

Computer Science Department

Computer Networks

CSIT 340-02

Members

Hanser Rodriguez

Lorraine Torpey

Gabriela Hass

George Hamwi

**Phase 1: Introduction:**

1. UDP Functionality
   1. Socket Programming
   2. Behaviors
2. UDP vs TCP
   1. Behaviors
3. Group Project
   1. Major Components
   2. Implementation
4. Main challenges
   1. Addressing Project

**Phase 2: Design & Implementation**

1. Components
   1. System Implementation
   2. Design
      1. Pseudo Code
      2. Skeleton of Code
2. Technology Needed
   1. Constructs
3. Basic Requirements
   1. Functional Requirements
   2. Other Features
4. Behaviors of server and client

**Phase 3:Testing**

1. Specific Requirement
   1. Test Cases
2. Screen shots
   1. Test Cases
3. Corner case
   1. Failed Cases
   2. Fixed them

**Phase 4: Conclusion**

1. Summary
   1. Project
   2. Requirements
      1. Met
      2. Not met
2. Learned of Project
   1. Specific Technologies
   2. Teamwork Skills
   3. Difficulty of Project

**Phase 5: Team Contribution**

**Phase 1: Introduction**

For our group project in CSIT 340, we are tasked to design and implement a chat room server and chat room client. . UDP socket enables simple IP communication using the user datagram protocol. UDP is a very simple protocol. Messages which are datagrams are sent to other hosts on an IP network without the need to set up a data path or transmission channel. UDP is connectionless, unreliable, datagram protocol. UDP and TCP differ in their Implementations, TCP socket enables reliable IP communication using the Transmission Control Protocol. TCP runs on top of the IP. TCP is a reliable and connection oriented protocol. The TCP connection must be established before exchanging data, TCP retransmits data that did not reach the final destination due to errors. While TCP requires a established connection before exchanging data, because UDP is very simple protocol the datagrams are sent to other hosts on an IP network without the need to set up special transmission channels.

For this project our chat room server maintains the chat room, this will always maintain the active clients in the chat room. Our chat room will send a welcome message whenever a new client joins the chat room, it will also add the new client to the active clients . The chat room will notify all existing clients when an existing licentious leaves the chat room it will also remove the client that left from the active client list. Our chat room will broadcast a message to all the existing clients when an existing client sends a message to the chat room. Some major components that our project we need to take in to account is to make a client-server model as a means to represent the communication in a network system

One of the main challenges we will face as a group would be scheduling time to meet as a group. Different from the other projects we have done the communctions would be one to one communication where this group project implements a one to many communication in which there is one server communicating with more than one clients. The client will need to know the addresses and client that are being added in the queue of the server. Once the server waits for incoming messages or connections the server will then know about the clients unique identifiers.

**Phase 2: Design & Implementation:**

During the design and implementation phase of the project we had to make sure that the ChatRoomSever maintained the chat room. And a set of list of active clients and with each client added to the queue and different name is appears in the chat box also maintain a wide range of messages from the different.

**Components**

Some components that are needed in this project in order to work. Is we would need to do some system implementations. Such as for the ChatUDPClient we added the is an interfacethat contains a single method: public void. A class that implements the interface must contain an actionPerformed() method. The ActionEvent parameter is an Event object that represents an event (a button click).ActionListener And we added the nickname, hostname, and port to help identify each client that will be chatting in the chat room. And this will fall in the same place for the Server side of the code as well.



**Pseudo Code**

Designing the code was not an easy task we decided to first at least the get the Pseudo code correct and then implement in Java.

**This the ChatUDPClient**

**Adding if statement here** if (inPacket.getLength() != 0) {

String inMsg = new String(inPacket.getData(), 0, inPacket.getLength());

chatView.append(inMsg);

catch (IOException e)

e.printStackTrace();

**Adding frame here**

setSize(500, 400);

setMinimumSize(new Dimension(500, 400));

setTitle(nickName + "'s chat box");

**Add default values here**

hostTF.setText(hostName);

portTF.setText(port + "");

nickNameTF.setText(nickName);

**Add chat view here**

chatView = new JTextArea();

chatView.setEditable(false)

**Add chat input here**

chatInput = new JTextArea(3, 10);

chatInput.setLineWrap(true)

break;

**This the ChatUDPServer**

public static void main(String[] args)

socket = new DatagramSocket(port);

print ("Server started at port " + port);

DatagramPacket inPacket = new DatagramPacket(inBuf, inBuf.length); **Adding While loop here** while (true) socket.receive(inPacket);

boolean firstJoin = true;

User user = new User(inPacket.getAddress(), inPacket.getPort());

System.out.println("User port: " + user.getPort());

**Adding for statement for** (User u : inUsers) {

if (u.getPort() == user.getPort() && u.getAddress().equals(user.getAddress())) {

firstJoin = false;

**send message to chat room**

StringTokenizer st = new StringTokenizer(inMsg, "\t");

String senderName = st.nextToken();

String msg = st.nextToken();

if (msg.equals("~leave")) {

**user leaves chat room**

for (User u : inUsers) {

if (u.getAddress().equals(user.getAddress()) && u.getPort() == user.getPort())

user = u;

inUsers.remove(user);

else

**notify first join of a sender**

if (firstJoin)

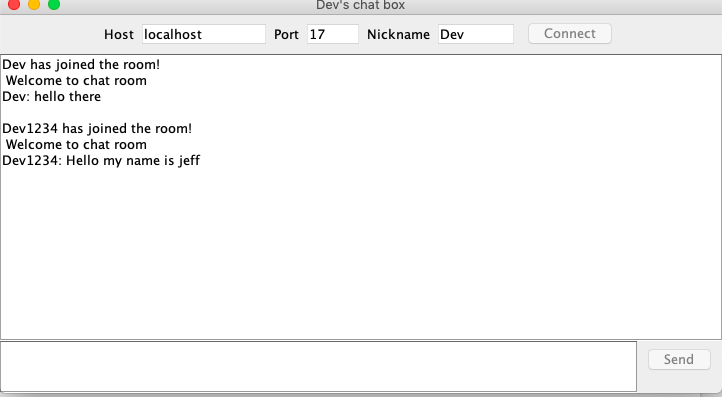
inUsers.add(user);

sendMsg(senderName + " has joined the room!\n");

**Functional Requirements**

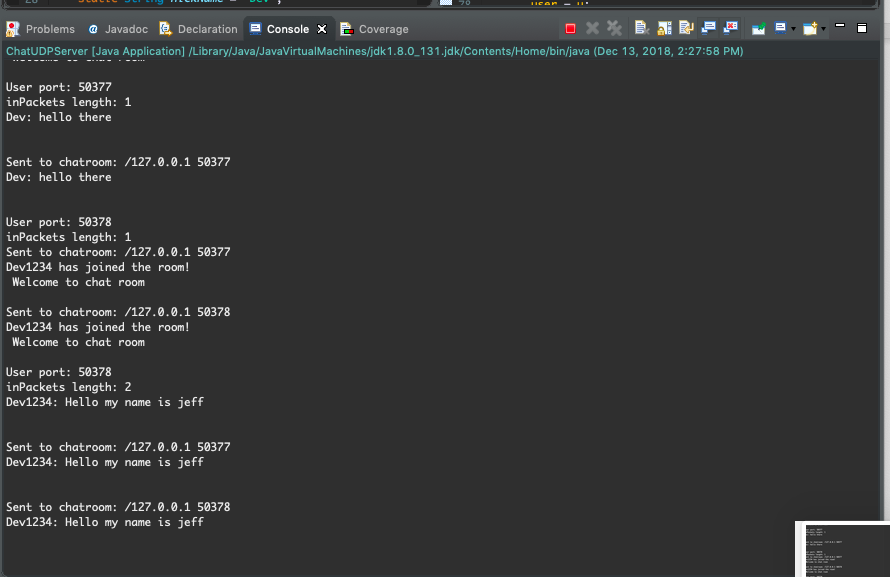
Some functional requirements we had to make sure was implemented in the program. Is that the chat box GUI appeared once running the client UDP that it automatically joins the chat room when the client program is run. The client is also allowed to input messages and appears with a unique identifier.

Here you can see the client has joined the chat room and is able to input a message. The host, port, and nickname is also unique identifiers to help identifier the different users and make sure that those users have the right port number to join the chat room.



**Behaviors of server and client**

Here you see the behavior of the server side of the program. On the server side of the UDPServer you can see the client is able to join the chat room. And see the welcome message to newly joined client. It also notifies all existing clients in the chat room that a new client has joined the chat. Also you can see the broadcast of what's happen in the server side.



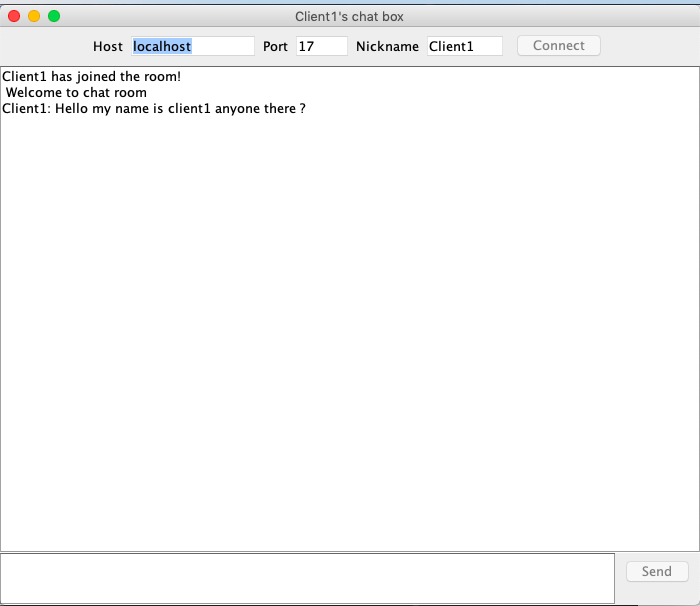
**Phase 3:Testing**

Running UDPserver

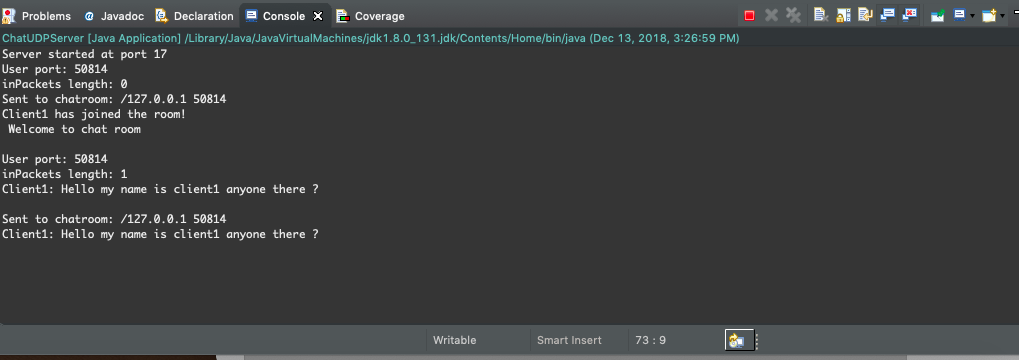


Running client

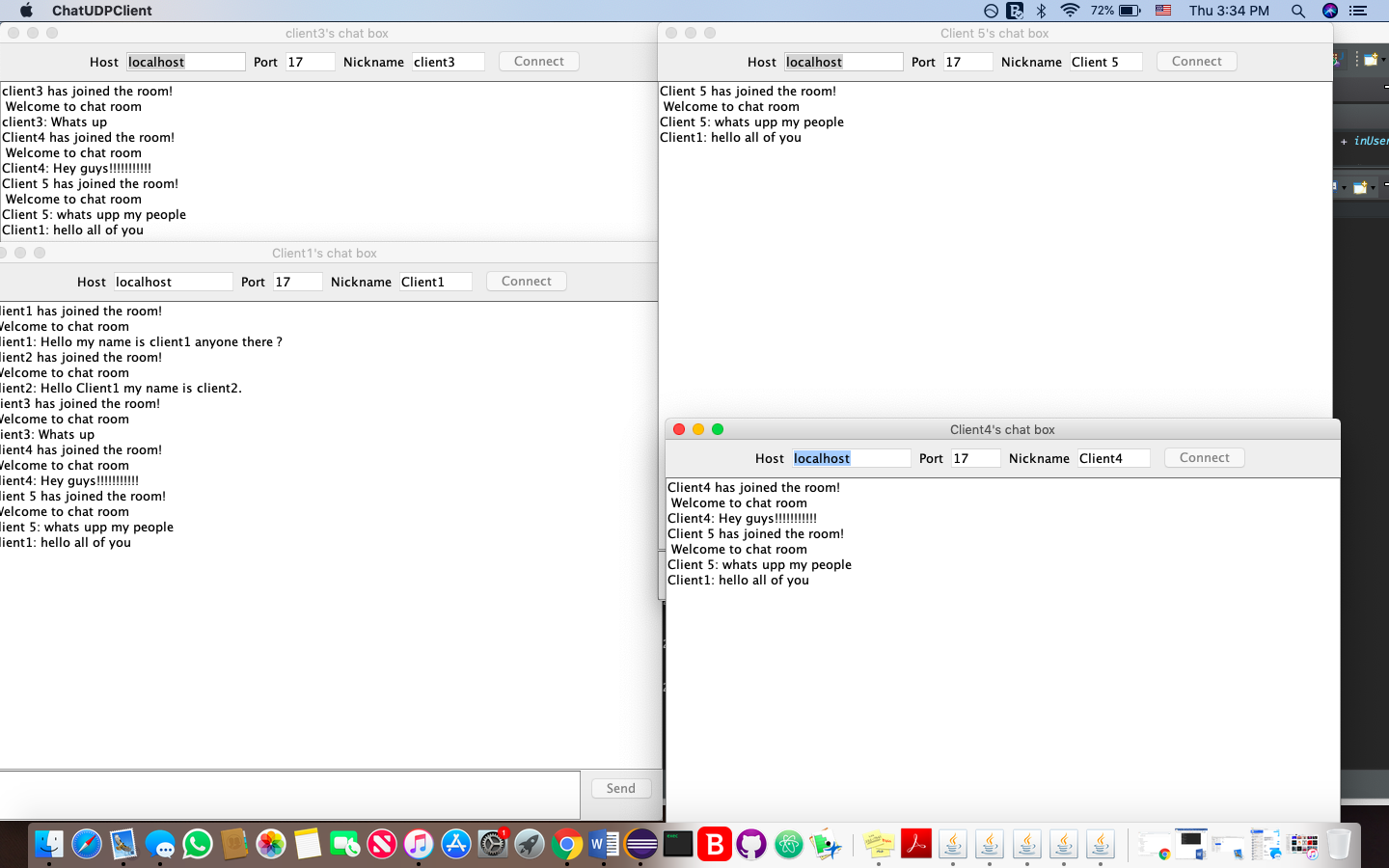
UDPClient is connected and able to input message.



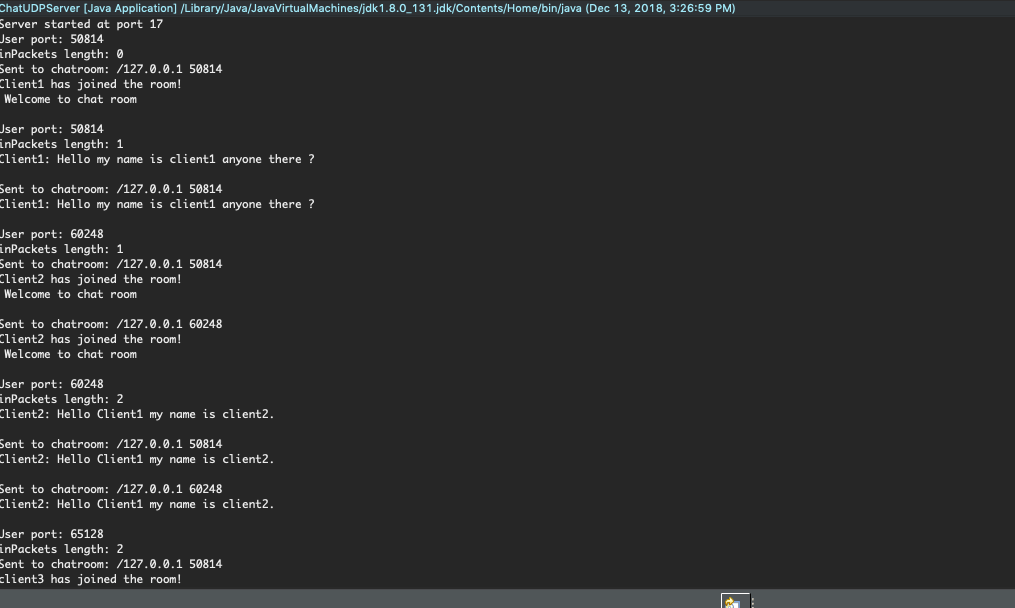
Server is able to receive message



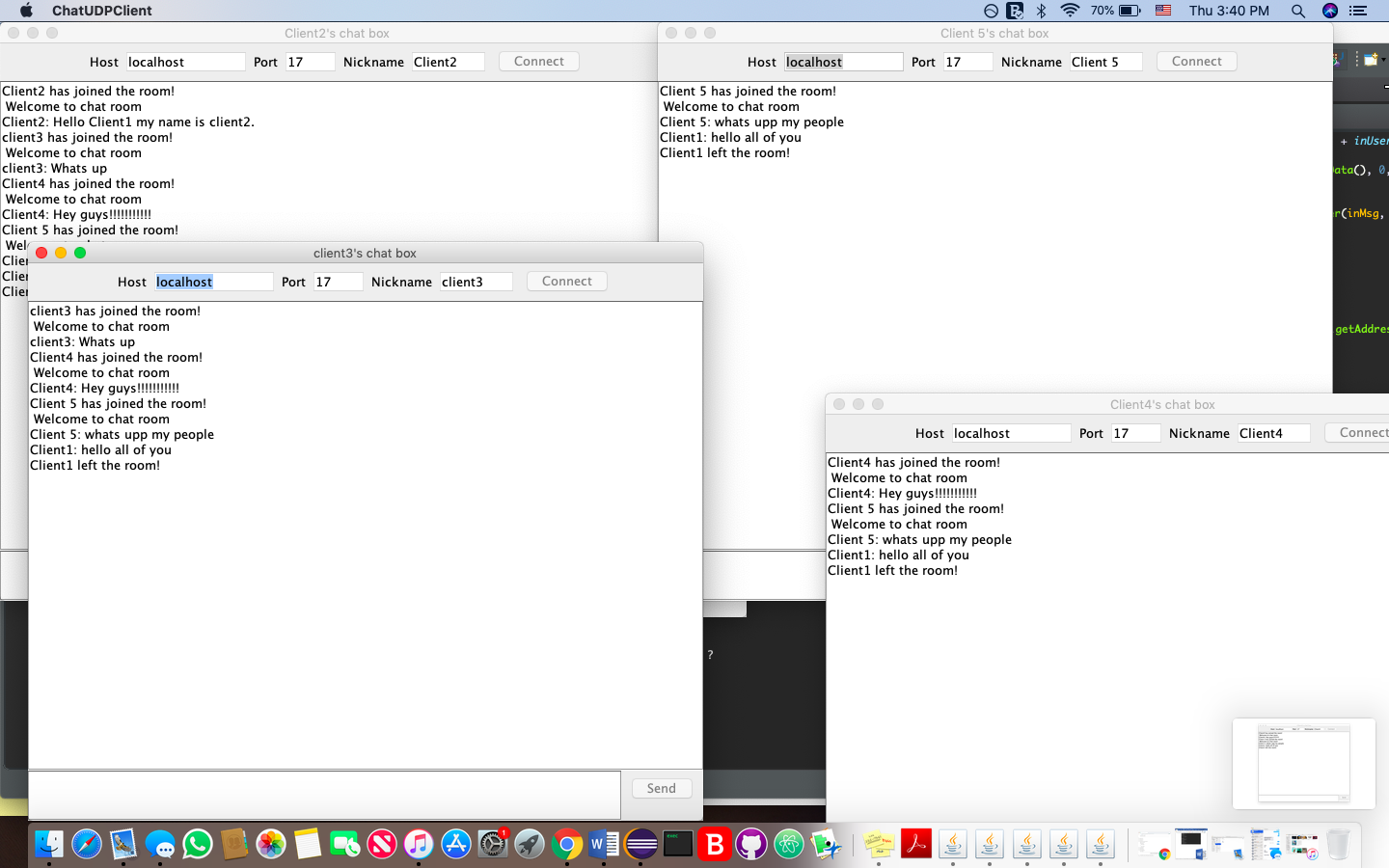
Here you multiple users. Are able to join the chatroom and each client is notified and able to input a message.



Server end on of multiple users.



When client1 leaves the chat room. Multiple user are notified that client1 has left the room.



**Phase 4: Conclusion**

Our group had great difficulty due to scheduling and the project was also difficult than the other projects. In the other projects normally can meet once to divide up all the work and everyone will work each other parts. For the time span for project time was short due to other assignments and classes that conflicting. During the testing phase which played an important role in the project was difficult due to the many errors we had of the code. Although this project wasn’t that difficult only if we had more time to go over the framework of the code.

**Phase 5: Team Contribution**

Hanser Rodriguez- report= design, testing, conclusion, client code

Lorraine Torpey- Client code and Server code, testing

Gabriela Hass- Client code, and Server code, testing

George Hamwi- Introduction report.