COMP3234 Computer and Communication Networks / ELEC3443 Computer Networks

Programming Assignment: A Simple Game House Application

Total mark is 100.

I. OVERVIEW OF THE ASSIGNMENT

In this assignment, you will implement a simple game house application using Python socket programming. A game house is an application in which multiple clients connect to a game server, get authorized, and then select a game room to enter and play a game with another player in the same room.

The game house application consists of two parts: the server program and the client program. The server program should always be running and use a welcome socket to wait for connection requests from clients. The client programs establish TCP connections with the server program. After a connection is set up, the client needs to send its user name and password to the server, and can enter the game hall after successful authentication. An authenticated user is able to query the status of the game rooms, and then pick a room to enter. To start a game, there should be exactly two players in the same room. Therefore, if the entering player is the first one in the room, the player has to wait; otherwise, the game starts. After the game has started, the server generates a random boolean value and each player is invited to guess the boolean value; the player who guesses the same as the randomly generated value is the winner, and the game results in a tie if the two players' guesses are the same. After notifying both players the game result, the game is over and both players return to the game hall. A user may leave the system when the user is in the game hall.

II. DETAILED DESIGN OF THE APPLICATION

A. Server Basics

The server program takes 2 parameters: (1) server's listening port; (2) path to a *UserInfo*.txt file, which contains user names and passwords for all users (clients) that may participate in the application.

Upon start, the server program listens at the specified port. Upon receipt of a TCP connection request from a client, the server creates a new socket, and a new thread that handles the socket dedicated to this client. The main server thread then continues listening on the port.

Each line of the *UserInfo*.txt file contains the user name and password of one user, in the following format:

user_name:password
(Note: user_name and password should be replaced by the actual user name and password.)

Upon start, the server reads all users' information from the .txt into its memory, and authenticates any new joining client using the information. The server is also responsible to maintain the list of players in each room.

B. Client Basics

The client program takes 2 parameters: (1) hostname or IP address of the server; (2) listening port of the server.

Upon start, the client connects to the server at the IP and port by setting up a TCP connection. After the connection is successfully established, the client program goes to the stage of "user authentication" described as follows.

1. User Authentication

The client prompts its user the following on its terminal:

Please input your user name:

and then waits for the user to input the user name. After the user has input the user name, the client prompts:

Please input your password:

and then waits for the input of the password. After the user has input its password, the client sends the user name and password to the server, with a message in the following format:

/login user_name password

(Note: user_name and password should be replaced by the actual user name and password.)

If the authentication at the server is successful, the server thread (which handles the user's connection) sends the following message back to the client, and goes to the stage of "in the game hall" described next.

1001 Authentication successful

Otherwise, the server responds with the following message and the authentication procedure is repeated (*i.e.*, the client should prompt the user to enter name and password again):

1002 Authentication failed

2. In the Game Hall

After successful authentication, the user can enter the following commands:

/list

/enter target_room_number

1) /list
When the user is in the game hall, it can send this command to query the number of users in each game room. Upon receiving a /list command from a client, the server responds with a status code and the number of players in all the game rooms in the following format:

3001 number_of_all_rooms number_of_players_in_room$_1$ ... number_of_players_in_room$_n$

(Note: numbers should be separated by a space; index of the rooms starts from 1.)

2) /enter

Upon receiving a /enter command from a client (e.g., "/enter 2" to enter room 2), the server first checks if the target room number is valid or not. If it is valid, the server checks if the target room is empty or not. If the room is empty, the server responds with the following message:

3011 Wait

Upon receiving the above message, the client waits for the server to send the message with status code 3012.

If the target room is not empty, *i.e.*, there is another user waiting, the server notifies the waiting user and the currently entering user with the following message and then enters the stage of "playing a game" described next.

3012 Game started. Please guess true or false

If the room has already been occupied by two players, the entering player will receive the following message from the server and stay in the game hall:

3013 The room is full

3. Playing a Game

Upon receiving a message 3012 from the server, the client waits for the user to enter a boolean value, i.e., true or false, and then sends the input to the server via the following message:

/guess true_or_false
(Note: true_or_false should be replaced by the entered true (or false).)

Upon receiving /guess messages from both of the players, if their guesses are the same, the result is a tie, and the server sends the following message to both players:

3023 The result is a tie

Otherwise, the server generates a random Boolean value, r. The player whose guess is r becomes the winner, and the other loses the game. The server sends the following the message to the winner:

3021 You are the winner

The server sends the following message to the loser:
3022 You lost this game

After receiving any of the above messages, the game ends and the states of both players return to "in the game hall".

4. Exit from the System

When in the hall, the player can input the command /exit to quit the system.

/exit

Upon receiving a /exit command from the client, the server responds with

4001 Bye bye

Upon receiving the response from the server, the client program closes its socket and then quits.

Whenever a player exits from the system, the server thread (that handles the user's connection) closes the corresponding socket and then ends.

5. Dealing with incorrect message format

If the server receives a command message in the wrong format, it simply responds with

4002 Unrecognized message

6. Summary of the States

The states of a game player and the commands/actions invoking the state transition are illustrated in Fig. 1.

