CS425: Course Project
ChatApp - Client-Server Messaging Application

Group No: 10 - Gurkirat Singh, 14258 Teekam Chand Mandan, 13744
Kshiitiz Singhal, 14330

December 15, 2017

Introduction

In this project, we have designed a client-server architecture based chat application named ChatApp. The application supports 2 modes of chat: Broadcast and Private messages. Such an Internet Relay Chat(IRC) application can be used to communicate between among team members and also extended to work for communication amongst other applications.

Objective

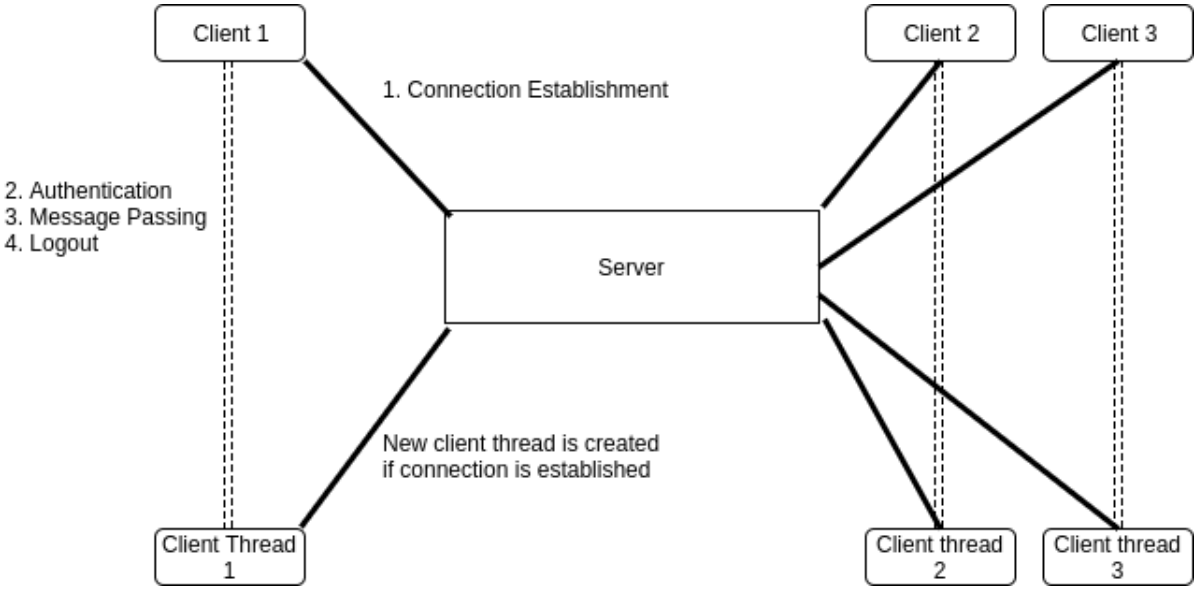
We wanted to design an application that behaved like modern day chat applications and is easy to use. Apart from the aforementioned features of one-to-one communication and broadcast following features too were envisioned:

- Easy to use user interface
- Asynchronous Messaging
- Responsive Server
- Special commands to communicate with Server such as - Change mode of communication, check who else is online, block unwanted conversations
- Channel Conversations
- Minimalistic configuration to set up the application
- Encryption of messages at various levels
- Safeguards against attempts to hack user credentials
- Appropriate Error Detection and Handling
- Logging of user activity

Assumptions

Currently, in the absence of graphical widgets we have assumed that the user is familiar with the basic commands used in the application. Also, the application requires the client end to be aware of the host server IP and Port. Also the client should be on the same network as the host. The number of connections at any time has been restricted to 100.

