|  |  |
| --- | --- |
| **Ex. No.: 1** | **NETWORK COMMANDS** |
| **26.12.2022** |

**Aim:**

Use network commands like tcpdump, netstat, ifconfig, etc., and create ping and trace route PDUs.

**Algorithm:**

1. Download the required packages, if not already installed.
2. Run the corresponding Linux commands on the terminal.

**Code:**

**1)** TCPDUMP**:** sudo tcpdump

**2)** NETSTAT: netstat -at, netstat -au, netstat -lt, netstat -lu

**3)** IFCONFIG: ifconfig

**4)** NSLOOKUP: nslookup www.google.com, nslookup lms.snuchennai.edu.in

**5)** TRACEROUTE: traceroute www.google.com

**6)** PING: ping 8.8.8.8

**Input/Output:**

**1) TCPDUMP**Text

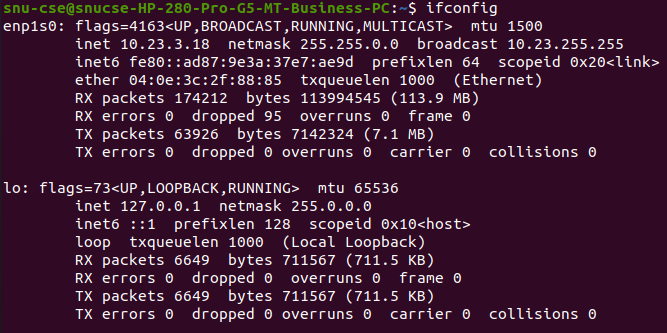
Description automatically generated with medium confidence

1. **NETSTAT**

Text

Description automatically generated

1. **IFCONFIG**



1. Text

   Description automatically generated**NSLOOKUP**
2. Text

   Description automatically generated**TRACEOUTE**
3. **PING**

A picture containing text

Description automatically generated

**Result:**

Network related commands were run on the Linux terminal.

|  |  |
| --- | --- |
| **Ex. No.: 2** | **SOCKET PROGRAMMING** |
| **02.01.2023** |

**Aim:**

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

**Algorithm:**

1. Create a URL object and pass URL as string to download the webpage. URL example = new URL (pass URL of webpage you want to download)
2. Create Buffered Reader object and pass openStream(). Method of URL in Input Stream object.
3. Create a string object to read each line one by one from stream.
4. Write each line in html file where webpage will be downloaded.
5. Close all objects.
6. Catch exceptions if URL failed to download.

**Program:**

import java.io.\*;

import java.net.\*;

public class exercise2

{

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

{

URL url = new URL("http://www.google.com");

BufferedReader reader = new BufferedReader(new InputStreamReader(url.openStream()));

BufferedWriter writer = new BufferedWriter(new FileWriter("data.html"));

String line;

while((line=reader.readLine())!=null)

{

System.out.println(line);

writer.write(line);

writer.newLine();

}

reader.close();

writer.close();

}

}

Graphical user interface, application, Teams

Description automatically generatedA picture containing text

Description automatically generated**Input/Output:**

**Result:**

A webpage has been downloaded using TCP sockets.

|  |  |
| --- | --- |
| **Ex. No.: 3** | **SOCKET PROGRAMMING** |
| **09.01.2023** |

**Aim:**

Socket program to implement ECHO and client-server application for chat using TCP.

**Algorithm:**

**ECHO CLIENT**

1. Start the program.
2. Create a socket which binds the Ip address of server and the port address to acquire service.
3. After establishing connection send a data to server.
4. Receive and print the same data from server.
5. Close the socket. End the program.

**ECHO SERVER**

1. Start the program.
2. Create a server socket to activate the port address.
3. Create a socket for the server socket which accepts the connection.
4. After establishing connection receive the data from client.
5. Print and send the same data to client.
6. Close the socket.
7. End the program.

**TCP CLIENT**

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

**TCP SERVER**

1. Start the program.
2. Include necessary package in java.
3. To create a socket in server to client
4. The server establishes a connection to the client.
5. The server accept the connection and to send the data from server to client and
6. vice versa.
7. The server communicates with the client to send the end of the message.
8. Stop the program.

**Program:**

**ECHO CLIENT**

import java.io.\*;

import java.net.\*;

public class eclient

{

public static void main(String args[])

{

Socket c=null;

String line;

DataInputStream is,is1;

PrintStream os;

try

{

c=new Socket("localhost",8080);

}

catch(IOException e)

{

System.out.println(e);

}

try

{

os=new PrintStream(c.getOutputStream());

is=new DataInputStream(System.in);

is1=new DataInputStream(c.getInputStream());

do

{

System.out.println("client");

line=is.readLine();

os.println(line);

if(!line.equals("exit"))

System.out.println("server:"+is1.readLine());

}while(!line.equals("exit"));

}

catch(IOException e)

{

System.out.println("socket closed");

}

}

}

**ECHO SERVER**

import java.io.\*;

import java.net.\*;

import java.lang.\*;

public class eserver

{

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

{

ServerSocket s=null;

String line;

DataInputStream is;

PrintStream ps;

Socket c=null;

try

{

s=new ServerSocket(8080);

}

catch(IOException e)

{

System.out.println(e);

}

try

{

c=s.accept();

is=new DataInputStream(c.getInputStream());

ps=new PrintStream(c.getOutputStream());

while(true)

{

line=is.readLine();

System.out.println("msg received and sent back to client");

ps.println(line);

}

}

catch(IOException e)

{

System.out.println(e);

}

}

}

**TCP CLIENT**

import java.net.\*;import java.io.\*;

public class TCPclient1

{

public static void main(String arg[])

{

Socket c=null;

String line;

DataInputStream is,is1;

PrintStream os;

try

{

c=new Socket("127.0.0.1",9999);

}

catch(IOException e)

{

System.out.println(e);

}

try

{

os=new PrintStream(c.getOutputStream());

is=new DataInputStream(System.in);

is1=new DataInputStream(c.getInputStream());

do

{

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

line=is.readLine();

os.println(line);

System.out.println("Server:" + is1.readLine());

}while(line.equalsIgnoreCase("quit")==false);

is1.close();

os.close();

}

catch(IOException e)

{

System.out.println("Socket Closed!Message Passing is over");

}

}

}

**TCP SERVER**

import java.net.\*;

import java.io.\*;

public class TCPserver1

{

public static void main(String arg[])

{

ServerSocket s=null;

String line;

DataInputStream is=null,is1=null;

PrintStream os=null;

Socket c=null;

try

{

s=new ServerSocket(9999);

}

catch(IOException e)

{

System.out.println(e);

}

try

{

c=s.accept();

is=new DataInputStream(c.getInputStream());

is1=new DataInputStream(System.in);

os=new PrintStream(c.getOutputStream());

do

{

line=is.readLine();

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

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

line=is1.readLine();

os.println(line);

}while(line.equalsIgnoreCase("quit")==false);

is.close();

os.close();

}

catch(IOException e)

{

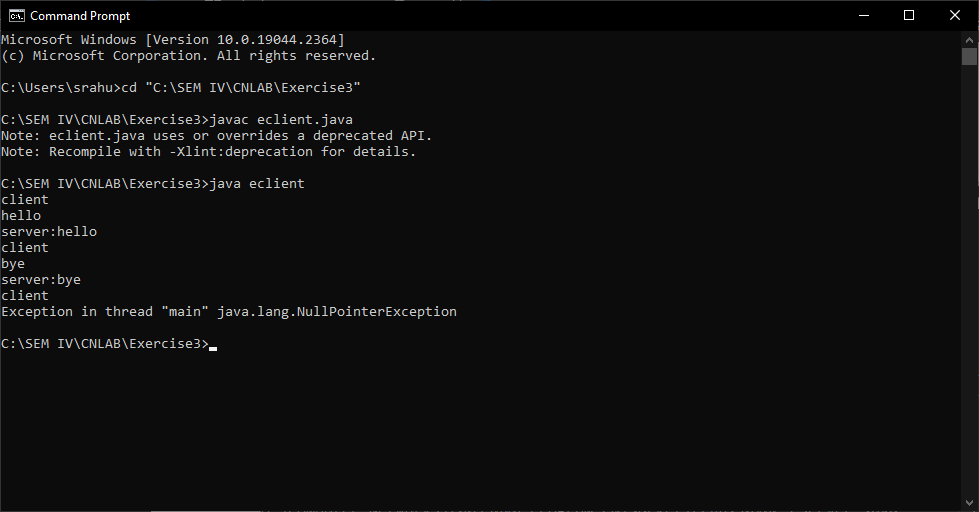
System.out.println(e);

