ELEC 3120: Computer Communication Networks
Prof. Tsang, Danny H. K.
(Fall 2020)

Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology

Programming Assignment 2

Notes:

1) In your source code, please write down proper comments. You may discuss ideas with others in the class, but you should implement your own program. Do not copy from others.

2) Although Python 3 is preferred, you are welcome to use any programming language.

3) **Source code submission:** Please compress the source code and executable files into a ZIP file, with your "`student ID-operating system-programming language`", for example, "`20510000-Windows-Python3`", as the file name. Upload the ZIP file to the "programming assignment 2" field, under the "assignments" section of the ELEC3120 Canvas website. The deadline for source code submission is hard at **11 a.m. on December 21, 2020**, with no late submission acceptable.

Simple Chatroom

The chatroom works in client/server mode. When the chat server starts, it should wait for clients' requests and can support N users at the same time, i.e., the maximum number of users in the chatroom is N, and you can set N=3 for convenience. When the client wants to send a message, she will send the message to the server. Then the server will broadcast any messages from a client to all the other connected clients.

Detailed requirements:

1. You are required to implement all the chatroom functions with both TCP and UDP. Each protocol takes up 5% of the final grades.
2. Chatroom functions:
    a. (User entering) When a client connects to the server, she will receive a greeting message from the server: "Welcome to the chatroom! Please choose a username:" After the client sends the selected username, the server should check if the username has been taken, if it is available, a notification message, "<username> has entered the chatroom…" needs to be printed at the server side and at any other clients' side, otherwise, the server replies to the client: "Sorry, this username has been taken, consider another one?".
    b. (Sending message) When a client sends a message to the server, the message needs to be printed at the server side and shown at any other online clients' side in real time, formatted as: "<username> message content". You are not required to print out the history messages for a user coming later, but you are encouraged to have a try if you are interested.
    c. (Server size) The server should be able to support no more than N users. When the server receives the request from the (N+1)-th user, the server will decline the connection request and show "<x,y> was blocked" at the server side, where x, y are the IP address and the port number of the rejected client, respectively. You are not required to print out a polite error message at the rejected client side, but you are encouraged to have a try if you are interested.
3. (Hint) For TCP, since it is a connection-based protocol, you should implement the function with multi-threading, which means every time a user connects to the server, a separate thread will be created for her at the server side. Once the server receives a message from a thread, he will send that message to all the other connected threads. For UDP, for displaying the messages in real time, we allow for several programs running for a single client,
or you can turn to multithreading.

Example:
Client 1,2,3,4 come sequentially.

At server side (N=3):

John has entered the chatroom…
<John> Hi everyone, my name is John!
Alisa has entered the chatroom…
<Alisa> Hi, my name is Alisa.
Mary has entered the chatroom…
<Mary> Aloha! This is Mary.
<John> Hi Alisa and Mary, how is it going?
<Alisa> Very well :)
<x, y> was blocked.

At client1 side:

Welcome to the chatroom! Please choose a username:
- John
  Welcome, John! Now you can start sending messages!
- Hi everyone, my name is John!
  Alisa has entered the chatroom…
  <Alisa> Hi, my name is Alisa!
  Mary has entered the chatroom…
  <Mary> Aloha! This is Mary.
- Hi Alisa and Mary, how is it going?
  <Alisa> Very well :)

At client2 side:

Welcome to the chatroom! Please choose a username:
- Alisa
  Welcome, Alisa! Now you can start sending messages!
- Hi, my name is Alisa!
  Mary has entered the chatroom…
  <Mary> Aloha! This is Mary.
  <John> Hi Alisa and Mary, how is it going?
- Very well :)

At client3 side:

Welcome to the chatroom! Please choose a username:
- Alisa
  Sorry, this username has been taken, consider another one?

- Mary
  Welcome, Mary! Now you can start sending messages!
- Aloha! This is Mary.
  <John> Hi Alisa and Mary, how is it going?
  <Alisa> Very well :)

Client4 with IP and port number <x, y> tries to connect but cannot.

Here are some useful references to use multithread in socket programming with Python:

1. https://docs.python.org/3.7/library/threading.html#
2. https://www.geeksforgeeks.org/multithreading-python-set-1/
3. https://realpython.com/intro-to-python-threading/

Notes: Although multithreading is a concept we don't cover in the lectures, it is surprisingly simple to understand. As shown in the examples of the references, to use multithreading in Python, you can do it in one or two line of codes.