**COMPUTER NETWORK LABORATORY(**15CSL57)

**PART A**

1. Implement three nodes point – to – point network with duplex links between them.

Set the queue size, vary the bandwidth and find the number of packets dropped.

2. 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.

3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot

congestion window for different source / destination.

4. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation

and determine the performance with respect to transmission of packets.

5. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or

equivalent environment.

6. Implement and study the performance of CDMA on NS2/NS3 (Using stack called

Call net) or equivalent environment.

**PART B**

**Implement the following in Java:**

7. Write a program for error detecting code using CRC-CCITT (16- bits).

8. Write a program to find the shortest path between vertices using bellman-ford

algorithm.

9. Using TCP/IP sockets, write a client – server program to make the client send the file

name and to make the server send back the contents of the requested file if present.

10. Write a program on datagram socket for client/server to display the messages on

client side, typed at the server side.

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

12. Write a program for congestion control using leaky bucket algorithm.

1. Implement three nodes point – to – point network with duplex links between them.

#Create Simulator

set ns [new Simulator]

#Open Trace file and NAM file

set ntrace [open prog1.tr w]

$ns trace-all $ntrace

set namfile [open prog1.nam w]

$ns namtrace-all $namfile

#Finish Procedure

proc Finish {} {

global ns ntrace namfile

#Dump all the trace data and close the files

$ns flush-trace

close $ntrace

close $namfile

#Execute the nam animation file

exec nam prog1.nam &

#Show the number of packets dropped

exec echo "The number of packet drops is " &

exec grep -c "^d" prog1.tr &

exit 0

}

#Create 3 nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

#Label the nodes

$n0 label "TCP Source"

$n2 label "Sink"

#Set the color

$ns color 1 blue

#Create Links between nodes

#You need to modify the bandwidth to observe the variation in packet drop

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

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

#Make the Link Orientation

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

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

#Set Queue Size

#You can modify the queue length as well to observe the variation in packet drop

$ns queue-limit $n0 $n1 10

$ns queue-limit $n1 $n2 10

#Set up a Transport layer connection.

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set sink0 [new Agent/TCPSink]

$ns attach-agent $n2 $sink0

$ns connect $tcp0 $sink0

#Set up an Application layer Traffic

set cbr0 [new Application/Traffic/CBR]

$cbr0 set type\_ CBR

$cbr0 set packetSize\_ 100

$cbr0 set rate\_ 1Mb

$cbr0 set random\_ false

$cbr0 attach-agent $tcp0

$tcp0 set class\_ 1

#Schedule Events

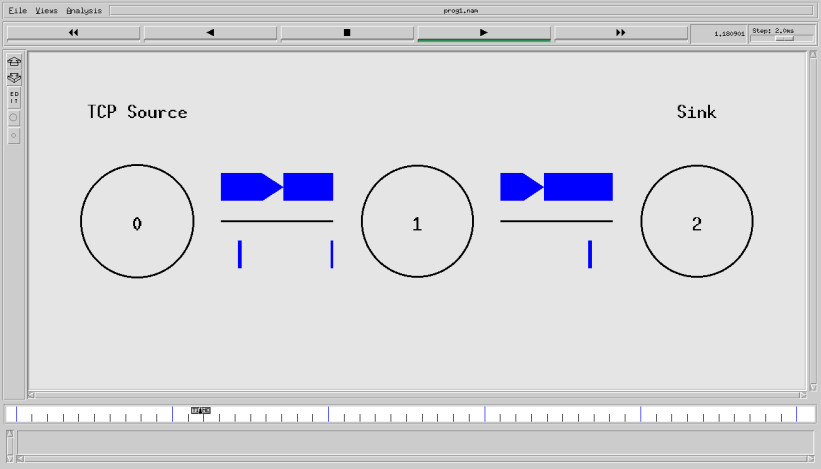
$ns at 0.0 "$cbr0 start"

$ns at 5.0 "Finish"

#Run the Simulation

$ns run

**Output**

****

2. 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.

