Computer Networks (CSD304) – Socket Programming Assignment

Due Date: November 25, 2020 (10 pm)

Guidelines

- ✓ This assignment aims to make the students familiar with socket programming in computer networks.
- ✓ **This assignment is to be completed individually.**
- ✓ **Programming Language to be used: Java**
- ✓ Use either UDP or TCP sockets for this assignment.
- ✓ Code should be easy to understand (make proper use of comments, don't overuse them).
- ✓ Assignment submitted after due date and time will not be evaluated and a score of zero will be awarded for this assignment.
- ✓ Materials copied from the Internet or otherwise will attract penalty.

Grading: This term paper has a **weightage of 10%** in your overall 100 points.

Submission

Each student must upload the following files on Blackboard:

a) Client.java file - The java file must contain your name and roll no (as comments).
b) Server.java file - The java file must contain your name and roll no (as comments).
c) Paste your code and screenshots of input and output screens (paste them in this file) - Name the document as Socket_CN2020_FirstName_LastName.pdf. [**You are required to strictly follow the naming convention**.]

Question

Write a program that involves a client and a server. The client sends server 4 values, for example *X, n, B, C* where, X is the adjacency matrix of a directed graph with 5 nodes A B C D E, and n is the length of the path from node B to node C.

The server responds back with two responses:

(a) positive Y response (or negative N response) if there exists (or doesn't exist) a path of length n from B to C.
(b) the image of the directed graph with nodes A B C D E proving the validity of the response.

For simplicity, assume a 5-node graph with nodes named A, B, C, D, E.

For example: Let's take a 3-node directed graph:

Case 1: Client sends the following to the server:

Input:

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

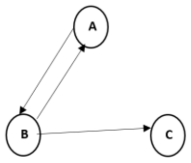
, 2, A, C

where, there is an adjacency matrix, 2 is the length of the path from node A to node C – that server has to check whether it exists or not.

Server should return the following:

Output 1: Yes, there exists a path of length 2 from node A to node C.

Output 2: Graph:



Case 2: Client sends the following to the server:

Input:

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

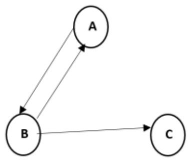
, 2, C, A

where, there is an adjacency matrix, 2 is the length of the path from node C to node A.

Server should return the following:

Output 1: No, there is no path of length 2 from node C to node A.

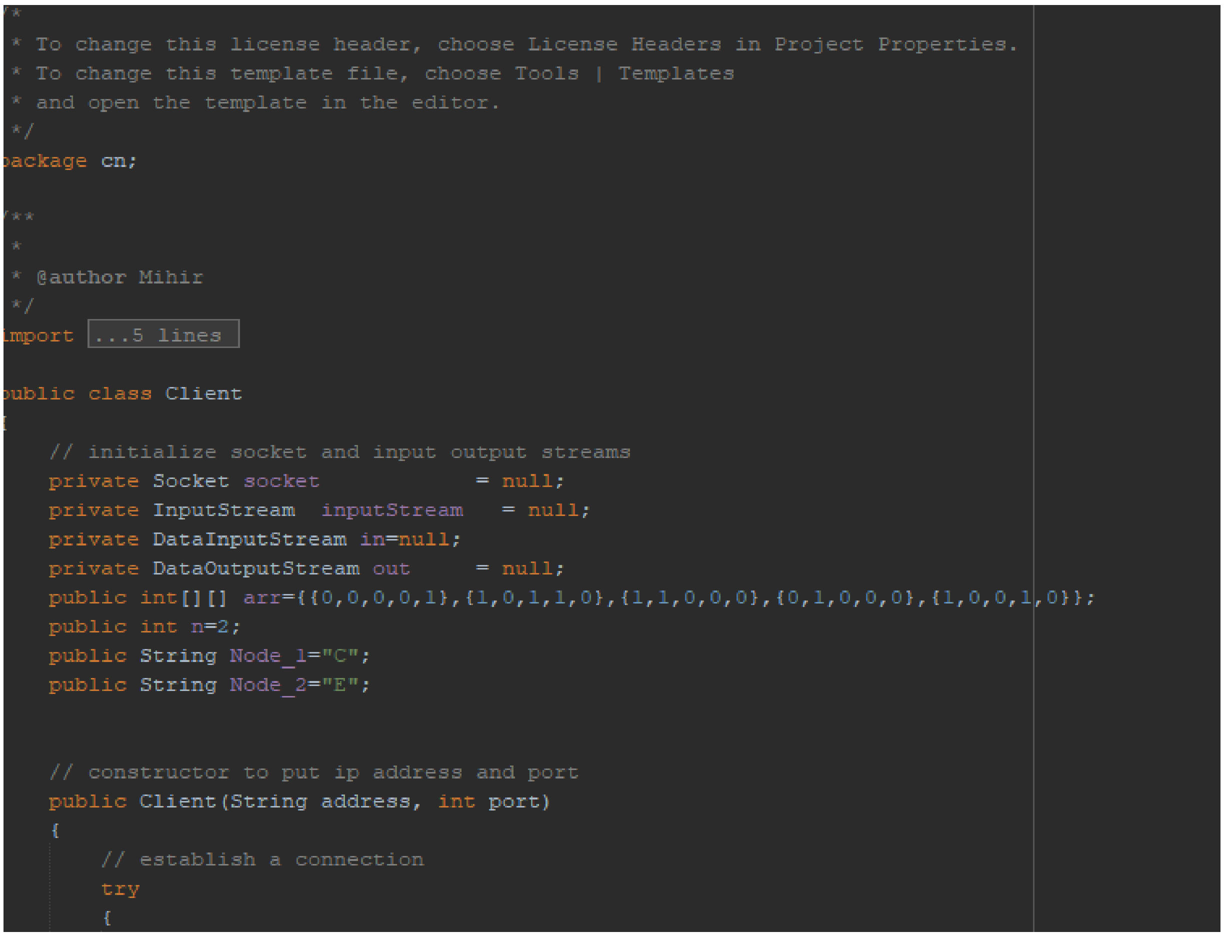
Output 2: Graph:



Submission Template

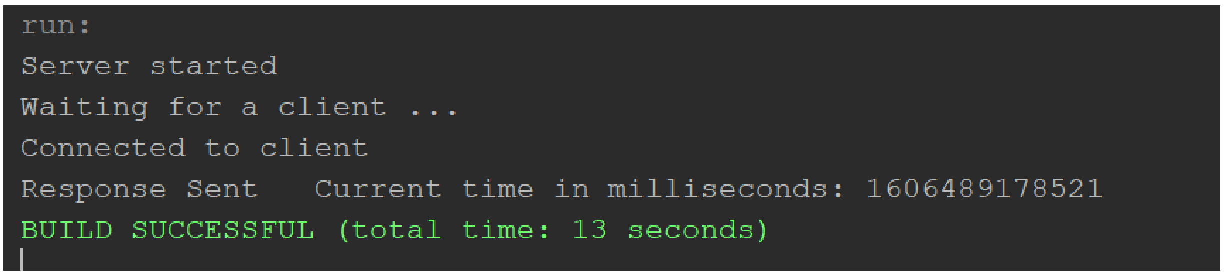
\\Screenshots of Input and Output Screens

Computer Networks (CSD304) – Socket Programming Assignment



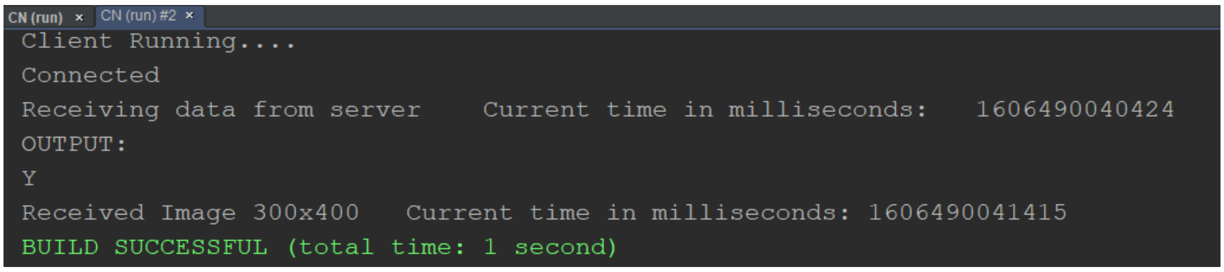
```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package cn;

/**
 *
 * @author Mihir
 */
import ...5 lines

public class Client
{
    // initialize socket and input output streams
    private Socket socket            = null;
    private InputStream  inputStream   = null;
    private DataInputStream in=null;
    private DataOutputStream out     = null;
    public int[][] arr={{0,0,0,0,1},{1,0,1,1,0},{1,1,0,0,0},{0,1,0,0,0},{1,0,0,1,0}};
    public int n=2;
    public String Node_1="C";
    public String Node_2="E";


    // constructor to put ip address and port
    public Client(String address, int port)
    {
        // establish a connection
        try
        {
```

