

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



MẠNG MÁY TÍNH (CO3093)
Đề tài

**DEVELOP A NETWORK APPLICATION
FILE SHARING SYSTEM**

Giảng viên hướng dẫn: Hoàng Lê Hải Thanh
Sinh viên thực hiện: Mai Phương Nhã - 2111892
Phan Phạm Thi - 2114857
Trần Hồ Thanh Vũ - 2220031
Phạm Lê Thế Anh - 1652026

Thành phố Hồ Chí Minh, Tháng 11/2023



Mục lục

1	Danh sách thành viên và công việc	2
2	Phân tích yêu cầu bài toán	2
2.1	Yêu cầu chức năng	2
2.2	Yêu cầu phi chức năng	2
3	Mô tả các function trong class Client và Server	3
4	Class Diagram	5
5	Đánh giá tổng hợp kết quả đạt được	5
5.1	Về lập trình	5
5.2	Kết quả	5
6	User manual	6
7	Extend	12
8	Source Code	15
	Tài liệu tham khảo	15

1 Danh sách thành viên và công việc

Thành viên	MSSV	Công việc
Mai Phương Nhã	2111892	Code Phase 2, Soạn báo cáo, Tổng hợp
Phan Phạm Thi	2114857	Code Phase 2, Soạn báo cáo, Tổng hợp
Trần Hồ Thanh Vũ	2220031	Phase 1, Soạn báo cáo
Phạm Lê Thế Anh	1652026	Phase 1, Soạn báo cáo

2 Phân tích yêu cầu bài toán

2.1 Yêu cầu chức năng

- Hệ thống cho phép người dùng tạo tài khoản/xóa tài khoản, đăng nhập/đăng xuất bằng Username, Password.
- Hệ thống có các chức năng quan trọng như **publish** (*publish 1 file lên hệ thống*), **fetch** (*yêu cầu tải 1 file từ 1 peer đến 1 client*), **ping** (*kiểm tra trạng thái hoạt động của client - đối với terminal server*), **discover** (*xem danh sách những file đã publish của một client cụ thể - đối với terminal server*).
- Trong điều kiện client được yêu cầu vẫn đang *online*, peer có thể fetch file và file sẽ được tự động tải về hoặc thực hiện những tác vụ khác một cách liên tục.
- Khi có đồng thời nhiều client kết nối với server, server sẽ có nhiều thread để đi xử lý các yêu cầu từ các client.
- Khi hệ thống nhận thấy một client đã kết nối với server nhưng đăng nhập không thành công sẽ không thể tiếp tục thực hiện các tác vụ.
- Trong khi client thực hiện tác vụ của mình, mỗi client đều có 1 port để nghe yêu cầu **fetch file** (*từ các peer khác*). Khi nhận được yêu cầu, Client trực tiếp gửi file cho peer đang gửi fetch file mà không cần thông qua Server.

2.2 Yêu cầu phi chức năng

- Hiệu suất: Đáp ứng được kết nối của 1 server - nhiều client (nhiều peer) qua kết nối TCP/IP cùng một mạng LAN. Trả về kết quả của các tác vụ không trên khoảng thời gian là 5s.
- Bảo mật: Đáp ứng vấn đề bảo mật cho người dùng hệ thống bằng cách quản lý tài khoản.
- Tính linh động và tương thích: Có thể thực hiện tác vụ cho cả server (tác vụ ping, discover) hoặc client (tác vụ login, publish,...)
- Khả năng sử dụng:
 - Khi có lỗi trong vấn đề truy cập cơ sở dữ liệu hay kết nối qua các socket, hệ thống trả về lỗi hoặc dòng thông báo đến người dùng và kết thúc phiên kết nối.
 - Hệ thống có hướng dẫn, thông báo, cảnh báo về các lựa chọn cho người dùng.

