# EXPERIMENT NO: - 01

## Implement three nodes’ point-to-point network with duplex links between them. Set thequeue size, vary the bandwidth, and find the number of packets dropped.

# This script is created by NSG2 beta1

# <<http://wushoupong.googlepages.com/nsg>> #================================

===

# Simulation parameters setup #================================

===

set val(stop) 10.0 ;# time of simulation end#===================================

# Initialization #================================

===

#Create a ns simulator set ns [new Simulator] #Open the NS trace file

set tracefile [open b1.tr w]

$ns trace-all $tracefile #Open the NAM trace file set namfile [open b1.nam w]

$ns namtrace-all $namfile #================================

===

# Nodes Definition #================================

===

#Create 3 nodes set n0 [$ns node]set n1 [$ns node]set n2 [$ns node]

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

# Links Definition #================================

===

#Createlinks between nodes

$ns duplex-link $n0 $n1 1.0Mb 10ms DropTail

$ns queue-limit $n0 $n1 5

$ns duplex-link $n2 $n1 1.0Mb 10ms DropTail

$ns queue-limit $n2 $n1 5 #Give node position (for NAM)

$ns duplex-link-op $n0 $n1 orient left-down

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

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

===

# Agents Definition #================================

===

#Setup a TCP connection set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set sink1 [new Agent/TCPSink]

$ns attach-agent $n2 $sink1

$ns connect $tcp0 $sink1

$tcp0 set packetSize\_ 2000 #================================

===

# 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" #================================

===

# Termination #================================

===

#Define a 'finish' procedureproc finish {} {

global ns tracefile namfile

$ns flush-

trace close

$tracefile close

$namfile

exec nam b1.nam &

exec echo "no of packets dropped is" & exec grep -c "^d" b1.tr &

exit 0

}

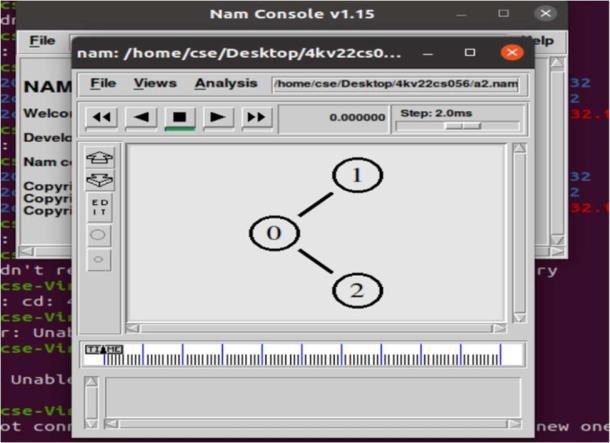
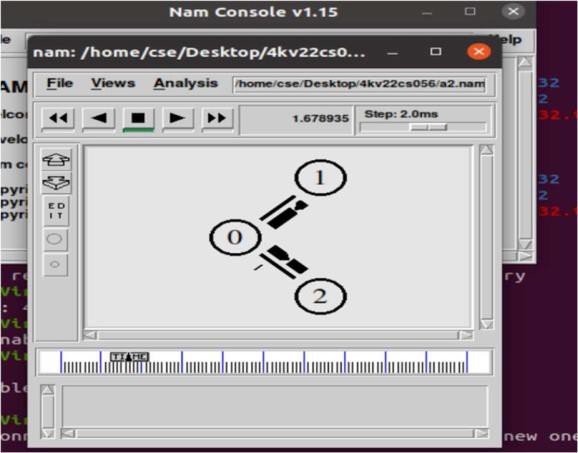
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "finish"

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

$ns run

**OUTPUT**:



# EXPERIMENT NO:- 02

## Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

# This script is created by NSG2 beta1

# <<http://wushoupong.googlepages.com/nsg>> #===================================

# Simulation parameters setup #===================================

set val(stop) 10.0 ;# time of simulation end #===================================

# Initialization #===================================

#Create a ns simulator set ns [new Simulator] #Open the NS trace file set tracefile [open a1.tr w]

$ns trace-all $tracefile #Open the NAM trace file set namfile [open a1.nam w]

