CS425 - Computer Networks: Mini Project, Chat Application with Gaming

Nishit Asnani (14433), Mayank Patel (14377), Sunny Kant (150740)

November 12, 2017

1 Introduction

The aim of our project was to get hands-on experience in developing client-server messaging applications, and to make a game playing interface that could interleave with the chat application like in popular internet based games.

Text message passing is the simplest form of communication over a network and only after understanding the basics of point to point message passing, broadcasting and asynchronous messaging can more complex applications be developed.

2 Objectives

We had started with the following objectives in mind:

- Users must authenticate before they can message
- No duplicate logins allowed
- IP blocked after 3 failed login attempts.
- Determine who else is currently logged in.
- Determine who else has been online in the last one hour.
- Send private messages to an online user
- Offline users receive messages sent to them once they log in.
- Broadcast message to all currently logged in users.
- Block/unblock a set of users
- Try to build a basic game playing interface if time permits

3 Assumptions

- Only basic authentication to be used without encrypting passwords.
- If a user is offline, any message addressed to them to be stored in a txt file
- Users would not indulge in providing inputs that differ a lot from the protocol specification.

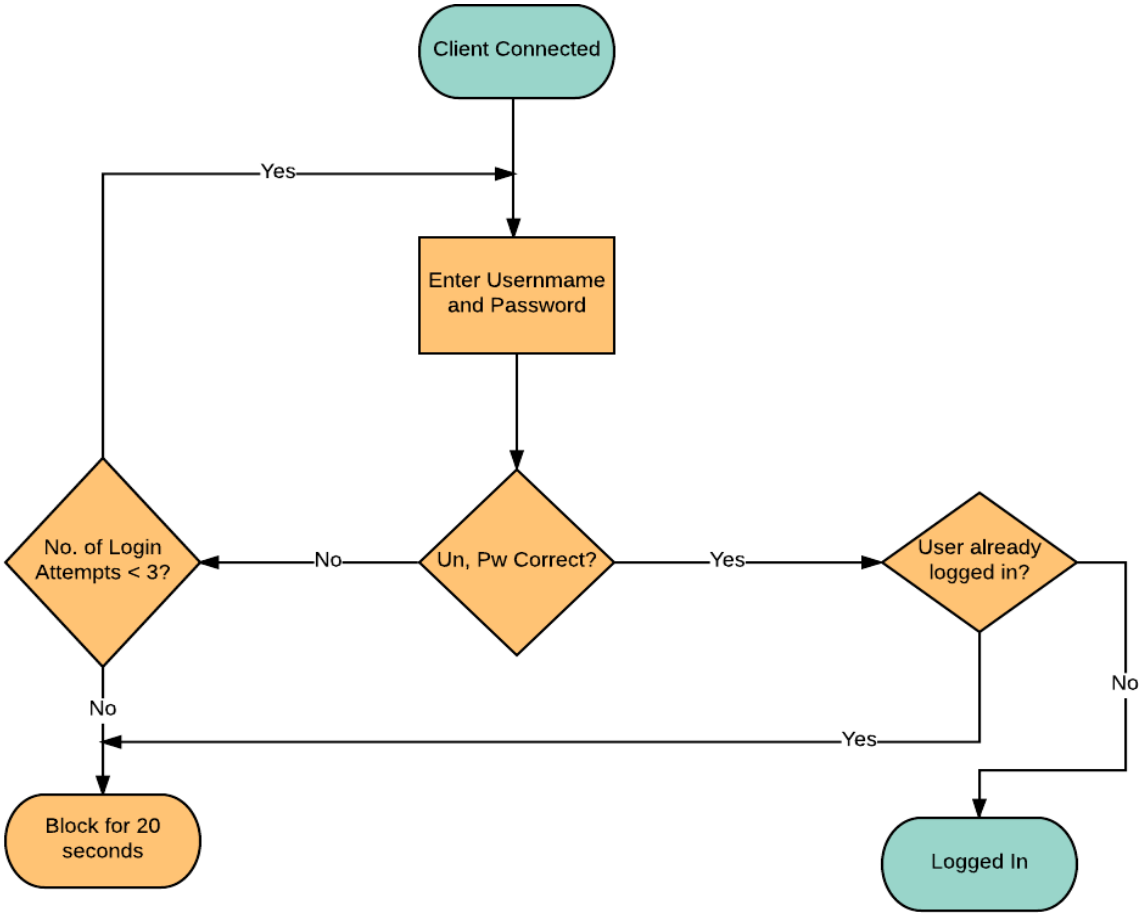

Figure 1: Login flow for chat client

4 Architecture

- Server starts
- Clients can connect using a username and the corresponding password. The client gets a maximum of three attempts to log in, after which their IP is blocked for 20 seconds. Duplicate logins from different or same machine are not allowed.
- Once logged in, the client receives a message from the server acknowledging the successful login. Also, it receives any messages that were sent to it while it was offline. It can then use one of the following commands:
  - *Un* > *msg* - Send message "msg" to usename "Un":
    - If username *Un* is online, the message is delivered.
    - If username *Un* is offline, the message is stored in a txt file, to be delivered later when *Un* comes online.
  - *broadcast* > *msg* - Message "msg" is delivered to all the users currently online.
  - *whoelse* - Displays the names of all the users currently logged in.
  - *wholasthr* - Displays the list of users who have been online in the last one hour.
  - *block Un* - puts username *Un* in this user's block list. *Un* can no longer send any message to the current user.
  - *unblock Un* - removes username "Un" from this user's block list. *Un* can now send messages to the current user again.
  - *playgame Un* - Request user *Un* if they would be willing to play a game of tic-tac-toe.
  - *playgame Un yes/no* - Accept or decline *Un's* invitation to play a game.