3 Mô tả các function trong class Client và Server

Class Name	Function	Description
Client	<code>__init__(self, sock, addr, request, request_argv)</code>	Khởi tạo
	<code>_read(self)</code>	Đọc dữ liệu từ socket và ghi vào receive buffer
	<code>_write(self)</code>	Ghi dữ liệu vào send buffer
	<code>read(self)</code>	Lấy những thông tin cần thiết theo format đã quy định từ receive buffer
	<code>write(self)</code>	Gửi dữ liệu từ send buffer đến kết nối socket
	<code>choose_peer(self)</code>	Chọn một peer để fetch file sau khi hiển thị danh sách peer
	<code>request_file_from_peer(self, peer)</code>	Kết nối đến peer và lấy file về
Server	<code>__init__(self, sock, addr)</code>	Khởi tạo
	<code>_read(self)</code>	Đọc dữ liệu từ socket và ghi vào receive buffer
	<code>read(self)</code>	Lấy những thông tin cần thiết theo format đã quy định từ receive buffer
	<code>write(self)</code>	Gửi dữ liệu từ send buffer đến kết nối socket
	<code>_create_message(self, response_type, res_code, res_msg, content_bytes, content)</code>	Tạo ra thông điệp gửi tới client và ghi vào send buffer
	<code>close(self)</code>	Đóng kết nối với client
	<code>process_request(self)</code>	Xử lý các yêu cầu của client dựa vào dữ liệu client gửi tới
	<code>reply_publish(self, fname, lname, file_size, extension)</code>	Phản hồi yêu cầu publish, thêm thông tin file chia sẻ khỏi hệ thống
	<code>reply_unpublish(self, fname)</code>	Phản hồi yêu cầu unpublish, xóa thông tin file chia sẻ khỏi hệ thống
	<code>reply_register(self, username, password, port)</code>	Phản hồi yêu cầu register, tạo một tài khoản mới cho Client yêu cầu
	<code>reply_fetch(self, fname)</code>	Phản hồi yêu cầu fetch
	<code>ping(self, ipAddress)</code>	Live check một địa chỉ IP
	<code>reply_check_ip</code>	Kiểm tra một địa chỉ IP có nằm trong hệ thống không
	<code>reply_unregister(self)</code>	Phản hồi yêu cầu unregister, xóa một tài khoản khỏi hệ thống
	<code>reply_login(self, password)</code>	Phản hồi yêu cầu login, kiểm tra password có khớp với địa chỉ IP đăng ký không
	<code>reply_view(self)</code>	Phản hồi yêu cầu view, lấy thông tin toàn bộ file mà một client đã chia sẻ

Hình 1: Bảng mô tả các chức năng của từng hàm tương ứng trong Class

1. Client

(a) `__init__(self, sock, addr, request, request_argv)`

- Constructor khai báo các thuộc tính của Class:

(b) `_read(self)`

- Đọc dữ liệu thông qua socket từ Server/Peer và ghi vào *receive buffer*

(c) `_write(self)`

- Ghi dữ liệu vào send buffer

(d) `read(self)`

- Lấy những thông tin cần thiết theo format đã quy định từ receive buffer

(e) `write(self)`

- Gửi dữ liệu từ send buffer đến kết nối socket

(f) `choose_peer(self)`

- Chọn một peer để fetch file sau khi hiển thị danh sách peer

(g) `request_file_from_peer(self, peer)`

- Kết nối đến peer và lấy file về

2. Server

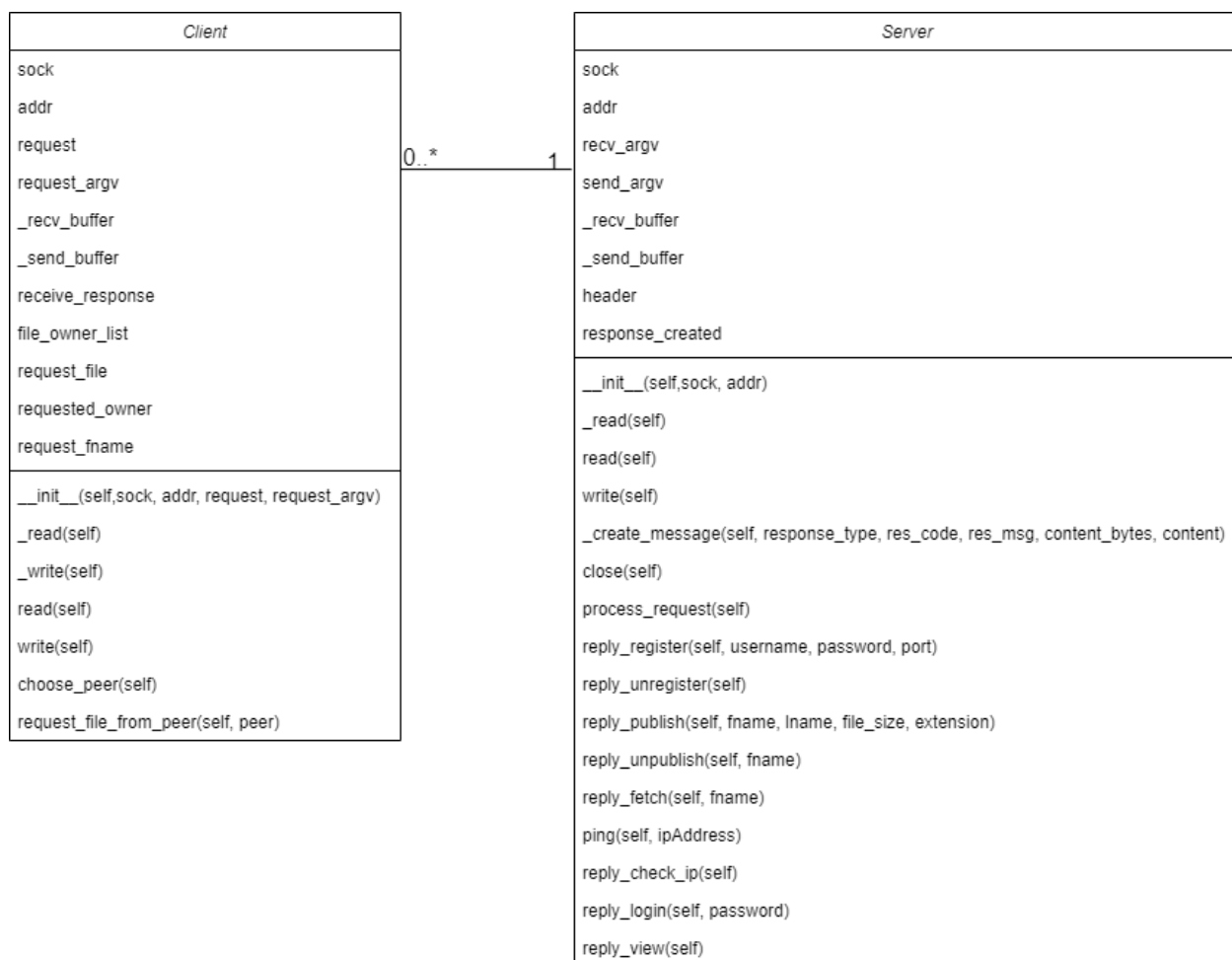
(a) `__init__(self, sock, addr)`

- Constructor khai báo các thuộc tính của Class:

(b) `_read(self)`

- Đọc dữ liệu từ socket và ghi vào receive buffer
- (c) *read(self)*
 - Lấy những thông tin cần thiết theo format đã quy định từ receive buffer
- (d) *write(self)*
 - Gửi dữ liệu từ send buffer đến kết nối socket
- (e) *_create_message(self, response_type, res_code, res_msg, content_bytes, content)*
 - Tạo ra thông điệp gửi tới client và ghi vào send buffer
- (f) *close(self)*
 - Đóng kết nối với client
- (g) *process_request(self)*
 - Xử lý các yêu cầu của client dựa vào dữ liệu client gửi tới
- (h) *reply_publish(self, fname, lname, file_size, extension)*
 - Phản hồi yêu cầu publish, thêm thông tin file chia sẻ khỏi hệ thống
- (i) *reply_unpublish(self, fname)*
 - Phản hồi yêu cầu unpublish, xóa thông tin file chia sẻ khỏi hệ thống
- (j) *reply_fetch(self, fname)*
 - Phản hồi yêu cầu fetch
- (k) *ping(self, ipAddress)*
 - Live check một địa chỉ IP
- (l) *reply_check_ip(self)*
 - Kiểm tra một địa chỉ IP có nằm trong hệ thống không
- (m) *reply_unregister(self)*
 - Phản hồi yêu cầu unregister, xóa một tài khoản khỏi hệ thống
- (n) *reply_login(self, password)*
 - Phản hồi yêu cầu login, kiểm tra password có khớp với địa chỉ IP đăng ký không
- (o) *reply_view(self)*
 - Phản hồi yêu cầu view, lấy thông tin toàn bộ file mà một client đã chia sẻ

4 Class Diagram



Hình 2: Class Diagram của hệ thống

5 Đánh giá tổng hợp kết quả đạt được

5.1 Về lập trình

- Thiết lập được cơ chế xử lý cho 1 Server đối với các Client và 1 Client với các Peer còn lại trong hệ thống.
- Hoàn thành các tác vụ riêng của Server: **Ping và Discover**.