$ns namtrace-all $namfile

$ns color 1 red

$ns color 2 blue

$ns color 3 green

$ns color 4 yellow #===================================

# 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]

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

# Links Definition #===================================

#Createlinks between nodes

$ns duplex-link $n0 $n1 1.0Mb 10ms DropTail

$ns queue-limit $n0 $n1 5

$ns duplex-link $n1 $n2 1.0Mb 10ms DropTail

$ns queue-limit $n1 $n2 5

$ns duplex-link $n2 $n3 1.0Mb 10ms DropTail

$ns queue-limit $n2 $n3 5

$ns duplex-link $n3 $n4 1.0Mb 10ms DropTail

$ns queue-limit $n3 $n4 5

$ns duplex-link $n4 $n5 1.0Mb 10ms DropTail

$ns queue-limit $n4 $n5 5 #Give node position (for NAM)

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

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

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

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

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

Agent/Ping instproc recv {from rtt} {

$self instvar node\_

puts "node [$node\_ id] recieved a ping answer from $from with rtt=$rtt ms"

} #===================================

# Agents Definition #===================================

#Setup a TCP connection set tcp0 [new Agent/TCP]

$tcp0 set class\_ 1

$ns attach-agent $n1 $tcp0

set sink1 [new Agent/TCPSink]

$ns attach-agent $n3 $sink1

$ns connect $tcp0 $sink1

$tcp0 set packetSize\_ 1500 #Setup a UDP connection set udp2 [new Agent/UDP]

$udp2 set class\_ 2

$ns attach-agent $n2 $udp2 set null3 [new Agent/Null]

$ns attach-agent $n4 $null3

$ns connect $udp2 $null3

$udp2 set packetSize\_ 1500 #setup a ping connection set p0 [new Agent/Ping]

$p0 set class\_ 3

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

$p1 set class\_ 4

$ns attach-agent $n5 $p1

$ns connect $p0 $p1 #===================================

# Applications Definition #===================================

#Setup a CBR Application over TCP connection set cbr0 [new Application/Traffic/CBR]

$cbr0 attach-agent $udp2

$cbr0 set packetSize\_ 1000

$cbr0 set rate\_ 1.0Mb

$cbr0 set random\_ null

$ns at 1.0 "$cbr0 start"

$ns at 2.0 "$cbr0 stop"

#Setup a FTP Application over UDP connection set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp0

$ns at 1.0 "$ftp1 start"

$ns at 2.0 "$ftp1 stop"

$ns at 1.0 "$p0 send"

$ns at 1.2 "$p1 send"

$ns at 1.4 "$p0 send"

$ns at 2.0 "$p1 send"

$ns at 2.2 "$p0 send"

$ns at 2.4 "$p1 send"

$ns at 3.0 "$p0 send"

$ns at 3.2 "$p1 send"

$ns at 3.4 "$p0 send"

$ns at 3.6 "$p1 send #===================================

# Termination #===================================

#Define a 'finish' procedure proc finish {} {

global ns tracefile namfile

$ns flush-trace close $tracefile close $namfile

exec nam a1.nam &

puts "no of packets droped"

exec grep "^d" a1.tr | grep -c "ping" & exit 0

}

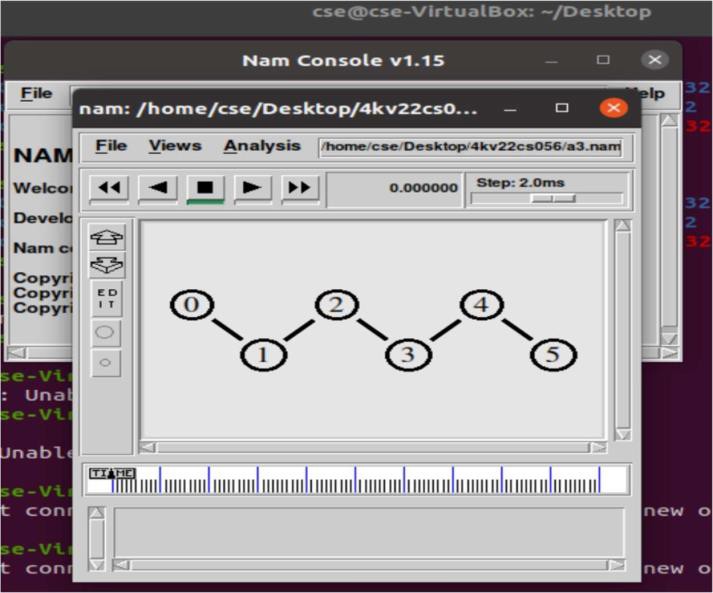
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "finish"

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

$ns run

**OUTPUT**



EXPERIMENT NO:- 03

## Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plotcongestion window for different source / destination.