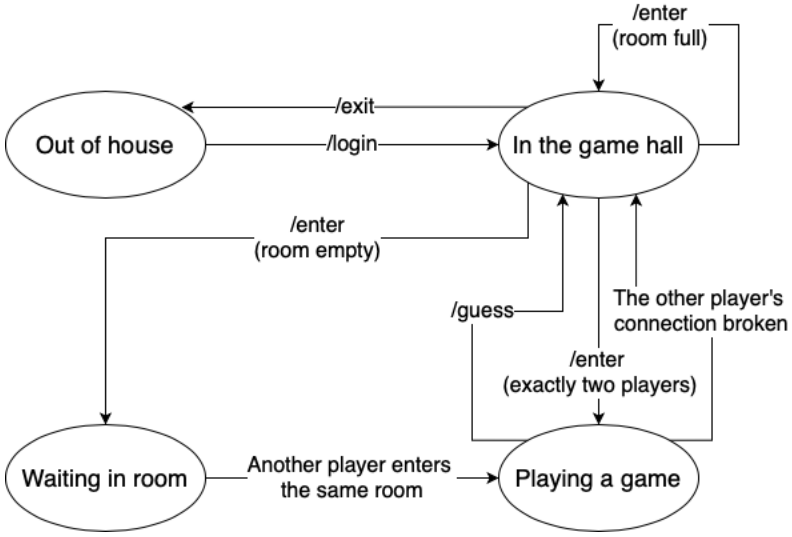

Fig. 1. State transition of a player.

For better understanding of the interactions between clients and the server, please refer to example interactions given in the Appendix.

III. IMPLEMENTATION NOTES

1. You will find the traceback module (https://docs.python.org/3.8/library/traceback.html) useful for printing out stack trace for debugging purpose.

2. You program should have sufficient robustness. In the networked environment, a TCP connection may be broken, or a remote end host may suddenly leave the system at any step of the dialog. Your program should be able to properly handle the exceptions thrown, print out error messages accordingly and end the client program or server thread accordingly. If the exception happens when two players are playing in a room such that one player is disconnected, the server should notify the remaining player that it is the winner and return it to the game hall.

3. You will need a state variable remembering which state each client is in, in the server thread handling this client's connection

4. You can create a room member list including players in different game rooms. Multiple server threads may try to access the room member list when processing commands /list, /enter, /guess, and when adding and removing players to and from the list. So your server thread needs to lock the list whenever it tries to access it, and release the lock afterwards. You will find Python threading lock class useful: https://docs.python.org/3.8/library/threading.html#threading.Lock.

In a server thread handling one client, to notify the other player when playing a game, make use of the other player's instance stored in the room member list.

5. You can assume a user can only exit when in the game hall, but not when waiting in a game room or playing a game (such that your client program only needs one thread for implementation).

6. You can specify the total number of game rooms in your program.

7. You can assume that user names and passwords do not consist of space(s).

8. You can assume one user does not log in at two or more terminals simultaneously.

9. Programming language and platform

- You should write your program using Python. You can implement it on Windows, Linux, Mac, etc., with any development environment that you prefer. You can use any Python modules that you deem useful to implement your program.
- Please pay attention to your programming style, and add comments wherever suitable for better readability.

IV. PROGRAM TESTING

We will test your programs using normal interaction cases like those described in the Appendix, and with various abnormal cases, such as incorrect message formats, broken network connections, unexpected crash of the server or the client program, etc. Therefore, you should test your program in similar fashion during development.

V. GROUP OR INDIVIDUAL

You should work on this assignment individually, or in a group with at most two students.

You will get 10% bonus mark if you work on the assignment individually, i.e., your mark will be your original mark * 1.1, upper bounded by the full mark of 100.

VI. SUBMISSION

You should submit the following files in a zip file, named as a1-yourstudentid.zip (individual work) or a1-studentid1-studentid2.zip (group work; only one submission by one of the group members is needed for each group):

(1) GameServer.py

(2) GameClient.py

(3) A readme file to provide any additional information on the implementation and execution of your programs that you deem necessary for us to know.

Please submit the .zip file on Moodle:

(1) Login Moodle.

(2) Find "Programming Assignment".

(3) Click "Add submission", browse your .zip file and save it. Done.

(4) You will receive an automatic confirmation email, if the submission was successful.

(5) You can "Edit submission" to your already submitted file, but ONLY before the deadline.

APPENDIX

Sample outputs on client terminals when Jacky, Bob and Michael interact with the same server:

```
cwus-MacBook-Pro:a1 cwu$ python3 GameClient.py localhost 22222
Please input your user name:
Jacky
Please input your password:
12345
1001 Authentication successful
/list
3001 10 0 0 0 0 0 0 0 0 0
/enter 1
3011 Wait
3012 Game started. Please guess true or false
/guess true
3023 The result is a tie
/exit
4001 Bye bye
Client ends
cwus-MacBook-Pro:a1 cwu$
```

```
cwus-MacBook-Pro:a1 cwu$ python3 GameClient.py localhost 22222
Please input your user name:
Bob
Please input your password:
12345
1002 Authentication failed
Please input your user name:
Bob
Please input your password:
54321
1001 Authentication successful
/enter 1
3012 Game started. Please guess true or false
true
4002 Unrecognized message
/guess true
3023 The result is a tie
/list
3001 10 0 1 0 0 0 0 0 0 0 0
/enter 2
3012 Game started. Please guess true or false
/guess false
3021 You are the winner
/exit
4001 Bye bye
Client ends
cwus-MacBook-Pro:a1 cwu$
```

```
cwus-MacBook-Pro:a1 cwu$ python3 GameClient.py localhost 22222
Please input your user name:
Michael
Please input your password:
abcde
1001 Authentication successful
/list
3001 10 2 0 0 0 0 0 0 0 0 0
/enter 1
3013 The room is full
/enter 2
3011 Wait
3012 Game started. Please guess true or false
/guess true
3022 You lost this game
/exit
4001 Bye bye
Client ends
cwus-MacBook-Pro:a1 cwu$
```