EX.NO:1 Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute. Capture ping an

Commands:

Tcpdump:

Display traffic between 2 hosts:

To display all traffic between two hosts (represented by variables host1 and host2): # tcpdump host host1 a

Display traffic from a source or destination host only:

To display traffic from only a source (src) or destination (dst) host: # tcpdump src host

tcpdump dst host

Display traffic for a specific protocol

Provide the protocol as an argument to display only traffic for a specific protocol, for example tcp, udp, icm

For example to display traffic only for the tcp traffic :

tcpdump tcp

Filtering based on source or destination port

To filter based on a source or destination port:

tcpdump src port ftp # tcpdump dst port http

Netstat

Netstat is a common command line TCP/IP networking available in most versions of Windows, Linux, UNIX

Netstat provides information and statistics about protocols in use and current TCP/IP network connections.

#netstat

ipconfig

prompt. This utility allows you to get the IP address information of a Windows computer.

Using ipconfig

From the command prompt, type ipconfig to run the utility with default options. The output of the default co

#ipconfig

nslookup

The nslookup (which stands for name server lookup) command is a network utility program used to obtain i

The nslookup command is a powerful tool for diagnosing DNS problems. You know you're experiencing a D

#nslookup

Trace route:

Traceroute uses Internet Control Message Protocol (ICMP) echo packets with variable time to live (TTL) va

Traceroute is a network diagnostic tool used to track the pathway taken by a packet on an IP network from

The response time of each hop is calculated. To guarantee accuracy, each hop is queried multiple times (u

For the first set of packets, the first router receives the packet, decrements the TTL value and drops the pa

Proceeding in this way, traceroute uses the returned ICMP Time Exceeded messages to build a list of route

With the tracert command shown above, we're asking tracert to show us the path from the local computer a

www.google.com. #tracert google.com

Ping:

The ping command sends an echo request to a host available on the network. Using this command, you ca

ping172.16.6.2

Ex.No: 2 Write a HTTP web client program to download a web page using TCP sockets

PROGRAM

Client

```
import javax.swing.*; import java.net.*; import java.awt.image.*; import javax.imageio.*; import java.io.*;
```

import java.awt.image.BufferedImage; import java.io.ByteArrayOutputStream; import java.io.File;

```
import java.io.IOException; import javax.imageio.ImageIO;
```

public class Client

{

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

{

");

Socket soc; BufferedImage img = null; soc=new

Socket("localhost",4000); System.out.println("Client is running.

```
try {
```

```
System.out.println("Reading image from disk. ");
```

img = ImageIO.read(new File("digital_image_processing.jpg")); ByteArrayOutputStream baos = new ByteA

```
baos.flush();
```

byte[] bytes = baos.toByteArray(); baos.close(); System.out.println("Sending image to server."); OutputStre

dos.write(bytes, 0, bytes.length); System.out.println("Image sent to server. "); dos.close();

```
out.close();
```

```
}
```

```
catch (Exception e)
```

{

```
System.out.println("Exception: " + e.getMessage());
```

```
soc.close();
```

```
}
```

```
soc.close();
```

```
}
```

```
}
```

Server

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

```
import java.awt.image.*; import javax.imageio.*; import javax.swing.*; class Server
```

{

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

{

ServerSocket server=null; Socket socket;

server=new ServerSocket(4000); System.out.println("Server Waiting for image");

socket=server.accept(); System.out.println("Client connected."); InputStream in = socket.getInputStream();

DataInputStream dis = new DataInputStream(in); int len = dis.readInt();

System.out.println("Image Size: " + len/1024 + "KB"); byte[] data = new byte[len]; dis.readFully(data);

```
dis.close();
```

```
in.close();
```

InputStream ian = new ByteArrayInputStream(data); BufferedImage bImage = ImageIO.read(ian); JFrame

ImageIcon icon = new ImageIcon(bImage); JLabel l = new JLabel();

l.setIcon(icon); f.add(l);

```
f.pack(); f.setVisible(true);
```

```
}
```

```
}
```

OUTPUT:

When you run the client code, following output screen would appear on client side.

