

Report

CMPE156 Name: Shuli He, ID:she77@ucsc.edu
Concurrent-File-Transfer

Usage

Client:

`./myclient <server-info.txt> <num-connections> <filename>`

example: `./myclient serverinfo.txt 3 set`

Server:

`./myserver <port>`

example: `./myserver 12345`

Design

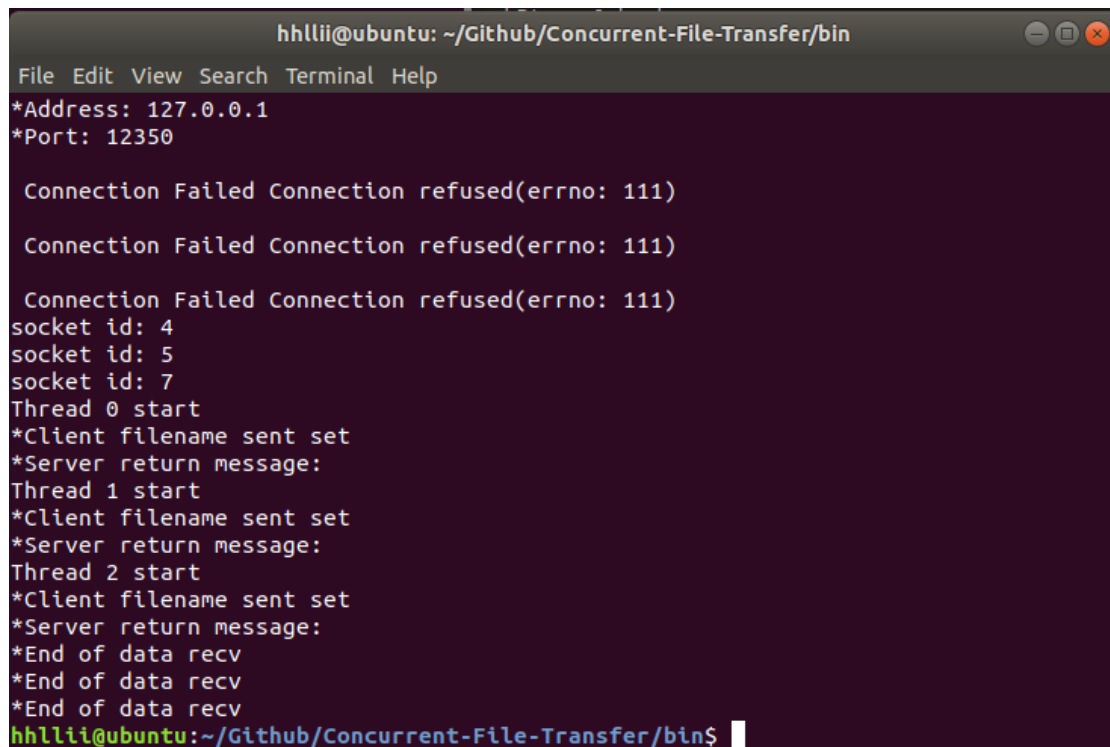
Client:

1. Handle the input argv and verify if they are valid. For port number is a valid port or number of connections is too large.
2. Get the available server list by trying to connect store the connected socket to a list.
3. Create threads and assigned with socket list which helps thread to get the socket connection.
4. Each thread has the file size (from server) and offset (by client) to determine

the file part they need to receive.

5. After all thread get the file part, main thread will assemble them to a single file. Files store at. /dest/ fold.
6. Clean all memory, temp files and close the socket before client exit.

Test:

A terminal window titled 'hhllii@ubuntu: ~/Github/Concurrent-File-Transfer/bin' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal output shows the server configuration: '*Address: 127.0.0.1' and '*Port: 12350'. It then displays three 'Connection Failed Connection refused(errno: 111)' messages. Following these, it shows 'socket id: 4', 'socket id: 5', and 'socket id: 7'. Then, 'Thread 0 start' is followed by '*Client filename sent set' and '*Server return message:'. This pattern repeats for 'Thread 1 start' and 'Thread 2 start'. Finally, it shows '*End of data recv' three times. The prompt 'hhllii@ubuntu:~/Github/Concurrent-File-Transfer/bin\$' is visible at the bottom with a cursor.

