FTP Protocol Design

Michael Carey, Mariah Harris, Kendrick Ngo, Brian Yadgarian

1. Design Summary

This protocol is designed to provide basic FTP functionality and facilitates connections between a client and a server through socket connections. The program has a basic CLI for the user to interact with the application by inputting the following commands: “get”, “put”, “ls”, and “quit”, thus giving users basic control over file transfers between a client and server. The data connection between the client and server is established through an ephemeral port, which is uniquely generated to establish a temporary connection for data transfer. Once the transfer is complete, the connection is terminated, and the ephemeral port is released.

2. System Responsibilities

2.1 Server Responsibilities

The server is responsible for creating and maintaining the control connection between the client and the server using ephemeral ports.

The server contains files: `sFile1.txt, sFile2.txt, sFile3.txt` for testing purposes. To execute, navigate to `Project Files/server/` and run `python3 pythonserv.py <port_number>`

- `get <file_name>` send file to client
- `put <file_name>` receive file from client
- `ls` receive ls command from client to list contents of current directory
    - Send this information to client
- `quit` or `ctrl+c` end server connection

2.2 Client Responsibilities

The client (cli.py) is responsible for connecting to an already created socket and providing a user interface for interacting with the system. The client's basic Command Line Interface (CLI) acts as an entry point into the program and incorporates basic user detection and user input guardrails.

The client contains files: `cFile1.txt, cFile2.txt, cFile3.txt` for testing purposes. To execute, navigate to `Project Files/client/` and run `python3 cli.py localhost <port_number>`

- Basic CLI
- Invoked as `ftp>` at runtime, indicating command acceptance
- Available commands
    - `get <file_name>` download a file from the server
    - `put <file_name>` upload a file to the server
    - `ls` list contents of server
    - `menu` display all commands
    - `quit` or `crtl+c` exit main program and ends connection to the client

3. Requirements

3.1 Major Requirements

- GET command downloads a file from the server using a data connection
- PUT command uploads a file to the server using a data connection
- LS command lists the files on the server
- A control connection used for connection between client and server
- A data connection used for the transfer of data between client and server
- When executed on the terminal, the server should have the following structure:
    - python pythonserv.py <port #>
- When executed on the terminal, the client should have the following structure:
    - python cli.py <server IP> <server port>
- When the user is on the client terminal, it should start with "ftp> "

3.2 Other Requirements

- System call integration
- When run, server will create a socket connection and wait for connections
- When run, the client will connect to the server socket connection

- Once connection the user will be presented with a simple CLI so they can interact with the system
- The CLI should print “ftp> “ at the beginning of each line
- The CLI should accept only 4 commands: get, put, ls, quit
- The “get” command should have the following structure: get <filename>
- The “put” command should have the following structure: put <filename>
- The client CLI should have basic error detection
- When either the “get” or “put” command is inputted, the client should send the command to the server
- When the server receives a “get” or “put” command, it should create an ephemeral socket connection (data) and notify the client to connect to this temporary connection
- Once the client receives the OK from the server for a data connection, it will connect to it and then start file transfer procedures
- Once file transfer is complete, the client and server should end the data connection
- When the client receives a “ls” command, it should send this command to the server
- When the server receives a “ls” command from the client, it will initiate that command and send the results back to the client
- When the client receives the “ls” data from the server, it should print out that data to the user
- When the client receives a “quit” command from the user, it should send that command to the server, then close its connection to the server
- When the server receives a “quit” from the client, it should close the connection to the client