# This script is created by NSG2 beta1

# <<http://wushoupong.googlepages.com/nsg>>

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

# Simulation parameters setup #===================================

set val(stop) 10.0 ;# time of simulation end

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

# Initialization #===================================

#Create a ns simulator set ns [new Simulator]

#Open the NS trace file

set tracefile [open lab3.tr w]

$ns trace-all $tracefile #Open the NAM trace file

set namfile [open lab3.nam w]

$ns namtrace-all $namfile set t [open tahoe w]

set r [open reno w] set v [open vegas w]

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

# Nodes Definition #===================================

#Create 11 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] set n10 [$ns node]

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

# Links Definition #===================================

#Createlinks between nodes

$ns duplex-link $n0 $n1 1.0Mb 10ms DropTail

$ns queue-limit $n0 $n1 50

$ns duplex-link $n2 $n1 1.0Mb 10ms DropTail

$ns queue-limit $n2 $n1 50

$ns duplex-link $n1 $n3 1.0Mb 10ms DropTail

$ns queue-limit $n1 $n3 50

set lan [$ns newLan "$n3 $n4 $n5 $n6 $n7 $n8 $n9 $n10" 0.5mb 100ms || Queue/DropTail MAC/802\_3 channel]

#Give node position (for NAM)

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

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

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

# Agents Definition #===================================

#Setup a TCP/Reno connection set tcp0 [new Agent/TCP/Reno]

$ns attach-agent $n0 $tcp0

set sink1 [new Agent/TCPSink]

$ns attach-agent $n10 $sink1

$ns connect $tcp0 $sink1

$tcp0 set packetSize\_ 1500

#Setup a TCP/FullTcp/Tahoe connection set tcp2 [new Agent/TCP/FullTcp/Tahoe]

$ns attach-agent $n2 $tcp2

set sink3 [new Agent/TCPSink]

$ns attach-agent $n7 $sink3

$ns connect $tcp2 $sink3

$tcp2 set packetSize\_ 1500 #Setup a TCP/Vegas connection set tcp4 [new Agent/TCP/Vegas]

$ns attach-agent $n4 $tcp4

set sink6 [new Agent/TCPSink]

$ns attach-agent $n9 $sink6

$ns connect $tcp4 $sink6

$tcp4 set packetSize\_ 1500 #===================================

# Applications Definition #===================================

#Setup a FTP Application over TCP/Reno connection set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

$ns at 1.0 "$ftp0 start"

$ns at 1.0 "plot $tcp0 $r"

$ns at 2.0 "$ftp0 stop"

#Setup a FTP Application over TCP/FullTcp/Tahoe connection set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp2

$ns at 1.0 "$ftp1 start"

$ns at 1.0 "plot $tcp2 $t"

$ns at 2.0 "$ftp1 stop"

#Setup a FTP Application over TCP/Vegas connection set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp4

$ns at 1.0 "$ftp2 start"

$ns at 1.0 "plot $tcp2 $v"

$ns at 2.0 "$ftp2 stop #===================================

# Termination #===================================

#Define a 'finish' procedure proc finish {} {

global ns tracefile namfile

$ns flush-trace

close $tracefile close $namfile

exec nam lab3.nam &

exec Xgraph reno vegas tahoe & exit 0

}

proc plot {tcps file} { global ns

set now [$ns now]

set cwnd [$tcps set cwnd\_] puts $file "snow $cwnd"

$ns at [expr $now+0.1] "plot $tcps $file"

