جامعة بيرزيت
BIRZEIT UNIVERSITY

Chat Application – UDP -P2P

COMPUTER – NETWORKS - 2022

Report Name : Chat Application Based on UDP -P2P

Course Name : Computer Networks

Teacher: Dr. Mohammed Helal

Student :

Tareq Khanfar – 1200265 sec 2

Table of Contents

Introduction :

This report is an explanation of the program UDP-based Chatting Application based on a Peer-to-Peer architecture

The application supports public chat and private chat with any active person .

Technical explanation .

The program in general consists of server and client :

Server : any client is activate will be send userID to server then server will be save UserID , IPAddress and PortNumber for user . And each client can communicate with any other client by sending its Username to the server, then server send IPAddress and port number to sender .When sender is accept this information , start connection with client .

Every five seconds, a message is automatically sent from the client to the server to prove its presence

Protocols used:

Server-specific protocols for sending messages :

First Protocol : "1"+ IP_USER + PORT_USER

1 : to distinguish the message as it is send IP Address and port Number for X user , to sender.

IP_USER: Contains the address of the User requested by the sender

PORT_USER : Contains the Port of the User requested by the sender

Second Protocol : "9"+ str

9 : This means that this message must be sent to all users

Str : The content of the message is all the names of the active users

Server-specific protocols for Receiving messages :

If the first character for message (messageFromClient[0]) was an 'A' This meaning that message to add user in database (UserID , IPAdress , PortNumber)

If the first character for message (messageFromClient[0]) was an '@' This message means that the address and port of the requested person must be sent to the sender

If the first character for message (messageFromClient[0]) was an 'P' This means a general message and must be sent to all users registered in the database.

If the first character for message (messageFromClient[0]) was an 'k' This means that this message is received from one of the users to prove his presence and the content of the message is his name

Client-specific protocols for sending messages :

```
85
86    public static void send(String Myusername, String str, int x) throws IOException {
87        if (x == 0) {
88            buffer = ("P" + Myusername + str).getBytes();
89            DatagramPacket packet = new DatagramPacket(buffer, buffer.length, IP_Server, 1288);
90            datagramSocket.send(packet);
91        }
92
93        else if (x == 1) {
94            buffer = str.getBytes();
95            DatagramPacket packet = new DatagramPacket(buffer, buffer.length, IP_Server, 1288);
96            datagramSocket.send(packet);
97        } else if (x == 2) {
98            InetAddress ipClient = InetAddress.getByName(ipToClient);
99            buffer = ("2" + Myusername + ": " + str).getBytes();
100           DatagramPacket packet = new DatagramPacket(buffer, buffer.length, ipClient, portToClient);
101           datagramSocket.send(packet);
102       }
103
104   }
```

If X = 0 this mean message must be to send all users

And add **'P' + MyUserName + str**

P : to understand the server that message public message .

MyUserName : to knowing all client Who is the sender .

Str :the message

Else if x = 1 this message must be send to server only , this message is contain two protocol

```
1 - Client.send("", "A" + username.getText(), 1);
```

```
2 - Client.send("", "@" + SelectUser.getText(), 1)
```

First protocol if the first charectar (str[0]) is 'A' This message means that it must be sent to the server to add the user

First protocol if the first charectar (str[0]) is '@' This message means that it must be sent to the server to get IPAdress and port number for selected user .

Else if x = 2

After finished previous step , The protocols in this step :

1-"2" + MyUserName + str

2 : To make the user understand that this message is a message from a user and not a server

MyUserNam: So that the user knows who the sender is

Str : the message

2- "k"

K : So that the server knows that this message is to prove the activity (presence)

Client-specific protocols for Receiving messages :

if the first charectar (str[0]) is ‘1’ It means that this message contains the address and port of the user you requested and came from server

if the first charectar (str[0]) is ‘2’ This means that this message came from a user

if the first charectar (str[0]) is ‘9’ This message came from the server and contains all active user names

Exaplain My Code :

Side Server : Main.class

```
32    public void start(Stage primaryStage) {
33        Thread reciveUsers = new Thread () { // this thread to receive any request from clients
34             public void run () {
35                 try {
36                     Server.recive() ;
37                 } catch (IOException e) {
38                     // TODO Auto-generated catch block
39                     e.printStackTrace();
40                 }
41             }
42            } ;
43            Thread OnlineClient = new Thread () { // this thread to  Send active usernames to all users
44                 public void run () {
45
46                     try {
47                         while (true) {
48
49                             Server.OnlineClient();
50                             Server.sendAllUsers(); // send  userNames to all client
51
52                             Thread.sleep(600);
53                         }
54                     } catch (IOException | InterruptedException e) {
55                         // TODO Auto-generated catch block
56                         e.printStackTrace();
57                     }
58                 }
59                } ;
60                // start threads
61           OnlineClient.start();
62            reciveUsers.start();
```

In this image, it was created two threads :
reciveUsers : this threads His job is to implement the function inside it, which is to receive any message coming from the client .

