**CN LAB**

**CODE-1:**

**SERVER:**

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class SimpleServerGUI {

private static ServerSocket serverSocket;

private static Socket clientSocket;

private static BufferedReader in;

private static PrintWriter out;

public static void main(String[] args) {

JFrame frame = new JFrame("Server");

JTextArea textArea = new JTextArea(10, 30);

JScrollPane scrollPane = new JScrollPane(textArea);

JTextField textField = new JTextField(20);

JButton sendButton = new JButton("Send");

frame.setLayout(new java.awt.BorderLayout());

frame.add(scrollPane, java.awt.BorderLayout.CENTER);

JPanel panel = new JPanel();

panel.add(textField);

panel.add(sendButton);

frame.add(panel, java.awt.BorderLayout.SOUTH);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setVisible(true);

try {

serverSocket = new ServerSocket(12345);

textArea.append("Server started. Waiting for a client...\n");

clientSocket = serverSocket.accept();

textArea.append("Client connected.\n");

in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

out = new PrintWriter(clientSocket.getOutputStream(), true);

sendButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String message = textField.getText();

out.println("Server: " + message);

textArea.append("Server: " + message + "\n");

textField.setText("");

}

});

String message;

while ((message = in.readLine()) != null) {

textArea.append(message + "\n");

}

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

if (clientSocket != null) clientSocket.close();

if (serverSocket != null) serverSocket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

**CLIENT:**

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class SimpleClientGUI {

private static Socket socket;

private static BufferedReader in;

private static PrintWriter out;

public static void main(String[] args) {

JFrame frame = new JFrame("Client");

JTextArea textArea = new JTextArea(10, 30);

JScrollPane scrollPane = new JScrollPane(textArea);

JTextField textField = new JTextField(20);

JButton sendButton = new JButton("Send");

frame.setLayout(new java.awt.BorderLayout());

frame.add(scrollPane, java.awt.BorderLayout.CENTER);

JPanel panel = new JPanel();

panel.add(textField);

panel.add(sendButton);

frame.add(panel, java.awt.BorderLayout.SOUTH);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setVisible(true);

try {

socket = new Socket("localhost", 12345);

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

out = new PrintWriter(socket.getOutputStream(), true);

sendButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String message = textField.getText();

out.println("Client: " + message);

textArea.append("Client: " + message + "\n");

textField.setText("");

}

});

String response;

while ((response = in.readLine()) != null) {

textArea.append(response + "\n");

}

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

if (socket != null) socket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

**CODE-2:**

**SERVER:**

import java.io.\*;

import java.net.\*;

public class TCPServer {

public static void main(String[] args) {

try (ServerSocket ss = new ServerSocket(4567)) {

System.out.println("Server is running and waiting for clients...");

while (true) {

Socket s = ss.accept(); // Establishes connection and waits for the client

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

// Handling client interaction

new ClientHandler(s).start();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

class ClientHandler extends Thread {

private Socket socket;

ClientHandler(Socket socket) {

this.socket = socket;

}

public void run() {

try {

DataInputStream dis = new DataInputStream(socket.getInputStream());

DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

String command = dis.readUTF();

if (command.equalsIgnoreCase("upload")) {

String filename = dis.readUTF();

int len = dis.readInt();

byte[] data = new byte[len];

dis.readFully(data);

// Save encrypted data

try (FileOutputStream fos = new FileOutputStream(filename)) {

fos.write(data);

System.out.println("File uploaded: " + filename);

}

} else if (command.equalsIgnoreCase("download")) {

String filename = dis.readUTF();

File file = new File(filename);

if (file.exists()) {

byte[] data = new byte[(int) file.length()];

try (FileInputStream fis = new FileInputStream(file)) {

fis.read(data);

}

dos.writeUTF("OK");

dos.writeInt(data.length);

dos.write(data);

} else {

dos.writeUTF("ERROR: File not found");

}

}

socket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

**CLIENT:**

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.io.\*;

import java.net.\*;

public class TCPClient {

private JFrame frame;

private JTextField fileField;

private JTextArea textArea;

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new TCPClient().createAndShowGUI());

}

private void createAndShowGUI() {

frame = new JFrame("Client");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(400, 300);

JPanel panel = new JPanel(new GridLayout(3, 1));

frame.add(panel, BorderLayout.CENTER);

fileField = new JTextField();

panel.add(new JLabel("Enter file name:"));

panel.add(fileField);

textArea = new JTextArea();

textArea.setEditable(false);

panel.add(new JScrollPane(textArea));

JPanel buttonPanel = new JPanel();

JButton uploadButton = new JButton("Upload");

JButton downloadButton = new JButton("Download");

buttonPanel.add(uploadButton);

buttonPanel.add(downloadButton);

frame.add(buttonPanel, BorderLayout.SOUTH);

uploadButton.addActionListener(e -> uploadFile());

downloadButton.addActionListener(e -> downloadFile());

frame.setVisible(true);

}

private void uploadFile() {

String filename = fileField.getText();

if (filename.isEmpty()) {

textArea.append("Please enter a file name.\n");

return;

}

try (Socket s = new Socket("localhost", 4567);

DataOutputStream dos = new DataOutputStream(s.getOutputStream())) {

dos.writeUTF("upload");

dos.writeUTF(filename);

