

CS 371 Project Report

Ngoc Phan

Abstract:

The main focus of this project is to implement a client-server application for a private remote model. The client and server can upload/download files and store them in their folders. The functionalities of this include upload, download, delete, and see the content of the shared folder (which contains the information about the files' sizes, uploading date and time, and the number of downloads). The average uploading speed is calculated to be 51.83 mbps, and the downloading speed is approximately 56.61 mbps. These are roughly estimated depending on the number of devices using the network at a time, these speeds could be slower if there are more devices and faster if there are less devices. The connection at my place is unstable and there are multiple devices in use at the same time, therefore the uploading and downloading speeds fluctuate as described in the graphs below.

Introduction and motivation:

The server and client have their own folders in which they store their files. The shared folder, called "Info", contains two files: info.txt and downloadRecord.txt. The file info.txt has the files' names, when they are uploaded, and their sizes. The file downloadRecord.txt contains the number of downloads each file has. This is a convenient way to control how to see the information related to these files.

The project design consists of a client and a server. The client can make connection to a specified port, upload a file to the server, download a file from the server, delete a file that exists either on the client's folder or the server's folder, and see the content of the shared folder. On the other hand, the server accepts an incoming connection and handles the commands coming from the client.

Project design and Implementation:

Starting with creating the socket s for both the server and client:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Connect: The client request to connect to a specific port number. If successful, it will show “Connection from: (<host>, <port>)” on the server side.

Upload: The client requests to upload a file to the server. Before starting the upload, the client will check the specified file exists in the client’s local folder first (using `isfile()`): if it does, start the upload; if it doesn’t, print out “Error!”

Download: The client request to download a file from the server. The filename is sent to the server to check if there’s such file in the server’s folder (using `isfile()`): if it does, the server will send back an ack saying that such file exists and confirm if the client still wants to download it. If yes, the download starts. If there’s no such file in the server’s folder, an “Error!” message is printed out.

Delete: The specified file name is sent from the client to the server, the server will then check if there’s such a file in either the server’s or the client’s folder (using `isfile()`). If there is, the file will be deleted and a message “<filename> removed!” will show up, if not, an error message “<filename> does not exist!” is displayed.

Dir: The directory name is sent from the client to the server where it is checked if the directory exists (using `isdir()`). If yes, the content of the client’s, server’s folder, or the shared folder is displayed, including the filenames, upload date and time (only if they’re uploaded, if the files have already been in the folder, this upload date and time will not be available) and the number of downloads made of each file.

To capture the upload date and time, every time a file is uploaded, I use `time.time()` to get the current time and store it in `info.txt` with the corresponding filename.

To get the number of downloads, every time a file is downloaded, the client would look into the `downloadRecord.txt` to see if it’s been downloaded before, if not, create a new entry with download number 1 and append it to the `.txt` file. If the file has been downloaded before, add 1 to the current number of downloads for the corresponding file.

To get the time upload/download speed, I use `time.time()` at the start of the uploading/downloading process to get the `start_time`, then use `time.time()` again every other second and calculate `time_lapsed = time.time() - start_time` for every other second. Finally, take the total number of megabytes transferred divided by `time_lapse` to get the download/upload speeds and display them.

Experiments: I ran with a file named “big” and a file that doesn’t exist called “random”

Connect: Enter “connect” to make the connection, then put in the port number.

```
[TRANS-Air-5:cs371 thttien$ python client.py
Enter "connect", "download", "upload", "delete", "dir" (or "q" to quit)
-> connect
Port: 5000
```

If successful, the result is shown below:

```
[TRANS-Air-5:cs371 thttien$ python server.py
Connection from : ('192.168.1.6', 61393)
```

Upload: Enter “upload” to upload a file, then type in the file name. Two scenarios will happen:

- If the file doesn’t exist in the client’s local folder, there’s nothing to upload --> a “Error!” message will show up:

```
Enter "connect", "download", "upload", "delete", "dir" (or "q" to quit)
-> upload
Filename: random.txt
Error!
```

- If the file exists in the client’s local folder, the upload process will begin. The upload speed is displayed on the screen also

```
Enter "connect", "download", "upload", "delete", "dir" (or "q" to quit)
-> upload
Filename: big
```