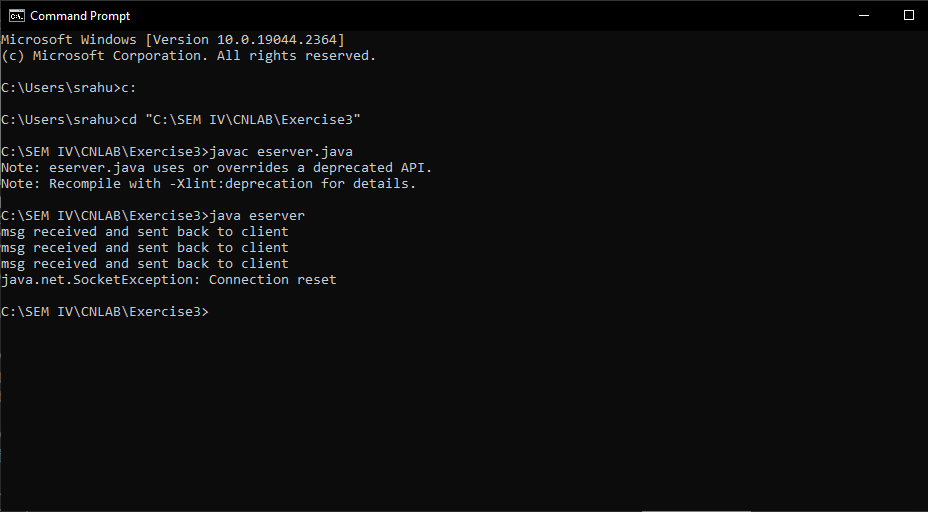
}

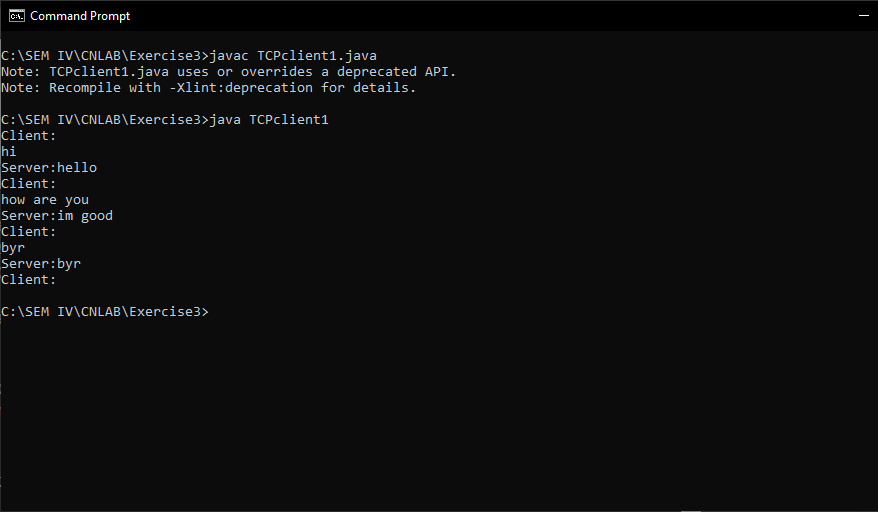
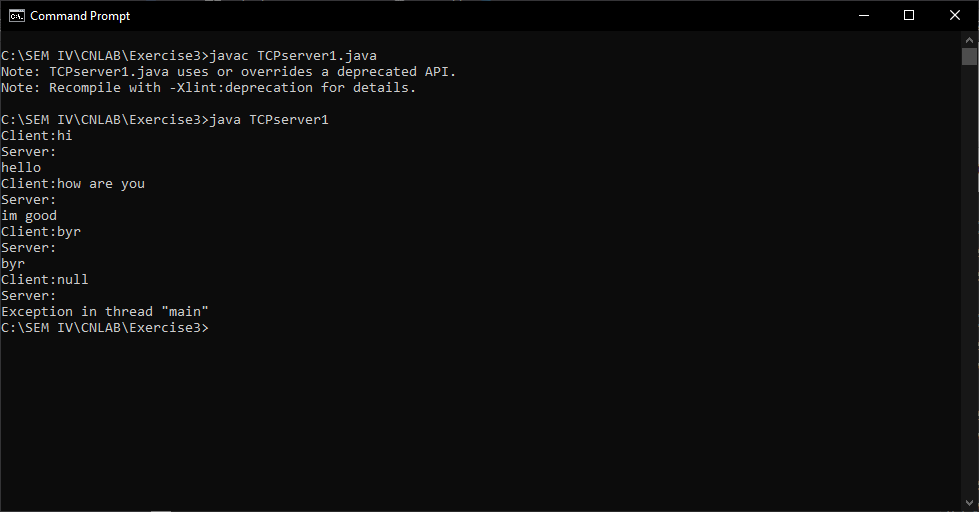
}

}

**INPUT/OUTPUT:**

**ECHO CLIENT  
**

**ECHO SERVER  
**

**TCP CLIENT  
TCP SERVER  
**

**Result:**

Sockets were programmed implement ECHO and client-server application for chat using TCP.

|  |  |
| --- | --- |
| **Ex. No.: 4** | **DOMAIN NAME SYSTEM SIMULATION** |
| **17.01.2023** |

**Aim:**

To simulate Domain Name System using UDP sockets.

**Algorithm:**

**SERVER**

1. Create an array of hosts and its ip address in another array.
2. Create a datagram socket and bind it to a port.
3. Create a datagram packet to receive client request.
4. Read the domain name from client to be resolved.
5. Lookup the host array for the domain name.
6. If found, then retrieve corresponding address.
7. Create a datagram packet and send ip address to client.
8. Repeat steps 3-7 to resolve further requests from clients.
9. Close the server socket.

**CLIENT**

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

**Program:**

**SERVER**

import java.io.\*;

import java.net.\*;

public class udpserver

{

private static int indexOf(String[] array, String str)

{

str = str.trim();

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

{

if (array[i].equals(str))

return i;

}

return -1;

}

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

{

String[] hosts = {"google.com", "wikipedia.org", "youtube.com", "facebook.com"};

String[] ip = {"68.180.206.184", "209.85.148.19", "80.168.92.140", "69.63.189.16"};

System.out.println("Press Ctrl + C to Quit");

while (true)

{

DatagramSocket serversocket=new DatagramSocket(1362);

byte[] senddata = new byte[1021];

byte[] receivedata = new byte[1021];

DatagramPacket recvpack = new

DatagramPacket(receivedata, receivedata.length);

serversocket.receive(recvpack);

String sen = new String(recvpack.getData());

InetAddress ipaddress = recvpack.getAddress();

int port = recvpack.getPort();

String capsent;

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

if(indexOf (hosts, sen) != -1)

capsent = ip[indexOf (hosts, sen)];

else

capsent = "Host Not Found";

senddata = capsent.getBytes();

DatagramPacket pack = new DatagramPacket(senddata,

senddata.length,ipaddress,port);

serversocket.send(pack);

serversocket.close();

}

}

**CLIENT**

import java.io.\*;

import java.net.\*;

public class udpclient

{

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

{

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

DatagramSocket clientsocket = new DatagramSocket();

InetAddress ipaddress;

if (args.length == 0)

ipaddress = InetAddress.getLocalHost();

else

ipaddress = InetAddress.getByName(args[0]);

byte[] senddata = new byte[1024];

byte[] receivedata = new byte[1024];

int portaddr = 1362;

System.out.print("Enter the hostname : ");

String sentence = br.readLine();

senddata = sentence.getBytes();

DatagramPacket pack = new DatagramPacket(senddata, senddata.length, ipaddress, portaddr);

clientsocket.send(pack);

