Distributed Chat Server System

Ishan Arora
Indian Institute of Information Technology, Allahabad
IIT2017501

Vikrant Singh
Indian Institute of Information Technology, Allahabad
IIT2017502

Srikar Anand
Indian Institute of Information Technology, Allahabad
IIT2017504

Akshay Gupta
Indian Institute of Information Technology, Allahabad
IIT2017505

***Abstract*—This document is a report of the project - 'Distributed Chat Server System'. In this document we present various modules used in the implementation of the project. We also explain architecture of the software and how to run the the project.**

***Index Terms*—Distributed Systems, Chat-Server, Distributed server, Socket Programming**

I. Introduction

In this project we have implemented a chat system in which users can connect to a server and join a chat room. The fact that this application is distributed allows us to scale up so that many users can use the application in real time.
This project has been made in Java8. The message protocol supports the transfer of arbitrary Java types over the network. For the purposes of the project, we implemented standard chat messages. The implementation is structured into three primary parts: the client, the server, and the protocol used to communicate between them.

II. Software Architecture

The implementation is structured into three primary parts: the client, the server, and the protocol used to communicate between them.

A. Client

Client is the structural model which would interfere with user with the GUI (Graphical User Interface). The client structure takes the message from the user, does any pre-processing that is required and sends the message to the Protocol model.
This structure handles establishing the connection to the server, handling messages retrieved from the server, and passing messages from the user to the server, while keeping the user informed of all activity in the user interface. When a successful connection is established, it creates different threads for different tasks such as send, receive messages so that it can see live messages.
These messages can be of two types : The messages sent by the client user to the chat rooms, or the commands sent by the user or client structure.

B. Server

Server is the back-end structural model, which would process the transmission of messages.It contains the metadata, dictionaries of clients, clients' names, IP addresses, dictionaries of chat rooms, names, and clients in that chat room. When a client connects to the server, the server creates a new thread that continuously keeps on listening to the client for new messages. When a new message is received from a client, the thread created redirects the message to the target client or chat room. When a command is received by the server, it will offload the handling of that command to a message handler much in the same way that clients handle their messages.

C. Protocol

Protocol is the structure that is shared between both client and server. This has to deal with structure of messages, encryption and decryption of messages, types of messages, and the methods by which client and server can handle the messages. As discussed, there are two types of messages that could be sent or received. One is the communication messages, other is the command messages. Communication messages can simply be directly passed to clients, which will be received by the message handlers at clients. However, commands will be handled by the message handler in the protocol.

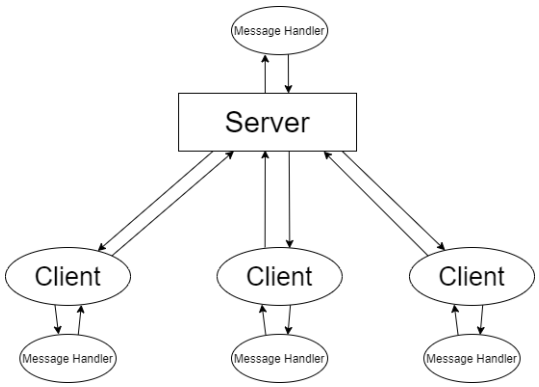

Fig. 1. Project Architecture

III. Implementation

A. One to One Messaging

The project implements peer-to-peer architecture where two clients send and receive message to each other through a server. At first , first client sends the message to server with the room id , then the server further sends this message to second client which has this room id. This implementation of One to One messaging is just the special type of group messaging when there are only 2 members in the chat room. Also , this implementation did not require any central server.

B. Group Messaging

Multiple servers can be created with this application , each of them having their own set of clients . Each client can create various room(s) with an specific id and other clients on that server can join those chat rooms through that id . Whenever a client creates a room with an id , it creates a thread , all those clients who have joined this room id keep listening to this thread from the server . A global room is where all the clients in a server interact . This is the default thread created when the server is allotted a port . Thus this implementation does not require central server and all the messages are synchronised.

C. Message Ordering

Messages in chat rooms need to be in order of their sent time , so that other client receive the message in order in which the messages were sent . Since server runs an infinite loop to keep accepting the requests and fires up a new thread whenever a new client is connected . Every time when a client sends a message its timestamp according to server's time is maintained . Messages are sent according to this before they are converted to byte-stream . This way ordering of message is ensured such that the other client receives the messages in order they were sent.

IV. Functionality and accepting Result

A. Functionality

- One to One Messaging
- Creating a chat room
- Joining a chat room
- Chat Room Messaging
- Audio Messaging
- File transfer
- Viewing chat messages in order
- Leaving chat rooms

B. Result

Lets see how the functionality of our chat system works.
The front page has two functionalities, first is to add the server and second is to add clients who can interact through these servers, as shown in Fig 2.

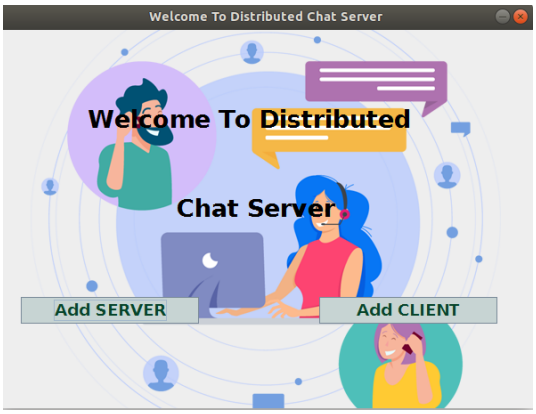

Fig. 2. Front page

For messaging user first need to create server by entering server port number as shown in Fig 3 and then he/she can add the clients.

For messaging user first need to create server by entering server port number and then he/she can add the clients. After creating the server, the clients can be added through add client page by entering

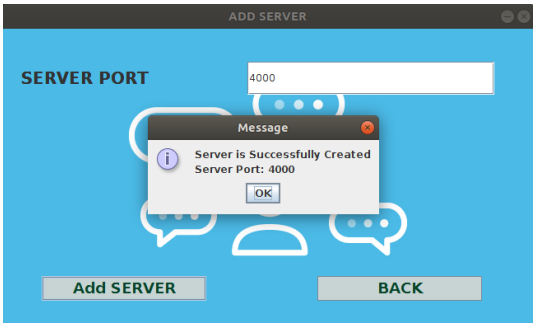

Fig. 3. Creating Server

the name of the client and the port number of the server to which we want to add the client. This is shown in Fig. 4.

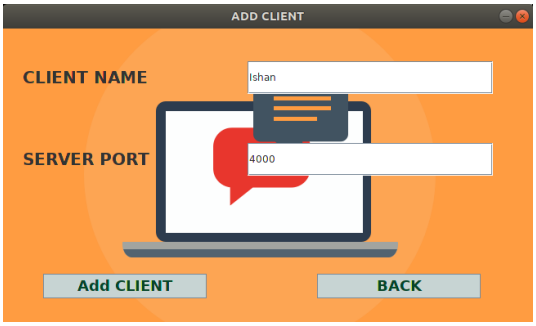

Fig. 4. Creating Client

Two clients can connect themselves to the same server to have one to one messaging.They can also create new chat rooms for this purpose. Clients can also send audio messages to each other by clicking on Start Recording, then record voice message and then click on Stop Send when finished. This is shown in Figure 5 below.

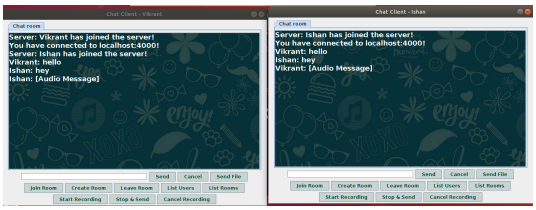

Fig. 5. One to One Messaging and Audio messaging

User can also create room for the chat by clicking on the Create Room button and entering the name for the room as shown in Fig. 6.

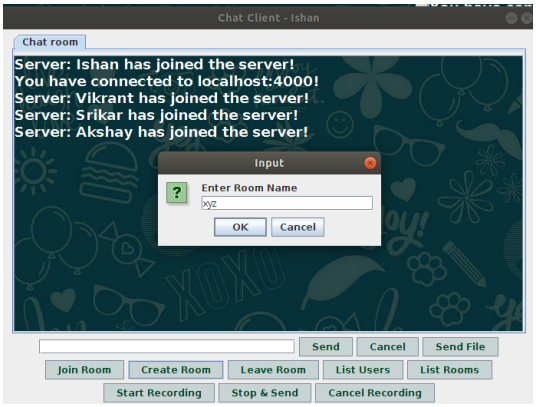

Fig. 6. Create room

Other clients can join the room by clicking on Join Room and then entering the valid room id to join. They can see room id on clicking the List Room button. This is shown in Fig. 7.

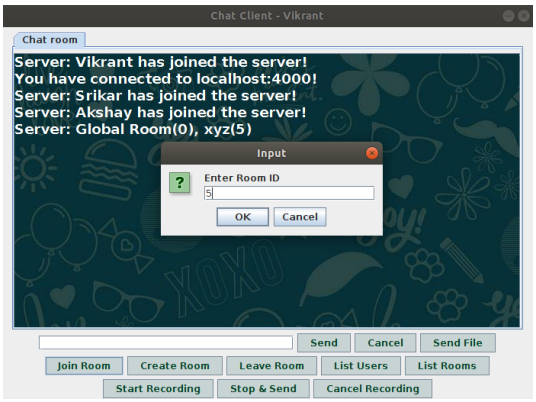

Fig. 7. Join Room

Group messaging can be done either through the global chat room or through other room which client may have created as shown in Fig. 8. Cients can also leave room whenever they want by clicking Leave Room button.

Clients can also send the files to each other by clicking on Send File button and selecting the require file. Other clients may download it and save it on their systems. This is shown in the Fig 9 and Fig 10.

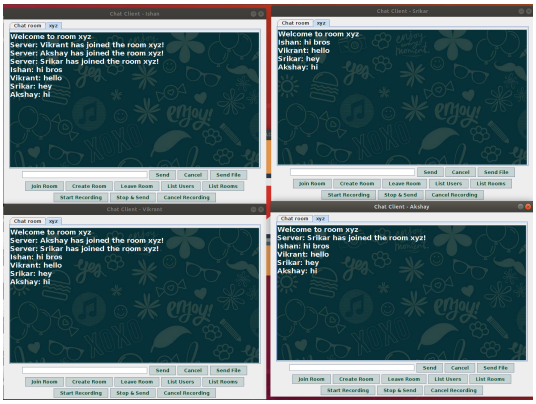

Fig. 8. Group messaging through Chat Rooms

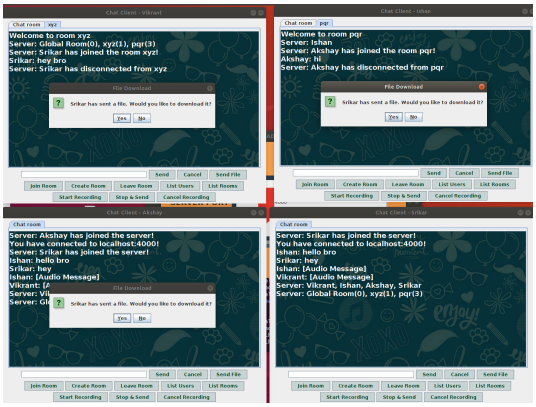

Fig. 9. Sending Files

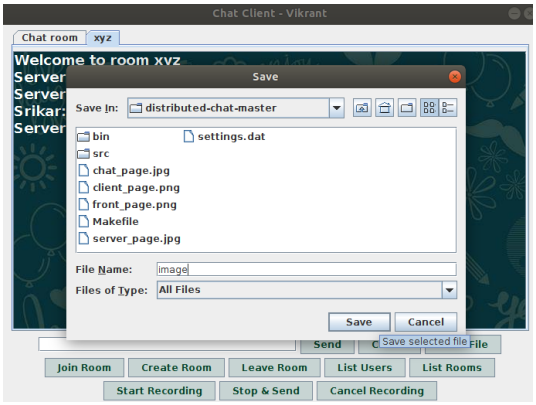

Fig. 10. Downloading Files

Also message ordering has been taken into consideration. The messages are in order of their sent time only, the message which is sent earlier is delivered earlier. This is clear from Fig. 5 and Fig. 8.

V. Technology Used

- Java Socket programming
- Java Swing
- Java AWT

VI. Conclusion

In this project we have managed to build a finely structured distributed chat-server system, which is a complex combination of various structures or modules like client, server, message handlers.
Here a Graphical User Interface has been provided through which User can control the client module. User can Log In, create rooms and enter a chat room of their choice. After entering, user can send text messages , audio messages or transfer files to client in the various chat group facility. The project can further be extended to implement audio call and video call features. In the current version of the project, there are no security features implemented. This can be upgraded by building an encryption of chat and key-protected chat rooms.

References

[1] Rolou Lyn R. Maata, Ronald Cordova, Balaji Sudramurthy and Alrence Halibas, “Design and Implementation of Client-Server Based Application using Socket Programming in a Distributed Computing Environment,” ,Conference: 2017 IEEE International Conference on Computational Intelligence and Computing Research, At India

[2] Motaz Daadoo., Developing and Implementation of Distributed Chat Applications using WPF and WCF , European Journal of Scientific Reseach ,Vol. 143 pp. 424-440 , Jan 2017