COMP3331 Assignment Report for Term 3, 2020

This report is written by Yip Jeremy Chung Lum, z5098112.

Program Design

Structure:

- client folder where client resides
  - client.py (client entry file)
- server folder where server resides
  - server.py (server entry file)
  - credentials.txt (contain username and passwords of authorised users)

Application Layer Message Format

Clients and server communicate over TCP. Messages are communicating with json format.

The basic structure of a message is built up with action and status. An action is just a field to indicate what phase is the message currently in. A status would store the detail information of the current phase.

An action only has three phases, which are Login, Command, and Shutdown.

Login phase is just indicating the client is still sending username and password to the server.

Following a successful login, it will switch to the Command phase. The user will now initiate one of the available commands (E.g. CRT, LST, MSG, DLT, etc).

The last one is the Shutdown phase. It only occurs when a user has successfully sent a SHT command. In other words, when the server is initiating a shutdown process, action will be change to the Shutdown phase.

An example for the status field, when action is set to Login, the server will reply a message with status field to tell the client does the username exist or not. If it exists, status will hold a string USER_EXIST, so that the client will send the password to the server. In the other hand, NEW_USER will be stored in status if the username could not be found in the credentials.txt file, so the user will now create a new account instead of login with an existing one.

Another example, during the Command phase, when a user sent a LST command, the server will either reply THREAD_ACTIVE or THREAD_INACTIVE in the status.

COMP3331 Assignment Report for Term 3, 2020

How do the system works

Command to start up a server:

```
cd server
python3 server.py server_port admin_password
```

Command to start up a client:

```
cd client
python3 client.py server_IP server_port
```

After the server start up, it will listen to every new connection and create two threads for it. One thread will listen to every incoming data requested by the client. All the messages will be processed within the connection_handler() function and an appropriate message will be sent back to the client immediately. The second thread will handle with the Shutdown phase.

The server will keep on waiting for clients to connect, but once a SHT command is received, it will send a shutdown message (i.e. action: Shutdown) to all active clients, which is processed by the second thread. It then removes all files that were created by the server program in the current working directory, close all sockets and shutdown the server itself.

After a client start up, it will initiate a TCP connection with the server and create two threads. One for displaying the server incoming messages, and another for handling the user input (i.e. action: Command) and send the messages to the server.

Design trade-offs

One of the trade-offs is to use multi-processing or multi-threading.

Multi threads were chosen because it can run faster on computer systems with multiple CPUs, since theses threads can be executed truly concurrent. Threads of a process can also share the memory of the global variables. Therefore, it helps to maintain the connection between multiple clients and the server.

Another trade-off is format of the message was chosen to be JSON because it is human readable and easy to move back between container and value.

COMP3331 Assignment Report for Term 3, 2020

Possible Improvements

The server currently uses the time.sleep() function to suspends execution for 0.1 seconds. In other words, the server checks for input in every client thread in a while loop without pausing and updates users every 0.1 second. It is in fact wasting many resources. Using time.sleep() may also impact the latency and CPU load.

A better implementation may be the server only updates whenever there is a new event, instead of looping infinitely, also known as event-driven programming.

Extensions to the Program

In this assignment, p2p message is implemented.

Particular circumstances for the program

This code was written under the CSE virtual lab. Hence, the program works well on CSE machine.

Sometimes the sever port may be occupied by someone else, then you can try a different port.

References of the Codes

- server.py and client.py is implemented based on the Multi-thread Code (Python) provided by COMP3331 assignment page.
- The TCP connection implementation is based on the TCPServer(python3).py and TCPClient(python3).py given by COMP3331 lecture page.