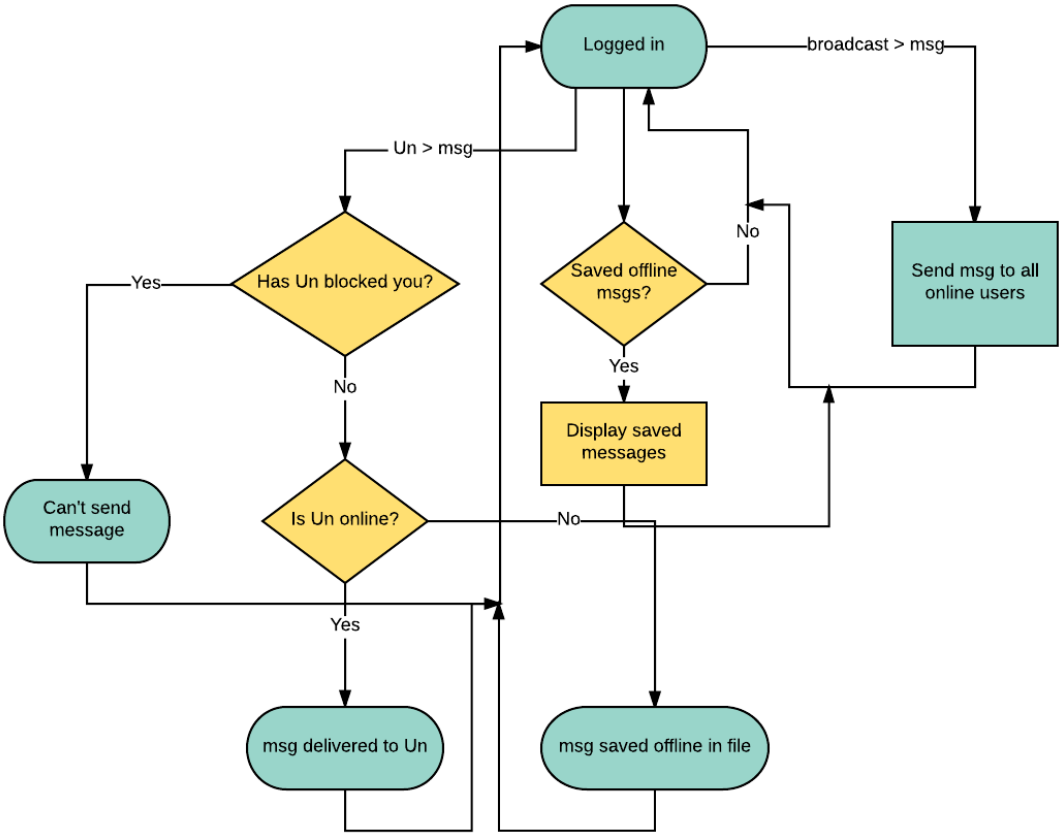

Figure 2: Messaging flow for chat client

– *move row col opponent* - In the current game against *opponent*, make a move (X or O, whichever corresponds to the current player) at (row, col).

5 Implementation Details

The chat client with game has been developed and tested on a Linux Machine. It uses Python 2.7 as its programming language and uses various Python libraries such as *socket, select, thread, os, numpy, time.*

6 Summary