DatagramPacket recvpack =new DatagramPacket(receivedata, receivedata.length);

clientsocket.receive(recvpack);

String modified = new String(recvpack.getData());

System.out.println("IP Address: " + modified);

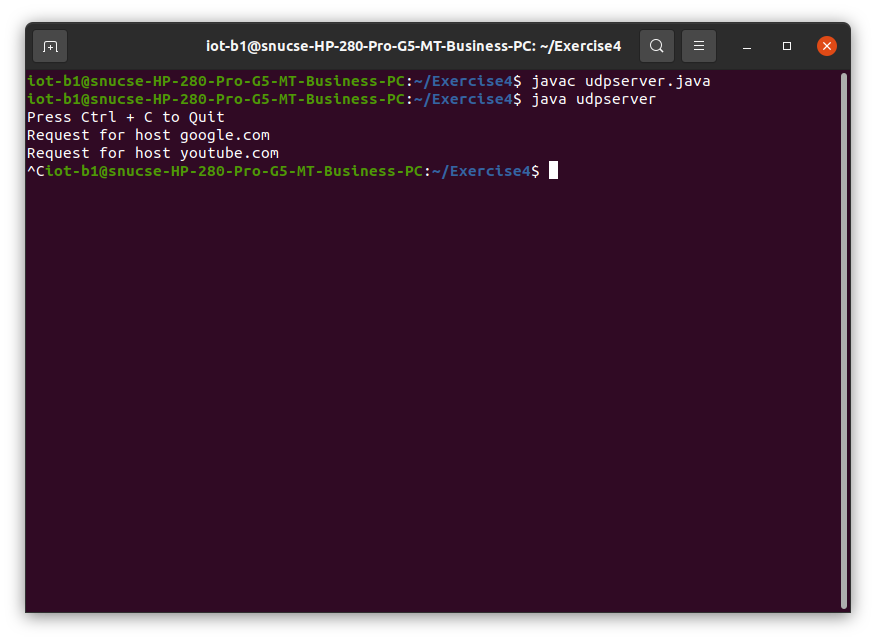
clientsocket.close();

}

}

**Input/Output:**

**SERVER**



**CLIENT**

Text

Description automatically generated

**Result:**

UDP sockets were used to simulate DNS.

|  |  |
| --- | --- |
| **Ex. No.: 5** | **ADDRESS RESOLUTION PROTOCOL PROGRAMMING** |
| **23.01.2023** |

**Aim:**

To simulate the working of address resolution protocol and reverse address resolution protocol using Java.

**Algorithm:**

**ARP SERVER**

1. Start the program.
2. Accept the socket which is created by the client.
3. Server maintains the table in which IP and corresponding MAC addresses are stored.
4. Read the IP address which is send by the client.
5. Map the IP address with its MAC address and return the MAC address to client.

**ARP CLIENT**

1. Start the program.
2. Using socket connection is established between client and server.
3. Get the IP address to be converted into MAC address.
4. Send this IP address to server.
5. Server returns the MAC address to client

**RARP SERVER**

1. Start the program.
2. Server maintains the table in which IP and corresponding MAC addresses are stored.
3. Read the MAC address which is send by the client.
4. Map the IP address with its MAC address and return the IP address to client.

**RARP CLIENT**

1. Start the program.
2. using datagram sockets UDP function is established.
3. Get the MAC address to be converted into IP address.
4. Send this MAC address to server.
5. Server returns the IP address to client.

**Program:**

**ARP SERVER**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Serverarp

{

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

{

ServerSocket obj=new ServerSocket(1800);

Socket obj1=obj.accept();

while(true)

{

DataInputStream din=new DataInputStream(obj1.getInputStream());

DataOutputStream dout=new DataOutputStream(obj1.getOutputStream());

String str=din.readLine();

String ip[]={"165.165.80.80","165.165.79.1"};

String mac[]={"6A:08:AA:C2","8A:BC:E3:FA"};

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

{

if(str.equals(ip[i]))

{

dout.writeBytes(mac[i]+'\n');

break;

}

}

obj.close();

}

}

}

**ARP CLIENT**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Clientarp

{

public static void main(String args[])

{

try

{

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

Socket clsct=new Socket("127.0.0.1",1800);

DataInputStream din=new DataInputStream(clsct.getInputStream());

DataOutputStream dout=new DataOutputStream(clsct.getOutputStream());

System.out.println("Enter the Logical address(IP):");

String str1=in.readLine();

dout.writeBytes(str1+'\n');

String str=din.readLine();

System.out.println("The Physical Address is:"+str);

clsct.close();

}

catch (Exception e)

{

System.out.println(e);

}

}

}

**RARP SERVER**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Serverrarp

{

public static void main(String args[])

{

try

{

DatagramSocket server=new DatagramSocket(1309);

while(true)

{

byte[] sendbyte=new byte[1024];

byte[] receivebyte=new byte[1024];

DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);

server.receive(receiver);

String str=new String(receiver.getData());

String s=str.trim();

InetAddress addr=receiver.getAddress();

int port=receiver.getPort();

String ip[]={"165.165.80.80","165.165.79.1"};

String mac[]={"6A:08:AA:C2","8A:BC:E3:FA"};

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

{

if(s.equals(mac[i]))

{

sendbyte=ip[i].getBytes();

DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,port);

server.send(sender);

break;

}

}

break;

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**RARP CLIENT**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Clientrarp

{

public static void main(String args[])

{

try

{

DatagramSocket client=new DatagramSocket();

InetAddress addr=InetAddress.getByName("127.0.0.1");

byte[] sendbyte=new byte[1024];

byte[] receivebyte=new byte[1024];

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

System.out.println("Enter the Physical address (MAC):");

String str=in.readLine(); sendbyte=str.getBytes();

DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,1309);

client.send(sender);

DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);

client.receive(receiver);

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

System.out.println("The Logical Address is(IP): "+s.trim());

client.close();

}

catch(Exception e)

{

System.out.println(e);

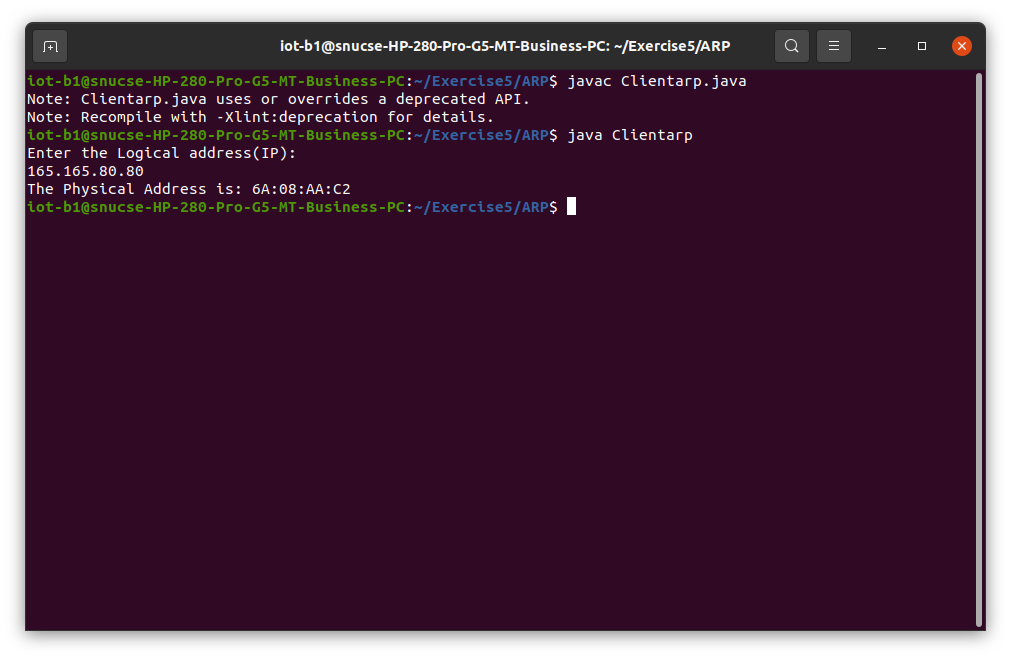
}

}

}

**Input/Output:**

**ARP SERVER  
Text

Description automatically generated  
ARP CLIENT  
  
RARP SERVER  
Text

Description automatically generated  
RARP CLIENT  
Text

Description automatically generated**

**Result:**

ARP and RARP were simulated using Java.