Ex.No: 3 Applications using TCP sockets like: Echo client and echo server, Chat and File Transfer

PROGRAM:

EchoServer.java import java.net.*; import java.io.*; public class EServer

{

public static void main(String args[])

{

ServerSocket s=null; String line; DataInputStream is; PrintStream ps; Socket c=null;

try

{

```
s=new ServerSocket(9000);
```

```
}
```

```
catch(IOException e)
```

{

```
}
```

try

{

```
System.out.println(e);
```

```
c=s.accept();
```

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

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

{

```
line=is.readLine(); ps.println(line);
```

```
}
```

```
}
```

```
catch(IOException e)
```

{

```
System.out.println(e);
```

```
}
```

```
}
```

```
}
```

EClient.java import java.net.*; import java.io.*; public class EClient

```
{ public static void main(String arg[])
```

{

```
Socket c=null; String line;
```

DataInputStream is,is1; PrintStream os;

try

{

InetAddress ia = InetAddress.getLocalHost(); c=new Socket(ia,9000);

```
}
```

```
catch(IOException e)
```

{

```
}
```

try

{

```
System.out.println(e);
```

os=new PrintStream(c.getOutputStream()); is=new DataInputStream(System.in);

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

{

System.out.println("Client:"); line=is.readLine();

```
os.println(line);
```

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

```
}
```

```
}
```

```
catch(IOException e)
```

{

```
System.out.println("Socket Closed!");
```

```
}
```

}}

OUTPUT

Server

C:\Program Files\Java\jdk1.5.0\bin>javac EServer.java C:\Program Files\Java\jdk1.5.0\bin>java EServer C

Client

C:\Program Files\Java\jdk1.5.0\bin>javac EClient.java C:\Program Files\Java\jdk1.5.0\bin>java EClient Clie

Server:Hai Server Client: Hello Server:Hello Client:end Server:end Client:ds

Socket Closed!

B . CHAT

PROGRAM

UDPserver.java import java.io.*; import java.net.*; class UDPserver

{

```
public static DatagramSocket ds;
```

public static byte buffer[]=new byte[1024]; public static int clientport=789,serverport=790;

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

{

ds=new DatagramSocket(clientport); System.out.println("press ctrl+c to quit the program");

BufferedReader dis=new BufferedReader(new InputStreamReader(System.in)); InetAddress ia=InetAddres

```
while(true)
```

{

DatagramPacket p=new DatagramPacket(buffer,buffer.length); ds.receive(p);

```
String psx=new String(p.getData(),0,p.getLength()); System.out.println("Client:" + psx); System.out.println("
```

String str=dis.readLine(); if(str.equals("end"))

break; buffer=str.getBytes();

```
ds.send(new DatagramPacket(buffer,str.length(),ia,serverport));
```

```
}
```

```
}
```

```
}
```

UDPclient.java import java .io.*; import java.net.*; class UDPclient

{

```
public static DatagramSocket ds;
```

```
public static int clientport=789,serverport=790;
```

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

{

byte buffer[]=new byte[1024]; ds=new DatagramSocket(serverport);

BufferedReader dis=new BufferedReader(new InputStreamReader(System.in)); System.out.println("server

InetAddress ia=InetAddress.getLocalHost(); while(true)

{

System.out.println("Client:"); String str=dis.readLine(); if(str.equals("end"))

break; buffer=str.getBytes();

ds.send(new DatagramPacket(buffer,str.length(),ia,clientport)); DatagramPacket p=new DatagramPacket(b

```
String psx=new String(p.getData(),0,p.getLength()); System.out.println("Server:" + psx);
```

```
}
```

```
}
```

```
}
```

OUTPUT:

Server

C:\Program Files\Java\jdk1.5.0\bin>javac UDPserver.java C:\Program Files\Java\jdk1.5.0\bin>java UDPser

press ctrl+c to quit the program Client:Hai Server

Server:Hello Client Client:How are You Server:I am Fine

Client

C:\Program Files\Java\jdk1.5.0\bin>javac UDPclient.java C:\Program Files\Java\jdk1.5.0\bin>java UDPclie

Client:Hai Server Server:Hello Clie Client:How are You Server:I am Fine Client:end

FILE TRANSFER

PROGRAM

File Server :

```
import java.io.BufferedInputStream; import java.io.File;
```

import java.io.FileInputStream; import java.io.OutputStream; import java.net.InetAddress; import java.net.S

```
public class FileServer
```

