null13

$ns connect $udp12 $null13

$udp12 set packetSize\_ 1500

#Setup a UDP connection

set udp14 [new Agent/UDP]

$ns attach-agent $n9 $udp14

set null15 [new Agent/Null]

$ns attach-agent $n0 $null15

$ns connect $udp14 $null15

$udp14 set packetSize\_ 1500

#Setup a UDP connection

set udp16 [new Agent/UDP]

$ns attach-agent $n5 $udp16

set null17 [new Agent/Null]

$ns attach-agent $n7 $null17

$ns connect $udp16 $null17

$udp16 set packetSize\_ 1500

#===================================

# Applications Definition

#===================================

#Setup a FTP Application over TCP connection

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

$ns at 1.0 "$ftp0 start"

$ns at 2.0 "$ftp0 stop"

#Setup a FTP Application over TCP connection

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp2

$ns at 1.0 "$ftp1 start"

$ns at 2.0 "$ftp1 stop"

#Setup a FTP Application over TCP connection

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp6

$ns at 1.0 "$ftp2 start"

$ns at 2.0 "$ftp2 stop"

#Setup a FTP Application over TCP connection

set ftp3 [new Application/FTP]

$ftp3 attach-agent $tcp4

$ns at 1.0 "$ftp3 start"

$ns at 2.0 "$ftp3 stop"

#Setup a CBR Application over UDP connection

set cbr4 [new Application/Traffic/CBR]

$cbr4 attach-agent $udp8

$cbr4 set packetSize\_ 1000

$cbr4 set rate\_ 1.0Mb

$cbr4 set random\_ null

$ns at 2.0 "$cbr4 start"

$ns at 4.0 "$cbr4 stop"

#Setup a CBR Application over UDP connection

set cbr5 [new Application/Traffic/CBR]

$cbr5 attach-agent $udp10

$cbr5 set packetSize\_ 1000

$cbr5 set rate\_ 1.0Mb

$cbr5 set random\_ null

$ns at 2.0 "$cbr5 start"

$ns at 4.0 "$cbr5 stop"

#Setup a CBR Application over UDP connection

set cbr6 [new Application/Traffic/CBR]

$cbr6 attach-agent $udp14

$cbr6 set packetSize\_ 1000

$cbr6 set rate\_ 1.0Mb

$cbr6 set random\_ null

$ns at 2.0 "$cbr6 start"

$ns at 4.0 "$cbr6 stop"

#Setup a CBR Application over UDP connection

set cbr7 [new Application/Traffic/CBR]

$cbr7 attach-agent $udp12

$cbr7 set packetSize\_ 1000

$cbr7 set rate\_ 1.0Mb

$cbr7 set random\_ null

$ns at 2.0 "$cbr7 start"

$ns at 4.0 "$cbr7 stop"

#Setup a CBR Application over UDP connection

set cbr8 [new Application/Traffic/CBR]

$cbr8 attach-agent $udp16

$cbr8 set packetSize\_ 1000

$cbr8 set rate\_ 1.0Mb

$cbr8 set random\_ null

$ns at 2.0 "$cbr8 start"

$ns at 4.0 "$cbr8 stop"

#===================================

# Termination

#===================================

#Define a 'finish' procedure

proc finish {} {

global ns tracefile namfile

$ns flush-trace

close $tracefile

close $namfile

exec nam out.nam &

exit 0

}

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "finish"

$ns at $val(stop) "puts \"done\" ; $ns halt"

$ns run

**DELAY.AWK**

\*\*\*\*\*\*\*\*\*

BEGIN {seqno = 0; droppedPackets = 0;receivedPackets = 0;count = 0;}{#packet delivery ratioif($1 == "s" && seqno < $6) {seqno = $6;}

else if($1 == "r") {receivedPackets++;} else if ($1 == "D" && $8 > 512){droppedPackets++; }#end-to-end delayif($1 == "s") {start\_time[$6] = $2;} else if(($1 == "r")) {end\_time[$6] = $2;} else if($1 == "D"){end\_time[$6] = -1;}}END { for(i=0; i<=seqno; i++)