|  |  |
| --- | --- |
| **Ex. No.: 6** | **NS2 PROGRAMMING** |
| **07.02.2023** |

**Aim:**

1. To create simple topology using Network Simulator.
2. To implement congestion control in TCP using NS-2.

**Algorithm:**

Exercise a

1. Start network simulator OTCL editor.
2. Create new simulator using  
   set ns [new Simulator]
3. Create Trace route to Network Animator set nf [open out.nam w] $ns namtrace-all $nf
4. Create procedure to trace all path  
   proc create\_testnet {} {  
   global s1 s2 r1 k1  
   set s1 [$ns node]  
   set s2 [$ns node]  
   set r1 [$ns node]  
   set k1 [$ns node]  
   }
5. Create full duplex connection using  
   $ns duplex-link $s1 $r1 8Mb 5ms drop-tail  
   $ns duplex-link $s2 $r1 8Mb 5ms drop-tail  
   set L [ns\_duplex $r1 $k1 800Kb 100ms drop-tail]
6. Connect with TCP and SINK command.  
   $ns connect  
   $tcp  
   $sink
7. Run and Execute the program.  
   $ns run

Exercise b:

1. Create a simulator object
2. Define different colors for data flows (for NAM)
3. Open the NAM trace file.
4. Define a 'finish' procedure- Close the NAM trace file and Execute NAM on the trace file.
5. Create 4 nodes and duplex links between them.
6. Set Queue Size of link (n2-n3) to 10 and give node positions. Monitor the queue for link(n2-n3). (for NAM)
7. Setup a TCP connection. Setup a FTP over TCP connection.
8. Setup a UDP connection. Setup a CBR over UDP connection.
9. Schedule events for the CBR and FTP agents
10. Call the finish procedure after 5 seconds of simulation time.
11. Print CBR packet size and interval.
12. Run the simulation.

**Code:**

Exercise a:  
set nf [open out.nam w]  
$ns namtrace-all $nf  
set tf [open out.tr w]  
$ns trace-all $tf  
  
proc finish {} {  
global ns nf  
$ns flush-trace  
  
close $nf  
exec nam out.nam &  
exit 0  
}  
  
set n0 [$ns node]  
set n1 [$ns node]  
set n2 [$ns node]  
set n3 [$ns node]  
  
$ns duplex-link $n0 $n2 5Mb 10ms DropTail  
$ns duplex-link $n1 $n2 5Mb 10ms DropTail  
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail  
  
$ns queue-limit $n2 $n3 2  
  
$ns duplex-link-op $n0 $n2 orient right-down  
$ns duplex-link-op $n1 $n2 orient right-up  
$ns duplex-link-op $n2 $n3 orient right  
  
$ns duplex-link-op $n2 $n3 queuePos 0.5  
  
set tcp [new Agent/TCP]

$tcp set class\_ 2  
  
$ns attach-agent $n0 $tcp  
set sink [new Agent/TCPSink]  
  
$ns attach-agent $n3 $sink  
$ns connect $tcp $sink  
  
$tcp set fid\_ 1  
set ftp [new Application/FTP]  
$ftp attach-agent $tcp  
$ftp set type\_ FTP  
set udp [new Agent/UDP]  
$ns attach-agent $n1 $udp  
  
set null [new Agent/Null]  
$ns attach-agent $n3 $null  
$ns connect $udp $null  
$udp set fid\_ 2  
  
set cbr [new Application/Traffic/CBR]  
$cbr attach-agent $udp  
$cbr set type\_ CBR  
  
$cbr set packet\_size\_ 1000  
  
$cbr set rate\_ 1mb  
  
$cbr set random\_ false  
$ns at 0.1 "$cbr start"  
$ns at 1.0 "$ftp start"  
$ns at 4.0 "$ftp stop"  
$ns at 4.5 "$cbr stop"

$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"  
$ns at 5.0 "finish"  
puts "CBR packet size = [$cbr set packet\_size\_]"  
puts "CBR interval = [$cbr set interval\_]"  
$ns run

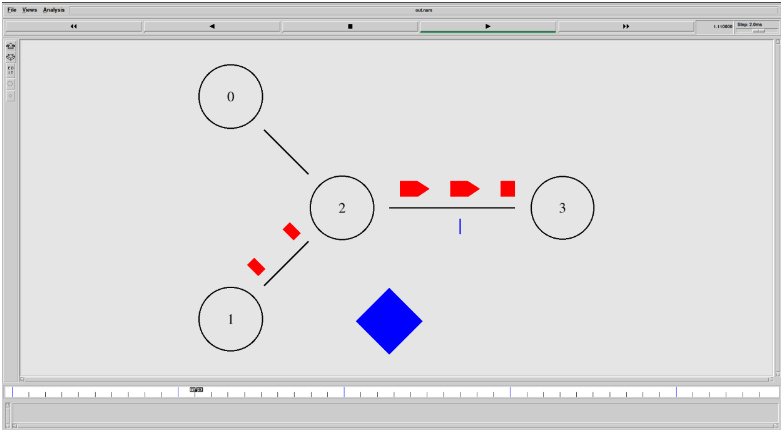
Exercise b:  
set ns [new Simulator]  
  
set n0 [$ns node]  
set n1 [$ns node]  
set n2 [$ns node]  
set n3 [$ns node]  
set n4 [$ns node]  
set n5 [$ns node]  
  
$ns duplex-link $n0 $n2 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 2Mb 10ms DropTail  
$ns duplex-link $n2 $n3 0.3Mb 200ms DropTail  
$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail  
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail  
set tcp1 [new Agent/TCP/Reno]  
$ns attach-agent $n0 $tcp1  
set sink1 [new Agent/TCPSink]  
$ns attach-agent $n4 $sink1  
$ns connect $tcp1 $sink1  
set ftp1 [new Application/FTP]  
$ftp1 attach-agent $tcp1  
$ftp1 set type\_ FTP  
$ns at 0.1 "$ftp1 start"  
$ns at 40.0 "$ftp1 stop"  
$ns at 50.0 "finish"  
proc plotWindow {tcpSource outfile} {  
global ns  
set now [$ns now]  
set cwnd [$tcpSource set cwnd\_]  
$ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"  
}  
set outfile [open "congestion.xg" w]  
$ns at 0.0 "plotWindow $tcp1 $outfile"  
proc finish {} {  
exec xgraph congestion.xg -geometry 300x300 &  
exit 0  
}

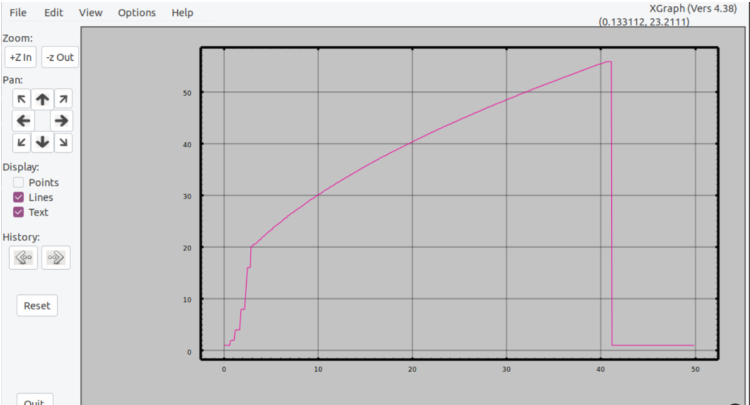
$ns run

**Input/Output:**

Exercise a:



Exercise b:



**Result:**

Successfully created simple topology using Network Simulator and implemented congestion control in TCP using NS-2.

|  |  |
| --- | --- |
| **Ex. No.: 7** | **ERROR CORRECTION IN COMPUTER NETWORKS** |
| **28.02.2023** |

**Aim:**

To create a program that simulates any error correction method on Java.