OnlineClient : this thread to send updated list to all users

Note : in line 52 the delay to minimized any loss in data

2-Side Server : Server.class

```
14
15 public class Server {
16    public static ArrayList<User>users = new ArrayList<>() ; // will be contain all users
17    public  static Queue<String> q   = new LinkedList<>(); //Contains the names of active users
18
19     private static final int sizeBuffer = 5000;
20     static byte[] buffer = new byte[ sizeBuffer];
21     static DatagramSocket datagramSocket ; // to send and receive messages for clients
22    public Server(DatagramSocket datagramSocket) {
23     this.datagramSocket = datagramSocket ;
24    }
25      public static void recive() throws IOException {
26             while (true) {
27                 // Prepare a package and assign it buffer and and its size
28             DatagramPacket datapacket = new DatagramPacket(buffer, buffer.length);
29             // Receiving the incoming packet from the client
30             datagramSocket.receive(datapacket);
31             InetAddress ipUser = datapacket.getAddress(); // get IP of sender
32             int portUser = datapacket.getPort(); // get Port number of sender
33             // show message on screen
34             Main.str += "Message from client : IP : " + ipUser  +"     PORT : " + portUser+"\n";
35             String temp = ipUser.getHostAddress()+"#"+portUser;
36             System.out.println("Inside :"+temp);
37
38             // convert the incoming data to String
39             String messageFromClient = new String(datapacket.getData(), datapacket.getOffset(), datapacket.getLength());
40             //Check the first letter to see the type of incoming message
41             char x = messageFromClient.charAt(0) ;
42
43             switch (x) {
44             case 'k' :
```

```
43              switch (x) {
44              case 'k' :
45
46                  temp = ipUser.getHostAddress()+"#"+portUser;
47                   // if the IP and Port of Client already exist in queue , not added
48                  if (!q.contains(temp)) {
49                  q.add(temp);
50                  }
51                  break ;
52                  case 'A':
53                      // show message on screen
54                  Main.str+="Message : " + messageFromClient.substring(1)+"\n";
55               // add new user to list
56                  users.add(new User (messageFromClient.substring(1) ,ipUser , portUser ));
57              Main.str += "Added User is done" + "#of" + users.size() +"\n";
58                  break ;
59                        // Here the requested user information is sent by the sender
60              case '@' : send(search(messageFromClient.substring(1)) , ipUser , portUser) ;
61                  break ;
62              case 'P' :
63                  // send public message to all users in the list
64                  brodCast(messageFromClient.substring(1));
65                  Main.str += "Send Message for all users " +"\n";
66
67
68
69                  break ;
70
71              }
72
73                Platform.runLater(new Runnable() { // to display any events (Strings) during run time
74                  @Override
80      }
81      public static void send (User u , InetAddress ip , int port ) throws IOException {
82         // if the u equal null ,  ip equal null or port equal 0 , refused it
83          if (u==null || ip == null || port ==0) {
84              System.out.println("u is null");
85          }else {
86          /* Put the IP address and port of the person requested
87              in the message and then send it to the sender */
88              String n = ("1"+(u.getIP_User().getHostAddress())+"#"+u.getPORT()); // put
89              System.out.println(n);
90             buffer = n.getBytes();
91             DatagramPacket packet = new DatagramPacket (buffer , buffer.length , ip , port) ;
92             datagramSocket.send(packet);
93
94             Main.str += "user :  "+ip.getHostAddress() +" he want to connect with user : " +u.getIP_User().getHostAddress()  +"\n";
95
96
97          }
98      }
99
100     private static void brodCast ( String message ) throws IOException {
101         // A general message is sent to all users and is received from one of the users
102         for (int i = 0 ; i < users.size() ; i++) {
103            buffer =(("2")+message).getBytes();
104            DatagramPacket packet = new DatagramPacket (buffer , buffer.length , users.get(i).getIP_User() , users.get(i).getPORT()) ;
105            datagramSocket.send(packet);
106        }
107     }
108
```

```
    private static User search (String id) {
    /* this Function accept user name then search it in list ,
     if exist then return object contain user info. */
        for (int i = 0 ; i < users.size() ; i++) {
            if (users.get(i).getId_User().equals(id)) {
                return users.get(i) ;
            }
        }
        return null ;
    }

    public static void OnlineClient() throws IOException, InterruptedException {
        /* Here the IP's and port's in the list are compared with the IP's and port's in the queue.
         * If the first IP and port in the list is in the queue,
         * it is deleted from the queue only and skipped in the list.
         *  But if the IP and port is not in the queue, this means that the user has become inactive
         * and therefore must be deleted from the list and the queue as well
         * */
        for (int i = 0 ; i < users.size() ; i++) {
            String s = users.get(i).getIP_User().getHostAddress()+"#" +users.get(i).getPORT() ;
            if (q.contains(s)) {
                q.remove(s);
            }
            else {
                q.remove(s);

                users.remove(i);
            }
        }


        ,
         else {
            q.remove(users.get(i).getId_User());

            users.remove(i);

         }
      }

        sendAllUsers(); // send  userNames to all client

    }
public static void sendAllUsers () throws IOException, InterruptedException {
    // Send active usernames to all users
    String str = "" ;
    for (int i = 0 ; i < users.size() ; i++) {
        str += users.get(i).getId_User()+"#" ;
    }
    System.out.println(str);
    byte [] buffer_ = new byte[str.length()*8 +20];
    for (int i = 0 ; i < users.size() ; i++) {
        buffer_ =("9"+str).getBytes();
        DatagramPacket packet = new DatagramPacket (buffer_ , buffer_.length , users.get(i).getIP_User() , users.get(i).getPORT()) ;
        datagramSocket.send(packet);
        Thread.sleep(500); // put delay half second to avoid any loss , aim : minimized loss in data
    }
}


}
```

