ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC BÁCH KHOA

□□□

MẠNG MÁY TÍNH (CO3094)

Bài tập lớn 1

File-sharing Application

GVHD: Bùi Xuân Giang

Lớp: L08

STT	MSSV	Họ và tên	Nhiệm vụ được phân công
1	2114531	Đậu Đức Quân	Thực hiện server.py, messageProtocol.py
2	2115257	Bùi Lê Văn	Thực hiện client.py, messageProtocol.py
3	2112048	Nguyễn Lê Phúc	Làm Assignment 2
4	2113499	Nguyễn Công Huy	Làm Assignment 2

Thành Phố Hồ Chí Minh, Tháng 11 -2023

MỤC LỤC

I. Function Description

1. MessageProtocol

Các loại tin nhắn (class Type):

- REQUEST: Yêu cầu từ client gửi đến server.
- RESPONSE: Phản hồi từ server trả về cho client.

Các loại tiêu đề (class Header):

- PING: Kiểm tra kết nối và Round-Trip Time giữa client và server.
- DISCOVER: Tìm kiếm thông tin về các file thuộc lưu trữ của hostname.
- PUBLISH: Đăng tải thông tin lên server.
- FETCH: Lấy dữ liệu và lưu trữ về cục bộ.
- REGISTER: Đăng ký thông tin lên server.
- RETRIEVE: Yêu cầu truy xuất thông tin tệp từ máy chủ.
- LEAVE: Ngắt kết nối với server.

Cấu trúc tin nhắn (class Message):

- __init__(self, header, type, info, json_string=None): Khởi tạo tin nhắn với tiêu đề, loại tin nhắn, thông tin và chuỗi json tùy chọn.
- get_header(self): Lấy tiêu đề của tin nhắn.
- get_type(self): Lấy loại tin nhắn.
- get_info(self): Lấy thông tin của tin nhắn.
- get_packet(self): Lấy gói tin nhắn dưới dạng từ điển.

2. Client.py

- **PORT, FTPPORT, SVHOST, SVPORT, CLNAME, CLHOST, CLPORT**: Đây là các biến toàn cục được sử dụng để cấu hình thông tin về máy chủ và máy khách.

Class Client:

- init(self, server_host, server_port, host, port): Phương thức khởi tạo này nhận vào địa chỉ IP và cổng của máy chủ, cũng như địa chỉ IP và cổng của máy khách. Nó gán các giá trị này cho các thuộc tính tương ứng của đối tượng và sau đó gọi phương thức register().
- run(self): Phương thức này khởi tạo một socket lắng nghe trên địa chỉ IP và cổng của máy khách. Nếu thư mục "downloads/" không tồn tại, nó sẽ tạo thư mục này. Sau đó, nó khởi tạo một máy chủ FTP và một luồng lắng nghe, và bắt đầu chúng. Cuối cùng, nó gọi phương thức command() để bắt đầu nhận lệnh từ người dùng.
- command(self): Phương thức này chạy trong một vòng lặp vô hạn, yêu cầu người dùng nhập một lệnh trong số các lệnh fetch, publish, leave.
- register(self): Phương thức này tạo một socket tạm thời và kết nối đến máy chủ. Nó sau đó tạo một thông điệp yêu cầu đăng ký với máy chủ, gửi thông điệp này qua socket, và sau đó chờ phản hồi từ máy chủ. Nếu máy chủ phản hồi với kết quả 'OK', nó sẽ gán tên máy chủ cho đối tượng và sau đó gọi phương thức run. Nếu không, nó sẽ ném ra một ngoại lệ với thông báo lỗi từ máy chủ.
- leave(self): Phương thức này tạo một socket tạm thời và kết nối đến máy chủ. Nó sau đó tạo một thông điệp yêu cầu rời khỏi máy chủ, gửi thông điệp này qua socket, và sau đó chờ phản hồi từ máy chủ.
- listen(self): Phương thức này bắt đầu lắng nghe các kết nối đến trên socket lắng nghe. Khi một kết nối được chấp nhận, nó sẽ tạo một luồng mới để xử lý kết nối đó bằng cách gọi phương thức reply_conn.
- reply_conn(self, rcv_sock, snd_addr): Phương thức này xử lý một kết nối đến. Nó nhận vào một socket và địa chỉ của máy gửi. Nó nhận thông điệp từ máy gửi và tạo một đối tượng `Message` từ dữ liệu nhận được. Nó sau đó kiểm tra tiêu đề của thông điệp và gọi phương thức tương ứng để trả lời thông điệp đó.