{

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

{

```
//Initialize Sockets
```

ServerSocket ssock = new ServerSocket(5000); Socket socket = ssock.accept();

//The InetAddress specification

```
InetAddress IA = InetAddress.getByName("localhost");
```

```
//Specify the file
```

File file = new File("e:\\Bookmarks.html"); FileInputStream fis = new FileInputStream(file);

BufferedInputStream bis = new BufferedInputStream(fis); //Get socket's output stream

OutputStream os = socket.getOutputStream(); //Read File Contents into contents array

```
byte[] contents;
```

long fileLength = file.length(); long current = 0;

```
long start = System.nanoTime(); while(current!=fileLength){
```

```
int size = 10000;
```

```
if(fileLength - current >= size) current += size;
```

else{

size = (int)(fileLength - current); current = fileLength;

```
}
```

```
contents = new byte[size];
```

bis.read(contents, 0, size); os.write(contents);

```
System.out.print("Sending file ... "+(current*100)/fileLength+"% complete!");
```

```
}
```

```
os.flush();
```

```
//File transfer done. Close the socket connection! socket.close();
```

```
ssock.close();
```

```
System.out.println("File sent succesfully!");
```

```
}}
```

File Client:

```
import java.io.BufferedOutputStream; import java.io.FileOutputStream; import java.io.InputStream;
```

```
import java.net.InetAddress; import java.net.Socket;
```

```
public class FileClient {
```

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

```
//Initialize socket
```

```
Socket socket = new Socket(InetAddress.getByName("localhost"), 5000); byte[] contents = new byte[10000
```

```
//Initialize the FileOutputStream to the output file's full path. FileOutputStream fos = new FileOutputStream(
```

BufferedOutputStream bos = new BufferedOutputStream(fos); InputStream is = socket.getInputStream();

```
//No of bytes read in one read() call int bytesRead = 0;
```

```
while((bytesRead=is.read(contents))!=-1) bos.write(contents, 0, bytesRead);
```

```
bos.flush(); socket.close();
```

```
System.out.println("File saved successfully!");
```

```
}
```

```
}
```

Output

server

E:\nwlab>java FileServer Sending file ... 9% complete! Sending file ... 19% complete! Sending file ... 28% c

E:\nwlab>client E:\nwlab>java FileClient File saved successfully!

E:\nwlab>

Ex.No: 4 Simulation of DNS using UDP Sockets

PROGRAM

DNS Server

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

public class udpdnsserver

{

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

{

```
str = str.trim();
```

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

{

```
if (array[i].equals(str)) return i;
```

```
}
```

```
return -1;
```

```
}
```

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

{

```
String[] hosts = {"yahoo.com", "gmail.com","cricinfo.com", "facebook.com"}; String[] ip = {"68.180.206.184",
```

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(

```
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.s

```
serversocket.close();
```

}}}

UDP DNS Client java import java.io.*; import java.net.*;

public class udpdnsclient

{

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

{

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

InetAddress ipaddress; if (args.length == 0)

```
ipaddress = InetAddress.getLocalHost();
```

else

ipaddress,portaddr);

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,
```

```
clientsocket.send(pack);
```

DatagramPacket recvpack =new DatagramPacket(receivedata,receivedata.length); clientsocket.receive(re

```
String modified = new String(recvpack.getData()); System.out.println("IP Address: " + modified); clientsock
```

}}

OUTPUT

Server

javac udpdnsserver.java java udpdnsserver

Press Ctrl + C to Quit Request for host yahoo.com Request for host cricinfo.com

Request for host youtube.com

Client

```
>javac udpdnsclient.java
```

```
>java udpdnsclient
```

Enter the hostname : yahoo.com IP Address: 68.180.206.184

```
>java udpdnsclient
```

Enter the hostname : cricinfo.com IP Address: 80.168.92.140

```
>java udpdnsclient
```

Enter the hostname : youtube.com IP Address: Host Not Found

Ex.No:5 Write a code simulating ARP /RARP protocols

PROGRAM

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("1

DataInputStream din=new DataInputStream(clsct.getInputStream()); DataOutputStream dout=new DataOu

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);
```

}}

```
}
```

Server:

```
import java.io.*; import java.net.*; import java.util.*; class Serverarp
```

{

public static void main(String args[])

{

try{

ServerSocket obj=new ServerSocket(139); Socket obj1=obj.accept(); while(true)

{

DataInputStream din=new DataInputStream(obj1.getInputStream()); DataOutputStream dout=new DataOu

```
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();
```

```
}
```

```
}
```

catch(Exception e)

{

```
System.out.println(e);
```

}}

```
}
```

Output:

E:\networks>java Serverarp E:\networks>java Clientarp Enter the Logical address(IP):

165.165.80.80

The Physical Address is: 6A:08:AA:C2

Program for Reverse Address Resolution Protocol (RARP) using UDP

PROGRAM:

Client:

```
import java.io.*; import java.net.*; import java.util.*; class Clientrarp12
```

{

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 th

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

```
DatagramPacket sender=newDatagramPacket(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.cl
```

DatagramSocket client=new DatagramSocket(); InetAddress addr=InetAddress.getByName("127.0.0.1"); b

```
byte[] receivebyte=new byte[1024];
```

BufferedReader in=new BufferedReader(new InputStreamReader(System.in)); System.out.println("Enter th

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

DatagramPacket sender=newDatagramPacket(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()); c

```
}
```

catch(Exception e)

{

```
System.out.println(e);
```

}}}

Server:

```
import java.io.*; import java.net.*; import java.util.*; class Serverrarp12
```

{

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;
```

}}}catch(Exception e)

{

```
System.out.println(e);
```

}}}

Output:

I:\ex>java Serverrarp12 I:\ex>java Clientrarp12

Enter the Physical address (MAC): 6A:08:AA:C2

The Logical Address is(IP): 165.165.80.8

Ex.No: 6 Study of Network simulator (NS) and Simulation of Congestion Control

Algorithms using NS

Program:

include <wifi_lte/wifi_lte_rtable.h> struct r_hist_entry *elm, *elm2;

```
int num_later = 1;
```

```
elm = STAILQ_FIRST(&r_hist_);
```

```
while (elm != NULL && num_later <= num_dup_acks_){ num_later;
```

```
elm = STAILQ_NEXT(elm, linfo_);
```

```
}
```

```
if (elm != NULL){
```

```
elm = findDataPacketInRecvHistory(STAILQ_NEXT(elm,linfo_));
```

```
if (elm != NULL){
```

elm2 = STAILQ_NEXT(elm, linfo_); while(elm2 != NULL){

```
if (elm2->seq_num_ < seq_num && elm2->t_recv_ <
```

time){

STAILQ_REMOVE(&r_hist_,elm2,r_hist_entry,linfo_); delete elm2;

} else

```
elm = elm2;
```

```
elm2 = STAILQ_NEXT(elm, linfo_);
```

```
}
```

```
}
```

```
}
```

```
}
```

```
void DCCPTFRCAgent::removeAcksRecvHistory(){ struct r_hist_entry *elm1 = STAILQ_FIRST(&r_hist_); s
```

```
int num_later = 1;
```

```
while (elm1 != NULL && num_later <= num_dup_acks_){ num_later;
```

```
elm1 = STAILQ_NEXT(elm1, linfo_);
```

```
}
```

```
if(elm1 == NULL)
```

```
return;
```

elm2 = STAILQ_NEXT(elm1, linfo_); while(elm2 != NULL){

if (elm2->type_ == DCCP_ACK){ STAILQ_REMOVE(&r_hist_,elm2,r_hist_entry,linfo_); delete elm2;

} else {

```
elm1 = elm2;
```

```
}
```

```
elm2 = STAILQ_NEXT(elm1, linfo_);
```

```
}
```

```
}
```

inline r_hist_entry

```
*DCCPTFRCAgent::findDataPacketInRecvHistory(r_hist_entry *start){ while(start != NULL && start->type_
```

```
start = STAILQ_NEXT(start,linfo_); return start;
```

```
}
```

Ex.No: 7 Study of TCP/UDP performance using Simulation tool.

