Lab 3: FTP and Tunneling

Due date: June 4, 2023 (Anywhere on Earth, i.e., June 5, 5 AM PDT)

TA Contact Info: Stan Lyakhov, lyakhovs@oregonstate.edu

The goal of this lab is to implement a simplified version of FTP (File Transfer Protocol) and use tunneling to be able to download/upload files between two laptops connected through an SSH tunnel.
All network code **must** be written with Python asyncio sockets. Feel free to use *async_server.py* and *async_client.py* from last lab as reference.

1. **Preliminary Setup** There's a few preparations we need to make on the client/server side.

   (a) While you will eventually be running this on separate computers, we want to create a setup that allows you to test the code on the same computer while you're still developing it.

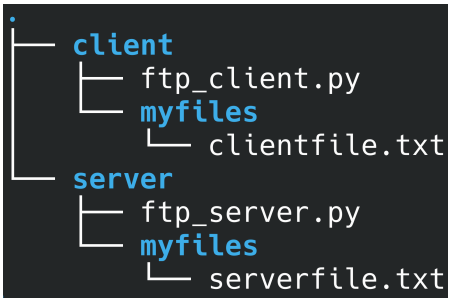
   (b) Create two directories (folders): "server" and "client".

   (c) Inside the "server" folder, create a python file "ftp_server.py". Inside the "client" folder, create a python file "ftp_client.py".

   (d) Inside both the "server" and "client" directory, create another directory called "myfiles".

   (e) Create some test files in "myfiles" for both the client and server to test your FTP functionality. Make sure to put some contents inside the files (can be anything you want). See an example file structure below.

   (f) In both your client and server programs put "import os" at the top of the file and then "os.chdir('myfiles')" right below it. Now you should be able to read/write files inside the "myfiles" directory easily without worrying about the fact that the python file is not inside that directory.



2. **FTP Protocol: Initializing Connection** There's a few steps we have to make before starting FTP functionaity.

   (a) When the client makes a connection to the server, the server should prompt it to enter a password. This password should be set ahead of time on the server.

   (b) The server should tell the client each time if the password is correct: after 3 incorrect tries, the server should close the connection. See Part 3 for the overview of how the server should communication any errors (including password errors).

   (c) If the login is successful, the server should respond with the introduction message (just like in the previous labs).

3. **FTP Protocol: Overview** Our FTP server/client will always upload/download things between the server's and client's "myfiles" directories. No other files in other directories should be modified by the server/client!

   (a) The client/server must support the following commands:
      i. "list": list the files in the server's directory
      ii. "put *file*": uploads *file* from the client to the server
      iii. "get *file*": downloads *file* from the server to the client
      iv. "remove *file*": remove *file* that's on the server
      v. "close": close the connection to the server

   (b) A significant portion of this lab involves handling errors: what should the server do if the client decides to download a file that doesn't exist?
      i. When a command can be executed, the server should respond with "ACK" and then carry on executing the command (in the case of the "list" command, the server would respond with "ACK" and then the list of files).
      ii. When a command sent by the client cannot be executed (e.g. downloading a file that doesn't exist), the server should respond with: "NAK *errorMessage*" where *errorMessage* should explain the type of error that occurred. The client should make sure that the user sees this message! A similar process should happen for password checking in Part 2.
      iii. Some errors the client should handle on its own. For example, if the user tries to us "put *file*" where *file* doesn't exist, the client should tell this to the user without having to communicate with the server.

   (c) Tips for implementation of the commands
      i. For all parts of the lab, you should make heavy use of "receive_long_message()" and "send_long_message()" from the previous labs. It helps the client/server agree on the length of the message to be sent/received.
      ii. For "list" you can use "os.listdir".
      See https://docs.python.org/3/library/os.html#os.listdir
      For "remove" you can use "os.remove".
      See https://docs.python.org/3/library/os.html#os.remove.
      iii. For "put" and "get", the client/server respectively should read the file and send its contents over the network. Then the server/client should write these contents into a new file.
      iv. For "close", both the server and client should just close their respective "writers", which just closes the connection.

4. **FTP Across Laptops** Now we get to run FTP across two separate computers to upload/download files between friends! We're going to be using SSH tunnels like we discussed in class. Let us assume that *Laptop1* will be running the FTP server and *Laptop2* will be running the FTP client. As usual, we will be tunneling through the *flip* servers.

   (a) Both laptops should start wireshark listening on the loopback interface.

   (b) First, *Laptop1* uses remote port forwarding to make the FTP service available on *flip*:
   ssh -NR PORT1:localhost:HOSTPORT1 ONID@flip2.engr.oregonstate.edu.
   Hint: *HOSTPORT1* will depend on the port that the FTP server is listening on (in previous labs 8080).

   (c) Open another terminal and start the FTP server

   (d) Now, *Laptop2* will have to use local port forwarding to complete the tunnel:
   ssh -NL PORT2:localhost:HOSTPORT2 ONID@flip2.engr.oregonstate.edu.
   Hint: This time *PORT2* will depend on the port that the FTP client tries to connect to (in previous labs 8080).

   (e) Open another terminal and start the FTP client.

   (f) The client should now run through some commands to upload, download, list, and remove some files, and then end in the client closing the connection.
   All this communication should be recorded by wireshark! Post screenshots of the TCP stream (by clicking follow TCP stream) from both the client and the server communicating. Feel free to provide any other screenshots/photos showing the communication between the laptops.