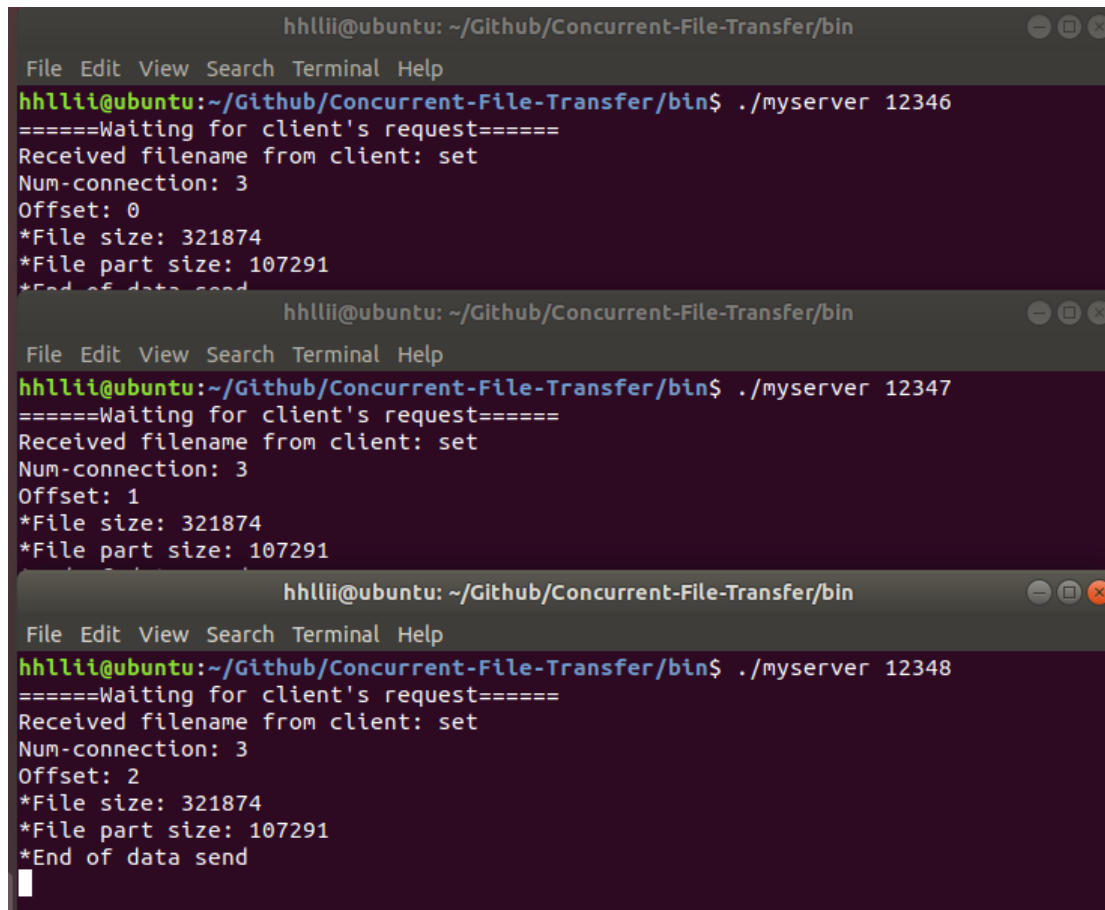
```
hhllii@ubuntu: ~/Github/Concurrent-File-Transfer/bin
File Edit View Search Terminal Help
*Address: 127.0.0.1
*Port: 12350

Connection Failed Connection refused(errno: 111)
Connection Failed Connection refused(errno: 111)
Connection Failed Connection refused(errno: 111)
socket id: 4
socket id: 5
socket id: 7
Thread 0 start
*Client filename sent set
*Server return message:
Thread 1 start
*Client filename sent set
*Server return message:
Thread 2 start
*Client filename sent set
*Server return message:
*End of data recv
*End of data recv
*End of data recv
hhllii@ubuntu:~/Github/Concurrent-File-Transfer/bin$
```

Server:

1. Receive the message from client and send back the file information.
2. Files in ./files/ fold. If no file return nothing and client get nothing.
3. Get thread download request and send the request part by offset and file size.

Test:

The image displays three overlapping terminal windows from an Ubuntu system, each running a custom server program named 'myserver'. The windows are titled 'hhllii@ubuntu: ~/Github/Concurrent-File-Transfer/bin'. The first window shows the server running on port 12346, receiving a request with filename 'set', 3 connections, and an offset of 0. The second window shows the server on port 12347, receiving a request with the same filename and connections but an offset of 1. The third window shows the server on port 12348, receiving a request with an offset of 2. All three requests specify a file size of 321874 and a part size of 107291. The output for each window ends with '*End of data send' followed by a cursor.

```
hhllii@ubuntu: ~/Github/Concurrent-File-Transfer/bin
File Edit View Search Terminal Help
hhllii@ubuntu:~/Github/Concurrent-File-Transfer/bin$ ./myserver 12346
=====Waiting for client's request=====
Received filename from client: set
Num-connection: 3
Offset: 0
*File size: 321874
*File part size: 107291
*End of data send

hhllii@ubuntu: ~/Github/Concurrent-File-Transfer/bin
File Edit View Search Terminal Help
hhllii@ubuntu:~/Github/Concurrent-File-Transfer/bin$ ./myserver 12347
=====Waiting for client's request=====
Received filename from client: set
Num-connection: 3
Offset: 1
*File size: 321874
*File part size: 107291

hhllii@ubuntu: ~/Github/Concurrent-File-Transfer/bin
File Edit View Search Terminal Help
hhllii@ubuntu:~/Github/Concurrent-File-Transfer/bin$ ./myserver 12348
=====Waiting for client's request=====
Received filename from client: set
Num-connection: 3
Offset: 2
*File size: 321874
*File part size: 107291
*End of data send
█
```

Protocol:

1. Create the SimpleAddress struct to store address information and SimpleChunk struct as the transfer structure through socket.
2. `vector<int> getActiveSockList(vector<SimpleAddress> list);` function to both create socket connection and socket list.
3. `void simpleSocketSend(int sockfd, SimpleChunk* chunk, int chunk_size);`
`int simpleSocketRecv(int sockfd, SimpleChunk* chunk, int chunk_size);`
these two functions to manage the socket communication.

Potential problem

1. Limited the maximum thread number. `#define MAX_SERVER 128` to modify the number of connections.
2. File name length be limited.
3. No thread return catch.
4. For servers failure need to download again.
5. If file not exist, the client will not get an error message it just got an empty file.