Input hardcoded in client.java



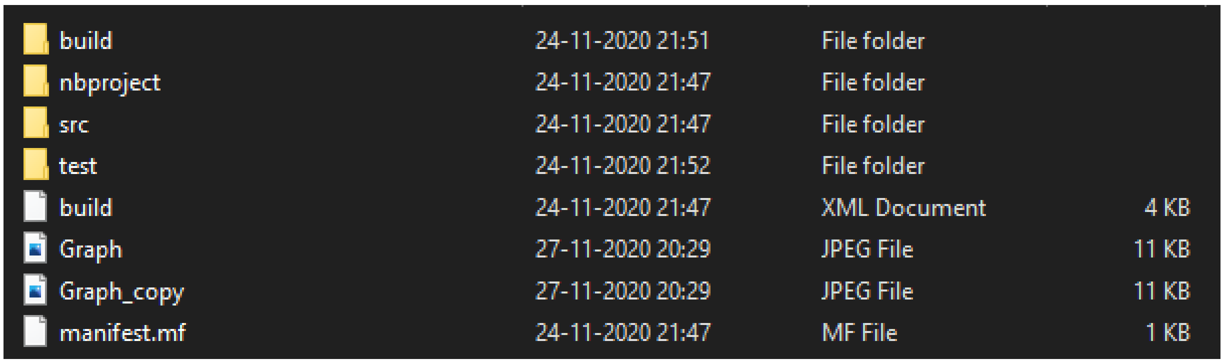
```
run:
Server started
Waiting for a client ...
Connected to client
Response Sent   Current time in milliseconds: 1606489178521
BUILD SUCCESSFUL (total time: 13 seconds)
```

Server output



```
Client Running....
Connected
Receiving data from server    Current time in milliseconds:   1606490040424
OUTPUT:
Y
Received Image 300x400   Current time in milliseconds: 1606490041415
BUILD SUCCESSFUL (total time: 1 second)
```

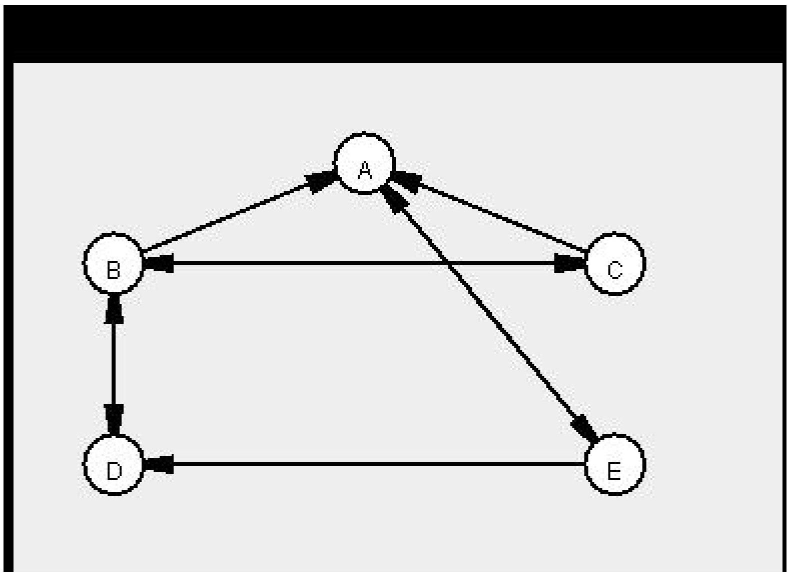
Client outpot



build	24-11-2020 21:51	File folder	
nbproject	24-11-2020 21:47	File folder	
src	24-11-2020 21:47	File folder	
test	24-11-2020 21:52	File folder	
build	24-11-2020 21:47	XML Document	4 KB
Graph	27-11-2020 20:29	JPEG File	11 KB
Graph_copy	27-11-2020 20:29	JPEG File	11 KB
manifest.mf	24-11-2020 21:47	MF File	1 KB

Graph saved in project folder

Graph_copy is the image received by client socket. Also in the project folder



Graph based on adjacency matrix

Computer Networks (CSD304) – Socket Programming Assignment

\\Server side code – put the code here

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package cn;

/**
 *
 * @author Mihir
 */

import java.net.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.geom.AffineTransform;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.nio.ByteBuffer;
import javax.imageio.ImageIO;
import javax.swing.*;

class Draw_Graph1 extends JPanel {
    int width;
    int height;
    Node[] nodes=new Node[5];
    ArrayList<edge> edges;
    public Draw_Graph1() { //Constructor

            edges = new ArrayList<>();
            width = 30;
            height = 30;
    }
    class Node {
            int x, y;
```