PROGRAM:

```
set ns [new Simulator]
```

$ns color 0 Blue

$ns color 1 Red

```
$ns color 2 Yellow set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node]
```

```
set f [open tcpout.tr w]
```

```
$ns trace-all $f
```

```
set nf [open tcpout.nam w]
```

$ns namtrace-all $nf

$ns duplex-link $n0 $n2 5Mb 2ms DropTail

$ns duplex-link $n1 $n2 5Mb 2ms DropTail

$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail

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

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

$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_ 1

```
set sink [new Agent/TCPSink]
```

```
$ns attach-agent $n1 $tcp
```

```
$ns attach-agent $n3 $sink
```

```
$ns connect $tcp $sink
```

```
set ftp [new Application/FTP]
```

$ftp attach-agent $tcp

```
$ns at 1.2 "$ftp start"
```

```
$ns at 1.35 "$ns detach-agent $n1 $tcp ; $ns detach-agent $n3 $sink"
```

```
$ns at 3.0 "finish" proc finish {} {
```

```
}
```

$ns run

Output

```
$ns flush-trace close $f
```

```
close $nf
```

```
puts "Running nam.."
```

exec xgraph tcpout.tr -geometry 600x800 & exec nam tcpout.nam &

```
exit 0
```

UDP Performance

PROGRAM:

```
set ns [new Simulator]
```

$ns color 0 Blue

$ns color 1 Red

```
$ns color 2 Yellow set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node]
```

```
set f [open udpout.tr w]
```

```
$ns trace-all $f
```

```
set nf [open udpout.nam w]
```

$ns namtrace-all $nf

$ns duplex-link $n0 $n2 5Mb 2ms DropTail

$ns duplex-link $n1 $n2 5Mb 2ms DropTail

$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail

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

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

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

$ns duplex-link-op $n2 $n3 queuePos 0.5 set udp0 [new Agent/UDP]

```
$ns attach-agent $n0 $udp0
```

```
set cbr0 [new Application/Traffic/CBR]
```

$cbr0 attach-agent $udp0 s

$

$ns attach-agent $n1 $null0 set null1 [new Agent/Null]

```
$ns attach-agent $n1 $null1
```

```
$ns connect $udp0 $null0
```

```
$ns connect $udp1 $null1
```

```
$ns at 1.0 "$cbr0 start"
```

```
$ns at 1.1 "$cbr1 start"
```

```
puts [$cbr0 set packetSize_] puts [$cbr0 set interval_]
```

```
$ns at 3.0 "finish" proc finish {} {
```

global ns f nf

```
$ns flush-trace close $f
```

```
close $nf
```

```
puts "Running nam.." exec nam udpout.nam & exit 0
```

```
}
```

$ns run

Output:

Ex.No: 8 Simulation of Distance Vector/ Link State Routing algorithm.

LINK STATE ROUTING PROTOCOL

PROGRAM

```
set ns [new Simulator]
```

$ns rtproto LS

```
set nf [open linkstate.nam w]
```

$ns namtrace-all $nf

```
set f0 [open linkstate.tr w]
```

```
$ns trace-all $f0 proc finish {} {
```

global ns f0 nf

```
$ns flush-trace close $f0 close $nf
```

```
exec nam linkstate.nam & exit 0
```

```
}
```

```
for {set i 0} {$i <7} {incr i} { set n($i) [$ns node]
```

```
}
```

```
for {set i 0} {$i <7} {incr i} {
```

$ns duplex-link $n($i) $n([expr ($i+1)%7]) 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(3) $null0
```

```
$ns connect $udp0 $null0
```

```
$ns at 0.5 "$cbr0 start"
```

$ns rtmodel-at 1.0 down $n(1) $n(2)

```
$ns at 5.0 "finish"
```

$ns run

Output:

DISTANCE VECTOR ROUTING ALGORITHM

PROGRAM

#Distance vector routing protocol ? distvect.tcl #Create a simulator object

```
set ns [new Simulator]
```

#Use distance vector routing

$ns rtproto DV

#Open the nam trace file set nf [open out.nam w]

```
$ns namtrace-all $nf # Open tracefile
```

```
set nt [open trace.tr w]
```

```
$ns trace-all $nt
```

#Define 'finish' procedur

```
proc finish {}
```

{

global ns nf

$ns flush-trace #Close the trace file close $nf