5.2 Kết quả

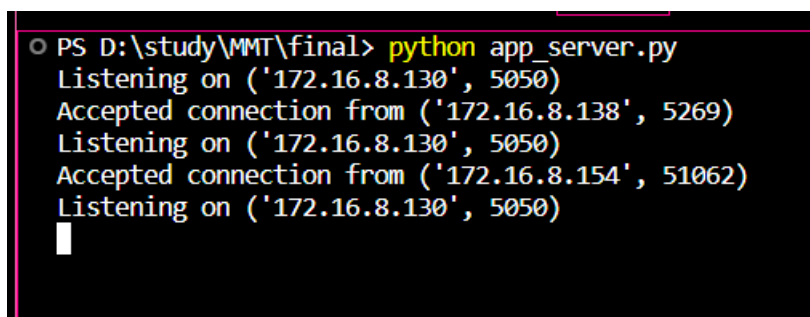
- Khởi động được Server. Server nhận được các yêu cầu của Client khi chúng được gửi đến, quá trình trích xuất yêu cầu tại Server hoạt động và cho ra kết quả như mong muốn.

- Client nhận được các responses mà Server gửi về và thực hiện các tác vụ như fetch, publish thành công.
- Hoạt động được trên nhiều máy tính khác nhau với cùng một mạng Internet để đảm bảo yêu cầu kết nối cùng mạng LAN và qua giao thức TCP/IP.
- Kết nối Client và Peer thành công. Chia sẻ file không cần thông qua Server: Không cần tạo bản sao để Server lưu trên cơ sở dữ liệu, Client trực tiếp truy cập vào đường dẫn có file được yêu cầu và gửi cho Peer. File xuất hiện ở folder hiện tại ở máy chủ của Peer.
- Với phần mở rộng của đề tài, nhóm có thiết lập cơ chế tạo tài khoản cho người dùng giải quyết cho vấn đề bảo mật.

6 User manual

Phần tiếp theo sẽ mô tả quá trình kiểm tra sử dụng với giả lập: *Client 1 là một người dùng đang sử dụng và nhận được tín hiệu yêu cầu fetch file từ Client 2 (Peer)*

- Bước 1:** Kiểm tra các IP Address của các máy tham gia hệ thống.
Thiết lập giá trị HOST, SERVER trong các file `app_client.py` và `app_server.py`
Trong ví dụ chạy thử dưới đây,
Server có địa chỉ IP là: **172.16.8.130**;
Client 1 có địa chỉ IP là **172.16.8.138**;
Client 2 có địa chỉ IP là **172.16.8.154**;
Ghi Chú: Đảm bảo 3 máy trên đều kết nối cùng một mạng LAN
- Bước 2:** *Khởi động Server.*
Chạy terminal trong thư mục chứa file `app_server.py` để server có thể bắt đầu nghe tín hiệu kết nối.



```
PS D:\study\MMT\final> python app_server.py
Listening on ('172.16.8.130', 5050)
Accepted connection from ('172.16.8.138', 5269)
Listening on ('172.16.8.130', 5050)
Accepted connection from ('172.16.8.154', 51062)
Listening on ('172.16.8.130', 5050)
```

Hình 3: Server khởi động và tạo cổng TCP/IP nghe các kết nối từ Clients

- Bước 3:** *Khởi động Client 1.*
Chạy terminal trong thư mục chứa file `app_client.py` để client có thể đăng nhập/đăng ký và bắt đầu các tác vụ.

```
PS C:\Users\Admin\Downloads\test> py app_client.py
Starting connection to ('172.16.8.130', 5050)
Type: RESPOND_CHECK_IP
Code: 205
Message: 205 "You haven't register for this device, please register!"
Type exit to exit, or press Enter to register...
> |
```

Hình 4: Client 1 khởi động và kết nối với Server

Sau khi Client 1 kết với Server, Server kiểm tra địa chỉ IP của Client 1 và thấy rằng máy chủ này không tồn tại trong hệ thống. Dòng thông báo cần đăng ký được in ra cho người dùng như hình trên.