- reply_ping(self, sock): Phương thức này được gọi khi máy chủ gửi một thông điệp PING. Nó tạo một thông điệp phản hồi với payload chứa chuỗi ‘Hello’ và gửi thông điệp này trở lại máy chủ qua socket.
- reply_discover(self, sock): Phương thức này được gọi khi máy chủ gửi một thông điệp DISCOVER. Nó tạo một danh sách các tệp mà máy khách hiện có và tạo một thông điệp phản hồi với payload chứa danh sách tệp này. Thông điệp phản hồi sau đó được gửi trở lại máy chủ qua socket.
- reply_retrieve(self, sock, fName): Phương thức này được gọi khi có máy gửi một thông điệp RETRIEVE với tên tệp fName. Nếu tệp fName tồn tại trong danh sách tệp của máy khách và tệp thực sự tồn tại trên đĩa, nó sẽ tạo một thông điệp phản hồi với payload chứa chuỗi ‘Accept’ và đường dẫn đến tệp. Nếu không, nó sẽ tạo một thông điệp phản hồi với payload chứa chuỗi ‘Deny’. Thông điệp phản hồi sau đó được gửi trở lại máy gửi qua socket.
- send(self, msg: Message, sock:socket): Phương thức này mã hóa thông điệp thành định dạng JSON và gửi nó qua socket đã cho. Nếu gửi thành công, nó sẽ in ra thông báo thành công; nếu không, nó sẽ in ra thông báo thất bại.
- fetch(self, fName): Phương thức này tạo một thông điệp yêu cầu với tên tệp fName và gửi nó đến máy chủ. Nó sau đó nhận phản hồi từ máy chủ, chứa danh sách các máy khách có tệp. Nếu không có máy khách nào có tệp, nó sẽ in ra ‘NO_AVAILABLE_HOST’. Nếu có, nó sẽ yêu cầu người dùng chọn một máy khách để tải tệp từ đó. Cuối cùng, nó sẽ gọi phương thức retrieve với tên tệp và địa chỉ IP của máy khách được chọn.
- retrieve(self, fName, host): Phương thức này tạo một thông điệp yêu cầu với tên tệp fName và gửi nó đến máy khách có địa chỉ IP host. Nó tạo một socket tạm thời, và cố gắng kết nối đến máy khách qua socket này. Sau khi gửi thông điệp, nó sẽ nhận phản hồi từ máy khách. Phản hồi từ máy khách sẽ chứa kết quả của yêu cầu. Nếu kết quả là ‘DENY’, nó sẽ thoát khỏi phương thức. Nếu kết quả không phải là ‘DENY’, nó sẽ tiếp tục quá trình truyền tệp bằng giao thức FTP. Nó sẽ tạo một kết nối FTP đến máy khách, mở tệp đích để ghi, và sau đó yêu cầu máy khách gửi tệp qua kết nối FTP. Cuối cùng, nó sẽ đóng kết nối FTP.

- exit(self): Phương thức này đóng socket lắng nghe, đặt thuộc tính active của đối tượng Client thành False, và sau đó chờ các luồng lắng nghe và FTP kết thúc. Cuối cùng, nó in ra "System exited" và thoát khỏi chương trình.
- publish(self, lName, fName) Tạo một thông điệp yêu cầu xuất bản với tên tệp fName và lName. Tạo một socket tạm thời và gửi thông điệp yêu cầu qua socket. Sau đó nhận phản hồi từ máy chủ chứa kết quả của yêu cầu. Nếu kết quả là 'ERROR', nó sẽ trả về 'ERROR' và không thêm tệp vào kho lưu trữ. Nếu kết quả không phải là 'ERROR', nó sẽ thêm tệp vào kho lưu trữ và trả về kết quả.