```
#Execute nam on the trace file exec nam -a out.nam &
```

```
exit 0
```

```
}
```

```
# Create 8 nodes set n1 [$ns node] set n2 [$ns node] set n3 [$ns node] set n4 [$ns node] set n5 [$ns node
```

Specify link characterestics

$ns duplex-link $n1 $n2 1Mb 10ms DropTail

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

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

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

$ns duplex-link $n5 $n6 1Mb 10ms DropTail

$ns duplex-link $n6 $n7 1Mb 10ms DropTail

$ns duplex-link $n7 $n8 1Mb 10ms DropTail

$ns duplex-link $n8 $n1 1Mb 10ms DropTail # specify layout as a octagon

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

```
$ns duplex-link-op $n2 $n3 orient up
```

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

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

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

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

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

$ns duplex-link-op $n8 $n1 orient left #Create a UDP agent and attach it to node n1 set udp0 [new Agent/U

```
$ns attach-agent $n1 $udp0
```

#Create a CBR traffic source and attach it to udp0

```
$cbr0 set packetSize_ 500
```

```
$cbr0 set interval_ 0.005
```

```
$cbr0 attach-agent $udp0
```

#Create a Null agent (a traffic sink) and attach it to node n4 set null0 [new Agent/Null]

```
$ns attach-agent $n4 $null0
```

#Connect the traffic source with the traffic sink

```
$ns connect $udp0 $null0
```

#Schedule events for the CBR agent and the network dynamics

```
$ns at 0.0 "$n1 label Source"
```

```
$ns at 0.0 "$n4 label Destination"
```

```
$ns at 0.5 "$cbr0 start"
```

$ns rtmodel-at 1.0 down $n3 $n4

$ns rtmodel-at 2.0 up $n3 $n4

```
$ns at 4.5 "$cbr0 stop"
```

#Call the finish procedure after 5 seconds of simulation time

```
$ns at 5.0 "finish" #Run the simulation
```

$ns run

OUTPUT

$ ns distvect.tcl

Ex.No:9 Performance Evaluation of Routing protocols using Simulation tool.

PROGRAM:

```
set ns [new Simulator]
```

#Define different colors for data flows (for NAM)

$ns color 1 Blue

```
$ns color 2 Red #Open the Trace file
```

```
set file1 [open out.tr w]
```

```
$ns trace-all $file1
```

```
#Open the NAM trace file set file2 [open out.nam w]
```

$ns namtrace-all $file2 #Define a 'finish' procedure proc finish {}

{

global ns file1 file2

```
$ns flush-trace close $file1 close $file2
```

```
exec nam out.nam &
```

```
# Next line should be commented out to have the static routing
```

$ns rtproto DV #Create six nodes set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n4 [$ns node] s

#Create links between the nodes

$ns duplex-link $n0 $n1 0.3Mb 10ms DropTail

$ns duplex-link $n1 $n2 0.3Mb 10ms DropTail

$ns duplex-link $n2 $n3 0.3Mb 10ms DropTail

$ns duplex-link $n1 $n4 0.3Mb 10ms DropTail

$ns duplex-link $n3 $n5 0.5Mb 10ms DropTail

$ns duplex-link $n4 $n5 0.5Mb 10ms DropTail

#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 up
```

$ns duplex-link-op $n1 $n4 orient up-left

$ns duplex-link-op $n3 $n5 orient left-up

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

#Setup a TCP connection

```
set tcp [new Agent/TCP/Newreno]
```

```
$ns attach-agent $n0 $tcp
```

```
set sink [new Agent/TCPSink/DelAck]
```

```
$ns attach-agent $n5 $sink
```

```
$ns connect $tcp $sink
```

$tcp set fid_ 1

#Setup a FTP over TCP connection set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ftp set type_ FTP

$ns rtmodel-at 4.5 up $n1 $n4

```
$ns at 0.1 "$ftp start"
```

$ns at 6.0 "finish"

$ns run

MULTICASTING ROUTING PROTOCOL

PROGRAM:

Create scheduler

```
#Create an event scheduler wit multicast turned on set ns [new Simulator -multicast on]
```

#$ns multicast #Turn on Tracing

```
set tf [open output.tr w]
```

```
$ns trace-all $tf
```

```
# Turn on nam Tracing set fd [open mcast.nam w]
```

$ns namtrace-all $fd # Create nodes

```
set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node] set n4 [$ns node] set n5 [$ns node
```