{if(end\_time[i] > 0){

delay[i] = end\_time[i] - start\_time[i];count++;}else{delay[i] = -1;

}

}

for(i=0; i<count; i++)

{

if(delay[i] > 0)

{

n\_to\_n\_delay = n\_to\_n\_delay + delay[i];

}

n\_to\_n\_delay = n\_to\_n\_delay/count;

}

printf( "\n")

printf("GeneratedPackets =%d\n", seqno+1)

printf("ReceivedPackets = %d\n", receivedPackets)

printf("Packet Delivery Ratio = %.2f\n", receivedPackets/(seqno+1)\*100"%")

printf("Packet Loss = %d\n", droppedPackets)

printf("Average End-to-End Delay= %.2f\n", n\_to\_n\_delay \* 1000 " ms")

printf("\n")

}

**THROUGUPUT**;

\*\*\*\*\*\*\*\*\*\*

BEGIN {

recvdSize = 0txsize=0drpSize=0startTime = 400stopTime = 0thru=0}{Event = $1time = $2node\_id = $4pkt\_size = $6level = $5# Store start timeif (event == "s" && pkt\_size >=512 ) {if (time < startTime) {startTime = time}}# Update total received packets size and store packets arrival timeif (event == "r" && pkt\_size >=512 ) {if (time > stopTime) {stopTime = time}# Rip off the header# hdr\_size = pkt\_size % 512# pkt\_size -= hdr\_size

# Store received packets sizerecvdSize += pkt\_size}}END {printf("Average Throughput[kbps] = %.2f\t\t StartTime=%.2f\tStopTime=%.2f\n",(recvdSize/(stopTime-startTime)\*3/1000),startTime,stopTime)}

**OUTPUT**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

dealy:

[root@localhost ~]# awk -f delay.awk out.tr

GeneratedPackets =10000

ReceivedPackets = 2376

Packet Delivery Ratio = 237600.00

Packet Loss = 0

Average End-to-End Delay= 03.00

[root@localhost ~]#

[root@localhost ~]# awk -f thro.awk out.tr

Average Throughput[kbps] = -1.13

StartTime=1.00

StopTime=2.00

StartTime=2.00

StopTime=4.00

[root@localhost ~]#

RESULT:

Thus the simulation of a wired network and measure the throughput,delay,packet loss is executed.

**EXNO:12 IMPLEMENT LINK STATE ROUTING AND DISTANCE VECTOR ROUTING**

**AIM:**

To Implement Link State routing and Distance Vector routing measure the following performance Metrics

**ALGORITHM**

**STEP 1:**Create a simulator object instance

**STEP 2**: open the nam file and tracefile

**STEP 3:**define the finsh procedure

**STEP 4:**declare distance vector algorithm

**STEP 5:**scheduled the event for cbr and tcp

**STEP 6:**add the agent simulator

**LINK STATE ROUTING**:

**STEP 1:**Create a simulator object instance

**STEP 2:**open the nam file and tracefile

**STEP 3:**define the finsh procedure

**STEP 4:**declare link state algorithm

**STEP 5:**scheduled the event for cbr and tcp

**STEP 6:**add the agent simulator

**PROGRAM:**

**DISTANCE VECTOR**:

set ns [new Simulator]

set nr [open thro.tr w]

$ns trace-all $nr

set nf [open thro.nam w]

$ns namtrace-all $nf

proc finish { } {

global ns nr nf

$ns flush-trace

close $nf

close $nr

exec nam thro.nam &

exit 0

}

for { set i 0 } { $i < 12} { incr i 1 } {

set n($i) [$ns node]}

for {set i 0} {$i < 8} {incr i} {

$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }

$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail

$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail

$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail

$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail

$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail

$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail

set udp0 [new Agent/UDP]

$ns attach-agent $n(0) $udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0

set null0 [new Agent/Null]

$ns attach-agent $n(5) $null0

$ns connect $udp0 $null0

set udp1 [new Agent/UDP]

$ns attach-agent $n(1) $udp1

set cbr1 [new Application/Traffic/CBR]

$cbr1 set packetSize\_ 500

$cbr1 set interval\_ 0.005

$cbr1 attach-agent $udp1

set null0 [new Agent/Null]

$ns attach-agent $n(5) $null0

$ns connect $udp1 $null0

$ns rtproto DV

$ns rtmodel-at 10.0 down $n(11) $n(5)

$ns rtmodel-at 15.0 down $n(7) $n(6)

$ns rtmodel-at 30.0 up $n(11) $n(5)

$ns rtmodel-at 20.0 up $n(7) $n(6)

$udp0 set fid\_ 1

$udp1 set fid\_ 2

$ns color 1 Red

$ns color 2 Green

$ns at 1.0 "$cbr0 start"

$ns at 2.0 "$cbr1 start"

$ns at 45 "finish"

$ns run

LINK STATE

set ns [new Simulator]

set nr [open thro.tr w]

$ns trace-all $nr

set nf [open thro.nam w]

$ns namtrace-all $nf

proc finish { } {

global ns nr nf

$ns flush-trace

close $nf

close $nr

exec nam thro.nam &

exit 0

}

for { set i 0 } { $i < 12} { incr i 1 } {

set n($i) [$ns node]}

for {set i 0} {$i < 8} {incr i} {

$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }

$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail

$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail

$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail

$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail

$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail

$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail

set udp0 [new Agent/UDP]

$ns attach-agent $n(0) $udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0

set null0 [new Agent/Null]

$ns attach-agent $n(5) $null0

$ns connect $udp0 $null0

set udp1 [new Agent/UDP]

$ns attach-agent $n(1) $udp1

set cbr1 [new Application/Traffic/CBR]

$cbr1 set packetSize\_ 500

$cbr1 set interval\_ 0.005

$cbr1 attach-agent $udp1

set null0 [new Agent/Null]

$ns attach-agent $n(5) $null0

$ns connect $udp1 $null0

$ns rtproto LS

$ns rtmodel-at 10.0 down $n(11) $n(5)

$ns rtmodel-at 15.0 down $n(7) $n(6)

$ns rtmodel-at 30.0 up $n(11) $n(5)

$ns rtmodel-at 20.0 up $n(7) $n(6)

$udp0 set fid\_ 1

$udp1 set fid\_ 2

$ns color 1 Red

$ns color 2 Green

$ns at 1.0 "$cbr0 start"

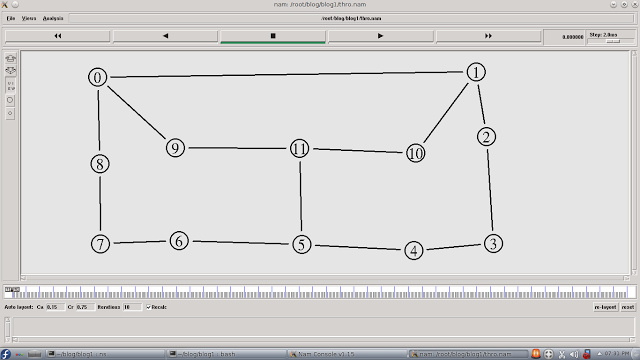
$ns at 2.0 "$cbr1 start"

$ns at 45 "finish"

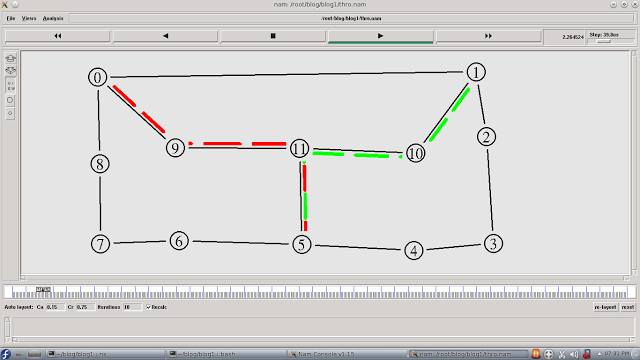
$ns run

OUTPUT:

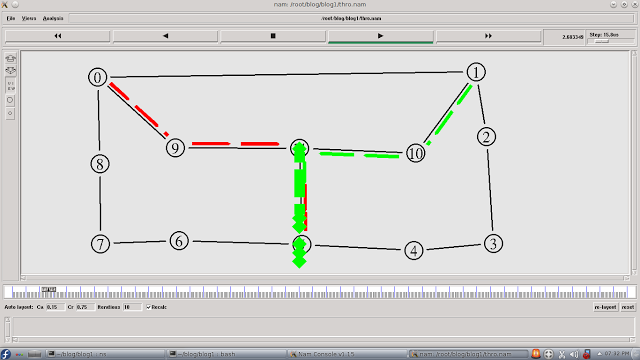
**NS2 : Simulating Distance Vector Routing**



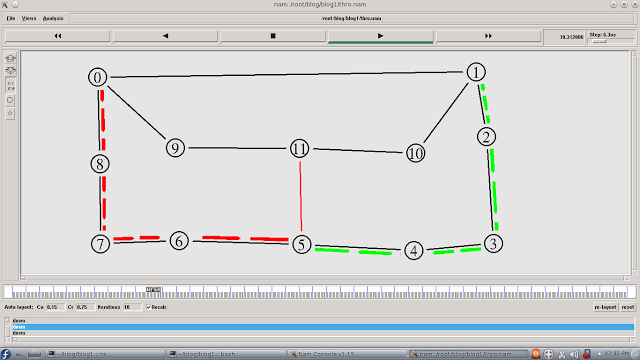
**Data Transfer in NAM**



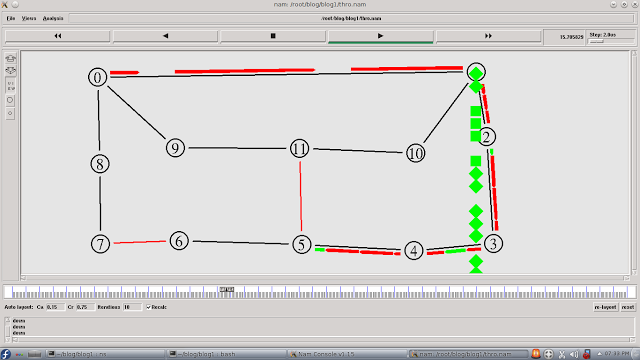
**First Link Fails**



**Data Transfer through Alternate Path**



**Second Link Fails & Data Transfer through Alternate Path**



THROUGHPUT:

[root@localhost ~]# awk -f thro.awk thro.tr

Average Throughput[kbps] = 0.00

StartTime=1.00

StopTime=2.00

Delay:

\*\*\*\*\*\*

[root@localhost ~]# awk -f delay.awk thro.tr

GeneratedPackets =200000

ReceivedPackets = 150000

Packet Delivery Ratio = 75.00

Packet Loss = 1.5

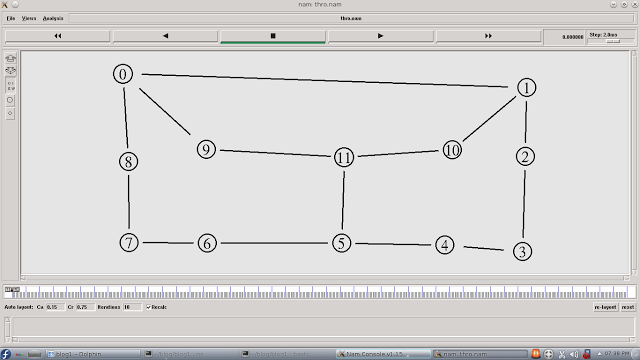
Average End-to-End Delay= 0.00001

[root@localhost ~]#

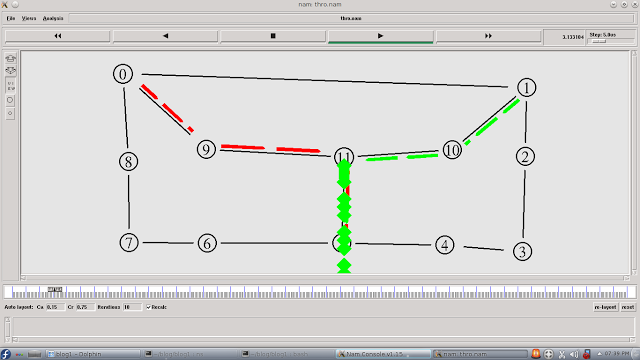
**LINKSTATE:**

**NS2 : Simulating Link State Routing**

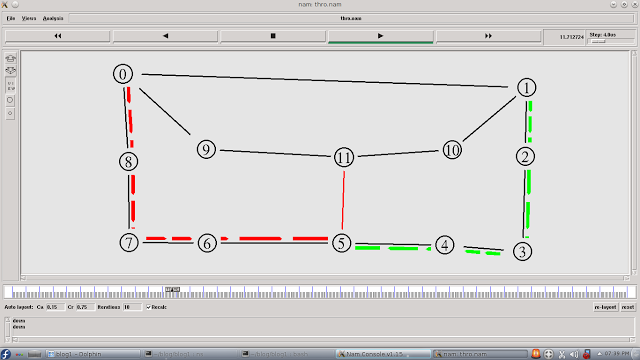
**Network Topology in NAM**



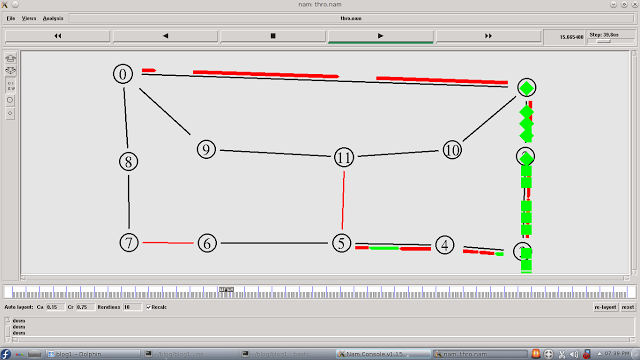
**Data Transfer & Link Failure**



**Data Transfer through Alternate Path**



**Second Link Fails & Data Transfer through Alternate Path**



THROUGHPUT:

[root@localhost ~]# awk -f thro.awk thro.tr

Average Throughput[kbps] = 0.00

StartTime=1.00

StopTime=2.00

Delay:

\*\*\*\*\*\*

[root@localhost ~]# awk -f delay.awk thro.tr

GeneratedPackets =200000

ReceivedPackets = 150000

Packet Delivery Ratio = 75.00

Packet Loss = 1.5

Average End-to-End Delay= 0.00001

**RESULT:**

Thus the simulation for distance vector and link state routing is implemented.

**EX NO:13 EXPERIMENT ON PACKET CAPTURE AND NETWORK TRAFFIC USING WIRE SHARK TOOL.**

**AIM:**

To implement simulation of packet capture and network traffic analysis suing Wire shark Tool.

**ABOUT WIRE SHARK TOOL**

Wireshark is a free application that allows you to capture and view the data traveling back and forth on your network, providing the ability to drill down and read the contents of each packet – filtered to meet your specific needs. It is commonly utilized to troubleshoot network problems as well as to develop and test software. This open-source protocol analyzer is widely accepted as the industry standard, winning its fair share of awards over the years. Wireshark can be downloaded at no cost from the [Wireshark Foundation website](https://wireshark.org/#download) for both macOS and Windows operating systems. Unless you are an advanced user, it is recommended that you only download the latest stable release. During the setup process (Windows only) you should choose to also install WinPcap if prompted, as it includes a library required for live data capture.