Class FTPServer(Thread):

- init(self, host_ip): Phương thức khởi tạo này nhận vào địa chỉ IP của máy chủ và khởi tạo máy chủ FTP. Nó tạo một DummyAuthorizer để xác thực người dùng, thêm một người dùng với tên và mật khẩu, và quyền truy cập đầy đủ. Nó sau đó tạo một FTPServer chạy trên địa chỉ IP và cổng đã cho, với số lượng kết nối tối đa là 256 và số lượng kết nối tối đa từ mỗi địa chỉ IP là 5.
- run(self): Phương thức này in ra thông báo rằng máy chủ FTP đã bắt đầu và sau đó gọi phương thức serve_forever của máy chủ để bắt đầu chấp nhận các kết nối.
- stop(self): Phương thức này gọi phương thức close_all của máy chủ để đóng tất cả các kết nối hiện tại và dừng máy chủ.

3. Server.py

Biến toàn cục:

- SERVER_TIMEOUT = 5: Đây là thời gian chờ tối đa (tính bằng giây) mà máy chủ sẽ chờ trước khi ngắt kết nối với một máy khách nếu không nhận được phản hồi.
- CLPORT là cổng mà máy khách sẽ sử dụng để kết nối đến máy chủ.
- SVPORT là cổng mà máy chủ lắng nghe các kết nối đến từ máy khách

Class Server:

- __init__(self, server_port): Khởi tạo máy chủ với cổng máy chủ được chỉ định. Nó cũng tạo ra các từ điển cho bảng TCP và bắt đầu máy chủ.
- start(self): Khởi tạo socket cho máy chủ, ràng buộc nó với địa chỉ IP của máy chủ và cổng máy chủ, và bắt đầu lắng nghe các kết nối đến. Nó cũng khởi tạo một luồng mới để lắng nghe các kết nối và bắt đầu luồng đó.
- command(self): Yêu cầu người dùng nhập một yêu cầu.

  Nếu yêu cầu là "ping", nó sẽ hiển thị danh sách các máy khách hiện tại và yêu cầu người dùng chọn một máy khách để ping.

  Nếu yêu cầu là "discover", nó cũng sẽ hiển thị danh sách các máy khách hiện tại và yêu cầu người dùng chọn một máy khách để thực hiện thao tác discover.

  Nếu yêu cầu là "exit", nó sẽ thực hiện hàm exit()

- listen(self): Lắng nghe các kết nối đến từ máy khách.

  Khi một máy khách kết nối, hàm sẽ tạo một luồng mới để xử lý máy khách đó.

  Hàm handle_client sẽ được gọi với socket của máy khách, tên máy chủ và địa chỉ IP của máy khách làm đối số.

Máy chủ này sử dụng giao thức TCP để giao tiếp với máy khách. Máy chủ sẽ lắng nghe các kết nối đến từ máy khách trên cổng máy chủ được chỉ định. Khi một máy khách kết nối, máy chủ sẽ thêm máy khách đó vào bảng TCP của mình.

- register(self, client_socket:socket, message, ip): Sử dụng để đăng ký một máy khách mới với máy chủ.

  Nó nhận vào một socket máy khách, một tin nhắn và một địa chỉ IP. Nếu tên máy chủ đã tồn tại trong danh sách máy khách, hàm sẽ trả về "DULICATED". Nếu không, nó sẽ thêm máy khách vào danh sách và trả về "OK". Sau đó, nó sẽ gửi một tin nhắn phản hồi đến máy khách với kết quả của quá trình đăng ký.