Create links

$ns duplex-link $n0 $n2 1.5Mb 10ms DropTail

$ns duplex-link $n1 $n2 1.5Mb 10ms DropTail

$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail

$ns duplex-link $n3 $n4 1.5Mb 10ms DropTail

$ns duplex-link $n3 $n7 1.5Mb 10ms DropTail

$ns duplex-link $n4 $n5 1.5Mb 10ms DropTail

$ns duplex-link $n4 $n6 1.5Mb 10ms DropTail

Routing protocol: say distance vector #Protocols: CtrMcast, DM, ST, BST set mproto DM

```
set mrthandle [$ns mrtproto $mproto {}]
```

Allocate group addresses set group1 [Node allocaddr] set group2 [Node allocaddr]

```
$ns attach-agent $n0 $udp0
```

```
$udp0 set dst_addr_ $group1
```

```
$udp0 set dst_port_ 0
```

```
set cbr1 [new Application/Traffic/CBR]
```

```
$cbr1 attach-agent $udp0
```

Transport agent for the traffic source set udp1 [new Agent/UDP]

```
$ns attach-agent $n1 $udp1
```

```
$udp1 set dst_addr_ $group2
```

```
$udp1 set dst_port_ 0
```

```
set cbr2 [new Application/Traffic/CBR]
```

```
$cbr2 attach-agent $udp1
```

Create receiver

```
set rcvr1 [new Agent/Null]
```

```
$ns attach-agent $n5 $rcvr1
```

$ns at 1.0 "$n5 join-group $rcvr1 $group1" set rcvr2 [new Agent/Null]

```
$ns attach-agent $n6 $rcvr2
```

$ns at 1.5 "$n6 join-group $rcvr2 $group1" set rcvr3 [new Agent/Null]

```
$ns attach-agent $n7 $rcvr3
```

$ns at 2.0 "$n7 join-group $rcvr3 $group1" set rcvr4 [new Agent/Null]

```
$ns attach-agent $n5 $rcvr1
```

$ns at 2.5 "$n5 join-group $rcvr4 $group2" set rcvr5 [new Agent/Null]

```
$ns attach-agent $n6 $rcvr2
```

$ns at 3.0 "$n6 join-group $rcvr5 $group2" set rcvr6 [new Agent/Null]

```
$ns attach-agent $n7 $rcvr3
```

```
$ns at 3.5 "$n7 join-group $rcvr6 $group2"
```

```
$ns at 4.0 "$n5 leave-group $rcvr1 $group1"
```

```
$ns at 4.5 "$n6 leave-group $rcvr2 $group1"
```

```
$ns at 5.0 "$n7 leave-group $rcvr3 $group1"
```

```
$ns at 5.5 "$n5 leave-group $rcvr4 $group2"
```

```
$ns at 6.0 "$n6 leave-group $rcvr5 $group2"
```

```
$ns at 6.5 "$n7 leave-group $rcvr6 $group2" # Schedule events
```

```
$ns at 0.5 "$cbr1 start"
```

```
$ns at 9.5 "$cbr1 stop"
```

```
$ns at 0.5 "$cbr2 start"
```

```
$ns at 9.5 "$cbr2 stop"
```

#post-processing

```
$ns at 10.0 "finish" proc finish {}
```

{

global ns tf

```
$ns flush-trace close $tf
```

```
exec nam mcast.nam & exit 0
```

```
}
```

For nam

#Colors for packets from two mcast groups

$ns color 10 red

$ns color 11 green

$ns color 30 purple

$ns color 31 green

```
# Manual layout: order of the link is significant! #$ns duplex-link-op $n0 $n1 orient right
```

#$ns duplex-link-op $n0 $n2 orient right-up #$ns duplex-link-op $n0 $n3 orient right-down # Show queue o

```
#$ns duplex-link-op $n2 $n3 queuePos 0.5
```

Group 0 source

```
$udp0 set fid_ 10
```

$n0 color red

$n0 label "Source 1"

Group 1 source

```
$udp1 set fid_ 11
```

$n1 color green

$n1 label "Source 2"

$n5 label "Receiver 1"

$n5 color blue

$n6 label "Receiver 2"

$n6 color blue

$n7 label "Receiver 3"

$n7 color blue

#$n2 add-mark m0 red #$n2 delete-mark m0"

Animation rate

$ns set-animation-rate 3.0ms

$ns run

Ex.No:10 Simulation of ErrorDetection Code (like CRC)

