A Simple Messaging Application

Bharati Khanijo
e-mail: bharatik@iisc.ac.in

Contents

1 Introduction

The motivation for creating a messaging application was from the first lab assignment. The application enables a user to register itself and then send messages to another registered user. It also provides the user with the feature to send messages to a group of registered users. Also in order to receive the message the receipts may not be online at the time the message is sent. They can retrieve the messages later when they are online. The report includes the detailed design of the application along with some examples.

2 Methods

Application is designed according to a client server architecture, where the message sent from one client is first sent to the server and the intended recipient(s) fetches the message from the server. It was developed with the help of Java Development Toolkit 12.0.2. An attempt was made to use the concept of sockets and multithreading, with emphasis on maintaining the consistency of shared variable in case of concurrent access by multiple threads.

2.1 Overall Design

The application can be logically divided into 2 parts. One part that runs on the client's machine (Client.java) and another that runs on the server. The server is always accepting new clients. When a client application is run, it attempts to connect to the server. The server on receiving the request, allocates a separate thread to handle all that client related requests and increments the number of active users. When the client exits the application,(i.e. will be treated as offline), the connection between them is closed and the number of active users is decremented at the server. To ensure that the increment and decrement operations are thread safe, they are placed inside synchronized block. Also this application uses a database to maintain registered users and their group membership along with the message details, thus removing the need of the clients to be simultaneously online while communicating/chatting.

2.2 Features

When the client application is run, the user is provided with 2 options, one to login and another to sign up. In case user is new to the application, it would be able to register itself with the server. Once registered the user will be able to send and receive messages with the help of command line interface. The user can send message to another registered user or a group of registered users with the help of SendMsg command. For example, to send message to another registered user (say Namita), the user will enter the following command on its terminal where Client application is running. SendMsg–Namita–Hi Namita, This is Sneha. Similarly,in order to send group messages the user will simply replace the recipient name with the group name. SendMsg–G1–Hello Everyone, This is Sneha. To view received messages the user will enter the following command RcvdMsg On doing so, the user will be able to see all messages either sent by him/her (To maintain coherence while reading messages) or intended to

be received by him/her either directly or as the recipient being a member of the group to which it was sent. For a message, the user is shown its id, it's sender, it's receiver, its sent time, presence of attachment, followed by the message itself. The list of messages are themselves sorted with the latest message at the top.

3 Conclusions

It was found that the application successfully enabled the users to register themselves and send and receive messages in both cases when the recipient is online and when it is offline. Also the message sent by the registered user to a registered group was visible to all the members of that group. The application server was able to cater to multiple clients simultaneously and also successfully able to maintain number of active users.

References

[1] LaTeX Sample IMRaD Format, http://www.astro.uu.se/ ulrike/StellEvol06/latex/report.tex

[2] Socket Programming in Java, https://www.geeksforgeeks.org/socket-programming-in-java/

[3] Introduction to Threads in Socket Programming, https://www.geeksforgeeks.org/introducing-threads-socket-programming-java/

[4] JDBC Prepared Statement, http://tutorials.jenkov.com/jdbc/preparedstatement.html