Pritan Barai z5309086

COMP3331/9331 Computer Networks and Applications: Assignment for Term 3, 2022

- **Language and Platform: Python 3.9**

Program Design

Client:

Each client runs independently from each other in their own folders simulating file transfer over the same separate terminals. The client will establish a TCP connection to the server and a UDP server capable of sending and receiving files from other edge devices.

The client should maintain at least two threads while active. One being a thread running the TCP connection to the server, processing the commands made to the client by the user. The other thread should initiate the UDP server running on the client for P2P communication.

Server:

The server uses a multi-thread class for the client, where for each connection request by a client, a new Thread is created allowing for multi-threading. The class will contain functions to login and process each command specified by the edge device.

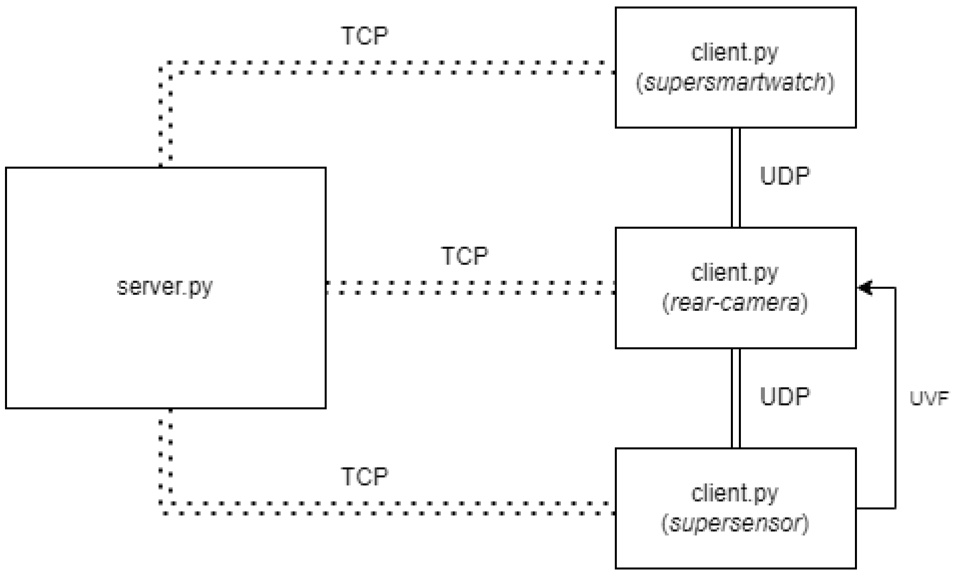
The server should retain a record of the failed consecutive login attempts for each edge device. When the number of failed attempts equals the servers defined value for unsuccessful authentication attempts, the device should be blocked for 10 seconds.