Explaining all function In the above pictures “Side Server – Server.class”

1 – Receive() : this function to receive any request (messages) from clients

Then it is answered according to the protocol in the message .

2-Send() : this function accepted Object User “ The user that the sender wishes to communicate with ," IPSender , Port NumberSender respectively . then send data Based on the transmission protocol.

3- broadcast() : A general message is sent to all users and is received from one of the users

4- search() : this function accept userName Then it is searched for in the list then return Object type User.

5- OnlineClient() : Here the IP’s and Ports (because ID of any Client) in the list are compared with the IP’s and Ports in the queue. If the first IP and Port in the list is in the queue, it is deleted from the queue only and skipped in the list. But if the IP and Port is not in the queue, this means that the user has become inactive and therefore must be deleted from the list and the queue as well

6- sendAllUsers() : Send active usernames to all users

3-Side Server : User.class

```
 5  public class User {
 6  private String id_User ; // contain user name
 7  private InetAddress IP_User ; // contain IP Address for user
 8  private int PORT ;// contain IP Address for user
 9  public User(String id_User, InetAddress iP_User, int pORT) { // Generate new User
10      this.id_User = id_User;
11      this.IP_User = iP_User;
12      this.PORT = pORT;
13  }
14  public String getId_User() {
15      return id_User;
16  }
17  public void setId_User(String id_User) {
18      this.id_User = id_User;
19  }
20  public InetAddress getIP_User() {
21      return IP_User;
22  }
23  public void setIP_User(InetAddress iP_User) {
24      IP_User = iP_User;
25  }
26  public int getPORT() {
27      return PORT;
28  }
29  public void setPORT(int pORT) {
30      PORT = pORT;
31  }
32  @Override
33  public String toString() {
34      return "User [id_User=" + id_User + ", IP_User=" + IP_User + ", PORT=" + PORT + "]";
35  }
36
```

1-Side Client : Client.class

```
import java.io.IOException;

public class Client {
    static DatagramSocket datagramSocket; // // to send and receive messages for clients and server
    static InetAddress IP_Server; // store ip server to connect with it
    private static final int sizeBuffer = 5000; // size of buffer in byte
    static byte[] buffer = new byte[sizeBuffer]; // decleartion array of byte , (buffer)
    private static int portToClient = 0; // It will contain the port of the user you want to communicate with
    private static String ipToClient = ""; // It will contain the ip address of the user you want to communicate with
    private static String str2 = ""; // contain message to showing on screen during run time
    static String s;

    public Client(DatagramSocket ds, InetAddress ip) { // assgin socket , and ip of server
        this.datagramSocket = ds;
        this.IP_Server = ip;
    }

    public static synchronized void recive() throws IOException {
        while (true) {
            DatagramPacket packet = new DatagramPacket(buffer, buffer.length);// Prepare a package and assign it buffer
                                                                              // and and its size
            datagramSocket.receive(packet); // Receiving the incoming packet from the client or server

            // convert the incoming data to String
            String messageFromServer = new String(packet.getData(), packet.getOffset(), packet.getLength());
            // Check the first letter to see the type of incoming message
            int x = Integer.parseInt(messageFromServer.charAt(0) + "");
            if (x == 1) {
                try {
                    // this message contain ip adddress and port number for requested user
                    String str[] = messageFromServer.substring(1).split("#");
```

```
                String str[] = messageFromServer.substring(1).split("#");
                str2 += "MESAGR From SERVER ## : " + str[0] + "      " + str[1] + "\n";
                ipToClient = str[0];
                portToClient = Integer.parseInt(str[1]);

            } catch (ArrayIndexOutOfBoundsException e) {
                e.printStackTrace();
            }
        }
        if (x == 2) {
            // message from client and showing on screen
            str2 += messageFromServer.substring(1) + "\n";
        }
        if (x == 9) {
            // message contain all online usernames
            System.out.println("MESSAGE : " + messageFromServer);

            String[] stt = messageFromServer.substring(1).split("#");
            s = "";
            for (int i = 0; i < stt.length; i++) {
                s += stt[i] + "\n";
            }
            Platform.runLater(new Runnable() {
                String s = Client.s;

                @Override
                public void run() {
                    Main.ShowAllUsers.setText(s);

                }
            });

