BLG433E - Computer Communication
Socket Programming Assignment

Client – Server Communication with a Simple Authentication Protocol

In the given assignment, you are required to write **Python (v3) code** for a client program and establish successful communication with the server, which runs on the IP **160.75.154.126**. Client program should contain two distinct phases in order to execute the communication successfully. First, it should authenticate itself to the server and then server is going to send a series of questions in the context of "Guess the Word" game and client program should provide the sufficient interface to a user to play the game. Details of the implementation are given below.

Part 1. Authentication Mechanism

An e-mail, which contains a unique ***hexKey*** has been sent to each student. In the following authentication mechanism, every student should use her/his own unique in order to authenticate herself/himself. Before further detail about the mechanism, we will briefly describe the main purpose of the authentication protocol. (Note: If you have not received your key information, please contact T.A.s Talip Tolga Sari – sarita@itu.edu.tr)

Preliminary Information about Authentication

The main objective of an authentication is to prevent anyone from imitating you by using your unique information. Simply assigning unique IDs and password for users to be used in authentication is not an efficient solution for computer networks. Because, anyone who listens (or shares) the same communication medium could obtain your id and password information to use them later in order to imitate you. Thus, in any secure communication protocol, users should avoid transmitting their critical information without an encryption mechanism to keep their identities safe.

With the given motivation described above, you need to avoid sending your unique ***hexKey*** plainly over the network. To encrypt your unique key, ***sha1*** hashing mechanism is going to be used in this project. You may use any method to do sha1 in Python using any library. Furthermore, you could use the following website to test your implementation http://www.sha1.cz/.

In the defined protocol, first, you need to initialize a TCP connection with the **port** numbered **2022** of the server, which runs on **160.75.154.126**. Then, client program should send the following string to initiate the protocol "**Start_Connection**". After receiving the start command, the server is going to send a random hex string ***(randomHex)***. Client program is

going to concatenate its unique ***hexKey*** and the received ***randomHex*** to create a string of hex which contains 64 chars. This string will be used as the input of sha1 hashing function to create a hash of the data. The sha1 function is going to return a byte stream (**unsigned char** in **C**) with size of 20 bytes. Then, you should convert this byte stream into a hex stream by using the functions in sha1.cpp. Conversion to hex is going to expand the size of the stream to 40 chars. You are going to use this char array to authenticate yourself to the server. However, lastly, you should add your student Id to the end of the stream by using '#' symbol between them. Thus, you will send a char array to the server with the size of 50 chars (40 sha1 result + 1 '#' symbol + 9 student ID). If the calculation and the transmission are successful, server is going to send to a message which informs you that you have successfully authenticated yourself and ask you whether you wish to proceed the second part or not. After receiving this message, the authentication part of the project is completed. Figure 1 shows the interactions between the client and the server during the authentication phase.

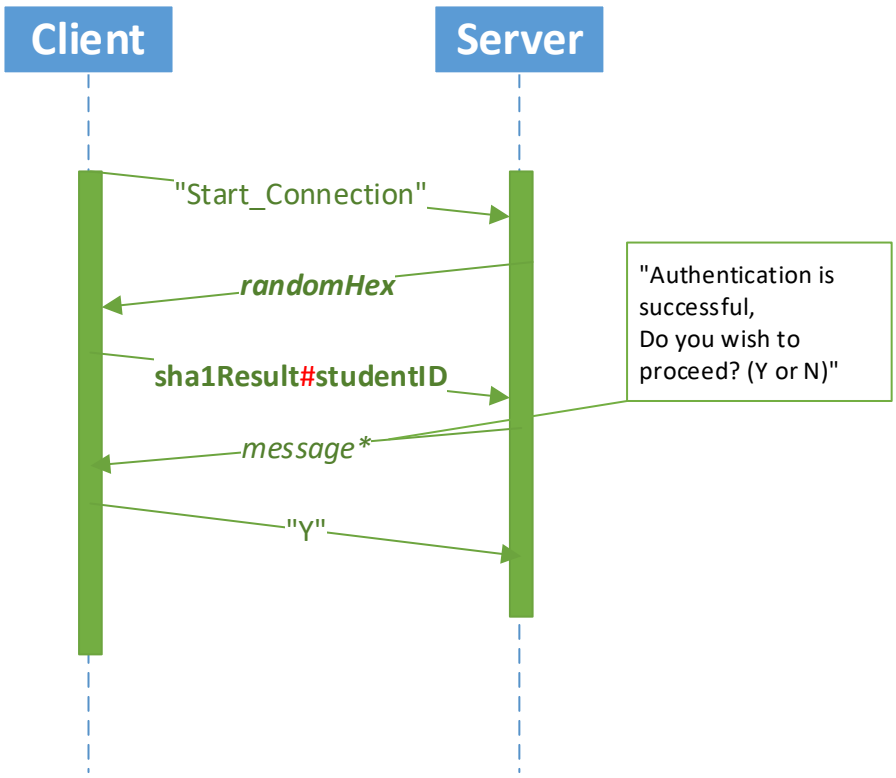


Figure 1: Interactions during the authentication

After, client sends "Y" to the server to notify the server to proceed, server is going to start the 2nd phase and start sending questions to the client.