**Algorithm:**

1. The original message (dataword) is considered as M(x) consisting of ‘k’ bits and the divisor as C(x) consists of ‘n+1’ bits.
2. The original message M(x) is appended by ‘n’ bits of zero’s. Let us call this zero-extended message as T(x).
3. Divide T(x) by C(x) and find the remainder.
4. The division operation is performed using XOR operation.
5. The resultant remainder is appended to the original message M(x) as CRC and sent by the sender(codeword).

**Program:**

import java.util.\*;

class exercise7

{

static int exorOperation(int x, int y)

{

if(x==y)

{

return 0;

}

return 1;

}

static void receiveData(int data[], int divisor[])

{

int rem[]=divideDataWithDivisor(data, divisor);

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

{

if(rem[i]!=0)

{

System.out.println("Corrupt data received...");

return;

}

}

System.out.println("Data received without error");

}

static int[] divideDataWithDivisor(int oldData[], int divisor[])

{

int rem[]=new int[divisor.length];

int i;

int data[]=new int[oldData.length+divisor.length];

System.arraycopy(oldData, 0, data, 0, oldData.length);

System.arraycopy(data, 0, rem, 0, divisor.length);

for(i=0; i<oldData.length; i++)

{

System.out.println((i+1)+".) First data bit is: "+rem[0]);

System.out.print("Remainder: ");

if(rem[0]==1)

{

for(int j=1; j<divisor.length; j++)

{

rem[j-1]=exorOperation(rem[j], divisor[j]);

System.out.print(rem[j-1]);

}

}

else

{

for(int j=1; j<divisor.length; j++)

{

rem[j-1]=exorOperation(rem[j], 0);

System.out.print(rem[j-1]);

}

}

rem[divisor.length-1]=data[i+divisor.length];

System.out.println(rem[divisor.length-1]);

}

return rem;

}

public static void main(String args[])

{

Scanner scan=new Scanner(System.in);

int size;

System.out.println("Enter size of the data array: ");

size=scan.nextInt();

int data[]=new int[size];

System.out.println("Enter data bits in the array one by one: ");

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

{

System.out.println("Enter bit "+(size-i)+":");

data[i]=scan.nextInt();

}

System.out.println("Enter the size of the divisor array: ");

size=scan.nextInt();

int divisor[]=new int[size];

System.out.println("Enter divisor bits in the array one by one: ");

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

{

System.out.println("Enter bit "+(size-i)+":");

divisor[i]=scan.nextInt();

}

int rem[]=divideDataWithDivisor(data, divisor);

for(int i=0; i<rem.length-1;i++)

{

System.out.print(rem[i]);

}

System.out.println("\nGenerated CRC code is:");

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

{

System.out.print(data[i]);

}

for(int i=0; i<rem.length-1; i++)

{

System.out.print(rem[i]);

}

System.out.println();

int sentData[]=new int[data.length+rem.length-1];

System.out.println("Enter bits in the array which you want to send: ");

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

{

System.out.println("Enter bit "+(sentData.length-1)+":");

sentData[i]=scan.nextInt();

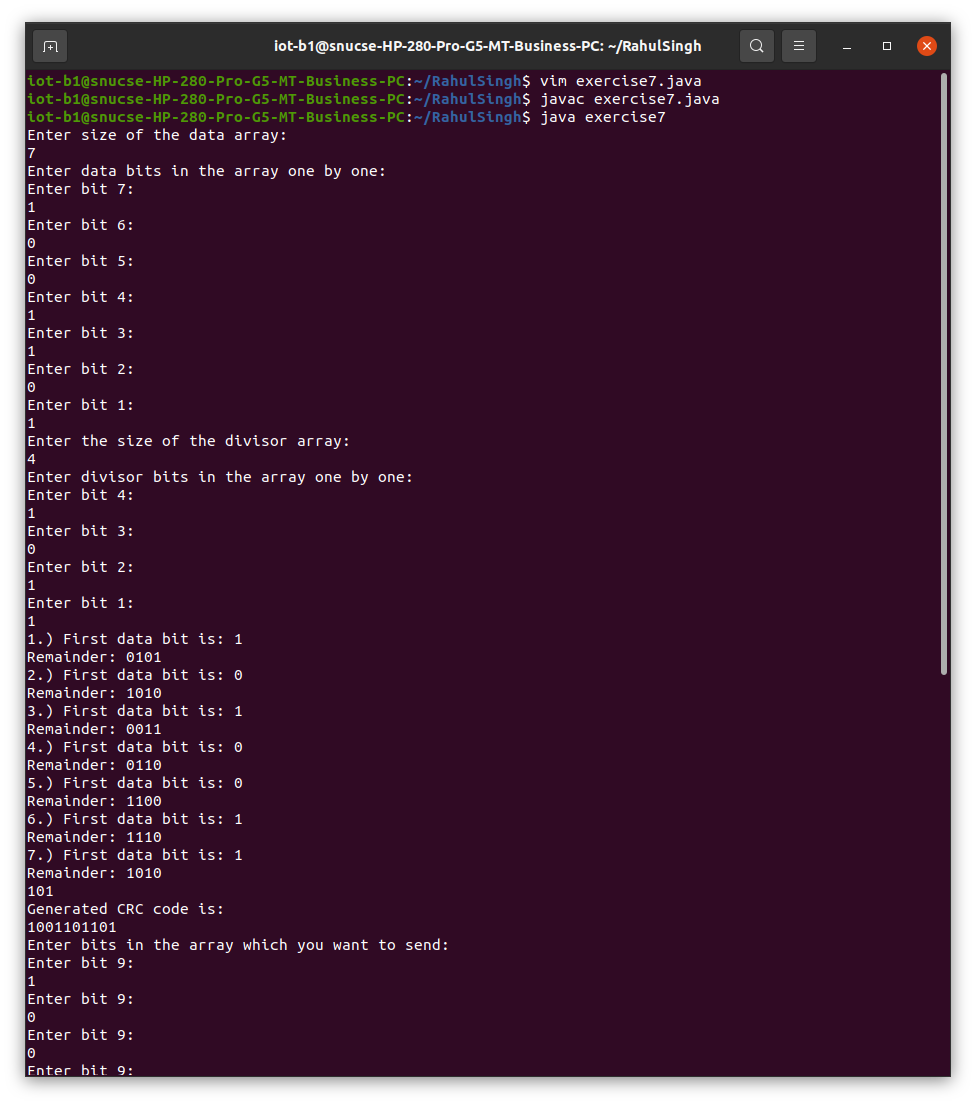
}

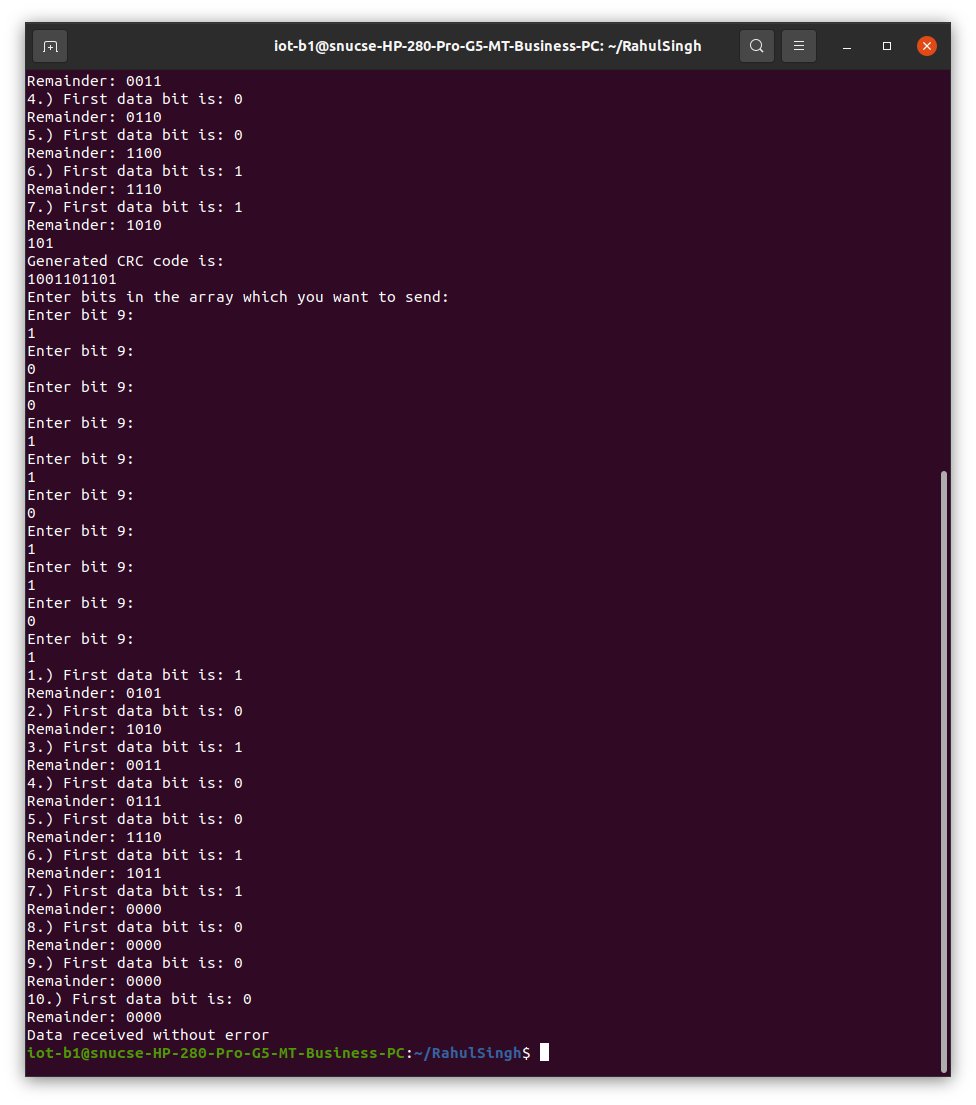
receiveData(sentData, divisor);