```

Explaining Receive() Function :

Its task is to receive any message coming to the client, whether it is from the server or another client. Then this message is dealt with according to the protocols for the reception process, which are found on page No. 6 and 7

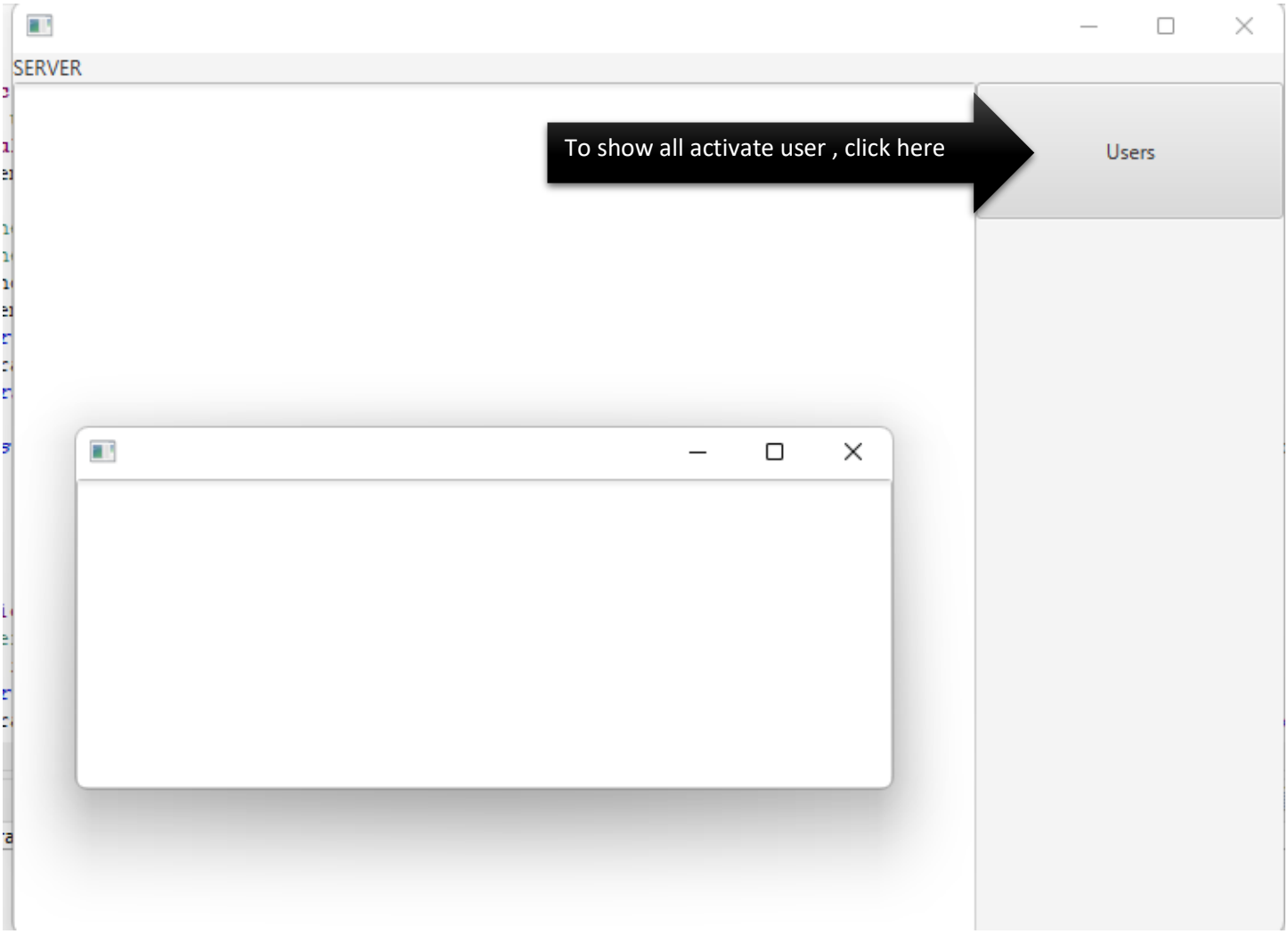
```
            }

            Platform.runLater(new Runnable() {
                @Override
                public void run() {
                    Main.MessageOnScreen.appendText(str2);
                    str2 = "################################################\n";
                    str2 = "";
                }
            });
        }

    }

    public static void send(String Myusername, String str, int x) throws IOException {
        if (x == 0) {
            // Send a message to the server to store the user in the list
            buffer = ("P" + Myusername + " :  " + str).getBytes();
            DatagramPacket packet = new DatagramPacket(buffer, buffer.length, IP_Server, 1288);
            datagramSocket.send(packet);
        }

        else if (x == 1) {
            /*
             * send message to server , and this message contain two protocol : 'A' This
             * message means that it must be sent to the server to add the user '@' This
             * message means that it must be sent to the server to get IPAdress and port
             * number for selected user .
             *
             */
            buffer = str.getBytes();
            DatagramPacket packet = new DatagramPacket(buffer, buffer.length, IP_Server, 1288);
            datagramSocket.send(packet);
        } else if (x == 2) {
            /*
             * send message to another user
             *
             */
            InetAddress ipClient = InetAddress.getByName(ipToClient);
            buffer = ("2" + Myusername + ": " + str).getBytes();
            DatagramPacket packet = new DatagramPacket(buffer, buffer.length, ipClient, portToClient);
            datagramSocket.send(packet);
        }

    }

    public static void sendMessageActive() throws IOException, InterruptedException {
        /*
         * Client keeps sending the aforementioned message every 5 seconds, Server keeps
         * updating the entry in the List, every time change has happened, it sends an
         * updated List to all online Clients.
         */
        byte[] buffer = ("k").getBytes();
        DatagramPacket packet = new DatagramPacket(buffer, buffer.length, IP_Server, 1288);
        while (true) {
            datagramSocket.send(packet);
            Thread.sleep(5000); // delay 5 second to send next message
            System.out.println("NEXT");

        }
    }
```

