Christopher Sweet 05/03/2018

1. Explain which transport layer protocol you used to implement FS and FC. Why?

   TCP was selected as the transport layer protocol. This was selected because of the requirements for a FTP server: Files must be transfered from the server to the client and vice-a-versa. This must be done in order, and packets cannot be dropped, two things that TCP ensures. Consider a counter example of an FTP server implemented using UDP. There would be no guarentee that all of the requested data would be transferred, nor that the files would be reconstructed in the correct order at the destination.

2. Explain how did you achieve concurrency? Please show your work.

   Concurrency was achieved by using python's threading library. All instances of get or push commands made use of classes inheritting from threading.Thread, allowing the class to be instantiated into it's own thread. The thread.start() command was then run to initialize the thread's operation. An additional parameter to the client class was added, titled cookie. This served to store the login credentials for each client to allow connections to only be maintained during file transfers. Servers created Server threads when a connection was created. These in turn spawn Put and Get Threads to handle the actual operations between the server and client. Figure 1 shown below illustrates this.

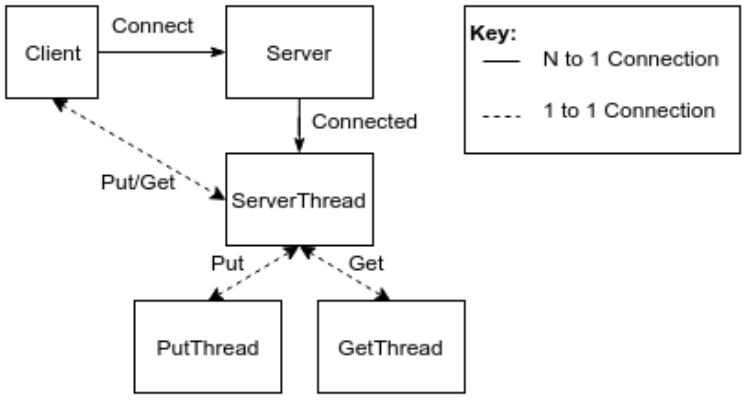

Figure 1: Server and Client Relationships

3. May you have synchronization problems in your design? If yes please explain how you resolved that in the implementation?

   Synchronization issues could occur in the event that two clients requested a put and a get command on the same file at the same time. Specifically, if one client requested a file as the other client were in the process of uploading a new copy of that file to the server a simultaneous access error would occur. In order to correct this files are uploaded and downloaded with a _server or _local appended to the filename. In a future update the sever could be designed to remove the old file and rename the swap file to the original filename.

4. Provide a sample run to demonstrate the execution of each command. Your program is expected to inform the user with either an error or status message after each command.

   Usage of this program requires python 3. Two terminal windows will be required, one to start the server and one to run the client. N many clients could be used if so chosen. All commands are given assuming you are running on a linux machine. Commands with the wrong number of arguments will print a USAGE message. Incorrectly entered commands or blank commands will terminate the connection.

Server Terminal:

```
$ python3 server.py
<Will list connections as they come in>
```

Client Terminal 1:

```
$ python3 client.py
<Enter an FTP command>
$ rftp localhost chras pa55word
<Enter an FTP command>
$ rput passwords_plain.txt
<Done sending data>
<Enter an FTP command>
$ rget passwords_plain.txt
<Downloaded File>
<Enter an FTP command>
```

Now start up a new Client:

Client Terminal 2:

```
$ python3 client.py
<Enter an FTP command>
$ rftp localhost chras pa55word
<Enter an FTP command>
$ rget passwords_plain.txt
<Downloaded File>
<Enter an FTP command>
$ rput passwords_plain.txt
<Done sending data>
<Enter an FTP command>
```

Feel free to go back and forth between the two clients as desired!