}

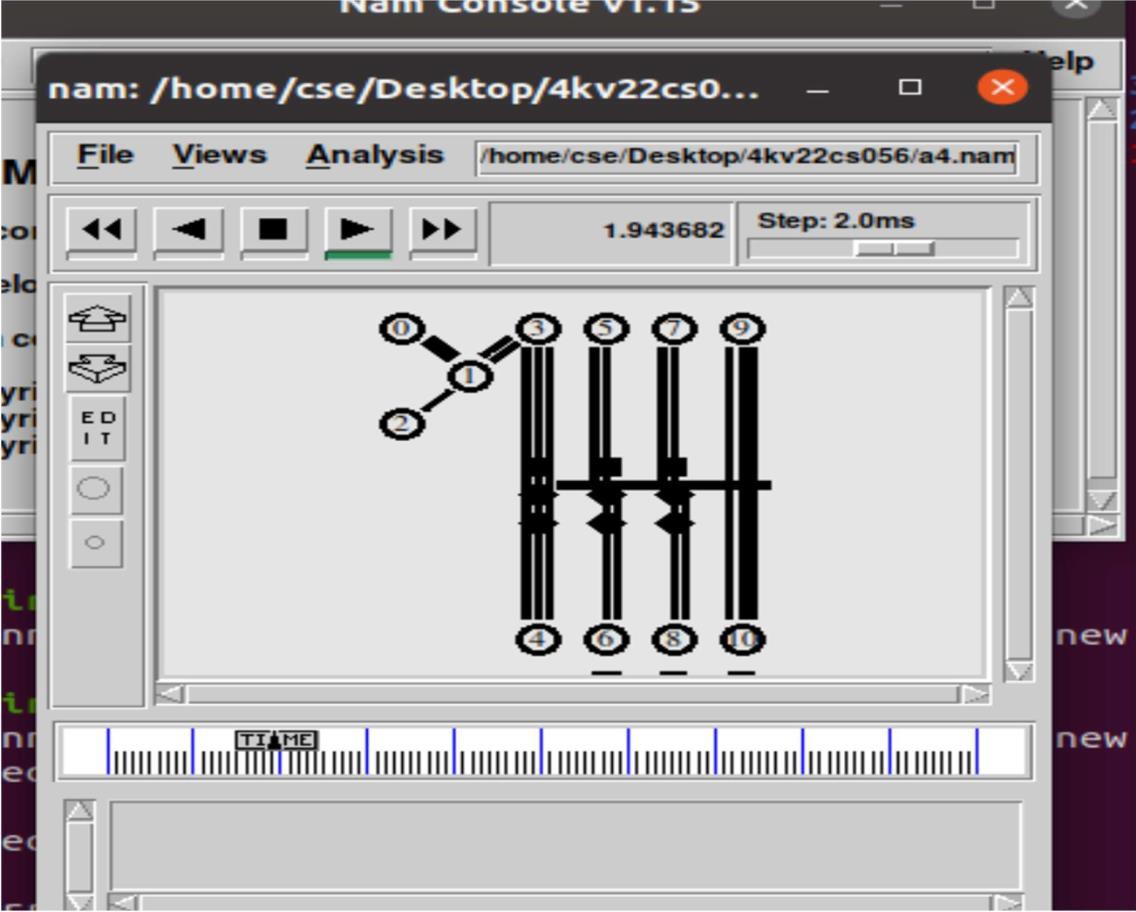
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "finish"

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

$ns run

**OUTPUT**:



EXPERIMENT NO:- 04

## Develop a program for error detecting code using CRC-CCITT (16-BITS).

import java.util.\*; class CRC{

public static void main(String args[])

{

Scanner scan=new Scanner(system.in); int n;

system.out.println(“enter size of data”); n=scan.nextInt();

int data[]=newint[n]; system.out.println(“entr the data bit by bit:”); for(int i=0;i<n;i++){

data[i]=scan.nextInt();} system.out.println(“enter the size of divisor”); n=scan.nextInt();

int divisor[]=ewint[n];

system.out.println(“enter the divisor,bit by bit”); for(int i=0;i<n;i++){

divisor[i]=scan.nextInt();}

int remainder[]=divide(data,divisor); system.out.print(“Remainder=”); for(int i=0;i<remainder.length-1;i++){ system.out.println(remainder[i]);

}

System.out.println(“\n The CRC code generated is:”); for(int i=0;i<data.length;i++){ System.out.print(data[i]);}

for(int i=0;i<remainder.length-1;i++){ System.out.print(remainder[i]);} System.out.println();

int sent data[]=new int[data.length+remainder.length-1];

system.out.println(“enter the data received at the receiver”); for(i=0;i<sent\_dta.length;i++){

sent\_data[i]=scan.nectInt();} receive(sent\_dta,divisor);