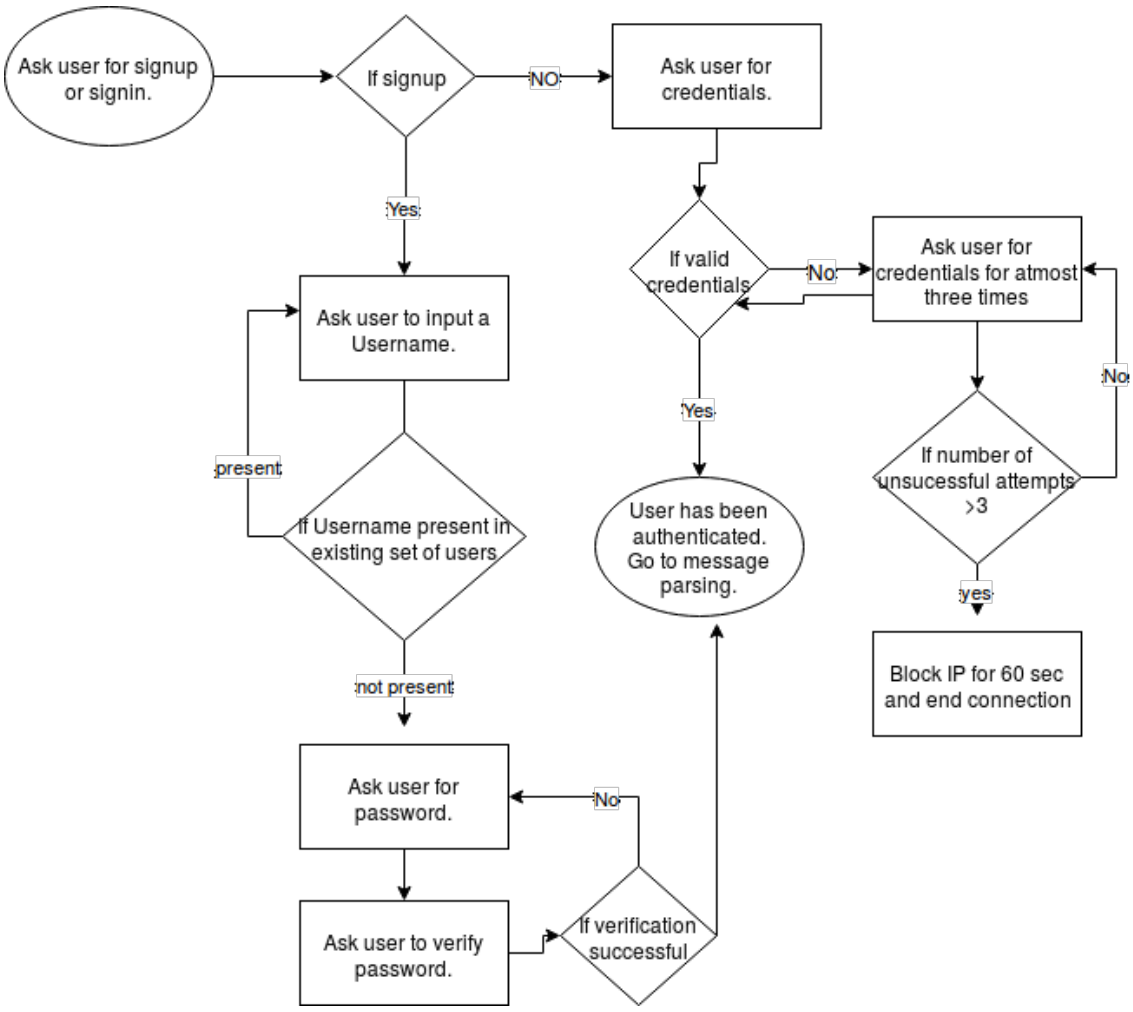
Architecture



- Connection Establishment:
  Once initialized the server is waiting for incoming connections. Once it establishes a connection, it creates a new thread to handle the client, while continuing to listen for other clients.

- Authentication:
  - Prompt user to either signup or login
  - In case of signup:
    1. Ask user to input a non-blank username
    2. If username is valid, i.e. non-blank and unique, go to next step, else repeat from step 1
    3. Ask user for non-empty password.
    4. Ask user to repeat password for confirmation. If passwords match, go to next step, else got to step 3
    5. User Created
  - In case of login:
    1. Prompt the user to enter credentials.
    2. If valid, login successful. Else, repeat for 2 more times.
    3. If all attempts are exhausted, block the connection IP for 60 seconds, to foil attempts to hack password.

- Message passing:
  User can now communicate with other users whether online or offline. The user is in broadcast mode by default. The messages along with sender and recipient are stored at server side to provide asynchronous messaging. Messages received when offline are printed when users comes back online next time. Thereafter the user can perform the following actions:
  - Send a message to everyone
  - Switch between broadcast and private messaging
  - Send a message to a particular user
  - List other online users.
  - List users who were recently online (in the last 1 hour)
  - Block a user/ Unblock a previously blocked user
  - Logout and signin as another user



Implementation Environment

The project has been implemented in the Python 2.7 environment. It uses the socket library for our TCP connection and RSA for encryption. All of the data is stored on the server end in JSON format text files.

Summary

The project involved basics of socket programming along with the creation of a protocol for communication. Addition of an additional window over the terminal makes the application much more appealing in comparison to other terminal based applications. To test our application, we ran the client on different laptops within the campus and communicated with our using our known server. As we were unable to complete all of the goals set in the beginning, there is potential in our project for improvement.