CSE 4226 — Network Programming Lab

Assignment 1

Developing a multi-threaded server-client application using TCP Sockets

Submitted by

Samin Shahriar Tokey

14.02.04.066

Section: B1

Submission Date: 6th June 2018

1 Implementation summary

Features	Status
User Registration and Login:	Implemented
Online User Lists:	Implemented
Friend Request:	Implemented
Unicast:	Implemented
Multicast:	Implemented
Broadcast:	Implemented
Friend list:	Implemented
Create Chat Room:	Partially Implemented

2 Implementation Challenges

I ran into various errors while trying to implement this problem. Most of them were null pointer exceptions and server connection resets. However I managed to get past most of them by following forums on Geeks for Geeks[1] and Stackoverflow[2]

3 Limitation and Future Scope of Improvements

The whole system is command based, thus very limited. In future I'd like to develop a multiclient chat server with UI and more features.

4 Discussion

I learned quite a lot about networking with multithreaded client class.

References

[1] Geek For Geeks,
`https://www.geeksforgeeks.org/, 6-6-2018`

[2] Stack Overflow,

`https://stackoverflow.com/, 6-6-2018`

Appendices

A CODE

```java
/*
 * 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 chat.server;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Scanner;
import java.util.StringTokenizer;

/**
 *
 * @author
 */
// ClientHandler class
class ClientHandler implements Runnable {

    String rqst = null;
    Scanner sc = new Scanner(System.in);
    private String username;
    private String password;
    final DataInputStream dIP;
    final DataOutputStream dOP;
    Socket soc;
    boolean isloggedin;
    String friends[] = new String[10];
    Room rooms[] = new Room[10];

    // constructor
    public ClientHandler(Socket soc, String name,
            DataInputStream dIP, DataOutputStream dOP) {
        this.dIP = dIP;
        this.dOP = dOP;
        this.username = name;
        this.soc = soc;
        this.isloggedin = false;
    }
    boolean fuse = true;

    @Override
    public void run() {

        String received;
        while (true) {
            try {

                if (fuse) {
                    dOP.writeUTF("Sign Up Stranger! Your Name?");
                    username = dIP.readUTF();
                    dOP.writeUTF("Your Password?");
                    password = dIP.readUTF();
                    dOP.writeUTF("Greetings! To Send Messages, you need to sign in
:)");
                    fuse = false;
                }
                received = dIP.readUTF();
                System.out.println(received);
                if (received.equals("signout")) {
                    this.isloggedin = false;
                } else if (received.equals("makeRoom")) {
                    for (int i = 0; i < rooms.length; i++) {
```

```java
                rooms[i] = new Room();
            }
            for (int i = 0; i < rooms.length; i++) {
                if (rooms[i].roomName.equals("Blank")) {
                    dOP.writeUTF("Room name?");
                    String nm = dIP.readUTF();
                    rooms[i].roomName = nm;
                    rooms[i].participants.add(this.username);

                    dOP.writeUTF("Room Created!");
                    break;
                }
            }
        } else if (received.equals("joinRoom")) {
            dOP.writeUTF("Room name?");
            String nm = dIP.readUTF();
            boolean f = true;
            for (int i = 0; i < rooms.length; i++) {
                if (rooms[i].roomName.equals(nm)) {
                    boolean f2 = true;
                    f = false;
                    for (String prt : rooms[i].participants) {
                        if (prt.equals(this.username)) {
                            dOP.writeUTF("You're already in the room :)");
                            f2 = false;
                            break;
                        }

                    }
                    if (f2) {
                        rooms[i].participants.add(nm);
                        dOP.writeUTF("You have been added in the room");
                        break;

                    }
                }
            }
```

```
                if (f) {
                    dOP.writeUTF("ROOM NOT FOUND!");
                }
            } else if (received.equals("messageRoom")) {
                dOP.writeUTF("Room name?");
                String nm = dIP.readUTF();
                dOP.writeUTF("Your message?");
                String msg = dIP.readUTF();
                boolean f1 = true;
                for (int i = 0; i < rooms.length; i++) {
                    if (rooms[i].roomName.equals(nm)) {
                        boolean f2 = true;
                        f1 = false;
                        for (String prt : rooms[i].participants) {
                            if (prt.equals(this.username)) {
                                for (String pr : rooms[i].participants) {
                                    if (pr.equals(this.username) == false) {
                                        for (ClientHandler CCH : Server.serverList) {
                                            if (CCH.username.equals(pr) && CCH.isloggedin) {
                                                CCH.dOP.writeUTF("From Room :" + nm + ":  " + msg);
                                            }
                                        }
                                    }
                                }
                                f2 = false;
                                break;
                            }

                        }
                        if (f2) {
                            dOP.writeUTF("YOU ARE NOT IN THE ROOM");
                        }
                    }

                }

                if (f1) {
                    dOP.writeUTF("ROOM NOT FOUND!");
                }
            } else if (received.equals("accept")) {
                if (rqst != null) {
                    int i;
                    for (i = 0; i < friends.length; i++) {
                        if (friends[i] == null) {
                            friends[i] = rqst;
                            rqst = null;
                            this.dOP.writeUTF("Friend Request Accepted!");
                            break;
                        }
                    }

                }

            } else if (received.equals("frndlst")) {

                int i;
                for (i = 0; i < friends.length; i++) {
                    if (friends[i] != null) {

                        this.dOP.writeUTF(friends[i]);

                    }
                }
            } else if (received.equals("denied")) {
                for (ClientHandler CCH : Server.serverList) {
                    if (CCH.username.equals(rqst)) {
                        CCH.dOP.writeUTF("Friend Request Denined :( :( Better
luck next time.");
                        for (int i = 0; i <= CCH.friends.length; i++) {
                            if (CCH.friends[i] == rqst) {
                                CCH.friends[i] = null;
                                break;
                            }
                        }
                        rqst = null;
                        break;
                    }
                }
            } else if (received.equals("rqst")) {

                int i;

                for (i = 0; i <= friends.length; i++) {
                    if (friends[i] == null) {
                        friends[i] = dIP.readUTF();
                        break;
                    }
                }

                for (ClientHandler CCH : Server.serverList) {
                    if (CCH.username.equals(friends[i])) {
                        CCH.dOP.writeUTF("Friend Request from : " + this.username + "Accept?");
                        CCH.rqst = this.username;
                        break;
                    }
                }
                dOP.writeUTF("REQUEST SENT!");
            } else if (received.equals("signin")) {
                String uname = dIP.readUTF();
                String pass = dIP.readUTF();
                boolean f = false;
                if (this.username.equals(uname) == true && this.password.equals(pass) == true) {
                    this.isloggedin = true;
                    f = true;
                    dOP.writeUTF(this.username + " Welcome!! You have succesfully signed in :)");
```

```java

                    }
                    if (!f) {
                        dOP.writeUTF("LOGIN FAILED :@ !");
                    }

                } else if (received.equals("online")) {

                    this.dOP.writeUTF("List of online people are: " + "\n");
                    for (ClientHandler CCH : Server.serverList) {
                        if (CCH.username.equals(this.username) == false && CCH.isloggedin == true) {
                            this.dOP.writeUTF(CCH.username + "\n");
                        }
                    }
                } else {

                    StringTokenizer st = new StringTokenizer(received, ":");
                    if (st.countTokens() == 2) {
                        String first = st.nextToken();
                        String second = st.nextToken();

                        if (second.equals("all")) {
                            for (ClientHandler CCH : Server.serverList) {
                                if (CCH.username.equals(this.username) == false && CCH.isloggedin == true) {

                                    for (int i = 0; i <= CCH.friends.length; i++) {
                                        if (CCH.friends[i].equals(this.username)) {
                                            CCH.dOP.writeUTF(this.username + " : " + first);
                                        }
                                        break;
                                    }
                                }
                            }

                        } else {
                            for (ClientHandler CCH : Server.serverList) {
                                if (CCH.username.equals(second) && CCH.isloggedin
    == true) {
                                    for (int i = 0; i <= CCH.friends.length; i++) {
                                        if (CCH.friends[i].equals(this.username)) {
                                            CCH.dOP.writeUTF(this.username + " : "
    + first);
                                        }
                                        break;
                                    }
                                    break;
                                }
                            }
                        }
                    } else {
                        String first = st.nextToken();
                        while (st.hasMoreTokens()) {
                            String second = st.nextToken();
                            for (ClientHandler CCH : Server.serverList) {
                                if (CCH.username.equals(second) && CCH.isloggedin
    == true) {
                                    for (int i = 0; i <= CCH.friends.length; i++) {
                                        if (CCH.friends[i].equals(this.username)) {
                                            CCH.dOP.writeUTF(this.username + " : "
    + first);
                                        }
                                        break;
                                    }
                                    break;
                                }
                            }
                        }
                    }
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }
}
```

```
/*
 * 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 chat.server;

/**
 *
 * @author
 */
// Java implementation for multithreaded chat client
// Save file as Client.java

import java.io.*;
import java.net.*;
import java.util.Scanner;

public class Client
{


    public static void main(String args[]) throws UnknownHostException, IOException
    {
        Scanner scn = new Scanner(System.in);
        InetAddress ip = InetAddress.getByName("localhost");

        Socket s = new Socket(ip, 7777);
        DataInputStream dis = new DataInputStream(s.getInputStream());
        DataOutputStream dos = new DataOutputStream(s.getOutputStream());

        Thread sendMessage = new Thread(new Runnable()
        {
            @Override
            public void run() {
                while (true) {

                    String msg = scn.nextLine();

                    try {
                        dos.writeUTF(msg);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        });

        Thread readMessage = new Thread(new Runnable()
        {
            @Override
            public void run() {

                while (true) {
                    try {
                        String msg = dis.readUTF();
                        System.out.println(msg);
                    } catch (IOException e) {

                        e.printStackTrace();
                    }
                }
            }
        });

        sendMessage.start();
        readMessage.start();