Explaining send() Function :
Its job is to send messages from the client to the server or another client
There are several protocols for this function and they are listed on page 6.
This function receives a message from the text boxes and is carried by its own protocol so that the message is handled correctly and sent to the right target.
Also, each message that is sent is carried by its own protocol so that the target can deal with the message correctly

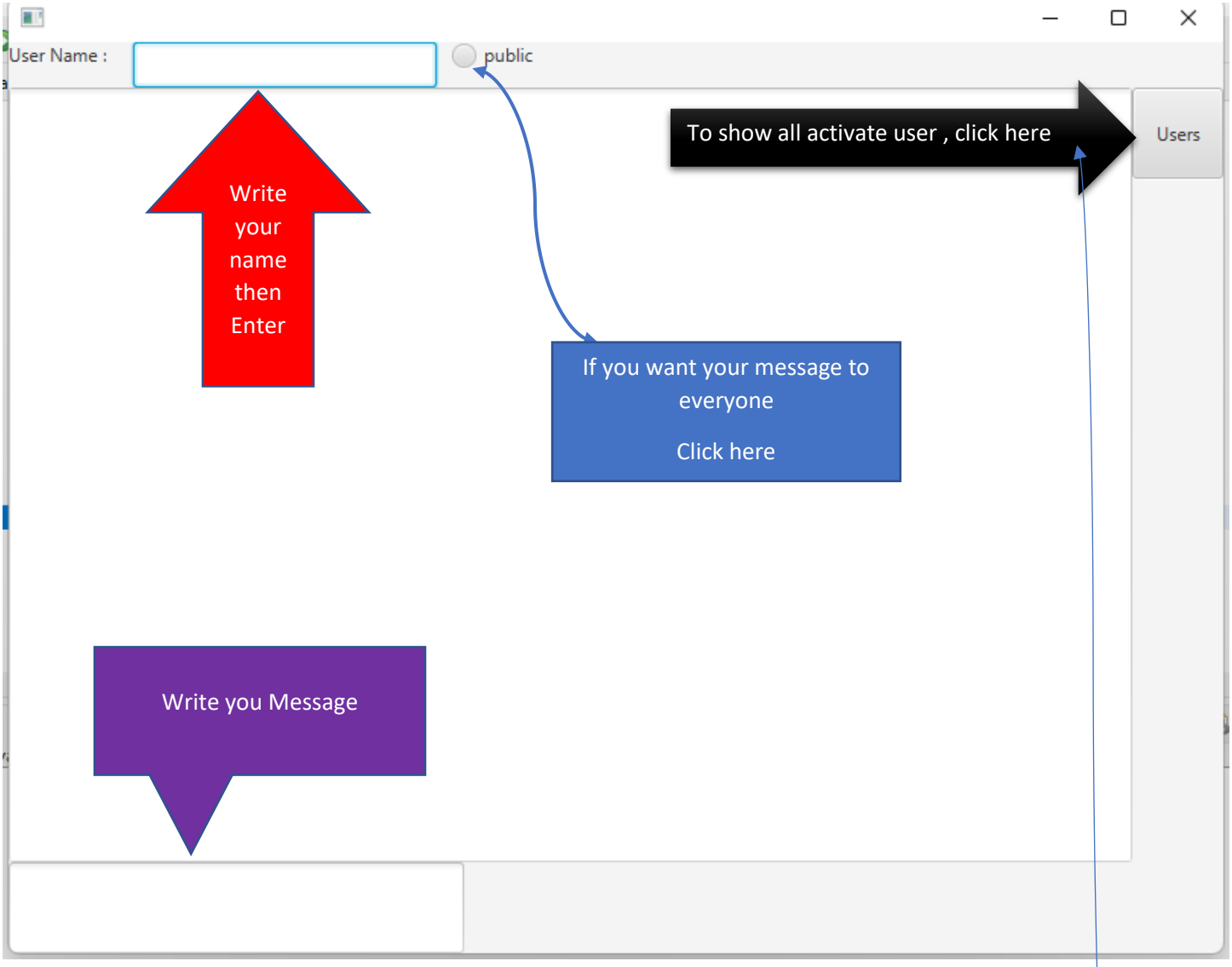
Explaining sendMessageActive() Function :
The client keeps sending a proof of attendance message, the server keeps updating the entry in the list, and every time a change occurs, it sends an updated list to all clients over the Internet.