#Create Simulator

set ns [new Simulator]

#Use colors to differentiate the traffic

$ns color 1 Blue

$ns color 2 Red

#Open trace and NAM trace file

set ntrace [open prog3.tr w]

$ns trace-all $ntrace

set namfile [open prog3.nam w]

$ns namtrace-all $namfile

#Finish Procedure

proc Finish {} {

global ns ntrace namfile

#Dump all trace data and close the file

$ns flush-trace

close $ntrace

close $namfile

#Execute the nam animation file

exec nam prog3.nam &

#Find the number of ping packets dropped

puts "The number of ping packets dropped are "

exec grep "^d" prog3.tr | cut -d " " -f 5 | grep -c "ping" &

exit 0

}

#Create six nodes

for {set i 0} {$i < 6} {incr i} {

set n($i) [$ns node]

}

#Connect the nodes

for {set j 0} {$j < 5} {incr j} {

$ns duplex-link $n($j) $n([expr ($j+1)]) 0.1Mb 10ms DropTail

}

#Define the recv function for the class 'Agent/Ping'

Agent/Ping instproc recv {from rtt} {

$self instvar node\_

puts "node [$node\_ id] received ping answer from $from with round trip time $rtt

ms"

}

#Create two ping agents and attach them to n(0) and n(5)

set p0 [new Agent/Ping]

$p0 set class\_ 1

$ns attach-agent $n(0) $p0

set p1 [new Agent/Ping]

$p1 set class\_ 1

$ns attach-agent $n(5) $p1

$ns connect $p0 $p1

#Set queue size and monitor the queue

#Queue size is set to 2 to observe the drop in ping packets

$ns queue-limit $n(2) $n(3) 2

$ns duplex-link-op $n(2) $n(3) queuePos 0.5

#Create Congestion

#Generate a Huge CBR traffic between n(2) and n(4)

set tcp0 [new Agent/TCP]

$tcp0 set class\_ 2

$ns attach-agent $n(2) $tcp0

set sink0 [new Agent/TCPSink]

$ns attach-agent $n(4) $sink0

$ns connect $tcp0 $sink0

#Apply CBR traffic over TCP

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set rate\_ 1Mb

$cbr0 attach-agent $tcp0

#Schedule events

$ns at 0.2 "$p0 send"

$ns at 0.4 "$p1 send"

$ns at 0.4 "$cbr0 start"

$ns at 0.8 "$p0 send"

$ns at 1.0 "$p1 send"

$ns at 1.2 "$cbr0 stop"

$ns at 1.4 "$p0 send"

