Protocol Design

Michael Carey, Mariah Harris, Kendrick Ngo, Brian Yadgarian

1. Design Summary

This protocol is designed to handle basic ftp functionality. It handles connections from a client to a server using socket connections. It has a basic CLI for the user to interact with the system wherein they may input the following commands: “get”, “put”, “ls”, “quit”. This allows the user basic control over file transfers between a client and server.

2. System Responsibilities

2.1 Server Responsibilities

- Create connections
    - Control connection
    - Data connection
- Download File (GET)
    - Send file to client
- Upload File (PUT)
    - Receive file from client
- LS
    - Receive ls command from client
    - List contents of current directory
        - Send this information to client

2.2 Client Responsibilities

- Connect to already created sockets
- Create UI for user to interact with the system
    - Basic CLI

        - Should print “ftp> “ at the beginning of each line
    - Accept 4 commands
        - GET
            - Download file from server
        - PUT
            - Upload file to server
        - LS
            - List contents of server
        - QUIT
            - End the connection to the server
    - GET/PUT should have the command structure as follows:
        - <command> <filename>

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