Network Application Development Project

A Simple File Transfer Service

By

Teng Zhao

40089560

Professor Chadi Assi

A report submitted in partial fulfilment of the requirements of COEN 366.

Concordia University

April 2022

This is a client-server program using Python stream sockets to partially simulate the file transfer protocol (FTP). The program is able to transfer any type of files. It will have the ability to download files from a server directory and upload files from a client directory.

Typical interaction:

* Start the server program and input IP, PORT, and DEBUG code.
* Start the client program and input IP, PORT and DEBUG code.
* When successful connection is established, the client program is able to upload files using the put command, download files using the get command, and change a file’s name on the server using a change command. The client will be parsing the commands, build and send requests to the server to perform these actions.
* The server will receive requests and perform actions after parsing them. The server will reply to the client with a response message.
* When a client quits, it will disconnect from the server and the server will return to listening for new connection requests.

**Table 1 Commands and its request message format**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Commands | Opcode | Description | Request format |
| 1. | put filename | 000 | Transfer file from client folder to server folder | Opcode: 3-bits  Filename length (FL): 5-bits  Filename: Byte 2 to Byte FL  File Size: Byte FL+1 to Byte FL+5 |
| 2. | get filename | 001 | Get file from server folder to client folder | Opcode: 3-bits  Filename length (FL): 5-bits  Filename: Byte 2 to Byte FL |
| 3. | change oldName newName | 010 | Change name of a file on the server folder | Opcode: 3-bits  oldFilename length (OFL): 5-bits  oldFilename: Byte 2 to Byte OFL  newFilename length (NFL): 1 Byte  newFilename: Byte OFL+1 to Byte OFL+1+NFL |
| 4. | help | 011 | Ask server for valid commands | Opcode: 3-bits  Unused: 5-bits |
| 5. | bye | - | Client closes connection with the server |  |

**Table 2 Response messages and its format**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Rescode | Response | Response format |
| 1. | 000 | Response for correct put and change request | Rescode: 3-bits  Unused: 5-bits |
| 2. | 001 | Response for correct get request | Rescode: 3-bits  Filename (FL): 5-bits  Filename: Byte 2 to Byte FL  File Size: Byte FL+1 to Byte FL+5 |
| 3. | 010 | File not found error | Rescode: 3-bits  Unused: 5-bits |
| 4. | 011 | Unknown request error | Rescode: 3-bits  Unused: 5-bits |
| 5. | 101 | Unsuccessful name change | Rescode: 3-bits  Unused: 5-bits |
| 6. | 110 | Help response | Rescode: 3-bits  Length: 5-bits  Help Data: Byte 2 to Byte length+1 |

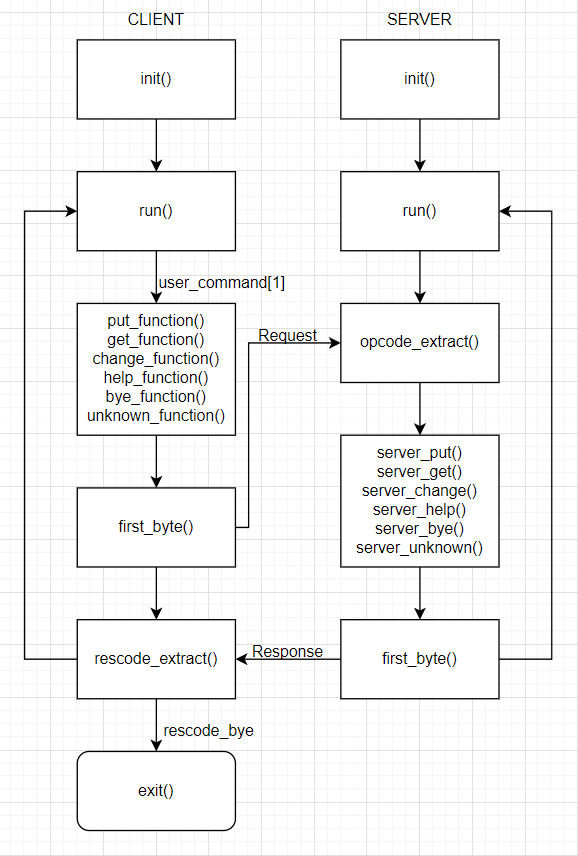


Figure Program flowchart diagram

Description of functions:

server.py

* to\_print(): prints out the send and receive logs when DEBUG is set to 1
* first\_byte(): creates the first byte of the response message with the rescode and file name size when applicable
* opcode\_extract(): extracts the opcode from the request to determine which action to take
* init(): creates socket and binds it to an IP and PORT
* run(): listens for client connections. When a client is connected, obtains the request, and processes it
* server\_put(): performs upload request and responds with rescode message
* server\_get(): performs download request and responds with rescode message
* server\_change(): performs file name change request and responds with rescode message
* server\_help(): returns message with valid requests
* server\_bye(): closes connection with client and returns to listening for client connection
* server\_unknown(): returns unknown request message

client.py

* to\_print(): prints out the send and receive logs when DEBUG is set to 1
* first\_byte(): creates the first byte of the request message with the opcode and file name size when applicable
* rescode\_extract(): extracts the rescode from the request to determine which action to take
* init(): creates socket and attempts to connect to server
* run(): takes user inputs, parses input, and sends request based on the input
* put\_function(): sends request for uploading a file and receives response message
* get\_function(): sends request for downloading a file and receives response message
* change\_function(): sends request for changing file name and received response message
* help\_function(): sends request for list of valid requests
* bye\_function(): closes connection with server
* unknown\_function(): handles server response of a invalid request