Tcp/udp socket forum

Introduction and program design

First, I would like to introduce the results of my homework. I did this assignment using Python3. According to the homework manual, we need to create a multi-threaded online discussion forum. The server and client should communicate using TCP/UDP.

I need to implement the corresponding functions according to the requirements, and can upload and download files.

Language: python3.8
Libraries that come with python:
socket
threading
sys, select
os
collections

Run steps:(for example)

python3 TCP_UDP_Server3.py 10005

```
python3 TCP_UDP_Client3.py 127.0.0.1 10005
```

Because I am more familiar with python, maybe I like it better, because it is easy to learn: Python language is a relatively easy to learn programming language compared to other programming languages, it focuses on how to solve problems rather than programming language syntax and structure. Graceful: The Python language strives to be concise and beautiful in code.

Our basic design idea is: first, the debugging is completed in the case of socket single thread, and then the functions that can be realized by udp are divided into functions to realize, and tcp can do it (such as file uploading and downloading) This part is left to tcp socket is implemented. Then consider the case of porting from threading to multithreading.

I am basically passing commands from client to server. Then make a judgment in the Server. In terms of time, the implementation of DLT and EDT is laborious. For MSG, I created an auxiliary related function and structure using a while loop. The first string of each line is a number, the message number plus 1 step. For DLT and EDT, I think the difficulty lies in matching the CRUD of the corresponding message number.

---

layer protocol and how your system works

This instant messaging application includes 2 major modules, the server program and the client program : client and server (the server extracts common codes and functions to Server_Common) and must use UDP and TCP to communicate.

Our engineering includes creating new user accounts, creating and deleting new threads, posting messages on threads, editing or deleting messages, uploading and downloading files to/from threads, reading threads and listing all threads. Most of the communication between the client and the server will be carried out through UDP, but more suitable for the stable communication function of TCP, we still use TCP for file upload and download. The server will listen on the port specified as a command line argument and wait for messages from clients. When the user executes the client, the authentication process will be started, mainly through the user name + " " + password for comparison.

Some working programing's snapchat:

This is credential check:

```
-----------------------------
===== Please type any messsage you want to send to server: =====

Enter username: test
Enter password: test
Welcome to the forum
===== Please type any messsage you want to send to server: =====


upd server get: b'UDP client send msg!'

===== Server is running =====
===== Waiting for connection request from clients...=====
Waiting for clients
test successful login
```

This is command : CRT is working

```
===== Server is running =====
===== Waiting for connection request from clients...=====
Waiting for clients
test successful login
test issued CRT command
Thread thread0 created
 10006
------------------------------
Traceback (most recent call last):
  File "TCP_UDP_Client3.py", line 194, in <module>
    main()
  File "TCP_UDP_Client3.py", line 150, in main
    clientSocket.connect(serverAddress)
ConnectionRefusedError: [Errno 61] Connection refused
(samaritan0) abel@AbeldeMBP b19313_socket % python3 TCP_UDP_Client3.py 127.0.0.1
 10006
------------------------------
===== Please type any messsage you want to send to server: =====

Enter username: test
Enter password: test
Welcome to the forum
===== Please type any messsage you want to send to server: =====

Enter one of the following commands: CRT, MSG, DLT, EDT, LST, RDT, UPD, DWN, RMV
, XIT:CRT thread0
Thread thread0 created
Enter one of the following commands: CRT, MSG, DLT, EDT, LST, RDT, UPD, DWN, RMV
, XIT:
```

Project design Trade-offs considered

Function realization and Trade-offs :

Most of the features on the list are implemented(CRT, MSG, DLT, EDT, LST, RDT, UPD, DWN, RMV, XIT), except 2 features:

Successful authentication of multiple concurrent existing and new users including all error handing. This will take more time to tune up, so I'll put it on hold for now.

Implementation of mechanisms to recover from occasional loss of 1 1 UDP segments , I am not very familiar with UDP segments, I think I may not done it this time, I 'll put it on hold for now.

not work under certain circumstances:

Our programming on cmd executing is not working all the time: when you use "EDT" Cmd, you should first CRT cmd to make thread, and have not delete "DLT" it, otherwise in some circumstance , programming may not work.

Refer

Socket usage : https://docs.python.org/3/library/socket.html

Ups and tcp in python programming : https://stackoverflow.com/questions/27893804/udp-client-server-socket-in-python

Orignial teacher 's TCPClient3.py and TCPServer3.py code