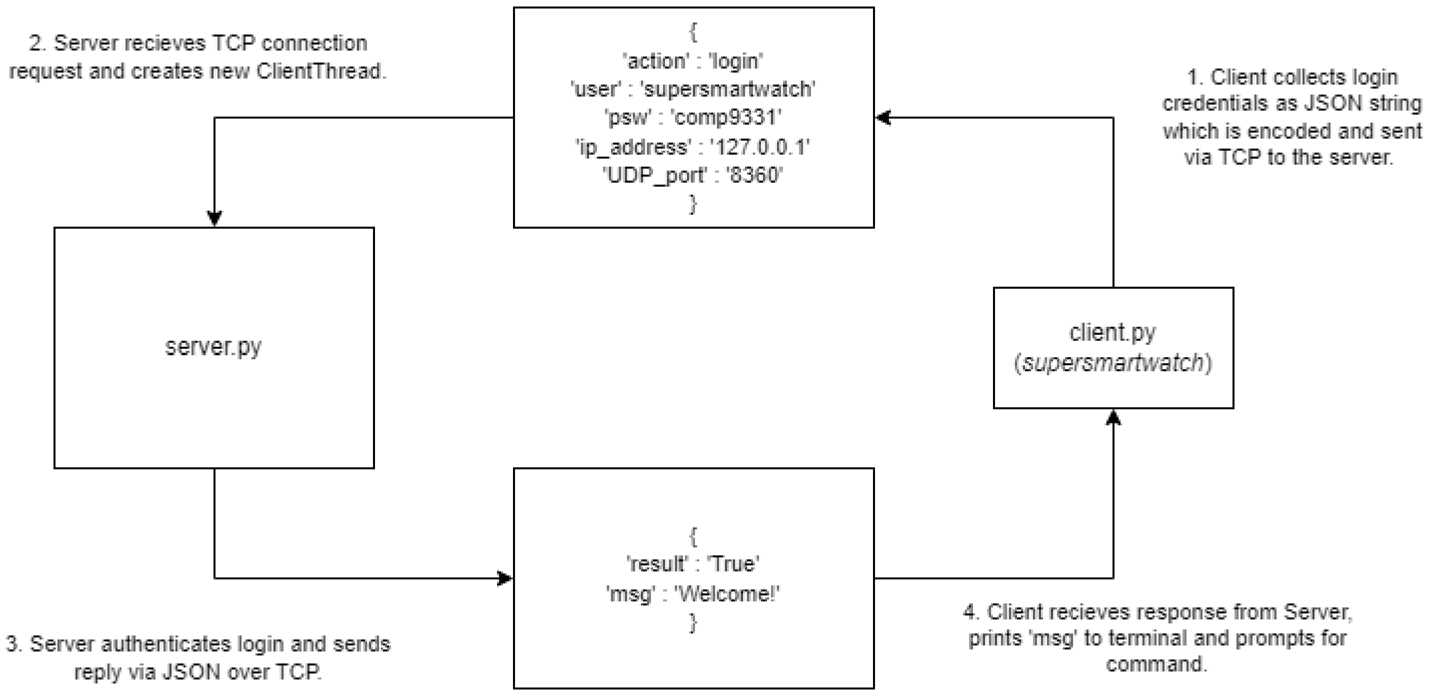
Application Layer Message Format

The message format used by the client references the example output given in the assignment spec.

The server outputs a log of new connections made, the received and sent requests made to the server and the reply sent to the client.

System Model




2. Server recieves TCP connection request and creates new ClientThread.
{
'action' : 'login'
'user' : 'supersmartwatch'
'psw' : 'comp9331'
'ip_address' : '127.0.0.1'
'UDP_port' : '8360'
}
1. Client collects login credentials as JSON string which is encoded and sent via TCP to the server.
server.py
client.py
(*supersmartwatch*)
{
'result' : 'True'
'msg' : 'Welcome!'
}
3. Server authenticates login and sends reply via JSON over TCP.
4. Client recieves response from Server, prints 'msg' to terminal and prompts for command.

How My System Is Run:

I have tested my assignment using 3 folders. One folder containing server,py and the files created by it, and two folders representing edge devices as client1 & client2 respectively. The client and server codes are run in separate terminals, with each edge device running client.py in their respective folder.

Design Trade-offs

I have assumed that during UVF, the video file will transfer completely with no packet loss. As there is no reliability implemented in UDP, I have added a simple check if the size of the file after being transferred by UVF is equal to the expected value.

Similarly, for the sake of simplicity, I have not created a UDP server class for client.py, as unlike server.py only one other thread is required for receiving UVF and have instead opted to use start_new_thread function from _thread.

Improvements & Extensions

I believe I could improve the organisation of the client initialisation and handling of authentication to streamline the code and improve readability. Likewise, I would have liked to extend the UVF command to correct the output if the client was issuing a command. Furthermore, I believe the server's command functions have a high degree of similarity between them, which could have been made more efficient and reduced the dependency on repeated code.

References

- assignment demo week7; RUI LI; https://www.youtube.com/watch?v=kPAz8w62wTI&t=2134s&ab_channel=RUILI
- Geeks for Geeks: Socket programming multi-threading python https://www.geeksforgeeks.org/socket-programming-multi-threading-python/