Bước 4: Đăng ký tài khoản cho Client 1.

```
Type exit to exit, or press Enter to register...
>
>register:
Username: thi
Password:
Repeat password:
Type: RESPOND_REGISTER
Code: 200
Message: 200 "Register successfully!"
Type exit to exit, or press Enter to login...
Client listen on ('172.16.8.138', 9999)....
> |
```

Hình 5: Đăng ký tài khoản với Username là *"thi"*

Ghi chú: Người dùng đăng ký thành công, sẽ bắt đầu tạo socket để listen các peer khác với PORT mặc định là 9999.

****** Trong trường hợp tài khoản đã tồn tại, giao diện hiển thị như sau:


```
Client listen on ('172.16.8.138', 9999)....  
  
>  
>login:  
Password:  
Type: RESPOND_LOGIN  
Code: 200  
Message: 200 "Login successfully!"  
Content length: 10  
Hello thi  
Enter a command to do something.  
> █
```

Hình 6: Đăng nhập thành công.

Bước 5: *Khởi động Client 2.*

Chạy terminal trong thư mục chứa file app_client.py để client 2 và bắt đầu các tác vụ.

```
Starting connection to ('172.16.8.130', 5050)  
Type: RESPOND_CHECK_IP  
Code: 205  
Message: 205 "You haven't register for this device, please register!"  
Type exit to exit, or press Enter to register...  
> █
```

Hình 7: Bắt đầu kết nối ở terminal của Client 2

Thực hiện đăng ký - đăng nhập như Client 1

Bước 6: *Client 1:* nhập câu lệnh "publish" để publish một file lên hệ thống. Ở đây, nhóm lấy ví dụ là file **hello.txt** và sử dụng câu lệnh "*publish share1 ./ex_publish/hello.txt*" để đặt tên cho file hello.txt trong hệ thống là share1. Lúc này, Server nhận yêu cầu publish từ người dùng và ghi vào cơ sở dữ liệu:

```
Enter a command to do something.  
> publish hello ./ex_publish/hello.txt  
Type: RESPOND_PUBLISH  
Code: 200  
Message: 200 "Successfully add!"  
> █
```

Hình 8: Publish một file tên "**hello.txt**" ở folder tên "**ex_publish**" với tên trên hệ thống là "**hello**"

Tương tự với các file khác:

```

v ex_publish
  2021_MT_KHM_KH...
  hello.txt
  plan-course.pdf
  app_client.py
  app_server.py
  db_interact.py
  libclient.py
  libserver.py
File path does not exist!
> publish course ./ex_publish/2021_MT_KHM_KHMT.pdf
Type: RESPOND_PUBLISH
Code: 200
Message: 200 "Successfully add!"
> publish plan ./ex_publish/plan-course.pdf
Type: RESPOND_PUBLISH
Code: 200
Message: 200 "Successfully add!"
> 
```

Hình 9: Minh họa hàm `publish()`

****** Ta có thể kiểm tra danh sách đã publish bằng lệnh `"view"`:

```

> view
Type: RESPOND_VIEW
Code: 200
Message: 200 View all your shared files
Content length: 138
NO.      FILE      LOCATION      TYPE      SIZE
1        course    ./ex_publish/2021_MT_KHM_KHMT.pdf .pdf      515089
2        hello     ./ex_publish/hello.txt      .txt       7
3        plan      ./ex_publish/plan-course.pdf .pdf     145335
> 
```

Hình 10: Dùng lệnh `"view"` để xem danh sách các file đã publish

Bước 7: *Client 2:* nhập câu lệnh `"fetch plan"` để yêu cầu Server trả về danh sách các Client đang có file tên `"plan"`