- handle_client(self, client_socket, hostname, ip): Hàm này được sử dụng để xử lý các yêu cầu từ máy khách.

  Nó nhận vào một socket máy khách, một tên máy chủ và một địa chỉ IP.

  Hàm sẽ lắng nghe các tin nhắn từ máy khách. Nếu không có tin nhắn nào, nó sẽ đóng kết nối với máy khách. Nếu có tin nhắn, hàm sẽ xác định loại tin nhắn và gọi hàm tương ứng để xử lý tin nhắn đó. Các loại tin nhắn mà hàm có thể xử lý bao gồm: "REGISTER", "PUBLISH", "FETCH" và "LEAVE".

- publish(self, client_socket, hostname, message): Sử dụng để đăng tải một tệp từ máy khách lên máy chủ.

  Nó nhận vào một socket máy khách, một tên máy chủ và một tin nhắn.

  Nếu tên tệp không tồn tại trong danh sách tệp của máy chủ, hàm sẽ thêm tên tệp vào danh sách và trả về "OK". Nếu không, nó sẽ trả về "DUPLICATE".Sau đó, nó sẽ gửi một tin nhắn phản hồi đến máy khách với kết quả của quá trình đăng tải.

- ping(self, hostname): Sử dụng để kiểm tra xem một máy khách có thể kết nối đến máy chủ hay không.

  Nó nhận vào một tên máy chủ. Nếu tên máy chủ không tồn tại trong danh sách máy khách, hàm sẽ trả về "PING: NOT FOUND THIS CLIENT". Nếu tên máy chủ tồn tại, hàm sẽ tạo một socket mới, kết nối đến máy khách và gửi một tin nhắn "PING" đến máy khách.

  Nó sẽ đo thời gian phản hồi (RTT) và in kết quả ra màn hình. Nếu không nhận được phản hồi từ máy khách, hàm sẽ in ra một thông báo lỗi.

- discover(self, hostname): Sử dụng để khám phá danh sách các tệp mà một máy khách đang chia sẻ.

  Nó nhận vào một tên máy chủ. Nếu tên máy chủ không tồn tại trong danh sách máy khách, hàm sẽ in ra "DISCOVER: NOT FOUND THIS CLIENT".

Nếu tên máy chủ tồn tại, hàm sẽ tạo một socket mới, kết nối đến máy khách và gửi một tin nhắn "DISCOVER" đến máy khách.

Nó sẽ đo thời gian phản hồi (RTT) và in ra danh sách tệp và RTT.

- fetch(self, client_socket, hostname, message): Sử dụng để lấy một tệp từ một máy khách.

  Nó nhận vào một socket máy khách, một tên máy chủ và một tin nhắn yêu cầu. Hàm sẽ tìm kiếm tên tệp trong tin nhắn và tìm kiếm tên tệp đó trong danh sách tệp của máy chủ.