}

Stractic int[] divide(int old\_data[],int divisor[]){ int remainder[],i;

int data[]=new int[old\_data.length+divisor.length]; system.arraycopy(data,0,remainder,0,divisor,length); for(i=0;i<old\_data.length;i++){ if(remainder[0]==1){

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

remainder[j-1]=exor(remainder[j],divisor[j]);}} else{

for(int j=1;i<divisor.length;j++){ remainder[j-1]=exor(remainder[j],0);

}

Remainder[divisor.length-1]=data[i+divisor.length]; Return remainder;

}

Static int exor(int a,int b){ if(a==b){

return 0;}

return 1;}

static void receive(int data[],int divisor[]){ int remainder[]=divide(data,divisor); for(int i=0;i<remainder.length;i++){ if(remainder[i]!=0){

system.out.println(“There is an error in therecieved data:”);}} system.out.println(“data was received without any error:”);

}

}

**OUTPUT:**

**Enter the size of the data:16**

**Enter the data,bit by bit:1 0 0 1 1 1 0 1 1 1 0 0 1 1 1 0 Enter the size of the divisor:5**

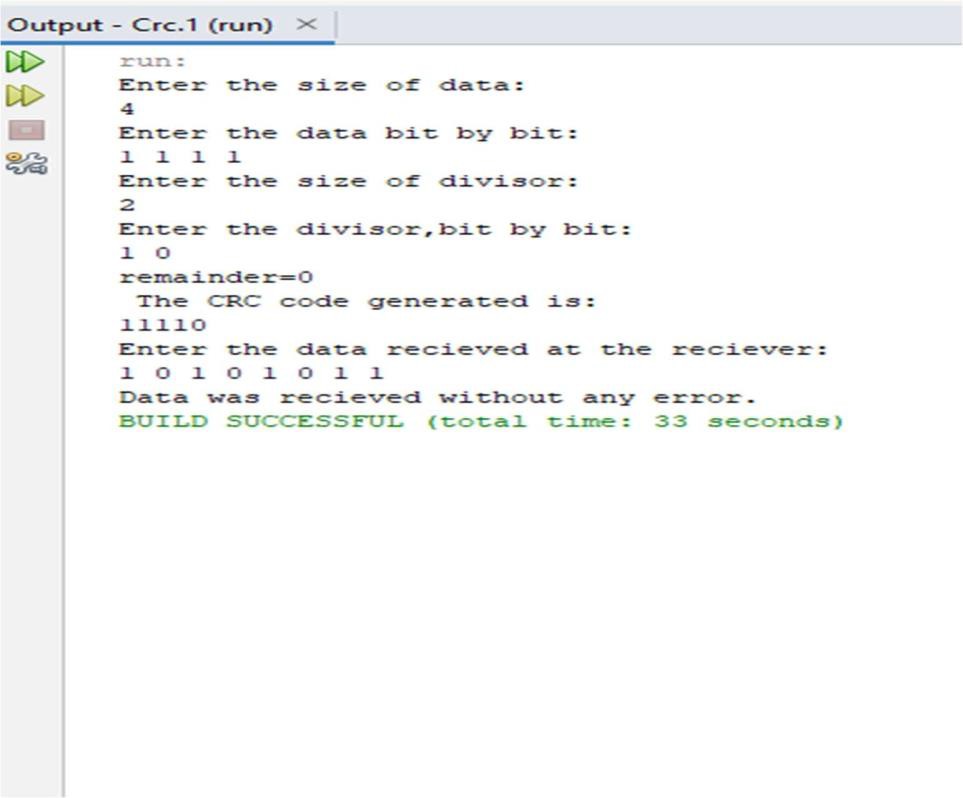
**Enter the divisor,bit by bit:1 0 1 1 1**

**Remainder=1 1 0 0**

**The CRC code generated is:1 0 0 1 1 1 0 1 1 1 0 0 1 1 1 0 1 1 0 0 Enter the data received at the receiver:**

**1 0 0 1 1 1 0 1 1 1 0 0 1 1 1 0 1 1 0 1**

**There is an error in received data.**



EXPERIMENT NO:- 05

## Develop a program to implement a sliding window protocol in the data link layer.

import java.util.Scanner; import java.util.Random;

public class SlidingWindowProtocol {

static int windowSize, totalFrames, sendBase, nextSeqNum; static boolean[] acknowledged;

static void sendFrame(int frameNumber) { System.out.println("Sending frame " + frameNumber);