Bước 8: *Server:* Truy cập danh sách các file đã được publish, tìm các file phù hợp với yêu cầu của Client 2 và lọc được danh sách các Client sở hữu file có tên `"plan"` và đang ở trạng thái online (sẵn sàng chia sẻ file) bằng hàm `ping(hostname)` với `hostname` là số IP Address của Client đã được convert sang dạng Integer.

```
> fetch plan
Type: RESPOND_FETCH
Code: 200
Message: 200 "Files founds: 1"
Content length: 75
NO.      FILE      TYPE      SIZE      OWNER      STATUS
1        plan      .pdf      145335     thi        online
Type online owner number you want to fetch file. Type "exit" if you don't want to fetch
>Choose file> █
```

Hình 11: Sử dụng hàm fetch để yêu cầu Server trả danh sách các chủ sở hữu đang online

****** Trong trường hợp Server không tìm thấy chủ sở hữu nào thỏa mãn, màn hình hiển thị tương tự như sau:

```
> fetch thi
Type: RESPOND_FETCH
Code: 204
Message: 204 "No match file found!"
```

Hình 12: Server không tìm thấy các chủ sở hữu file cần tìm

Bước 9: *Client 2:* Chọn 1 client trong danh sách được Server trả về bằng cách enter số thứ tự của Client đó trong danh sách ở cột No:

```
plan(copy)
>Choose file> 1
Invalid: Please choose online owner no.
>Choose file> exit
> fetch plan
Type: RESPOND_FETCH
Code: 200
Message: 200 "Files founds: 1"
Content length: 75
NO.      FILE      TYPE      SIZE      OWNER      STATUS
1        plan      .pdf      145335     thi        online
Type online owner number you want to fetch file. Type "exit" if you don't want to fetch
>Choose file> 1
Type: RESPOND_FILE
Code: 300
Message: 300 "Fetch file successfully!"
Content length: 5
<END>
> █
```