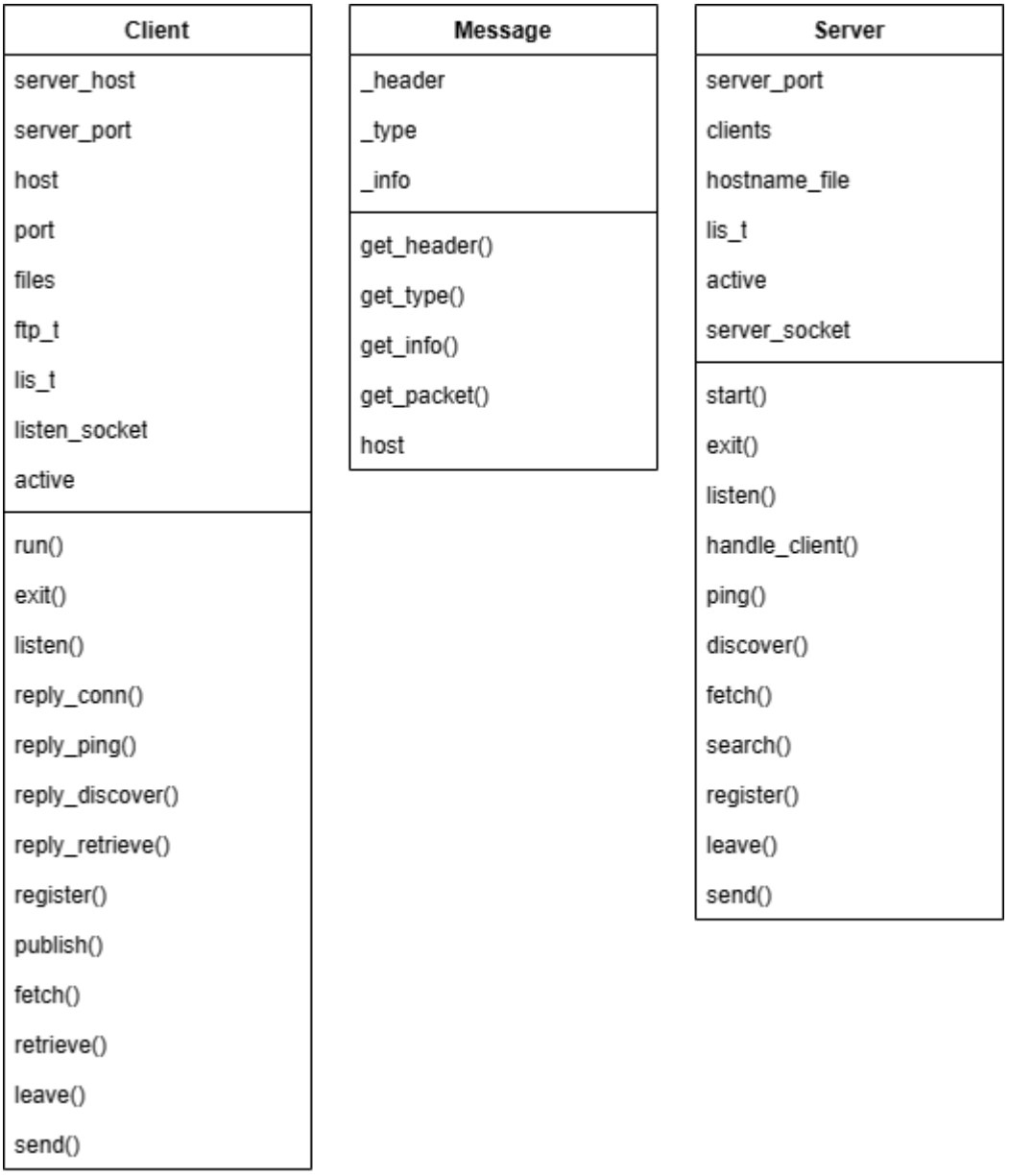
  Nếu tìm thấy tên tệp, hàm sẽ gửi một tin nhắn phản hồi đến máy khách với tên tệp và danh sách IP của các máy chủ có tệp. Nếu không tìm thấy tên tệp, hàm sẽ gửi một tin nhắn phản hồi với kết quả là "NOT FOUND".

- search(self, fname, asking_hostname): Sử dụng để tìm kiếm một tệp trong danh sách tệp của các máy khách đã kết nối đến máy chủ.

  Nó nhận vào tên tệp (fname) và tên máy chủ đang yêu cầu (asking_hostname). Hàm sẽ duyệt qua danh sách tệp của mỗi máy khách. Nếu tên tệp tồn tại trong danh sách tệp của một máy khách và máy khách đó không phải là máy khách đang yêu cầu, hàm sẽ thêm địa chỉ IP của máy khách đó vào danh sách ip_list. Cuối cùng, hàm sẽ trả về danh sách ip_list, chứa các địa chỉ IP của các máy khách có tệp được yêu cầu.