}

}

**Input/Output:**

****

****

**Result:**

Cyclic redundancy check was performed on Java.

|  |  |
| --- | --- |
| **Ex. No.: 8** | **TCP/UDP PERFORMANCE ON NETWORK SIMULATOR** |
| **07.03.2023** |

**Aim:**

To implement the study of TCP/UDP performance using Network Simulator.

**Algorithm:**

1. Create a simulation object.
2. Set routing protocol to routing.
3. Trace packets and all links onto NAM trace and to trace file.
4. Create right nodes.
5. Describe their layout topology.
6. Add a sink agent to node.
7. Connect source and sink.
8. Observe the traffic route when link is up and down.
9. View the simulated events and trace file analyse it.
10. Start the scheduler.

**Code:**

set ns [new Simulator]

set nr [open thro\_dt.tr w]

$ns trace-all $nr

set nf [open thro.nam w]

$ns namtrace-all $nf

proc finish { } {

global ns nr nf

$ns flush-trace

close $nf

close $nr

exec nam thro.nam &

exit 0 }

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

set n6 [$ns node]

set n7 [$ns node]

$ns duplex-link $n0 $n3 1Mb 10ms DropTail

$ns duplex-link $n1 $n3 1Mb 10ms DropTail

$ns duplex-link $n2 $n3 1Mb 10ms DropTail

$ns duplex-link $n3 $n4 1Mb 10ms DropTail

$ns duplex-link $n4 $n5 1Mb 10ms DropTail

$ns duplex-link $n4 $n6 1Mb 10ms DropTail

$ns duplex-link $n4 $n7 1Mb 10ms DropTail

$ns duplex-link-op $n0 $n3 orient right-up

$ns duplex-link-op $n1 $n3 orient right

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

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

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

$ns duplex-link-op $n4 $n7 orient right-down

$ns duplex-link-op $n6 $n4 orient left

set udp0 [new Agent/UDP]

$ns attach-agent $n2 $udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0

set null0 [new Agent/Null]

$ns attach-agent $n5 $null0

$ns connect $udp0 $null0

set udp1 [new Agent/UDP]

$ns attach-agent $n1 $udp1

set cbr1 [new Application/Traffic/CBR]

$cbr1 set packetSize\_ 500

$cbr1 set interval\_ 0.005

$cbr1 attach-agent $udp1

set null0 [new Agent/Null]

$ns attach-agent $n6 $null0

$ns connect $udp1 $null0

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set cbr2 [new Application/Traffic/CBR]

$cbr2 set packetSize\_ 500

$cbr2 set interval\_ 0.005

$cbr2 attach-agent $tcp0

set tcpsink0 [new Agent/TCPSink]

$ns attach-agent $n7 $tcpsink0

$ns connect $tcp0 $tcpsink0

$udp0 set fid\_ 1

$udp1 set fid\_ 2

$tcp0 set fid\_ 3

$ns color 1 Red

$ns color 2 Green

$ns color 3 blue

$ns at 0.2 "$cbr0 start"

$ns at 3.5 "$cbr0 stop"

$ns at 0.3 "$cbr1 start"

$ns at 4.5 "$cbr1 stop"

$ns at 0.4 "$cbr2 start"

$ns at 4.5 "$cbr2 stop"

$ns at 5.0 "finish"

$ns run

**Input/Output:**

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

**Result:**

TCP/UDP performance was studied on NS2.

|  |  |
| --- | --- |
| **Ex. No.: 9** | **ROUTING ALGORITHMS IN COMPUTER NETWORKS** |
| **14.03.2023** |

**Aim:**

To simulate Distance Vector Routing and Link State Routing Algorithms.

**Algorithm:**

1. A router transmits its distance vector to each of its neighbours in a routing packet.
2. Each router receives and saves the most recently received distance vector from each of its neighbours.
3. A router recalculates its distance vector when:
   1. It receives a distance vector from a neighbour containing different information than before.
   2. It discovers that a link to a neighbour has gone down.

**Code:**

**DISTANCE VECTOR ROUTING**

import java.io.\*;

public class dvr

{

static int graph[][];

static int via[][];

static int rt[][];

static int v;

static int e;

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

{

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

System.out.println("Please enter the number of Vertices: ");

v = Integer.parseInt(br.readLine());

System.out.println("Please enter the number of Edges: ");

e = Integer.parseInt(br.readLine());

graph = new int[v][v];

via = new int[v][v];

rt = new int[v][v];

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

for(int j = 0; j < v; j++)

{

if(i == j)

graph[i][j] = 0;

else

graph[i][j] = 9999;

}

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

{

System.out.println("Please enter data for Edge " + (i + 1) + ":");

System.out.print("Source: ");

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

s--;

System.out.print("Destination: ");

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

d--;

System.out.print("Cost: ");

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

graph[s][d] = c;

graph[d][s] = c;

}

dvr\_calc\_disp("The initial Routing Tables are: ");

System.out.print("Please enter the Source Node for the edge whose cost has changed: ");

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

s--;

System.out.print("Please enter the Destination Node for the edge whose cost has changed: ");

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

d--;

System.out.print("Please enter the new cost: ");

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

graph[s][d] = c;

graph[d][s] = c;

dvr\_calc\_disp("The new Routing Tables are: ");

}

static void dvr\_calc\_disp(String message)

{

System.out.println();

init\_tables();

update\_tables();

System.out.println(message);

print\_tables();

System.out.println();

}

static void update\_table(int source)

{

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

{

if(graph[source][i] != 9999)

{

int dist = graph[source][i];

for(int j = 0; j < v; j++)

{

int inter\_dist = rt[i][j];

if(via[i][j] == source)

inter\_dist = 9999;

if(dist + inter\_dist < rt[source][j])

{

rt[source][j] = dist + inter\_dist;

via[source][j] = i;

}

}

}

}

}

static void update\_tables()

{

int k = 0;

for(int i = 0; i < 4\*v; i++)

{

update\_table(k);

k++;

if(k == v)

k = 0;

}

}

static void init\_tables()

{

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

{

for(int j = 0; j < v; j++)

{

if(i == j)

{

rt[i][j] = 0;

via[i][j] = i;

}

else

{

rt[i][j] = 9999;

via[i][j] = 100;

}

}

}

}

static void print\_tables()

{

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

{

for(int j = 0; j < v; j++)

{

System.out.print("Dist: " + rt[i][j] + " ");

}

System.out.println();