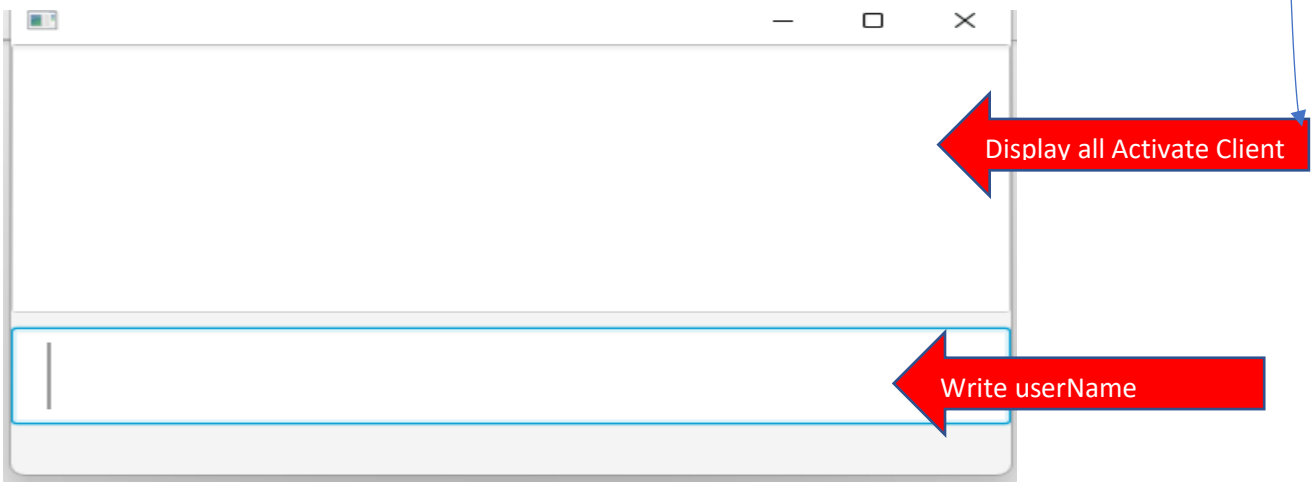
How to use the program ? Side Server :

1- Server Side



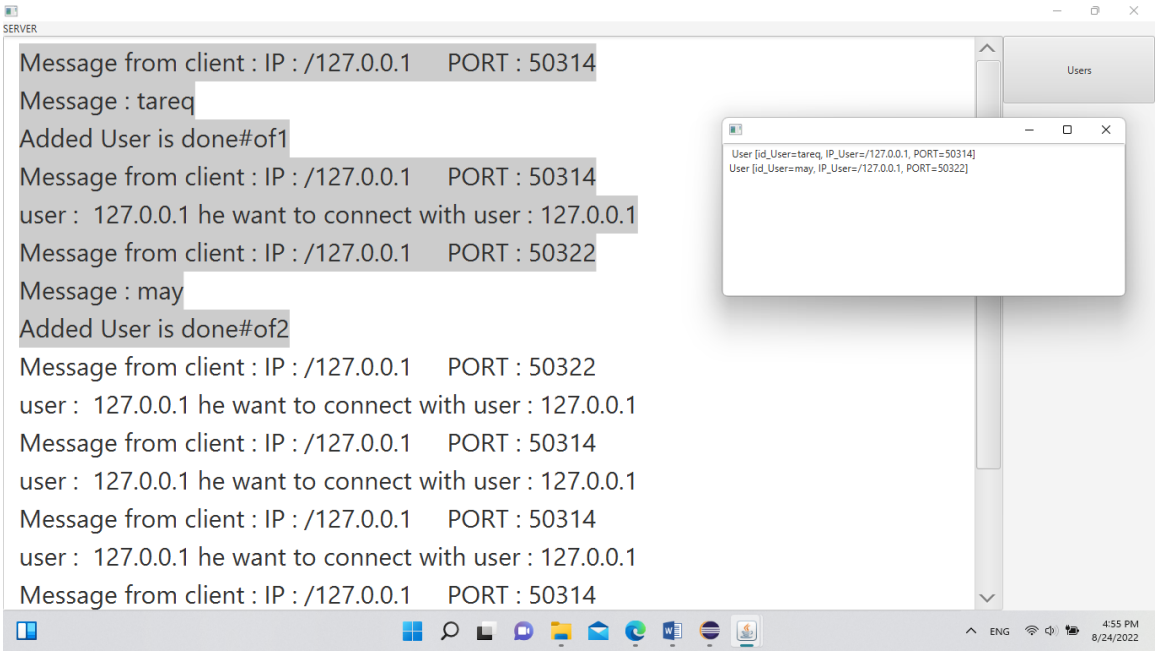
Client Side:


User Name :
public
Users
To show all activate user , click here
Write your name then Enter
If you want your message to everyone
Click here
Write you Message


Display all Activate Client
Write userName

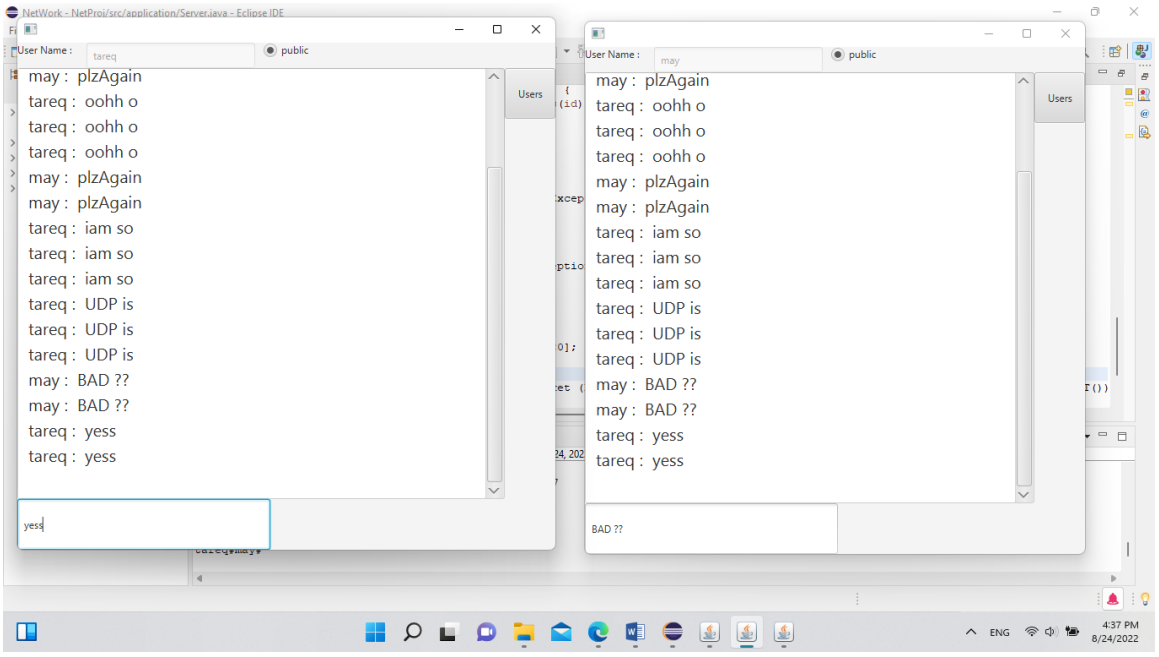
Run program :

Server Side :

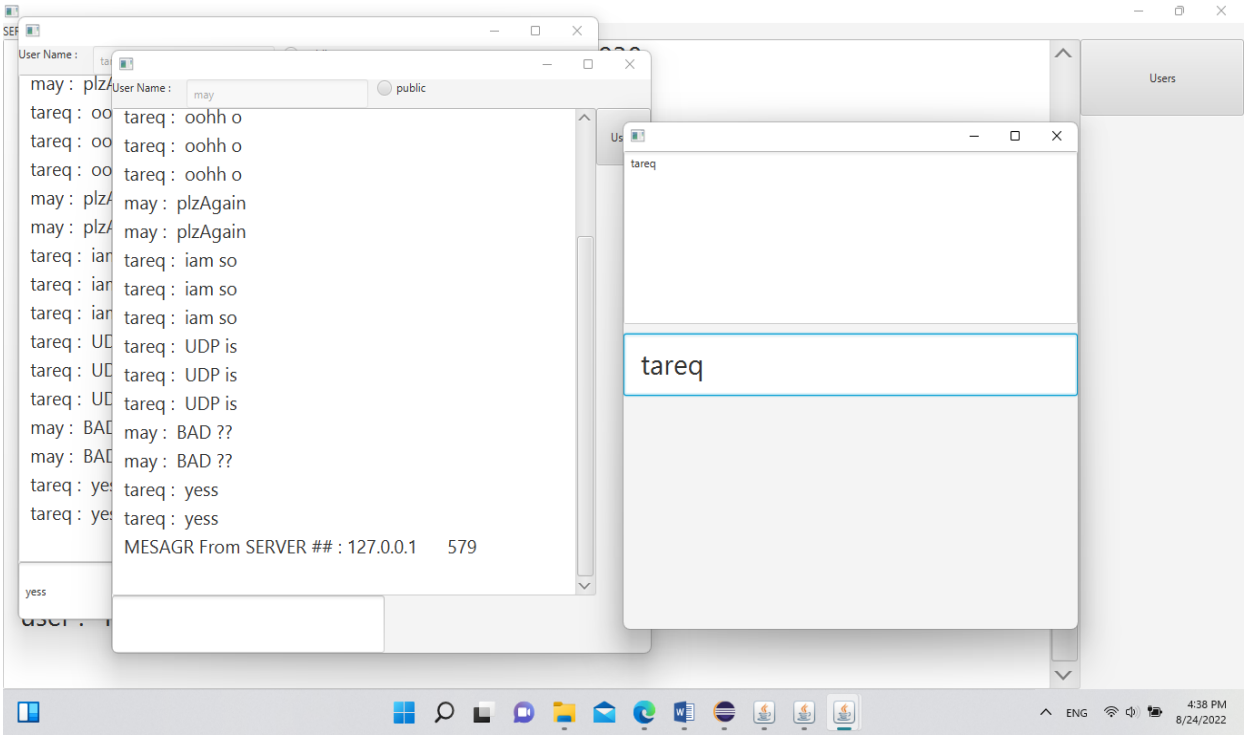


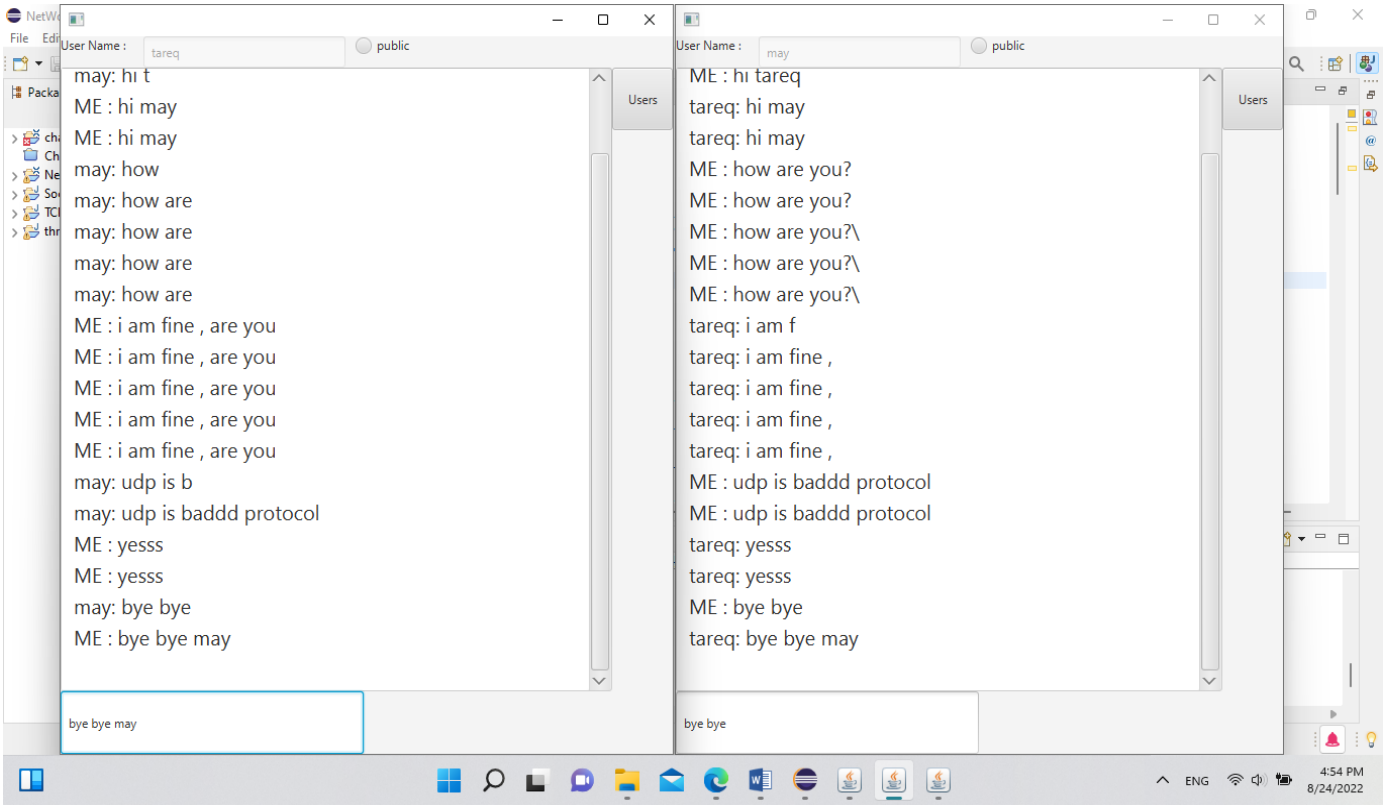
CLIENT_SIDE:

Public Chat :



Private Chat :





When one of client is leave the chat :

Example : May left the chat.