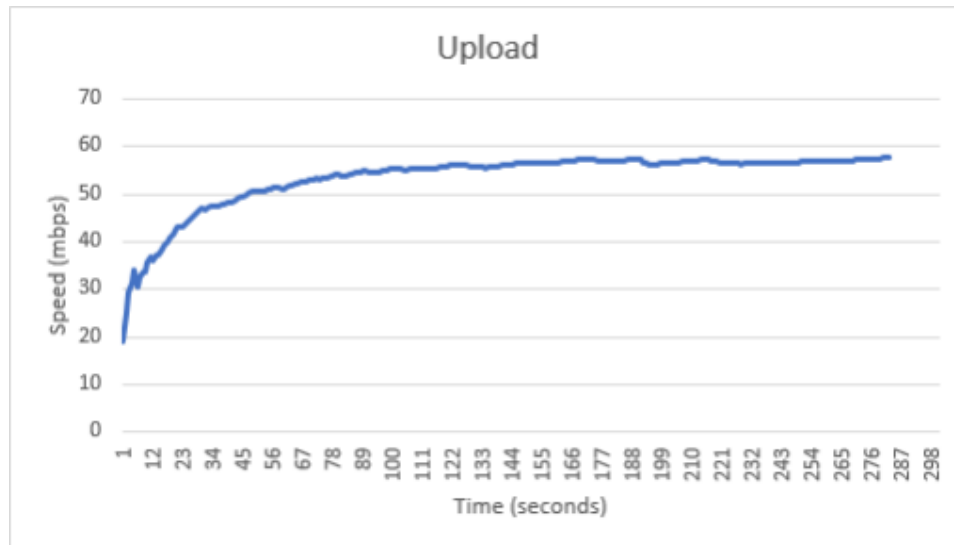
The upload speed is displayed on the terminal as below:

```
57.41
55.74
55.28
54.97
54.74
54.72
54.69
54.69
53.36
51.58
51.67
51.86
Average upload speed = 51.83 mbps
Upload completed!
```

The file “big” is now uploaded to the server’s folder as “uploaded_big”:

Folders	Documents
Client	download.txt
Info	Images
Server	photo.JPG
Documents	PDF Documents
hw1.docx	book.pdf
hw2.docx	
hw3.docx	Other
hw4	uploaded_big
hw7.docx	
PDF Documents	
hw6.pdf	
Tanenbaum...all (2011).pdf	
Spreadsheets	
Graphs	
Developer	
client.py	
server.py	

The graph for upload speed is as follow:



Graph 1: The upload speed has an average of 51.83 mbps. The vertical axis displays upload speed in megabytes per second and the horizontal axis displays the time in every other second with the unit of 10^{-1} seconds, for example:

$$298 * 10^{-1} \text{ seconds} = 29.8 \text{ seconds}$$

This graph makes sense because in the beginning, the data window size increases exponentially every roundtrip time, hence the upload speed, until it reaches a threshold, then it increases linearly after that until there's a packet loss.

Download: Enter “download” and a file name to download. Two scenarios will happen:

- If the file doesn’t exist in the client’s local folder, there’s nothing to upload --> a “Error!” message will show up:

```
Enter "connect", "download", "upload", "delete", "dir" (or "q" to quit)
-> download
Filename: random
Error!
```

- If the file exists in the server’s folder, the download process will begin. The download speed is displayed on the screen also:

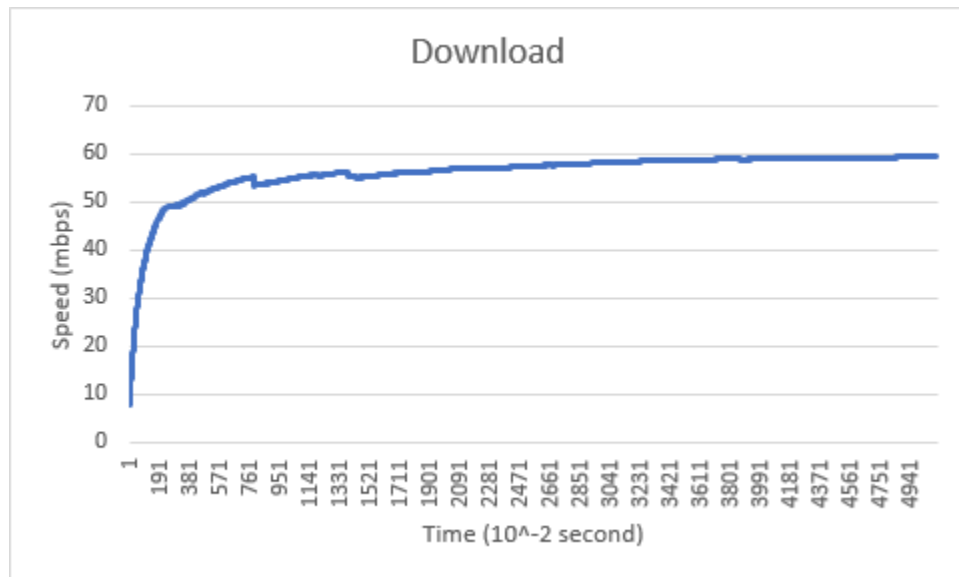
```
Enter "connect", "download", "upload", "delete", "dir" (or "q" to quit)
-> download
Filename: big
File exists, download? (y/n): y
43.95
53.69
56.01
57.37
57.73
58.01
55.96
55.8
56.59
57.02
57.59
58.13
58.48
58.7
58.9
59.23
59.72
60.22
60.54
60.97
61.29
61.37
61.67
61.85
61.65
61.7
61.77
61.94
62.09
62.14
62.18
62.33
62.45
62.48
60.99
59.71
59.29
58.97
58.27
57.83
57.5
56.97
Average download speed = 56.61 mbps
Download completed!
```

