**Qn. 5 Develop a program to implement a sliding window protocol in the data link layer.**

**Program :**

import java.util.Scanner;

import java.util.Random;

public class SlidingWindowProtocol {

static int windowSize, totalFrames, sendBase, nextSeqNum;

static boolean[] acknowledged;

// Simulate sending a frame

static void sendFrame(int frameNumber) {

System.out.println("Sending frame " + frameNumber);

}

// Simulate receiving acknowledgment with random loss

static int receiveAck() {

Random rand = new Random();

// 90% chance of receiving ACK successfully

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; // No acknowledgment

}

}

// Slide the window when an ACK is received

static void slideWindow(int ackFrame) {

while (sendBase<= ackFrame&&sendBase<totalFrames) {

acknowledged[sendBase] = true;

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

sendBase++;

}

}

// Run the sliding window protocol

static void runSlidingWindow() {

while (sendBase<totalFrames) {

// Send frames within the window

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

sendFrame(nextSeqNum);

nextSeqNum++;

}

// Simulate receiving an acknowledgment

int ackFrame = receiveAck();

if (ackFrame != -1) {

slideWindow(ackFrame);

} else {

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

}

// Simulate delay

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

System.out.println(e);

}

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Input the window size and total number of frames

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();

// Initialize variables

acknowledged = new boolean[totalFrames];

sendBase = 0;

nextSeqNum = 0;

// Run the sliding window protocol

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**

**Qn. 7 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.**

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);

}

}

}

/\*server\*/

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();

}

}

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

**client side, typed at the server side.**

/\*client 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();

}

}

/\*server side\*/

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);

}

}

}

**Qn.10. 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**