}

}

}

**LINK STATE ROUTING**

import java.util.\*;

public class lsr

{

public int distance[] = new int[10];

public int cost[][] = new int[10][10];

public void calc(int n,int s)

{

int flag[] = new int[n+1];

int i,minpos=1,k,c,minimum;

for(i=1;i<=n;i++)

{

flag[i]=0;

this.distance[i]=this.cost[s][i];

}

c=2;

while(c<=n)

{

minimum=99;

for(k=1;k<=n;k++)

{

if(this.distance[k]<minimum && flag[k]!=1)

{

minimum=this.distance[i];

minpos=k;

}

}

flag[minpos]=1;

c++;

for(k=1;k<=n;k++)

{

if(this.distance[minpos]+this.cost[minpos][k] < this.distance[k] && flag[k]!=1 )

this.distance[k]=this.distance[minpos]+this.cost[minpos][k];

}

}

}

public static void main(String args[])

{

int nodes,source,i,j;

Scanner in = new Scanner(System.in);

System.out.println("Enter the Number of Nodes \n");

nodes = in.nextInt();

lsr d = new lsr();

System.out.println("Enter the Cost Matrix Weights: \n");

for(i=1;i<=nodes;i++)

for(j=1;j<=nodes;j++)

{

d.cost[i][j]=in.nextInt();

if(d.cost[i][j]==0)

d.cost[i][j]=999;

}

System.out.println("Enter the Source Vertex :\n");

source=in.nextInt();

d.calc(nodes,source);

System.out.println("The Shortest Path from Source \t"+source+"\t to all other vertices are : \n");

for(i=1;i<=nodes;i++)

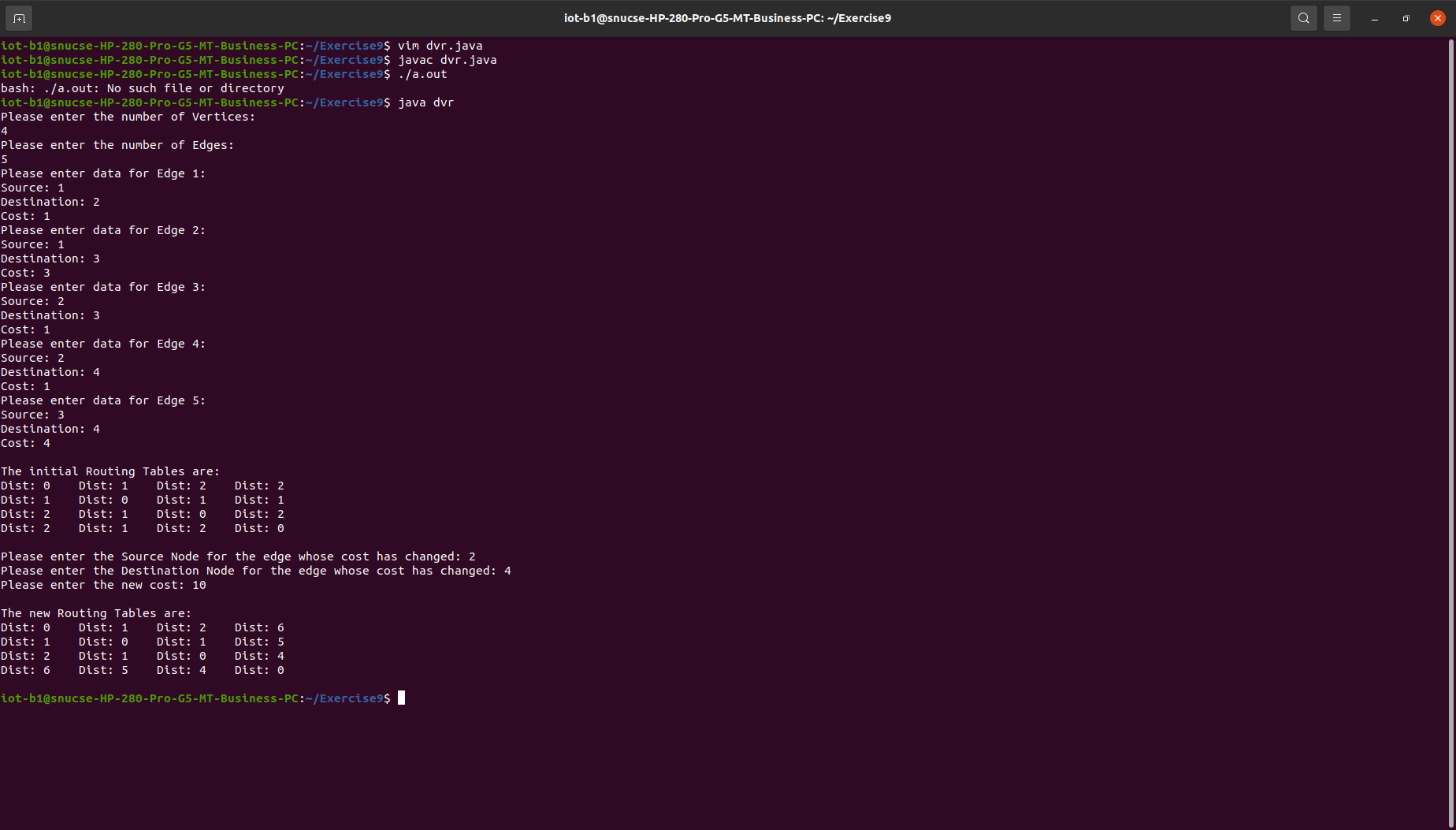
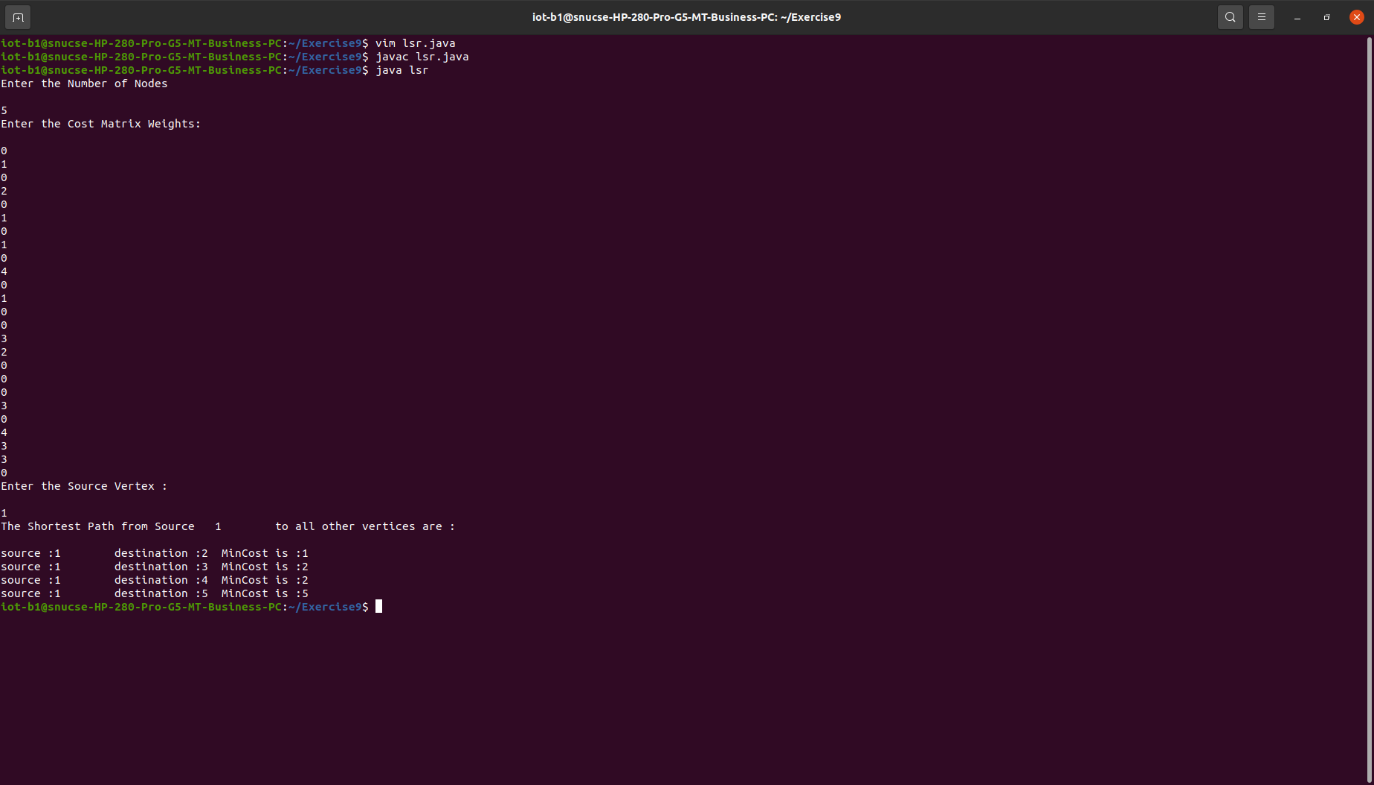
if(i!=source)