```

```
    }
}
```

```
/*
 * 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 chat.server;

/**
 *
 * @author
 */
// Java implementation of  Server side
// It contains two classes : Server and ClientHandler
// Save file as Server.java

import java.io.*;
import java.util.*;
import java.net.*;

// Server class
public class Server
{

    static Vector<ClientHandler> serverList = new Vector<>();

    static int i = 0;

    public static void main(String[] args) throws IOException
    {
        ServerSocket ss = new ServerSocket(7777);

        Socket s;

        while (true)
```

```java
        {
            s = ss.accept();

            System.out.println("New client request received : " + s);

            DataInputStream dIP = new DataInputStream(s.getInputStream());
            DataOutputStream dOP = new DataOutputStream(s.getOutputStream());

            System.out.println("Creating a new handler for this client...");

            ClientHandler CCH = new ClientHandler(s,"client " + i, dIP, dOP);

            Thread t = new Thread(CCH);



            serverList.add(CCH);

            t.start();

            i++;

        }
    }
}
```

```java
/*
 * 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 chat.server;

import java.util.Vector;


public class Room implements Runnable{
    Vector<String> participants = new Vector<>();
    public String roomName=new String("Blank");

    @Override
    public void run() {
        throw new UnsupportedOperationException("Not supported yet."); //To change
    body of generated methods, choose Tools | Templates.
    }


}
```