Part 2. “Guess the Word” Game over TCP

In this part of the communication, the server is going to send 8 random questions, containing a formal definition of random words, and wait an answer or request a letter from the client. Once, the game has start client will have 300 seconds to fetch all question and answer them. Server will periodically send remaining time to the client and conclude the game when the time is up. Thus, as oppose to the first part of this assignment, the communication between peers (client and server) is **asynchronous**. The applications should follow a pre-defined format in order to successfully communicate with the server. Details and purposes of these packet formats are given bellows:

Client-side packets

Client application has 6 different instructions as listed:

- Start a Game
- Terminate the Game
- Fetch Next Question
- Buy a Letter
- Take a Guess
- Get Remaining Time

All of these instructions, except GUESS instruction, contain a single byte. The format of these instructions are given bellow:

Guess Instruction:

```
*0-----------------7------------------*
| Instruction_type |     Payload      |
|      . . .              . . .       |
*-------------------------------------*
# Instruction_type Field: <UInt8> 04: Take a guess
# Payload field: <char array>: Guessed Word
```

Others:

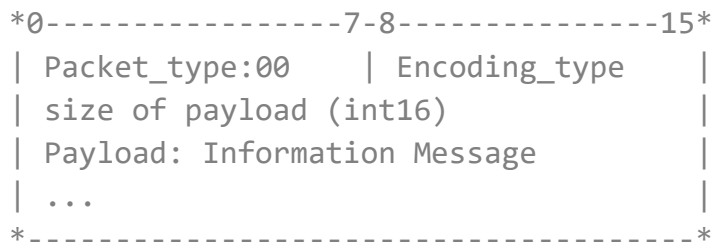
```
*0-----------------7*
|  Instruction_type  |
*--------------------*
# UInt8
00: Start Game
01: Terminate the game
02: Fetch the next question
03: Buy a letter
05: Get rem. time
```

Server-side packets

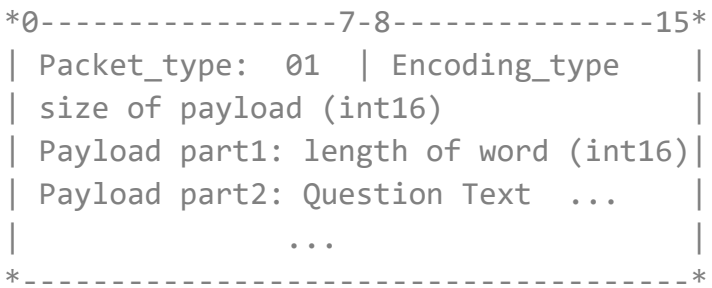
The server application has **5 types of message** related with the game. Your client application should be able to parse these messages and generate outputs for the players if necessary. Format and descriptions for these messages are listed below:

About* asynchronous *communication
However, as stated earlier, some of these messages could be sent asynchronously by the server without any incoming data event. For example, after game is started, server may send remaining time periodically without receiving any "Get Remaining Time" instruction from a client. So, any client application should be able to receive and send data simultaneously.

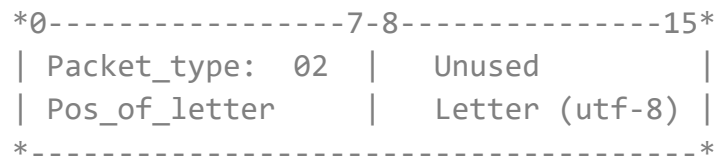
1. Information:
   This message type contains control information guiding you about the state of your connection. This is a control message, which are sent for the application developer, not for the players. So, your client application may/may not use them to create any output on UI.



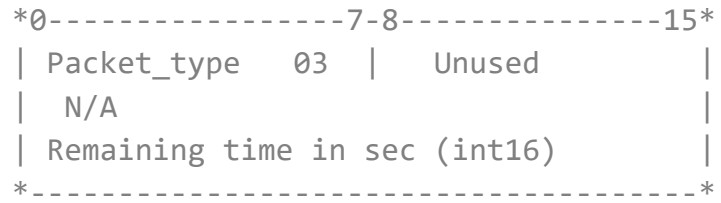
2. Question:
   This message contains the question, which is the definition of the word that should be guessed by the user. Having different encoding types available may change the size of the message for the same question. "Size of payload" field always states the number of characters in the definition, not the number bytes in the message.



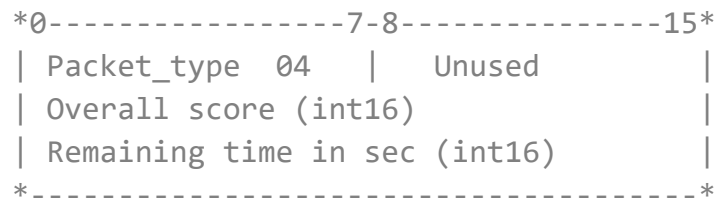
3. Letter from the word:
   Server sends a random letter from the word upon receiving “Buy a Letter” instruction using this message type. This message should be visualized by the client-side application when it is received, so the player can decide to buy another letter or take a guess.



4. Remaining time:
   Server uses this message format to update client about its remaining time in the game.



5. End of game
   This message will be send by the server, when the given time is up or upon receiving “Terminate the game” instruction.



About common fields:

```
Encoding_type: 00 for utf8, 01 for utf16
Fields containing int16: Contents of these fields are little-endian.
```

BONUS: Providing a simple GUI that separates incoming/outgoing data will earn bonus points up to 10.

Submission Deadline: 27.11.2022

Important Notes:

- You are expected to work individually on this homework. All forms of collaboration are discouraged and will be treated as plagiarism. This includes actions such as, but not limited to, submitting the work of others as one's own (even if in part and even with modifications) and copy/pasting from other resources (including Internet resources) even with proper reference. Such offenses are reported to the administration for disciplinary measures. All parties involved in the act will be treated equally.

- You should submit your homework through Ninova system. Late submissions are not accepted.

- Your code must be written in Python (minimum version 3.7).
- HINT: Use different threads for incoming/outgoing data.

Submission Guideline:

- You should submit homework file through Ninova
- If you have further questions about the assignment, you may contact the course assistant