PROGRAM:

```
import java.io.*; class crc_gen
```

{

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

BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); int[] data;

```
int[] div; int[] divisor; int[] rem;
```

```
int[] crc;
```

```
int data_bits, divisor_bits, tot_length;
```

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

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

System.out.println("Enter number of bits in divisor : "); divisor_bits=Integer.parseInt(br.readLine()); divisor=

```
for(int i=0; i<divisor_bits; i++) divisor[i]=Integer.parseInt(br.readLine()); System.out.print("Data bits are : ");
```

```
System.out.print("divisor bits are : ");        for(int i=0; i< divisor_bits; i++) System.out.print(divisor[i]); Sys
```

*/ tot_length=data_bits+divisor_bits-1; div=new int[tot_length];

```
rem=new int[tot_length]; crc=new int[tot_length];
```

```
/* CRC GENERATION */ for(int i=0;i<data.length;i++)
```

```
div[i]=data[i];
```

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

```
System.out.println();
```

```
for(int j=0; j<div.length; j++){ rem[j] = div[j];
```

```
}
```

rem=divide(div, divisor, rem); for(int i=0;i<div.length;i++)

{

```
//append dividend and remainder crc[i]=(div[i]^rem[i]);
```

```
}
```

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

```
/* ERROR DETECTION */ System.out.println();
```

System.out.println("Enter CRC code of "+tot_length+" bits : "); for(int i=0; i<crc.length; i++) crc[i]=Integer.pa

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

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

```
}
```

rem=divide(crc, divisor, rem); for(int i=0; i< rem.length; i++)

{

```
if(rem[i]!=0)
```

{

System.out.println("Error"); break;

```
}
```

```
if(i==rem.length-1) System.out.println("No Error");
```

```
}
```

```
System.out.println("THANK YOU. )");
```

```
}
```

```
static int[] divide(int div[],int divisor[], int rem[])
```

{

int cur=0; while(true)

{

```
for(int i=0;i<divisor.length;i++) rem[cur+i]=(rem[cur+i]^divisor[i]); while(rem[cur]==0 && cur!=rem.length-1) c
```

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

```
}
```

```
return rem;
```

```
}
```

```
}
```

OUTPUT :

Enter number of data bits :

Enter data bits :

1

1

1

1

Enter number of bits in divisor :

Enter Divisor bits :

1

1

Dividend (after appending 0's) are :

CARRIER SENSE MULTIPLE ACCESS

PROGRAM:

```
set ns [new Simulator]
```

$ns color 1 blue

$ns color 2 red

```
set fi1 [open out.tr w]
```

```
set winfile [open WinFile w]
```

```
$ns trace-all $fi1
```

```
set fi2 [open out.nam w]
```

```
$ns namtrace-all $fi2 proc finish {}
```

{

global ns fi1 fi2

```
$ns flush-trace close $fi1 close $fi2
```

```
exec nam out.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]
```

$n1 color red

$n1 shape box

$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] set tcp [new

```
$ns attach-agent $n0 $tcp
```

```
set sink [new Agent/TCPSink/DelAck]
```

```
$ns attach-agent $n4 $sink
```

```
$ns connect $tcp $sink
```

$tcp set fid_ 1

$tcp set window_ 8000

```
$tcp set packetsize_ 552
```

```
set ftp [new Application/FTP]
```

$ftp set type_ FTP

```
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

```
set cbr [new Application/Traffic/CBR]
```

$cbr attach-agent $udp

$cbr set type_ CBR

```
$cbr set packet_size_ 1000
```

```
$cbr set rate_ 0.01mb
```

```
$cbr set random_ false
```

```
$ns at 0.1 "$cbr start"
```

```
$ns at 1.0 "$ftp start"
```

```
$ns at 24.0 "$ftp stop"
```

```
$ns at 24.5 "$cbr stop"
```

```
proc plotwindow { tcpSource file }
```

{

```
global ns set time 0.1
```

```
set now [$ns now]
```

```
set cwnd [$tcpSource set cwnd_] set wnd [$tcpSource set window_] puts $file "$now $cwnd"
```

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

```
}
```

$ns at 1.0 "plotwindow $tcp $winfile"

```
$ns at 5 "$ns trace-annotate \"packet drop\""
```

$ns at 125.0 "finish"

$ns run

OUTPUT: