COL334 Assignment2

Abhinav Kumar (2019CS50415)

September 2021

1 Introduction

In this assignment, we build a chat application that allows users to send plain text messages with one another. Users can direct messages to other users using an @ prefix, and the server needs to forward these messages to the intended recipients. The message could be intended to be sent for a single client (Unicast) or all clients(Broadcast). The messages would be communicated as plain text, so the server will be able to read the messages.

2 Client Side

The client side application take a command-line input for the username and the server's IP address (localhost or 127.0.0.1 if you are running the server locally), and then start with opening two TCP sockets to the server, one for sending messages to other users and one for receiving messages from other users. When the sockets are opened, the client send REGISTER messages and read the acknowledgements.
After that it will start two thread, one for sending message to other users and one for receiving message from other users.

3 Server Side

The server side application begin with listening for connections on the port. Upon receiving new socket acceptances from a client application, the server starts a thread for that socket. If the thread is started for the socket which is registering for TORECV then the server add the user and the socket connection to a dictionary and end the thread. If it is for TOSEND then it will wait for messages to come from that user.

4 Brodcasting

If a user send a message with @all then it is a brodcast message and we used stop and wait implementation for this. At the server, we try to send the message to all the user one by one. If it's successful, try the same with B and so on. If at any point the server receive a failure while sending to any recipient, it report the error to the sender and stop sending the message to the rest of the recipients. If only all recipients successfully receive the message, a success message sent to the sender.

5 Error Handling

ERROR 100 : If the user send a registration message to server with the wrong type of username then the server will send ERROR 100 to user.
ERROR 101 : Server to client message in response to any communication until registration is complete.
ERROR 102 : If the message send by some client is not delivered to the other user by server due to the other user not registered or the other user send the server ERROR 103, then the server sends ERROR 102 to the sender client telling that your message is not delivered.
ERROR 103 : If the header file of the message send by the client to server or by the server to client is found malformed by the server or the client, then the server will send ERROR 103 to the client and the client will send ERROR 103 to the server respectively. In both the cases it will close the connection with the client. In first case where the server found that the message is malformed then it will send ERROR 103 to the sender client and close the connection with registering the user. In the second case where the client found that the message is malformed, then the client will send the ERROR 103 to the server and the server will close connection with this client who send the ERROR 103 not with the sender client.