}

static int receiveAck() { Random rand = new Random(); if (rand.nextInt(100) < 90) {

int ackFrame = sendBase + rand.nextInt(windowSize); System.out.println("Acknowledgment received for frame " + ackFrame); return ackFrame;

} else { System.out.println("Acknowledgment lost!"); return -1;

} }

static void slideWindow(int ackFrame) {

while (sendBase<= ackFrame&&sendBase<totalFrames) { acknowledged[sendBase] = true;

System.out.println("Frame " + sendBase + " acknowledged."); sendBase++;

} }

static void runSlidingWindow() { while (sendBase<totalFrames) {

while (nextSeqNum<sendBase + windowSize&&nextSeqNum<totalFrames) { sendFrame(nextSeqNum);

nextSeqNum++;

}

int ackFrame = receiveAck(); if (ackFrame != -1) { slideWindow(ackFrame);

} else {

System.out.println("Timeout! Resending frames...");

}

try { Thread.sleep(1000);

} catch (InterruptedException e) { System.out.println(e);

}

}

}

public static void main(String[] args) { Scanner sc = new Scanner(System.in); System.out.println("Enter the window size: "); windowSize = sc.nextInt();

System.out.println("Enter the total number of frames to be sent: "); totalFrames = sc.nextInt();

acknowledged = new boolean[totalFrames]; sendBase = 0;

nextSeqNum = 0; runSlidingWindow(); sc.close();

}

}

**OUTPUT**

**Enter the window size:**

**4**

**Enter the total number of frames to be sent: 10**

**Sending frame 0**

**Sending frame 1**

**Sending frame 2**

**Sending frame 3**

**Acknowledgment received for frame 2 Frame 0 acknowledged.**

**Frame 1 acknowledged.**

**Frame 2 acknowledged.**

**Sending frame 4**

**Sending frame 5**

**Sending frame 6**

**Sending frame 7**

**Acknowledgment received for frame 5 Frame 3 acknowledged.**

**Frame 4 acknowledged.**

**Frame 5 acknowledged**

EXPERIMENT NO:- 06

1. **Develop a program to find the shortest path between vertices using the Bellman-Fordand path vector routing algorithm.**

import java.util.Scanner; public class BellmanFord{ private int dist[];

private int n;

public static final int MAX\_VALUE=999; public BellmanFord(int n);{

this.n=n;

dist=new int[n+1];}

public void BellmanFordEvaluation(int src,int adj[][]){ for(int nd=1;nd<=n;nd++){

dist[nd]=MAX\_VALUE;} dist[src]=0;

for(int nd=1;nd<=n-1;nd++)

{

for(int sn=1;sn<=n;sn++)

{

for(int dn=1;dn<=n;dn++)

{

if(adj[sn][dn]!=MAX\_VALUE)

{

if(dist[dn]>dist[sn]+adj[sn][dn])

dist[dn]=dist[sn]+adj[sn][dn];}}}} for(int sn=1;sn<=n;sn++){

for(int dn=1;dn<=n;dn++)

{

if(adj[sn][dn]!=MAX\_VALUE){ if(dist[dn]>dist[sn]+adj[sn][dn])

System.out.println(“The graph contains negative edge cycle”);}}} for(int vertex=1;vertex<=n;vertex++){

System.out.println(“Distance of src “+src+”to”+vertex+”is”+dist[vertex]);}} public static void main(String [] args){

int n=0; int src;

Scanner sc=new Scanner(System.in); System.out.println(“Enter number of vertices”); n=sc.nextInt();

int adj[][]=new int[n+1][n+1]; System.out.println(“Enter adjacency matrix”); for(int sn=1;sn<=n;sn++){

for(int dn=1;dn<=n;dn++){ adj[sn][dn]=sc.nextInt(); if(sn==dn)

{

adj[sn][dn]=0; continue;} if(adj[sn[][dn]==0){

adj[sn][dn]=MAX\_VALUE;}}}

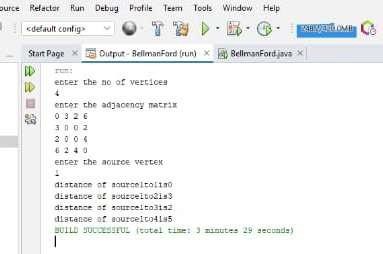
System.out.println(“Enter the src vertex:”);