**What Wireshark Is Used For?**

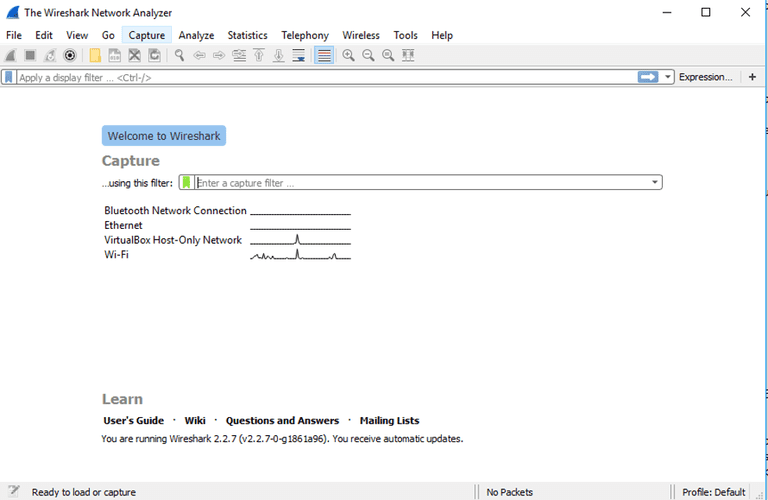
Wireshark has quite an extensive application or use. Here are a few examples of what people use Wireshark for:

* Network administrators use it to troubleshoot network problems
* Network security engineers use it to examine security problems
* Developers use it to debug protocol implementations
* Others use it to learn network protocol internals

**Features at a Glance**

* The following are some of the many features Wireshark provides:  
  Capture live packet data from a network interface.
* Open files containing packet data captured with tcpdump/WinDump, Wireshark, and a number of other packet capture programs.
* Import packets from text files containing hex dumps of packet data.
* Display packets with very detailed protocol information.
* Save packet data captured.
* Export some or all packets in a number of capture file formats.
* Filter packets on many criteria.
* Search for packets on many criteria.
* Colorize packet display based on filters.
* Create various statistics.

**How to Capture Data Packets**



When you first launch Wireshark a welcome screen similar to the one shown above should be visible, containing a list of available network connections on your current device. In this example, you'll notice that the following connection types are shown: Bluetooth Network Connection, Ethernet, VirtualBox Host-Only Network, Wi-Fi. Displayed to the right of each is an EKG-style line graph that represents live traffic on that respective network.

To begin capturing packets, first select one or more of these networks by clicking on your choice(s) and using the Shift or Ctrl keys if you'd like to record data from multiple networks simultaneously. Once a connection type is selected for capturing purposes, its background will be shaded in either blue or gray. Click on Capturefrom the main menu, located towards the top of the Wireshark interface. When the drop-down menu appears, select the Start option.

You can also initiate packet capturing via one of the following shortcuts.

Keyboard: Press Ctrl + E

Mouse: To begin capturing packets from one particular network, simply double-click on its name

Toolbar: Click on the blue shark fin button, located on the far left-hand side of the Wireshark toolbar

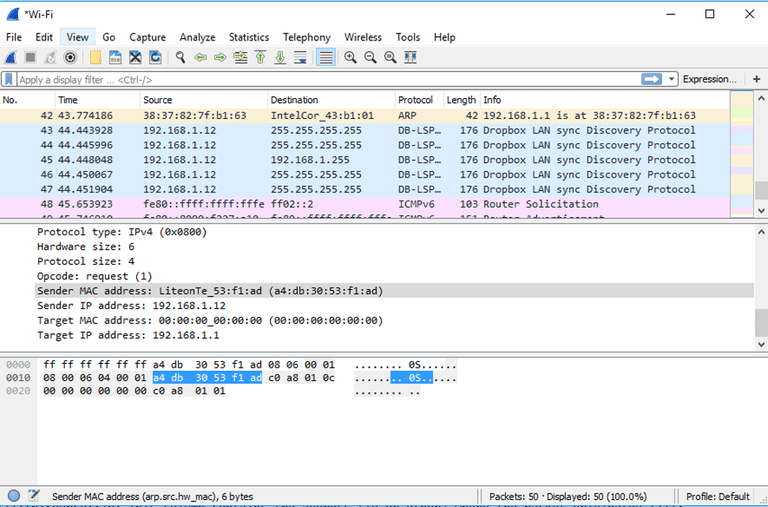
The live capture process will now begin, with packet details displayed in the Wireshark window as they are recorded. Perform one of the actions below to stop capturing.

Keyboard: Press Ctrl + E

Toolbar: Click on the red stop button, located next to the shark fin on the Wireshark toolbar

### OUTPUT:

### Viewing and Analyzing Network Traffic



**RESULT:**

Thus simulation of packet capture and network traffic analysis using Wire shark Tool is implemented.