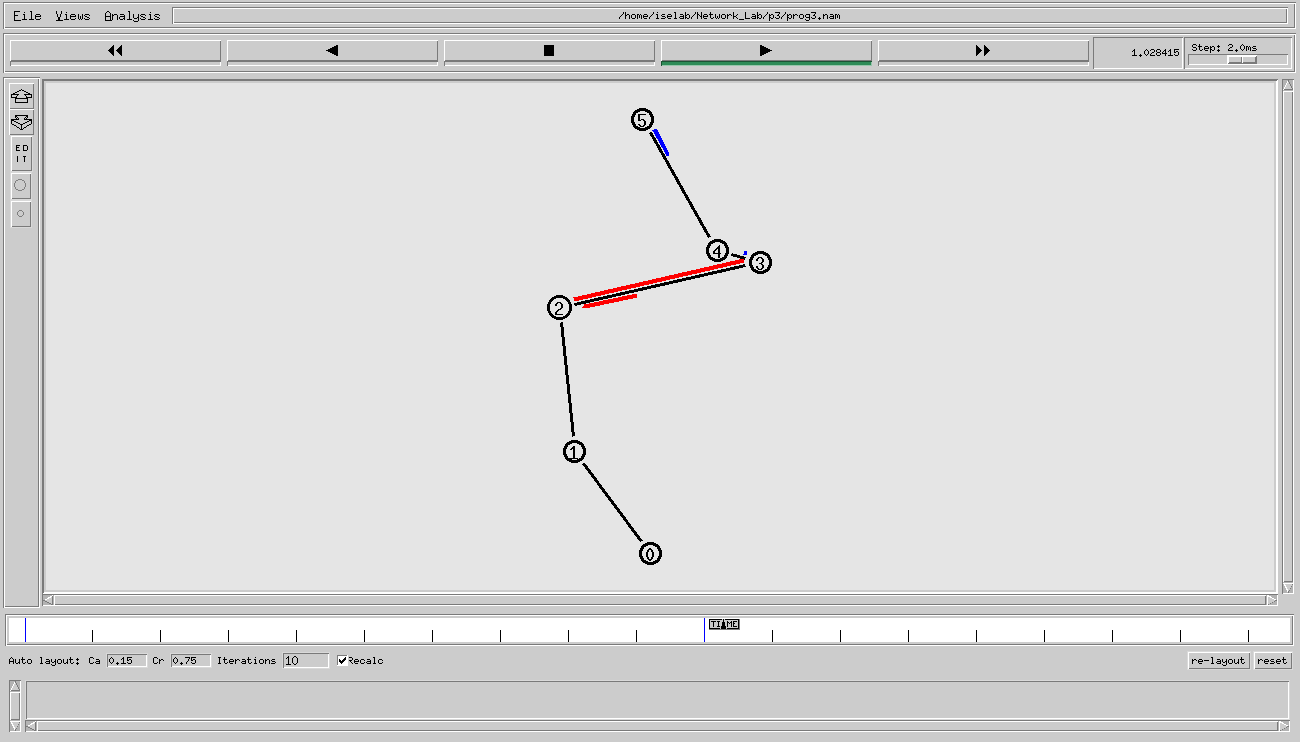
$ns at 1.6 "$p1 send"

$ns at 1.8 "Finish"

#Run the Simulation

$ns run

**Output**

****

3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot

congestion window for different source / destination.

set ns [new Simulator]

$ns color 1 Red

$ns color 2 Blue

set na [open Lab3.nam w]

$ns namtrace-all $na

set nt [open Lab3.tr w]

$ns trace-all $nt

set ng1 [open tcp1.xg w]

set ng2 [open tcp2.xg w]

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$ns make-lan "$n0 $n1 $n2" 1Mb 10ms LL Queue/DropTail Mac/802\_3

$ns make-lan "$n3 $n4 $n5" 2Mb 10ms LL Queue/DropTail Mac/802\_3

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

set tcp1 [new Agent/TCP]

set tcp2 [new Agent/TCP]

set cbr1 [new Application/Traffic/CBR]

set cbr2 [new Application/Traffic/CBR]

$ns attach-agent $n4 $tcp1

$cbr1 attach-agent $tcp1

$ns attach-agent $n1 $tcp2

$cbr2 attach-agent $tcp2

set sink1 [new Agent/TCPSink]

set sink2 [new Agent/TCPSink]

$ns attach-agent $n2 $sink1

$ns attach-agent $n5 $sink2

$ns connect $tcp1 $sink1

$ns connect $tcp2 $sink2

$tcp1 set class\_ 1

$tcp2 set class\_ 2

proc End {} {

global ns na nt

$ns flush-trace

close $na

close $nt

exec nam Lab3.nam &

exec xgraph tcp1.xg tcp2.xg &

exit 0

}

proc Draw {Agent File} {

global ns

set Cong [$Agent set cwnd\_]

set Time [$ns now]

puts $File "$Time $Cong"

$ns at [expr $Time+0.01] "Draw $Agent $File"

}

$ns at 0.0 "$cbr1 start"

$ns at 0.7 "$cbr2 start"

$ns at 0.0 "Draw $tcp1 $ng1"

$ns at 0.0 "Draw $tcp2 $ng2"

$ns at 10.0 "End"

$ns run

Output

4. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation

and determine the performance with respect to transmission of packets.

set ns [new Simulator]

set na [open Lab4.nam w]

$ns namtrace-all-wireless $na 500 500

set nt [open Lab4.tr w]

$ns trace-all $nt

set topo [new Topography]

$topo load\_flatgrid 500 500

$ns node-config -adhocRouting DSDV

$ns node-config -llType LL

$ns node-config -macType Mac/802\_11

$ns node-config -ifqType Queue/DropTail

$ns node-config -ifqLen 50

$ns node-config -phyType Phy/WirelessPhy

$ns node-config -channelType Channel/WirelessChannel

$ns node-config -propType Propagation/TwoRayGround

$ns node-config -antType Antenna/OmniAntenna

$ns node-config -topoInstance $topo

$ns node-config -agentTrace ON

$ns node-config -routerTrace ON

create-god 4

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

$n0 set X\_ 250.0

$n0 set Y\_ 250.0

$n0 set Z\_ 0.0

$n1 set X\_ 200.0

$n1 set Y\_ 250.0

$n1 set Z\_ 0.0

$n2 set X\_ 250.0

$n2 set Y\_ 250.0

$n2 set Z\_ 0.0

$n3 set X\_ 250.0

$n3 set Y\_ 250.0

$n3 set Z\_ 0.0

$ns at 0.0 "$n0 setdest 400.0 300.0 50.0"

$ns at 0.0 "$n1 setdest 50.0 100.0 20.0"

$ns at 0.0 "$n2 setdest 75.0 180.0 5.0"

$ns at 0.0 "$n3 setdest 100.0 100.0 25.0"

set tcp1 [new Agent/TCP]

$ns attach-agent $n0 $tcp1

set tcp2 [new Agent/TCP]

$ns attach-agent $n2 $tcp2

set sink1 [new Agent/TCPSink]

$ns attach-agent $n1 $sink1

set sink2 [new Agent/TCPSink]

$ns attach-agent $n3 $sink2

set cbr1 [new Application/Traffic/CBR]

$cbr1 attach-agent $tcp1

set cbr2 [new Application/Traffic/CBR]

$cbr2 attach-agent $tcp2

$ns connect $tcp1 $sink1

$ns connect $tcp2 $sink2

proc End {} {

global ns nt na

$ns flush-trace

close $na

close $nt

exec nam Lab4.nam &

}

$ns at 0.0 "$cbr1 start"

$ns at 0.0 "$cbr2 start"

$ns at 10.0 "End"

$ns run

AWK Code

|

BEGIN{Num\_of\_pkts=0;}

{

if ($1 == "r" && $3 == "\_1\_" && $4 == "AGT" && $7 == "tcp")

{

Num\_of\_pkts = Num\_of\_pkts + $8;

}

}

END{

Throughput = Num\_of\_pkts \* 8 / $2 /1000000;

printf("\n\n\tThroughput = %fbpms\n\n\n",Throughput);

}

**Output**

5. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or

equivalent environment.

set bwDL(gsm) 9600

set bwUL(gsm) 9600

set propDL(gsm) .500

set propUL(gsm) .500

set buf(gsm) 10

set ns [new Simulator]

set nt [open Lab5.tr w]

$ns trace-all $nt

set nodes(c1) [$ns node]

set nodes(ms) [$ns node]

set nodes(bs1) [$ns node]

set nodes(bs2) [$ns node]

set nodes(c2) [$ns node]

proc cell\_topo {} {

global ns nodes

$ns duplex-link $nodes(c1) $nodes(bs1) 3Mbps 10ms DropTail

$ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED

$ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED

$ns duplex-link $nodes(bs2) $nodes(c2) 3Mbps 50ms DropTail

}

switch gsm {

gsm -

gprs -

umts {cell\_topo}

}

$ns bandwidth $nodes(bs1) $nodes(ms) $bwDL(gsm) simplex

$ns bandwidth $nodes(ms) $nodes(bs1) $bwUL(gsm) simplex

$ns bandwidth $nodes(bs2) $nodes(ms) $bwDL(gsm) simplex

$ns bandwidth $nodes(ms) $nodes(bs2) $bwUL(gsm) simplex

$ns delay $nodes(bs1) $nodes(ms) $propDL(gsm) simplex

$ns delay $nodes(ms) $nodes(bs1) $propDL(gsm) simplex

$ns delay $nodes(bs2) $nodes(ms) $propDL(gsm) simplex

$ns delay $nodes(ms) $nodes(bs2) $propDL(gsm) simplex

$ns queue-limit $nodes(bs1) $nodes(ms) $buf(gsm)

$ns queue-limit $nodes(ms) $nodes(bs1) $buf(gsm)

$ns queue-limit $nodes(bs2) $nodes(ms) $buf(gsm)

$ns queue-limit $nodes(ms) $nodes(bs2) $buf(gsm)

$ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]

$ns insert-delayer $nodes(bs1) $nodes(ms) [new Delayer]

$ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]

$ns insert-delayer $nodes(bs2) $nodes(ms) [new Delayer]

set tcp [new Agent/TCP]

$ns attach-agent $nodes(c1) $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $nodes(c2) $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns connect $tcp $sink

proc End {} {

global ns nt

$ns flush-trace

close $nt

exec awk -f Lab5.awk Lab5.tr &

exec xgraph -P -bar -x TIME -y DATA gsm.xg &

exit 0

}

$ns at 0.0 "$ftp start"

$ns at 10.0 "End"

$ns run

AWK Code

|

BEGIN {Total\_no\_of\_pkts=0;}

{

if($1 == "r")

{

Total\_no\_of\_pkts = Total\_no\_of\_pkts + $6;

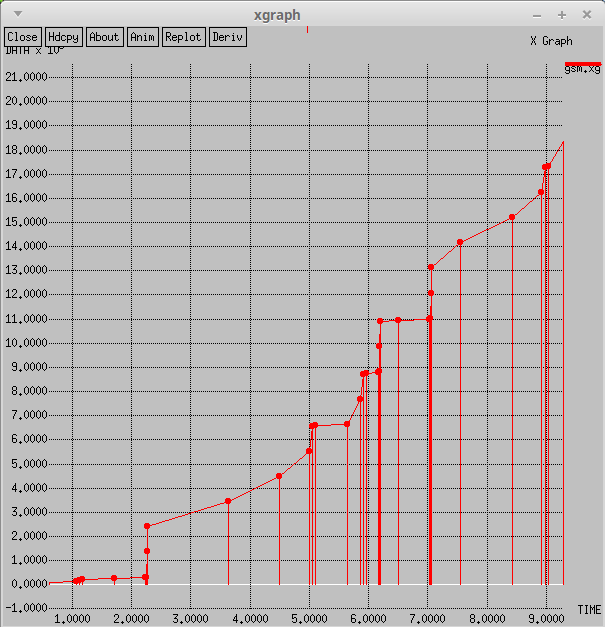
printf("%f %d\n",$2,Total\_no\_of\_pkts) >> "gsm.xg"

}

}

END{}

Output



6. Implement and study the performance of CDMA on NS2/NS3 (Using stack called

Call net) or equivalent environment.

set bwDL(cdma) 384000

set bwUL(cdma) 64000

set propDL(cdma) .150

set propUL(cdma) .150

set buf(cdma) 20

set ns [new Simulator]

set nt [open Lab6.tr w]

$ns trace-all $nt

set nodes(c1) [$ns node]

set nodes(ms) [$ns node]

set nodes(bs1) [$ns node]

set nodes(bs2) [$ns node]

set nodes(c2) [$ns node]

proc cell\_topo {} {

global ns nodes

$ns duplex-link $nodes(c1) $nodes(bs1) 3Mbps 10ms DropTail

$ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED

$ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED

$ns duplex-link $nodes(bs2) $nodes(c2) 3Mbps 50ms DropTail

}

switch umts {

umts {cell\_topo}

}

$ns bandwidth $nodes(bs1) $nodes(ms) $bwDL(cdma) simplex

$ns bandwidth $nodes(ms) $nodes(bs1) $bwUL(cdma) simplex

$ns bandwidth $nodes(bs2) $nodes(ms) $bwDL(cdma) simplex

$ns bandwidth $nodes(ms) $nodes(bs2) $bwUL(cdma) simplex

$ns delay $nodes(bs1) $nodes(ms) $propDL(cdma) simplex

$ns delay $nodes(ms) $nodes(bs1) $propDL(cdma) simplex

$ns delay $nodes(bs2) $nodes(ms) $propDL(cdma) simplex

$ns delay $nodes(ms) $nodes(bs2) $propDL(cdma) simplex

$ns queue-limit $nodes(bs1) $nodes(ms) $buf(cdma)

$ns queue-limit $nodes(ms) $nodes(bs1) $buf(cdma)

$ns queue-limit $nodes(bs2) $nodes(ms) $buf(cdma)

$ns queue-limit $nodes(ms) $nodes(bs2) $buf(cdma)

$ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]

$ns insert-delayer $nodes(bs1) $nodes(ms) [new Delayer]

$ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]

$ns insert-delayer $nodes(bs2) $nodes(ms) [new Delayer]

set tcp [new Agent/TCP]

$ns attach-agent $nodes(c1) $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $nodes(c2) $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns connect $tcp $sink

proc End {} {

global ns nt

$ns flush-trace

close $nt

exec awk -f Lab6.awk Lab6.tr &

exec xgraph -P -bar -x TIME -y DATA cdma.xg &

exit 0

}

$ns at 0.0 "$ftp start"

$ns at 10.0 "End"

$ns run

AWK Code

|

BEGIN {Total\_no\_of\_pkts=0;}

{

if($1 == "r")

{

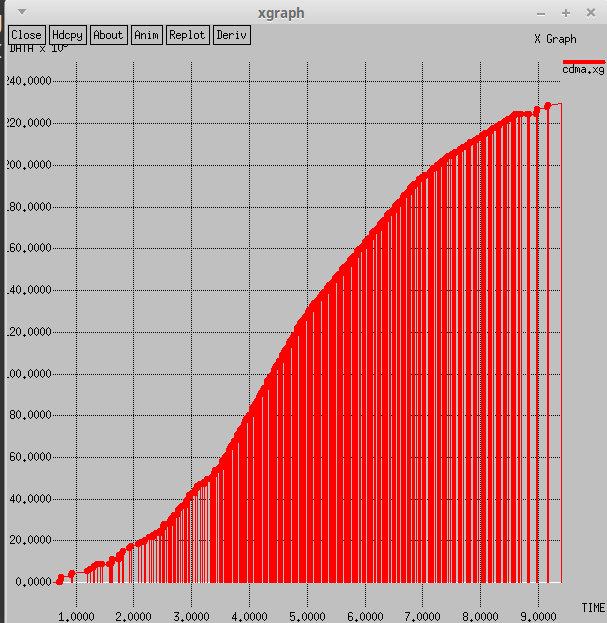
Total\_no\_of\_pkts = Total\_no\_of\_pkts + $6;

printf("%f %d\n",$2,Total\_no\_of\_pkts) >> "cdma.xg"

}

}

END{}



**PART B**

7. Write 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 the size of the data:");

n = scan.nextInt();

int data[] = new int[n];

System.out.println("Enter the data, bit by bit:");

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

data[i] = scan.nextInt();

}

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

n = scan.nextInt();

int divisor[] = new int[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);

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

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

}

System.out.println("\nThe 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 to be sent:");

for(int i=0 ; i < sent\_data.length ; i++) {

System.out.println("Enter bit number " + (sent\_data.length-i)

+ ":");

sent\_data[i] = scan.nextInt();

}

receive(sent\_data, divisor);

}

static int[] divide(int old\_data[], int divisor[]) {

int remainder[] , i;

int data[] = new int[old\_data.length + divisor.length];

System.arraycopy(old\_data, 0, data, 0, old\_data.length);

remainder = new int[divisor.length];

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

for(i=0 ; i < old\_data.length ; i++) {

System.out.println((i+1) + ".) First data bit is : "

+ remainder[0]);

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

if(remainder[0] == 1) {

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

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

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

}

}

else {

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

remainder[j-1] = exor(remainder[j], 0);

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

}

}

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

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

}

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 received data...");

return;

}

}

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

}

}

Output

8. Write a program to find the shortest path between vertices using bellman-ford

Algorithm

import java.util.Scanner;

public class BellmanFord

{

private int D[];

private int num\_ver;

public static final int MAX\_VALUE = 999;

public BellmanFord(int num\_ver)

{

this.num\_ver = num\_ver;

D = new int[num\_ver + 1];

}

public void BellmanFordEvaluation(int source, int A[][])

{

for (int node = 1; node <= num\_ver; node++)

{

D[node] = MAX\_VALUE;

}

D[source] = 0;

for (int node = 1; node <= num\_ver - 1; node++)

{

for (int sn = 1; sn <= num\_ver; sn++)

{

for (int dn = 1; dn <= num\_ver; dn++)

{

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

{

if (D[dn] > D[sn]+ A[sn][dn])

D[dn] = D[sn] + A[sn][dn];

}

}

}

}

for (int sn = 1; sn <= num\_ver; sn++)

{

for (int dn = 1; dn <= num\_ver; dn++)

{

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

{

if (D[dn] > D[sn]+ A[sn][dn])

System.out.println("The Graph contains negative egde cycle");

}

}

}

for (int vertex = 1; vertex <= num\_ver; vertex++)

{

System.out.println("distance of source " + source + " to "+ vertex + "is" + D[vertex]);

}

}

public static void main(String[ ] args)

{

int num\_ver = 0;

int source;

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number of vertices");

num\_ver = scanner.nextInt();

int A[][] = new int[num\_ver + 1][num\_ver + 1];

System.out.println("Enter the adjacency matrix");

for (int sn = 1; sn <= num\_ver; sn++)

{

for (int dn = 1; dn <= num\_ver; dn++)

{

A[sn][dn] = scanner.nextInt();

if (sn == dn)

{

A[sn][dn] = 0;

continue;

}

if (A[sn][dn] == 0)

{

A[sn][dn] = MAX\_VALUE;

}

}

}

System.out.println("Enter the source vertex");

source = scanner.nextInt();

BellmanFord b = new BellmanFord (num\_ver);

b.BellmanFordEvaluation(source, A);

scanner.close();

}

}

9. Using TCP/IP sockets, write a client – server program to make the client send the file

name and to make the server send back the contents of the requested file if present.

tcp program server side

import java.net.\*;

import java.io.\*;

public class server1

{

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

{ // establishing the connection with the server

ServerSocket sersock = new ServerSocket(4000);

System.out.println("Server ready for connection");

Socket sock = sersock.accept(); // binding with port: 4000

System.out.println("Connection is successful");

// reading the file name from client

InputStream istream = sock.getInputStream( );

BufferedReader fileRead =new BufferedReader(new InputStreamReader(istream));

String fname = fileRead.readLine( );

// reading file contents

BufferedReader contentRead = new BufferedReader(new FileReader(fname) );

// keeping output stream ready to send the contents

OutputStream ostream = sock.getOutputStream( );

PrintWriter pwrite = new PrintWriter(ostream, true);

String str;

while((str = contentRead.readLine()) != null) // reading line-by-line from file

{

pwrite.println(str); // sending each line to client

}

sock.close(); sersock.close(); // closing network sockets

pwrite.close(); fileRead.close(); contentRead.close();

}

}

tcp program client side

import java.net.\*;

import java.io.\*;

public class client1

{

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

{

Socket sock = new Socket( "127.0.0.1", 4000);

// reading the file name from keyboard. Uses input stream

System.out.print("Enter the file name");

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

String fname = keyRead.readLine();

// sending the file name to server. Uses PrintWriter

OutputStream ostream = sock.getOutputStream( );

PrintWriter pwrite = new PrintWriter(ostream, true);

pwrite.println(fname);

// receiving the contents from server. Uses input stream

InputStream istream = sock.getInputStream();

BufferedReader socketRead = new BufferedReader(new InputStreamReader(istream));

String str;

while((str = socketRead.readLine()) != null) // reading line-by-line

{

System.out.println(str);

}

pwrite.close(); socketRead.close(); keyRead.close();

}

}

10. Write a program on datagram socket for client/server to display the messages on

client side, typed at the server side.

import java.net.\*;

import java.io.\*;

class userver {

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

Socket s=new Socket("localhost",3333);

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

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

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

String str="",str2="";

while(!str.equals("stop")){

str=br.readLine();

dout.writeUTF(str);

dout.flush();

str2=din.readUTF();

System.out.println("client says: "+str2);

}

dout.close();

s.close();

}}

udp program client side

import java.net.\*;

import java.io.\*;

class uclient{

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

ServerSocket ss=new ServerSocket(3333);

Socket s=ss.accept();

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

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

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

String str="",str2="";

while(!str.equals("stop")){

str=din.readUTF();

System.out.println("server says: "+str);

str2=br.readLine();

dout.writeUTF(str2);

dout.flush();

}

din.close();

s.close();

ss.close();

}}

11. 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;

private BigInteger q;

private BigInteger N;

private BigInteger phi;

private BigInteger e;

private BigInteger 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);

phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));

e = BigInteger.probablePrime(bitlength / 2, r);

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

{

e.add(BigInteger.ONE);

}

d = e.modInverse(phi);

}

public rsa(BigInteger e, BigInteger d, BigInteger N)

{

this.e = e;

this.d = d;

this.N = N;

}

@SuppressWarnings("deprecation")

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);

System.out.println("String in Bytes: "

+ bytesToString(teststring.getBytes()));

// encrypt

byte[] encrypted = rsa.encrypt(teststring.getBytes());

// decrypt

byte[] decrypted = rsa.decrypt(encrypted);

System.out.println("Decrypting Bytes: " + bytesToString(decrypted));

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

}

private static String bytesToString(byte[] encrypted)

{

String test = "";

for (byte b : encrypted)

{

test += Byte.toString(b);

}

return test;

}

// Encrypt message

public byte[] encrypt(byte[] message)

{

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

}

// Decrypt message

public byte[] decrypt(byte[] message)

{

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

}

}

12. Write a program for congestion control using leaky bucket algorithm.

import java.io.\*;

import java.util.\*;

class Queue

{

int q[],f=0,r=0,size;

void insert(int n)

{

Scanner in = new Scanner(System.in);

q=new int[10];

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

{

System.out.print("\nEnter " + i + " element: ");

int ele=in.nextInt();

if(r+1>10)

{

System.out.println("\nQueue is full \nLost Packet: "+ele);

break;

}

else

{

r++;

q[i]=ele;

}

}

}

void delete()

{

Scanner in = new Scanner(System.in);

Thread t=new Thread();

if(r==0)

System.out.print("\nQueue empty ");

else

{

for(int i=f;i<r;i++)

{

try

{

t.sleep(1000);

}

catch(Exception e){}

System.out.print("\nLeaked Packet: "+q[i]);

f++;

}

}

System.out.println();

}

}

class Licky extends Thread

{

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

{

Queue q=new Queue();

Scanner src=new Scanner(System.in);

System.out.println("\nEnter the packets to be sent:");

int size=src.nextInt();

q.insert(size);

q.delete();

}

}