Src=sc.nextInt();

BellmanFord bellmanford=new BellmanFord(n); bellmanford.BellmanFordEvaluation(src,adj);

}}

**OUTPUT**:



EXPERIMENT NO:- 07

## Using TCP/IP sockets, write a client – server program to make the client send the filename and to make the server send back the contents of the requested file if present.

import java.net.\*; import java.io.\*; import java.util.\*; public class TCPClient

{

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

{

Socket sock = new Socket("127.0.0.1",4000); System.out.println("Enter the filename");

BufferedReader BufRead1 =new BufferedReader(new InputStreamReader(System.in)); String fname =BufRead1.readLine();

OutputStream ostream =sock.getOutputStream(); PrintWriter pw=new PrintWriter(ostream,true); pw.println(fname);

InputStream istream =sock.getInputStream();

BufferedReader BufRead2 =new BufferedReader(new InputStreamReader(istream)); String str;

while((str=BufRead2.readLine())!=null)

{

System.out.println(str);

}

}

}

import java.net.\*; import java.io.\*; public class TCPServer

{

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

{

ServerSocket sersock=new ServerSocket(4000);

System.out.println("server ready for transmission"); Socket sock=sersock.accept();

System.out.println("connection is successful and waiting for chatting"); InputStream istream=sock.getInputStream();

BufferedReader bufRead1=new BufferedReader(new InputStreamReader(istream)); String fname=bufRead1.readLine();

System.out.println("the requested file from client is:"+fname); BufferedReader bufRead2=new BufferedReader(new FileReader(fname)); OutputStream ostream =sock.getOutputStream();

PrintWriter pw = new PrintWriter(ostream,true); String str; while((str=bufRead2.readLine())!=null)

{

pw.println(str);

}

sock.close();

}

}

**OUTPUT:**

**Enter the file name Data.json**

**from client:”Hello ,how are you”**

EXPERIMENT NO:- 08

## Write a program on datagram socket for client/server to display the messages on clientside, typed at the server side.

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

public class UDPClient{

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

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

String message=in.readLine();

DatagramSocket clientSocket=new DatagramSocket(); InetAddress IPAddress=InetAddress.getByName("localhost"); byte[] sendData=new byte[1024];

byte[] receiveData=new byte[1024]; sendData=message.getBytes(); DatagramPacket sendPacket=new

DatagramPacket(sendData,sendData.length,IPAddress,9876); clientSocket.send(sendPacket);

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

String ServerMessage=new String(receivePacket.getData()); System.out.println("from server:"+ServerMessage); clientSocket.close();

}

}

import java.io.\*; import java.util.\*;

import java.net.\*;

public class UDPServer{

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

BufferedReader in = new BufferedReader(new InputStreamReader(System.in)); DatagramSocket ss = new DatagramSocket(9876);

byte[] sendData=new byte[1024];

byte[] receiveData=new byte[1024]; while(true){

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

String clientmessage=new String(receivePacket.getData()); System.out.println("from client:"+clientmessage); InetAddress IPAdd=receivePacket.getAddress();

int port=receivePacket.getPort(); String message= in.readLine(); sendData=message.getBytes();

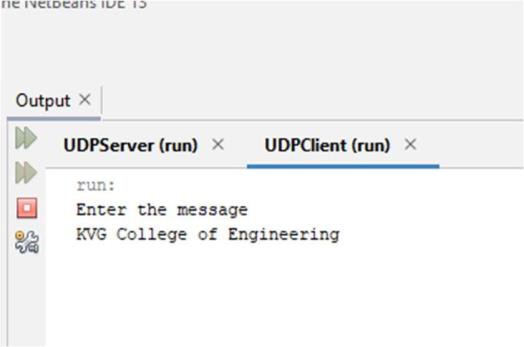
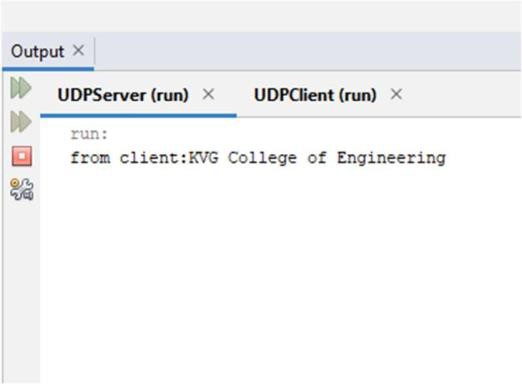
DatagramPacket sendPacket=new DatagramPacket(sendData,sendData.length,IPAdd,port); ss.send(sendPacket);

