Steve Hirabayashi

PID: 2247504

CNT4713 Project 1 – File Transfer Protocol (FTP) Client and Server Report

1. Introduction

In this project, I was given the task of building both an FTP client and server that has implemented the requested FTP client and server commands. In order to successfully complete this assignment, I needed to become familiar with the Python programming language as well as the various modules needed to establish functionality like Unix-style directory management, reading from configuration files and supporting program arguments.

Quite a bit of the knowledge needed for completing both the FTP client and server I derived from the FTP client program given by the professor. For most of my questions regarding Python, I referred to the "Introducing Python" textbook, as well as the online API. For most of my questions regarding FTP, I first referred to the RFC 959 document. If anything was unclear in the document, I would then proceed to search for the needed information online, with the references outlined in the References section.

Overall, I feel that the current implementation of the FTP client and server successfully meets the specifications outlined by the professor, although I was not able to conduct a sufficient test on every feature of the program.

2. Problem Statement

The objective of this project was to implement a functional client and server that completely follows File Transfer Protocol as specified in the RFC 959 document. The FTP server must have compatibility with any FTP client by correctly implementing the FTP commands. The server must also use multi-threading for each different client connected to the server, and must be able to be suspended or shut down by another program through a separate service connection. The FTP client program must have included most of the commonly implemented FTP client commands, and must be able to specify in the program arguments the configuration file, test file, and other settings that can override the default configuration settings.

3. Methodology

The project was started from a given FTP client program that had implemented a portion of the FTP client commands. Following the given FTP client program, the control connection between the client and server is done with an active TCP connection, and the corresponding FTP commands needed to run the FTP client commands were implemented, such as USER, PASS, LIST, PORT, STOR, RETR, etc.

Then the configuration files for both the client and server were created, the program reads these files using the ConfigParser module. Similarly, the program arguments for both the client and server program were handled using the ArgParse module; most notably allowing for the user to test the application using a given test file (or a default one), instead of manually inputting FTP commands. After that, the remaining FTP client and corresponding server commands (if any) were implemented one by one, such as PASV (to allow for passive mode for establishing data connections) and TYPE (to establish what type of file is being transmitted through the TCP data connection).

Then a logging feature was added that records the FTP commands and server responses for each client and server session. Thread locks were included to protect the log file from multiple simultaneous writes from the client threads. Lastly, a separate server run program was created to allow the user to suspend, resume and terminate the FTP server. Socket timeouts were used to ensure that servers can listen for both control connections from FTP clients, and service connections from the server run program.

4. Results

Running the default test file tests among other things getting and putting a jpeg file back and forth in active mode. Here is the results recorded in the client log:

```
   Server Response:
   220 Service ready
Input: login user1 us1
   Server Response:
   331 user1 accepted, need password
   230 Password ok, user1 logged in
Input: image
   Server Response:
   200 File Transfer set to Image
Input: active
   Client Response: Set Data Connections to Active Mode
Input: PUT tests/test_items/image.jpg cube.jpg
   Server Response:
   200 PORT command OK
   226 cube.jpg was successfully transmitted
Input: GET cube.jpg tests/test_items/cube.jpg
   Server Response:
   200 PORT command OK
   150 File status ok, ready to transfer cube.jpg
   226 cube.jpg was successfully transmitted
Input: lcd tests/test_items/
   Client Response: New directory: "E:\CNT 4713\FTP\tests\test_items"
Input: lls
   Client Response: Succ. displayed "E:\CNT 4713\FTP\tests\test_items"
Input: quit
   Server Response:
   221 Server closing connection
```

Following FTP protocol, every FTP command the client sends to the server is responded with a three digit code indicating success, failure or error states. The command PUT for example, will first send the PORT command (in active mode) and then do the STOR command. The transfer of the file is shown to be successful with the two response codes being ‘200’ for the PORT command and ‘226’ for the STOR command. Other client commands, like LCD and LLS do not involve any server commands, as shown above.

5. Analysis

As my first larger scale Python project, I was able to learn much of the functionality and features of the Python programming language, as well as some of the powerful modules available such as the os, socket, threading, sys, shutil, configparser, and argparse modules. The project also my first time implementing threads and sockets, giving me exposure to those concepts in computer science. As such, much of my struggles originated from unfamiliarity with both the Python language and the corresponding modules as well as to multi-threading and establishing TCP connections using sockets.

6. References

Aside from the given FTP client program and the RFC 959 document, other supplementary websites and textbooks I used:

Bernstein, D. J. (n.d.). *FTP: File Transfer Protocol*. Retrieved from Bernstein Internet publication: https://cr.yp.to/ftp.html

Colorado State University. (n.d.). *Basic FTP Commands*. Retrieved from CSU Computer Science Department: https://www.cs.colostate.edu/helpdocs/ftp.html

Lubanovic, B. (2015). *Introducing Python.* Sebastopol: O'Reilly Media Inc.