- leave(self, client_socket, hostname): Sử dụng để xử lý việc một máy khách ngắt kết nối với máy chủ. Hàm sẽ xóa máy khách khỏi danh sách máy khách của máy chủ và gửi một tin nhắn phản hồi đến máy khách với kết quả là "OK".
- exit(self): Sử dụng để dừng hoạt động của đối tượng, chờ luồng lis_t kết thúc, đóng socket máy chủ và kết thúc chương trình.
- send(self, msg: Message, sock: socket): Sử dụng để gửi một tin nhắn từ máy chủ đến một máy khách thông qua một socket. Nó nhận vào một đối tượng Message và một socket. Hàm sẽ chuyển đổi đối tượng Message thành một chuỗi JSON, sau đó mã hóa chuỗi JSON thành dạng byte và gửi nó đến máy khách thông qua socket.

II. Class Diagram



III. A summative Evaluation of Results Achieved

- Hoàn thành được giao thức publish và fetch ở phía client
- Hoàn thành được giao thức discover và ping ở phía server

Kiểm tra các chức năng của server

- Kết quả ping ở 2 client kitepea và jarvis:

```
Enter your request:ping
Current clients:{'kitepea': '192.168.7.21', 'jarvis'
: '192.168.7.37'}
Chose hostname: kitepea
Succesfull to send a PING message to 192.168.7.21
PING OK: Hello
Round-Trip Time: 0.00275970 (s)

Enter your request:ping
Current clients:{'kitepea': '192.168.7.21', 'jarvis'
: '192.168.7.37'}
Chose hostname: jarvis
Succesfull to send a PING message to 192.168.7.37
PING OK: Hello
Round-Trip Time: 0.11403465 (s)

Enter your request:
```

- Kết quả discover client jarvis:

```
Enter your request:discover
Current_clients:{'kitepea': '192.168.7.21', 'jarvis'
: '192.168.7.37'}
Chose hostname: jarvis
Succesfull to send a DISCOVER message to 192.168.7.3
7
DISCOVER OK: ['jv.txt', 'jv.mp3']
Round-Trip Time: 0.00095248 (s)

Enter your request:
```

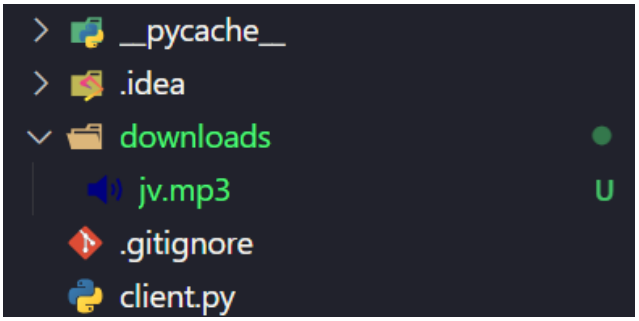
Kiểm tra các chức năng của client

- Kiểm tra kết quả publish file jv.mp3 ở client jarvis

```
Enter your request:publish
lName=jv.mp3
fName=jv.mp3
Succesfull to send a PUBLISH message to server
Enter your request:Accept Retrieve
Succesfull to send a RETRIEVE message to server
[I 2023-11-27 22:10:04] 192.168.7.21:50151-[] FTP session opened (connect)
[I 2023-11-27 22:10:04] 192.168.7.21:50151-[admin] USER 'admin' logged in.
[I 2023-11-27 22:10:04] 192.168.7.21:50151-[admin] RETR C:\Users\JS\Desktop\NetworkApp\jv.mp3 completed=1 bytes=8591 seconds=0.0
[I 2023-11-27 22:10:04] 192.168.7.21:50151-[admin] FTP session closed (disconnect).
```