(This is a shorten version of the number of displayed download speeds because I couldn't fit all data points used for the graph in the screenshot)

The file “big” is now downloaded to the client's folder as “download_big”:

Folders	Documents
▶ Client ▶	upload.txt
▶ Info ▶	Images
▶ Server ▶	photo.JPG
Documents	PDF Documents
▶ hw1.docx	book.pdf
▶ hw2.docx	
▶ hw3.docx	Other
▶ hw4	big
▶ hw7.docx	downloaded_big
PDF Documents	
hw6.pdf	
Tanenbaum...all (2011).pdf	
Spreadsheets	
Graphs	
Developer	
client.py	
server.py	

The graph for download speed is as follow:



Graph 2: The download speed has an average of 56.61 mbps. The vertical axis displays download speed in megabytes per second and the horizontal axis displays

the time in every other second with the unit of 10^{-2} seconds, for example:
 $4941 * 10^{-2}$ seconds = 49.41 seconds.

Similar to the Upload speed graph, this graph makes sense because in the beginning, the data window size increases exponentially every roundtrip time, hence the download speed, until it reaches a threshold, then it increases linearly after that until there's a packet loss.

Delete: Enter “delete” and a file name to delete.

```
Enter "connect", "download", "upload", "delete", "dir" (or "q" to quit)
-> delete
Filename: uploaded_big
```

- If the file is found, the deletion process begins and the result is shown below:

```
Connection from : ('192.168.1.6', 62269)
uploaded_big removed!
```

- If the file does not exist:

```
Enter "connect", "download", "upload", "delete", "dir" (or "q" to quit)
-> delete
Filename: random

Connection from : ('192.168.1.6', 62421)
random does not exists!
```

Dir: Enter “dir” and a folder name to see the content of that folder.

- If the folder doesn't exist, an “Error” message is displayed:

```
Enter "connect", "download", "upload", "delete", "dir" (or "q" to quit)
-> dir
Folder name: notReal

Connection from : ('192.168.1.6', 64104)
Folder does not exist!
```

- If the folder exists, the content of the folder is displayed as below:

```
Enter "connect", "download", "upload", "delete", "dir" (or "q" to quit)
-> dir
Folder name: Client

Enter "connect", "download", "upload", "delete", "dir" (or "q" to quit)
-> dir
Folder name: Client
```




















```

Connection from : ('192.168.1.6', 63764)
book.pdf 8595127 bytes 2020-04-24 17:50:37.038470
book.pdf 1
downloaded_big
.DS_Store
big 524288000 bytes 2020-04-23 21:26:15.823274
big 1
photo.JPG 28886223 bytes 2020-04-23 20:23:50.048161
photo.JPG 1
upload.txt

```

Explanation: The number next to a file is the number of times that file has been downloaded. Next to some of the files are the number of bytes, which is the file sizes, and the date and time, which is when those files are uploaded. If a file (for example: “upload.txt”) doesn’t have a number, or the file size/date and time next to it, that means the file has already been there from the beginning (it wasn’t uploaded or downloaded), so there’re no such data for them. Also, the file .DS_Store is a default file that’s used by Mac OS X to store folder specific metadata information so it doesn’t show up on the folder itself but still appears when I list all files to terminal.

For reference, the content of the Client folder is below:

Folders	Documents
 Client	 upload.txt
 Info	 Images
 Server	 photo.JPG
Documents	PDF Documents
 hw1.docx	 book.pdf
 hw2.docx	
 hw3.docx	
 hw4	
 hw7.docx	
PDF Documents	Other
 hw6.pdf	 big
 Tanenbaum...all (2011).pdf	 downloaded_big
Spreadsheets	
 Graphs	
Developer	
 client.py	
 server.py	

Conclusions:

In summary, the server and client are able to connect to a specified port. The download and upload speeds vary depending on the stability of the wireless connection. All required functionalities of the program are satisfied.