INDIAN INSTITUTE OF TECHNOLOGY, DELHI

ASSIGNMENT REPORT

COL334 : Assignment 2

APAR AHUJA | ENTRY NO. 2019CS10465

Course - COL334 | Prof. Abhijnan Chakraborty

Compiled on September 17, 2021

Contents

Chapter 1

Application Design

1.1 Client Side

The client side takes username as input from the terminal. This is later used to register the user at the server. The client side opens two sockets, one for receiving messages and one for sending messages. This is done to simplify the process and removing the need of concurrency management. Both the sockets are registered with the username. If the registration is not successful the program shows the error and quits. Note that the username must be alpha-numeric and should be available at the server at that moment, otherwise an error message is shown, **'Username already taken'**. Further, the username cannot be the keyword **ALL**, that is used for broadcasting messages.
After successful registration the client side starts two threads - one for the sending socket and one for the receiving socket.

1. The sending thread - 'write' takes in input from the command line and sends the message to the server. If the program is unable to parse the input, INPUT ERROR is raised and the user is requested to type again. Further, if the keyword **bye** is entered or the server is down, the sockets and threads are closed. If an incomplete header error is received the sockets are closed. This is because in this case the server will not know how to parse any further data it receives on the TCP connection. The client may then reattempt to open a new connection and register again.

2. The receiving thread - receive' takes messages sent by the server. If the keyword **bye** is received, the sockets and threads are closed. Now, there may be two types of header errors possible, ERROR 103: if the **FORWARD** message received has an header error then ERROR 103 is raised and sent to the server. ERROR 104: if the **RECEIVED** acknowledgement had an error, ERROR 104 is raised. Note that all the errors are also displayed on the command line.

1.2 Server Side

The server opens a **server** socket and starts accepting connections. If a connection request is made the server starts the user registration and registers the username and the send and receive sockets at the client side into dictionaries. If the username is already being used or the keyword **ALL** and ERROR 200 message is sent back and user may try to register again. Similarly malformed usernames are handled and ERROR 100 is sent to the client for this case. Note that clients aren't allowed to communicate over the server without registering themselves.
Post registration a **handle** thread is created for the sending socket. The handle thread receives messages from the clients and acts accordingly. If a **bye** message is received, the user is deregistered and it's thread and sockets are closed. Similarly if an incomplete header message is received the user is deregistered and it's thread and sockets are closed. This is because in this case the server will not know how to parse any further data it receives on the TCP connection.

1. Broadcast messages - If the username is **ALL** then the server tries to broadcast the messages to all the users registered in it's database. If any error occurs during the process, ERROR 102 is raised and is sent to the client.

2. Unicast messages - If the recipient user is not found then an error is raised. Otherwise the message is forwarded to the recipient and the server waits for an acknowledgement. If acknowledgement is received it tells the sender that their message has been successfully sent.

Note that the message length sent by the sender can't be more than 1024 bytes. This limit helps us to protect our server from buffer overflow attacks from hackers, who might try to bring down the server with large messages.