All of the required features from our objectives have been implemented with a couple of drawbacks. Authentication is done by reading from a txt file which is quite unsafe. This can be easily remedied by using a secure database instead of simple txt file. Also unsafe messages can be easily sent to harm/hijack server.

We also went a step further and implemented a fully functional gaming interface that can interleave with the chat. A user can play multiple games at a time as well.

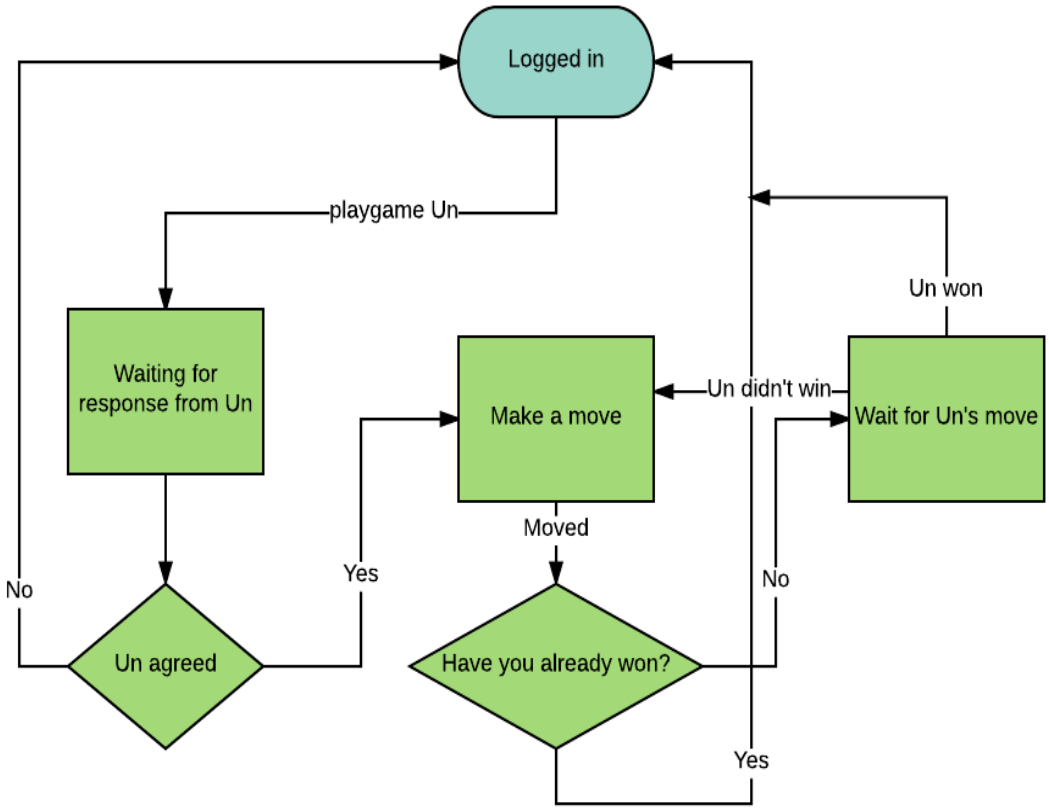

Figure 3: Flow for the tic tac toe game