File file = new File(filename);

if (!file.exists()) {

textArea.append("File not found: " + filename + "\n");

return;

}

byte[] data = new byte[(int) file.length()];

try (FileInputStream fis = new FileInputStream(file)) {

fis.read(data);

}

// Encrypt data using Caesar cipher

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

data[i] = (byte) ((data[i] + 3) % 256);

}

dos.writeInt(data.length);

dos.write(data);

textArea.append("File uploaded: " + filename + "\n");

} catch (IOException e) {

e.printStackTrace();

}

}

private void downloadFile() {

String filename = fileField.getText();

if (filename.isEmpty()) {

textArea.append("Please enter a file name.\n");

return;

}

try (Socket s = new Socket("localhost", 4567);

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

DataInputStream dis = new DataInputStream(s.getInputStream())) {

dos.writeUTF("download");

dos.writeUTF(filename);

String response = dis.readUTF();

if (response.equalsIgnoreCase("OK")) {

int len = dis.readInt();

byte[] data = new byte[len];

dis.readFully(data);

// Decrypt data using Caesar cipher

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

data[i] = (byte) ((data[i] - 3 + 256) % 256);

}

try (FileOutputStream fos = new FileOutputStream(filename)) {

fos.write(data);

}

textArea.append("File downloaded: " + filename + "\n");

} else {

textArea.append(response + "\n");

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

**CODE-3:**

**INTF:**

//function prototype

import java.rmi.\*;

public interface MyServerIntf extends Remote //remote interface

{ int i=0;

double add(double a, double b) throws RemoteException;

}

**IMPL:**

import java.rmi.\*;

import java.rmi.server.\*;

// UnicastRemoteObject supports for point-to-point active object references (invocations, parameters, and // results) using TCP streams.

public class MyServerImpl extends UnicastRemoteObject implements MyServerIntf

{

MyServerImpl() throws RemoteException

{}

public double add(double a, double b) throws RemoteException

{

return(a+b);

}

}

**SERVER:**

import java.net.\*;

import java.rmi.\*;

public class MyServer

{

public static void main(String[] arg)

{

try

{

MyServerImpl asi = new MyServerImpl();

Naming.rebind("RMServer",asi);

System.out.println("\nServer Started...");

}

catch(Exception e)

{

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

}

}

}

**CLIENT:**

import java.net.\*;

import java.rmi.\*;

public class MyClient

{

public static void main(String[] arg)

{

try

{

String sName = "rmi://"+arg[0]+"/RMServer";

MyServerIntf asif = (MyServerIntf)Naming.lookup(sName); // requesting remote objects on // the server

double d1=2000,d2=500;

System.out.println("Addition: "+asif.add(d1,d2));

}

catch(Exception e)

{

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

}

}

}

**CODE-4:**

**NS2(WIRED):**

set ns [new Simulator]

$ns color 0 blue

$ns color 1 black

set tr [open droptail-queue-out.tr w]

$ns trace-all $tr

set ftr [open droptail-queue-out.nam w]

$ns namtrace-all $ftr

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

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

$ns duplex-link $n1 $n2 10Mb 5ms DropTail

$ns duplex-link $n2 $n3 4Mb 3ms DropTail

$ns duplex-link $n3 $n4 100Mb 2ms DropTail

$ns duplex-link $n3 $n5 15Mb 4ms DropTail

set udp [new Agent/UDP]

$udp set fid\_ 1

set null [new Agent/Null]

$ns attach-agent $n0 $udp

$ns attach-agent $n4 $null

$ns connect $udp $null

set tcp [new Agent/TCP]

$tcp set fid\_ 0

set sink [new Agent/TCPSink]

$ns attach-agent $n1 $tcp

$ns attach-agent $n5 $sink

$ns connect $tcp $sink

$ns connect $tcp $sink

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set interval 0.020

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ftp set interval 0.020

proc finish {} {

global ns tr ftr

$ns flush-trace

close $tr

close $ftr

exec nam droptail-queue-out.nam &

exec gawk -f analysis.awk droptail-queue-out.tr &

exit

}

$ns at 0.1 "$cbr start"

$ns at 2.0 "$cbr stop"

$ns at 0.1 "$ftp start"

$ns at 2.0 "$ftp stop"

$ns at 2.1 "finish"

$ns run

**THROUGHPUT FILE:**

BEGIN {

rec= 0

drp=0

tot=0

rat=0.0

sum=0

sum1=0

throughput=0.0

throughput1=0.0

}

{

if($1== "r" && $4== 4)

{

rec++

}

if($1== "d" && $4 ==4 )

{

drp++

}

if($2<1.00 && $4==4)

{

sum=sum+$6

}

if($2<1.00 && $4==5)

{

sum1=sum1+$6

}

}

END {

tot = rec + drp

rat = (rec/tot) \*100

throughput= (sum\*8)/1000000

throughput1=(sum1\*8)/1000000

printf(" \n Packets received %d ", rec)

printf(" \n Packets dropped %d ", drp)

printf("\n Packets delivery ratio %f",rat)

printf("\n Throughput for udp is %f",throughput)

printf("\n Throughput for tcp is %f",throughput1)

}

EXECUTION:

~$ cd Desktop/

ns wired-net.tcl

nam wired.nam

awk -f sample.awk wired.tr