```
        String name;

        public Node(String myName, int myX, int myY) {
           x = myX;
           y = myY;
           name = myName;
        }
}
class edge {
        int i,j;

        public edge(int start_node, int dest_node) {
           i = start_node;
           j = dest_node;
        }
}
public void addNodes() {
        //add a node at pixel (x,y)
        nodes[0]=(new Node("A",175,50));
   nodes[1]=(new Node("B",50,100));
   nodes[2]=(new Node("C",300,100));
   nodes[3]=(new Node("D",50,200));
   nodes[4]=(new Node("E",300,200));
        this.repaint();
}
public void addEdge(int i, int j) {
        //add an edge between nodes i and j
        edges.add(new edge(i,j));
        this.repaint();
}
private static final Polygon ARROW_HEAD = new Polygon();
static {
   ARROW_HEAD.addPoint(0, 0);
   ARROW_HEAD.addPoint(-5, -30);
   ARROW_HEAD.addPoint(5, -30);
}
@Override
protected void paintComponent(Graphics g) { // draw the nodes and edges
        FontMetrics f = g.getFontMetrics();
        int nodeHeight = Math.max(height, f.getHeight());

        g.setColor(Color.black);
        for (edge e : edges) {
            Graphics2D g2 = (Graphics2D) g;
            g2.setStroke(new BasicStroke(2));
            double angle = Math.atan2(nodes[e.j].y - nodes[e.i].y, nodes[e.j].x - nodes[e.i].x);
            g2.drawLine(nodes[e.i].x, nodes[e.i].y, (int) (nodes[e.j].x - 10 * Math.cos(angle)), (int)
(nodes[e.j].y - 10 * Math.sin(angle)));
            AffineTransform tx1 = g2.getTransform();
            AffineTransform tx2 = (AffineTransform) tx1.clone();
            tx2.translate(nodes[e.j].x, nodes[e.j].y);
            tx2.rotate(angle - Math.PI / 2);
            g2.setTransform(tx2);
            g2.fill(ARROW_HEAD);
            g2.setTransform(tx1);


        }

        for (Node n : nodes) {
            int nodeWidth = Math.max(width, f.stringWidth(n.name)+width/2);
            g.setColor(Color.white);
            g.fillOval(n.x-nodeWidth/2, n.y-nodeHeight/2,
                       nodeWidth, nodeHeight);
            g.setColor(Color.black);
            g.drawOval(n.x-nodeWidth/2, n.y-nodeHeight/2,
                       nodeWidth, nodeHeight);

            g.drawString(n.name, n.x-f.stringWidth(n.name)/2,
                         n.y+f.getHeight()/2);
        }
    }

    }


public class Server
{
    //initialize socket and input stream
    private Socket          socket   = null;
    private ServerSocket    server   = null;
    private DataInputStream in       =  null;
    private DataOutputStream outputStream=null;
    static int[][] arr=new int[5][5];
    static int[][][] r1=new int[5][5][5];
    boolean exist;
    int n,nod1,nod2;

    public int str2int(String s){
        if (s.charAt(0)=='A'){
        return 0;}
        else if (s.charAt(0)=='B'){
        return 1;}
        else if (s.charAt(0)=='C'){
        return 2;}
        else if (s.charAt(0)=='D'){
        return 3;}
        else
            return 4;
    }

    public boolean pathfind(int n, int node1, int node2){
        for (int h=0;h<n;h++){
            for (int i=0;i<5;i++){
                for (int j=0;j<5;j++){
                    r1[h][i][j]=0;

                    for (int k=0;k<5;k++){
                        if(h==0){
                            r1[h][i][j]+=arr[i][k]*arr[k][j];

                        }
                        else{
                        r1[h][i][j]+=arr[i][k]*r1[h-1][k][j]; }
                    }
                }
            }
        }

        if(r1[n-1][node1][node2]!=0){
            return true;
        }
        else{
        return false;
        }

    }
```

```
    // constructor with port
    public Server(int port)
    {
        // starts server and waits for a connection
        try
        {
            server = new ServerSocket(port);
            System.out.println("Server started");

            System.out.println("Waiting for a client ...");