System.out.println("source :"+source+"\t destination :"+i+"\t MinCost is :"+d.distance[i]+"\t");

}

}

**Input/Output:**

**DISTANCE VECTOR ROUTING  
  
LINK STATE ROUTING  
**

**Result:**

Link State Routing and Distance Vector Routing Algorithms were implemented on Java.

|  |  |
| --- | --- |
| **Ex. No.: 10** | **ROUTING ALGORITHMS IN NS2** |
| **21.03.2023** |

**Aim:**

To evaluate routing algorithms on Network Simulation Tool.

**Algorithm:**

**Code:**

set val(chan) Channel/WirelessChannel;

set val(prop) Propagation/TwoRayGround;

set val(netif) Phy/WirelessPhy;

set val(mac) Mac/802\_11;

set val(ifq) Queue/DropTail/PriQueue;

set val(ll) LL;

set val(ant) Antenna/OmniAntenna;

set val(ifqlen) 50;

set val(rp) AODV;

set val(nn) 11;

set val(x) 500;

set val(y) 400;

set val(stop) 3;

set val(energymodel) EnergyModel;

set val(initialenergy) 1000;

set ns [new Simulator]

set tf [open ns\_aodv.tr w]

$ns trace-all $tf

set nf [open ns\_aodv.nam w]

$ns namtrace-all-wireless $nf $val(x) $val(y)

set topo [new Topography]

$topo load\_flatgrid $val(x) $val(y)

create-god $val(nn)

set chan\_1\_ [new $val(chan)]

$ns node-config -adhocRouting $val(rp) \

-llType $val(ll) \

-macType $val(mac) \

-ifqType $val(ifq) \

-ifqLen $val(ifqlen) \

-antType $val(ant) \

-propType $val(prop) \

-phyType $val(netif) \

-channel $chan\_1\_ \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace OFF \

-movementTrace ON \

-energyModel $val(energymodel) \

-initialEnergy $val(initialenergy) \

-rxPower 0.4 \

-txPower 1.0 \

-idlePower 0.6 \

-sleepPower 0.1 \

-transitionPower 0.4 \

-transitionTime 0.1

for {set i 0} {$i < $val(nn)} {incr i} {

set node\_($i) [$ns node]

$node\_($i) set X\_ [ expr 10+round(rand()\*480) ]

$node\_($i) set Y\_ [ expr 10+round(rand()\*380) ]

$node\_($i) set Z\_ 0.0

}

for {set i 0} {$i < $val(nn)} {incr i} {

$ns at [ expr 0.2+round(rand()) ] "$node\_($i) setdest [ expr 10+round(rand()\*480) ] [expr 10+round(rand()\*380) ] [expr 60+round(rand()\*30) ]"

}

set udp [new Agent/UDP]

$ns attach-agent $node\_(5) $udp

set null [new Agent/Null]

$ns attach-agent $node\_(2) $null

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set packetSize\_ 512

$cbr set interval\_ 0.1

$cbr set rate\_ 1mb

$cbr set maxpkts\_ 10000

$ns connect $udp $null

$ns at 0.4 "$cbr start"

for {set i 0} {$i < $val(nn)} {incr i} {

$ns initial\_node\_pos $node\_($i) 30

}

for {set i 0} {$i < $val(nn)} {incr i} {

$ns at $val(stop) "$node\_($i) reset";

}

$ns at $val(stop) "finish"

$ns at 3.1 "puts \"end simulation\"; $ns halt"

proc finish {} {

global ns tf nf

$ns flush-trace

close $tf

close $nf

exec nam ns\_aodv.nam &

exit 0

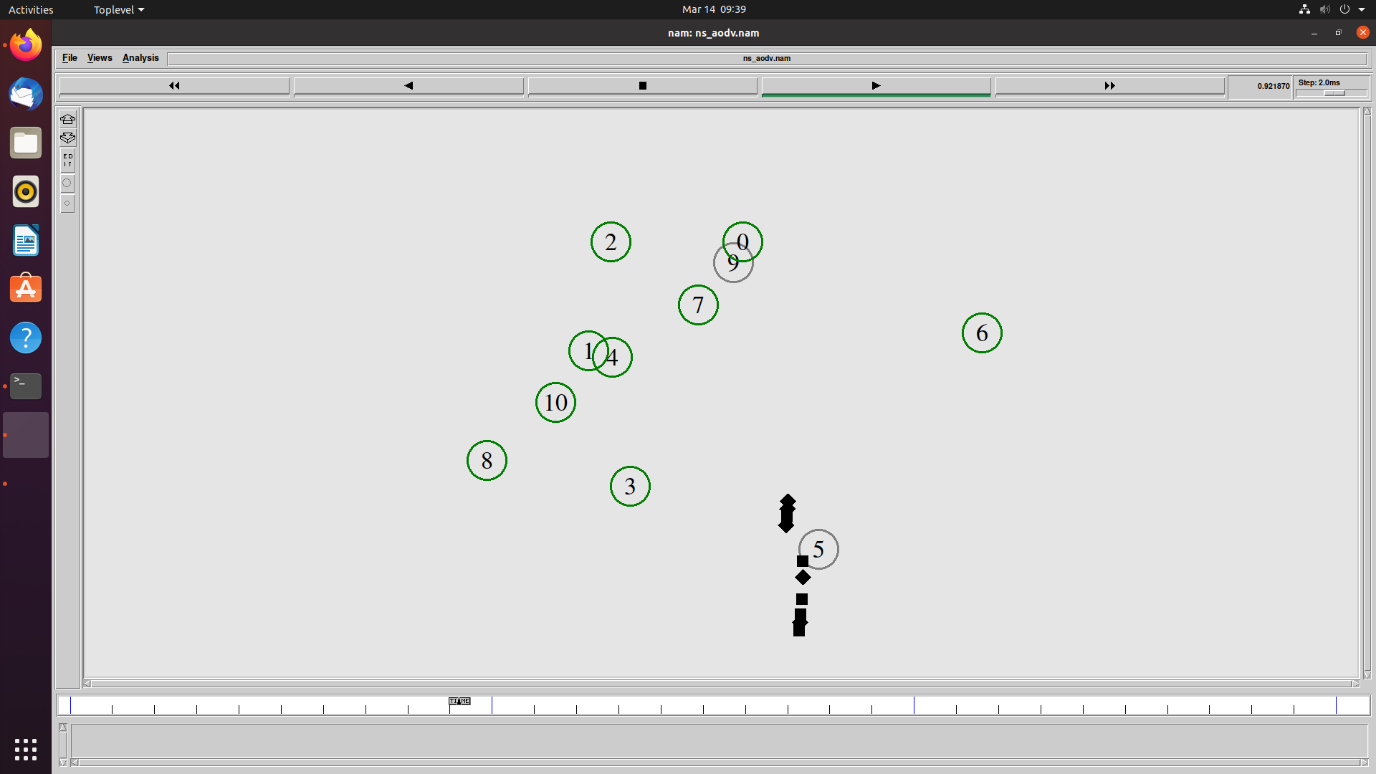
}

puts "CBR packet size = [$cbr set packetSize\_]"

puts "CBR interval = [$cbr set interval\_]"

$ns run

**Input/Output:**

****

**Result:**

Routing algorithms were evaluated using a network simulation tool.