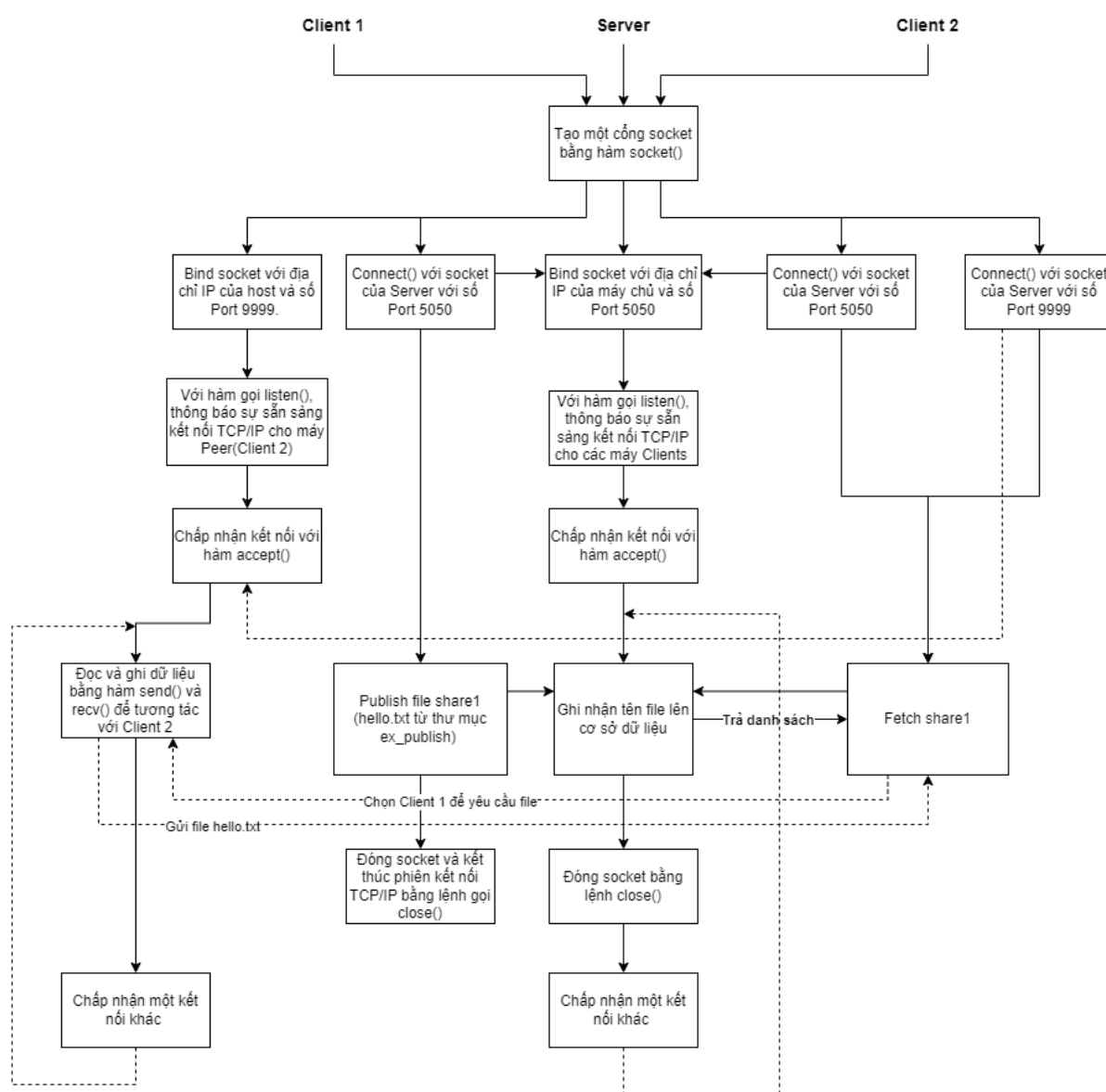
Hình 13: Tùy chọn Client bằng số thứ tự

Bước 10: *Client 1:* Nhận được tín hiệu yêu cầu file của Client 2 và tự động truy cập vào đường dẫn trong cột lname của bảng dữ liệu mà Client 1 đã dùng để publish lên Server trước đó để truy cập vào thư mục tên "ex_publish", lấy file tên "hello.txt" đi gửi cho Client 2 như ở bước 9

```
> Accepted connection from ('172.16.8.154', 51320)
Client listen on ('172.16.8.138', 9999)....

FILE ./ex_publish/plan-course.pdf
into file
Close connection to ('172.16.8.154', 51320)
```

Hình 14: Chấp nhận kết nối và gửi file cho Client 2



Hình 15: Quá trình kết nối để publish và fetch file

7 *Extend*

Phần Extend bao gồm quá trình sử dụng các chức năng:

- **Đối với Client:** Xác thực, Xem danh sách file đã publish
- **Đối với Server:** Ping (Kiểm tra trạng thái hoạt động của một Client), Discover (Xem danh sách file đã publish của một Client)

1. Client

- Xác thực
 - Đăng ký
 - * Server tự động kiểm tra địa chỉ IP của Client đã kết nối và đưa ra thông báo register nếu không tồn tại hoặc login nếu đã tồn tại tài khoản

```
PS C:\Users\Admin\Downloads\test> py app_client.py
Starting connection to ('172.16.8.130', 5050)
Type: RESPOND_CHECK_IP
Code: 205
Message: 205 "You haven't register for this device, please register!"
Type exit to exit, or press Enter to register...
> |
```

Hình 16: Đăng ký

- * Enter để nhập Username, Password, Repeat Password

```
Type exit to exit, or press Enter to register...
>
>register:
Username: thi
Password:
Repeat password:
Type: RESPOND_REGISTER
Code: 200
Message: 200 "Register successfully!"
Type exit to exit, or press Enter to login...
Client listen on ('172.16.8.138', 9999)....
> |
```

Hình 17: Đăng ký

- * Người dùng nhập password và lặp lại password sau khi dòng chữ "Password" và "Repeat Password" hiện ra. Lưu ý: password và repeat password được nhập vào sẽ không hiển thị để đảm bảo tính bảo mật

- * Sau khi enter, hệ thống trả về kết quả thành công nếu tài khoản đã được thêm vào cơ sở dữ liệu. Thất bại nếu kết nối mạng giữa các máy hoặc giữa server và database gặp sự cố hoặc tài khoản có Username/IP Address đã tồn tại.

```
>register:
Username: nha
Password:
Repeat password:
Type: RESPOND_REGISTER
Code: 206
Message: 206 "Username exists, please choose another username!"
Type exit to exit, or press Enter to register...
> █
```

Hình 18: Đăng ký thất bại vì Username đã tồn tại.