           // while(true){
                socket = server.accept();
                System.out.println("Connected to client");
                in= new DataInputStream(socket.getInputStream());
                outputStream=new DataOutputStream(socket.getOutputStream());
                for(int i=0;i<5;i++) {
                    for(int j=0;j<5;j++) {
                    arr[i][j] = in.readInt();
                    //System.out.print(String.valueOf(arr[i][j]));
                }
                    //System.out.println("");
                }
                n=in.readInt();
                nod1=str2int(in.readUTF());
                nod2=str2int(in.readUTF());
//                System.out.println(n);
//                System.out.println(nod1);
//                System.out.println(nod2);
                exist =pathfind(n,nod1,nod2);
                if(exist){
                    //System.out.println("YESS");
                outputStream.writeChar('Y');
                }
                else{
                    //System.out.println("NOOO");
                    outputStream.writeChar('N');
                }
                //BufferedImage image = ImageIO.read(("Graph.jpg"));
            JFrame f = new JFrame();
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(400, 300);
             Draw_Graph1 panel = new Draw_Graph1();
             panel.setSize(350,250);
             panel.setVisible(true);
             panel.addNodes();
             for(int i=0;i<5;i++){
            for(int j=0;j<5;j++){
                if (arr[i][j]>0){
                    panel.addEdge(i, j);
                }
            }
        }
        f.add(panel);

        f.setVisible(true);

        try
        {
            BufferedImage image = new BufferedImage(400,300, BufferedImage.TYPE_INT_RGB);
            Graphics2D graphics2D = image.createGraphics();
            f.paint(graphics2D);
            ImageIO.write(image,"jpeg", new File("Graph.jpeg"));


                    BufferedImage image1 = ImageIO.read(new File("Graph.jpeg"));

        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
        ImageIO.write(image1, "jpeg", byteArrayOutputStream);

        byte[] size = ByteBuffer.allocate(4).putInt(byteArrayOutputStream.size()).array();
        outputStream.write(size);
        outputStream.write(byteArrayOutputStream.toByteArray());
        outputStream.flush();
        System.out.println("Response Sent   Current time in milliseconds: " +
System.currentTimeMillis());

        Thread.sleep(5000);
        //System.out.println("Closing: " + System.currentTimeMillis());
```

```
        socket.close();
        }
        catch(Exception exception)
        {
          //code
          exception.printStackTrace();
        }

          //}



          // close connection

        }
        catch(IOException i)
        {
          System.out.println(i);
        }
    }

    public static void main(String args[])
    {
        Server server = new Server(5000);

    }
}
```

Computer Networks (CSD304) – Socket Programming Assignment

\\Client Side Code – put the code here

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package cn;

/**
 *
 * @author Mihir
 */
import java.awt.image.BufferedImage;
import java.net.*;
import java.io.*;
import java.nio.ByteBuffer;
import javax.imageio.ImageIO;

public class Client
{
    // initialize socket and input output streams
    private Socket socket            = null;
    private InputStream  inputStream   = null;
    private DataInputStream in=null;
    private DataOutputStream out     = null;
    public int[][] arr={{0,0,0,0,1},{1,0,1,1,0},{1,1,0,0,0},{0,1,0,0,0},{1,0,0,1,0}};
    public int n=2;
    public String Node_1="C";
    public String Node_2="E";


    // constructor to put ip address and port
    public Client(String address, int port)
    {
        // establish a connection
        try
        {
            socket = new Socket(address, port);
            System.out.println("Client Running....");
            System.out.println("Connected");



            inputStream  = socket.getInputStream();
            in=new DataInputStream(socket.getInputStream());



            out    = new DataOutputStream(socket.getOutputStream());
        }
        catch(UnknownHostException u)
        {
            System.out.println(u);
        }
        catch(IOException i)
        {
            System.out.println(i);
        }
        try{
            for(int i=0;i<5;i++){
                for(int j=0;j<5;j++){
                out.writeInt(arr[i][j]);//send adjacency matrix
                }
            }
            out.writeInt(n);//sending N
            out.writeUTF(Node_1);//send Node 1
            out.writeUTF(Node_2);//send node 2

            System.out.println("Receiving data from server    Current time in milliseconds:   " +
System.currentTimeMillis());
            char output=in.readChar();
            System.out.println("OUTPUT:");
            System.out.println(output);



        byte[] sizeAr = new byte[4];
        inputStream.read(sizeAr);
        int size = ByteBuffer.wrap(sizeAr).asIntBuffer().get();

        byte[] imageAr = new byte[size];
        inputStream.read(imageAr);

        BufferedImage image = ImageIO.read(new ByteArrayInputStream(imageAr));
        System.out.println("Received Image " + image.getHeight() + "x" + image.getWidth() + "   Current
time in milliseconds: " + System.currentTimeMillis());
        ImageIO.write(image, "jpeg", new File("Graph_copy.jpeg"));




            }
        catch(IOException i)
            {
                System.out.println(i);
            }
        try
        {
//          input.close();
        }
        catch(Exception i)
        {
            System.out.println(i);
        }
}




    public static void main(String args[])
    {
        Client client = new Client("127.0.0.1", 5000);
    }
}
```