}

}

}

**OUTPUT:**



EXPERIMENT NO:- 09

## Write a program for simple RSA algorithm to encrypt and decrypt the data.

import java.io.DataInputStream; import java.io.IOException; import java.math.BigInteger; import java.util.Random;

public class RSA

{

private BigInteger p,q,N,z,e,d; private int bitlength = 1024; private Random r;

public RSA()

{

r = new Random();

p = BigInteger.probablePrime(bitlength, r);

q = BigInteger.probablePrime(bitlength, r); N = p.multiply(q);

z = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE)); e = BigInteger.probablePrime(bitlength / 2, r);

while (z.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(z)<0)

{

e.add(BigInteger.ONE);

}

d = e.modInverse(z);

}

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

{

RSA rsa = new RSA();

DataInputStream in = new DataInputStream(System.in); String teststring;

System.out.println("Enter the plain text:"); teststring = in.readLine();

System.out.println("Encrypting String: " + teststring); byte[] bytes = teststring.getBytes();

byte[] encrypted = rsa.encrypt(bytes); System.out.println("String in encrypted: "+encrypted); byte[] decrypted = rsa.decrypt(encrypted);

System.out.println("Decrypted String: " + new String(decrypted));

}

public byte[] encrypt(byte[] message)

{

return (new BigInteger(message)).modPow(e, N).toByteArray();

}

public byte[] decrypt(byte[] message)

{

return (new BigInteger(message)).modPow(d, N).toByteArray();

}

}

**OUTPUT:**

**Enter a Plain Text :**

**Hello**

**Encrypting String : Hello**

**String in Encrypted : [B@70dea4e Decrypted String: Hello**

EXPERIMENT NO:- 10

## Develop a program for congestion control using a leaky bucket algorithm.

import java.util.Scanner; public class LBucket

{ static int random(int a)

{ return (int) (Math.random() \* a);

} public static void main(String[] args) throws InterruptedException { int[] packetSize = new int[5]; for (int i = 0; i < 5; i++) {

packetSize[i] = random(10); System.out.println("PacketSize[" + i + "] : " + packetSize[i]);

}

Scanner scanner = new Scanner(System.in); System.out.print("Enter output rate : ");

int outputRate = scanner.nextInt(); System.out.print("Enter bucket size : "); int bucketSize = scanner.nextInt(); scanner.close();

int p\_zsz\_rm = 0;

for (int i = 0; i < 5; i++) {

if (packetSize[i] + p\_zsz\_rm > bucketSize)

{ if (packetSize[i] > bucketSize)

{ System.out.println("Incoming packet" + i + " of size " + packetSize[i] + " is greater than bucket capacity : PACKET REJECTION");

}

else {

System.out.println("Packet " + i + ": Bucket capacity exceeded : REJECTING new packet");

} }

else {

p\_zsz\_rm += packetSize[i];

System.out.println("Incoming packet " + i + " of size : " + packetSize[i]); System.out.println("Bytes remaining for transmission : " + p\_zsz\_rm);

int p\_time = random(6) \* 10;

System.out.println("Time left for transmission is " + p\_time); for (int clk = 10; clk <= p\_time; clk += 10)

{

Thread.sleep(1); if (p\_zsz\_rm > 0)

{

if (p\_zsz\_rm <= outputRate)

{

System.out.println("Packet of size " + p\_zsz\_rm + " transmitted"); p\_zsz\_rm = 0;

} else

{

System.out.println("Packet of size " + outputRate + " transmitted");

p\_zsz\_rm -= outputRate; System.out.println("Bytes remaining after transmission " + p\_zsz\_rm); System.out.println("Time left : " + (p\_time - clk)); }

} else {

System.out.println("No packets to transmit");

}

}

} }}}

**OUTPUT:**