– Đăng nhập

- * Server tự động kiểm tra địa chỉ IP của Client đã kết nối và đưa ra thông báo login vì tài khoản đã có trong hệ thống

```
>login:
Password:
Type: RESPOND_LOGIN
Code: 200
Message: 200 "Login successfully!"
Content length: 16
Hello phuongnha
Enter a command to do something.
> █
```

Hình 19: Đăng nhập

- * Hệ thống hiển thị "Password" để lấy dữ liệu từ người dùng và Server kiểm tra trên database.
- * Trả về kết quả thành công hoặc thất bại cho người dùng.

– Đăng xuất

- * Sử dụng dòng lệnh "logout" để thông báo đến hệ thống yêu cầu đăng xuất.
- * Thoát tạm thời và chờ người dùng login

– Xóa tài khoản

- * Sử dụng dòng lệnh "unregister" để xóa tài khoản ra khỏi hệ thống
- * Hệ thống trả lại kết quả thành công hoặc thất bại trên màn hình

```
> unregister
Type: RESPOND_UNREGISTER
Code: 200
Message: 200 "Delete account successfully!"
```

Hình 20: Xóa tài khoản

- Xem danh sách file đã publish
 - Sử dụng dòng lệnh "view" để xem danh sách các file mà người dùng đã publish lên hệ thống
 - Server nhận yêu cầu và tra trong bảng dữ liệu danh sách các file tương ứng với địa chỉ IP của máy Client yêu cầu trả về cho người dùng

```
> view
Type: RESPOND_VIEW
Code: 200
Message: 200 View all your shared files
Content length: 100
```

NO.	FILE	LOCATION	TYPE	SIZE
1	hello	hello.txt	.txt	7
2	share1	share_file\hello.txt	.txt	23
3	share2	share_file\dbs.pdf	.pdf	736392

```
> |
```

Hình 21: Xem danh sách file đã publish

2. Server

- Ping
 - Ở terminal của Server, sử dụng dòng lệnh "ping <hostname>" để kiểm tra trạng thái online của client có địa chỉ là hostname
 - Server sử dụng hàm ping của hệ điều hành với format "*ping -n 1 <hostname>*" và kiểm tra rồi trả về kết quả True - False

```
ping 172.16.8.154

Pinging 172.16.8.154 with 32 bytes of data:
Reply from 172.16.8.130: Destination host unreachable.

Ping statistics for 172.16.8.154:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
    True
[COMMAND]Please enter "help" to show options>
```

Hình 22: Hàm ping ở terminal Server

- Discover
 - Sử dụng dòng lệnh "discover hostname" để xem danh sách file đã publish của client có địa chỉ hostname.

- Server truy cập cơ sở dữ liệu tìm các file có chủ sở hữu là hostname và trả về danh sách trên terminal.

```
[COMMAND]Please enter "help" to show options>
discover 172.16.8.154
NO.      FILE      LOCATION      TYPE      SIZE
1        hello      hello.txt     .txt      7
2        share1    share_file\hello.txt .txt      23
3        share2    share_file\dbs.pdf  .pdf      736392
[COMMAND]Please enter "help" to show options>
```

Hình 23: Hàm discover ở terminal Server

8 Source Code

Link github: <https://github.com/PhamThi1710/mmt-ass1-p2p.git>

Tài liệu tham khảo

- [1] harleenk_99, *Socket in Computer Network*.
<https://www.geeksforgeeks.org/socket-in-computer-network/>
- [2] Nathan Jennings, *Socket Programming in Python*.
<https://realpython.com/python-sockets/#background>
- [3] pankajcodehere, *File Transfer using TCP Socket in Python*.
<https://www.geeksforgeeks.org/file-transfer-using-tcp-socket-in-python/>
- [4] GeeksforGeeks, *Socket Programming with Multi-threading in Python*.
<https://www.geeksforgeeks.org/socket-programming-multi-threading-python/>
- [5] Darryn Campbell, *Python Socket Programming: Client, Server, Peer Libraries*.
<https://www.pubnub.com/blog/socket-programming-in-python-client-server-p2p/>