CS 425A: Introduction to Computer Networks
ChatRoom
Team 1

Rahul Gupta(14519) Shubham Sharma(14629) Ameya Loya(14080)

Introduction

Our ChatRoom is client-server messaging application. It supports private messages, broadcasting, asynchronous messaging. It has basic the security feature that requires each user to authenticate before he/she starts using the ChatRoom. In addition to basic security, our ChatRoom provides End To End encryption to the clients so that only the communicating users can read messages. A user also has been provided with facility of Blocking/Unblocking other users as he/she pleases.

Objectives

- To create a client server application messaging application
- To provide a user with a user interface that allows the user to interact with the application smoothly
- Implement various fundamental features like Signup, Private messaging, Broadcast messaging, Asynchronous messaging, showing status of various users(online or offline) and basic username password based authentication for security.
- To provide additional features like End To End encryption, Logout functionality, Blocking and Unblocking any other user.

Assumptions

- There will not be a lot of users so no database is required. In particular, we can use a text file to authenticate that consists of list of user names and their passwords.
- Message queues will not overflow.
- There are as many sockets available as the number of connecting users.
- End to End Encryption cannot be broken owing to the fact that P vs NP is an open question.

Architecture

Our ChatRoom is based on client server architecture. Multiple clients can communicate to each other via the centralized server. A friendly interface has been provided to the user to aid him/her in using the application. The user first authenticates to the server after which he/she uses the application platform to make use of various functionalities which is evident from the figure below.

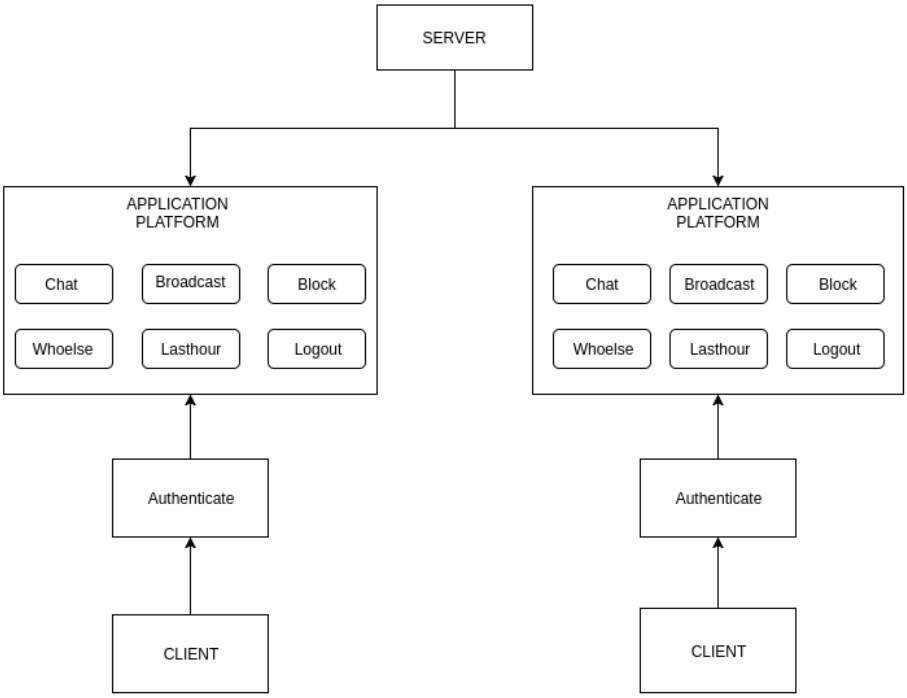

Figure 1: Application Architecture

Implementation

- We have used Python3 to implement our code.
- We have used tkinter library in python to create the user interface. On the client, a seperate thread runs tkinter and maintains dynamnic user interface while the other thread listens for incoming messages.
- A client connects to server using a TCP connection. We have implemented this using socket programming in Python3. A new socket can be created using the socket() call. Similarly, one can send and read data from a socket using send() and recv() calls respectively.
- MultiThreading has been used to allow for more than one client to connect to the server. Each time a new client connects, we create a new thread for the client on server. The advantage of a creating a new thread over creating a new process is the shared memory space with its parent.
- We have also used select() call to monitor over multiple file descriptors(sockets in this case). select() calls waits until one or more sockets are ready for I/O operation. Basically, we have used select() call to poll over the sockets.
- Keys of the intended receivers are distributed by the central server for encryption. When a user signs up, he/she generates and uploads a public key automatically while private key is stored locally on client's computer.
- End to End encryption has been provided using Crypto library.

Summary

We have been able to provide all the basic features that have been listed in the objectives. In addition these, we have also implemented the additional features that were the part of objectives. While doing so, we have made some reasonable assumptions as listed above. Through this project we learned:

- We learned how to handle I/O from multiple file descriptors aysnchronously using select call (which uses interrupt handling vectors).
- We learned to create user interface in python using tkinter.
- We learned about asymmetric end to end encryption and applied it in our application.