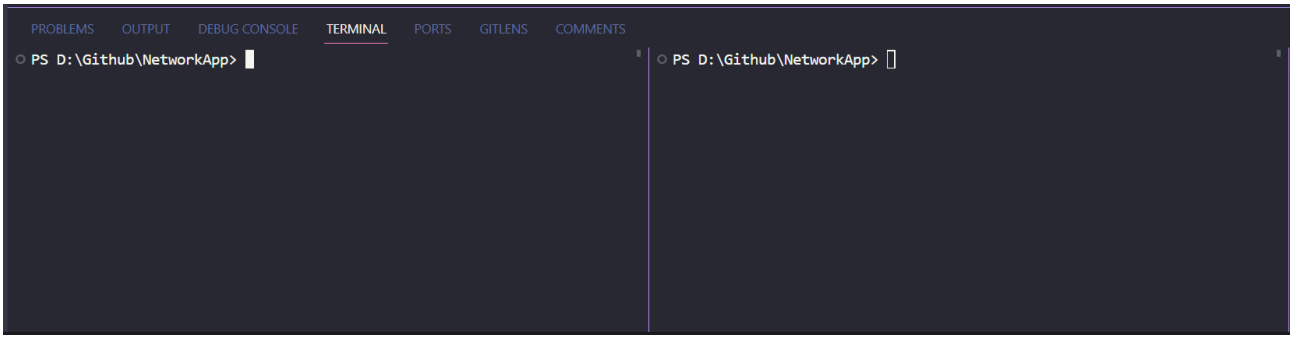
- Kiểm tra kết quả fetch của client kitepea

```
Enter your request:fetch
Type file name that you want:jv.mp3
Succesfull to send a FETCH message to server
['192.168.7.37']
Choose a host to retrieve: 192.168.7.37
Succesfull to send a RETRIEVE message to 192.168.7
.37
```



IV. User Manual

- Cd vào mục NetworkApp, mở 2 terminal cho server và client



- Tại terminal thứ nhất, tiến hành chạy dòng lệnh python server.py. Có được địa chỉ IP của server, trong ví dụ là 192.168.7.21

```
PS D:\Github\NetworkApp> python server.py
Server's running on 192.168.7.21, port: 8888
Running ... Waiting for connection
Enter your request:
```

- Vào cuối file client.py và tìm đến vị trí như hình, thay địa chỉ IP của server vừa lấy được vào SVHOST

```
#
SVHOST = '192.168.7.21'
SVPORT = 8888
CLNAME = socket.gethostname()
CLHOST = socket.gethostbyname(CLNAME)
CLPORT = 1111
```

- Tiến hành chạy dòng lệnh python client.py ở terminal thứ 2

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  COMMENTS

PS D:\Github\NetworkApp> python server.py
Server's running on 192.168.7.21, port: 8888
Running ... Waiting for connection
Enter your request:Here
Running ... Waiting for connection
OK
Curent clients active:  {'kitepea': '192.168.7.21'}
Succesfull to send a REGISTER message to 192.168.7.
21
```

```
PS D:\Github\NetworkApp> python client.py
Succesfull to send a REGISTER message to server
Running..
FTP Server start on 192.168.7.21:21
[I 2023-11-27 09:42:04] concurrency model: multi-t
hread
[I 2023-11-27 09:42:04] masquerade (NAT) address:
None
Start getting input
[I 2023-11-27 09:42:04] passive ports: None
Enter your request:
```

Ở phía server

- Để thực hiện ping, gõ “ping” và chọn hostname bằng cách gõ tên hostname cần ping

```
Enter your request:ping
Current clients:{'kitepea': '192.168.7.21'}
Chose hostname: kitepea
Succesfull to send a PING message to 192.168.7.21
PING OK: Hello
Round-Trip Time: 0.00264883 (s)

Enter your request:
```

- Để thực hiện discover, gõ "discover" và cung cấp tên hostname cần discover. Kết quả discover hostname "kitepea" hiện ra ở đây là kp.mp3

```
Enter your request:discover
Current_clients:{'kitepea': '192.168.7.21'}
Chose hostname: kitepea
Succesfull to send a DISCOVER message to 192.168.7.
21
DISCOVER OK: ['kp.mp3']
Round-Trip Time: 0.00161409 (s)
```

Ở phía client

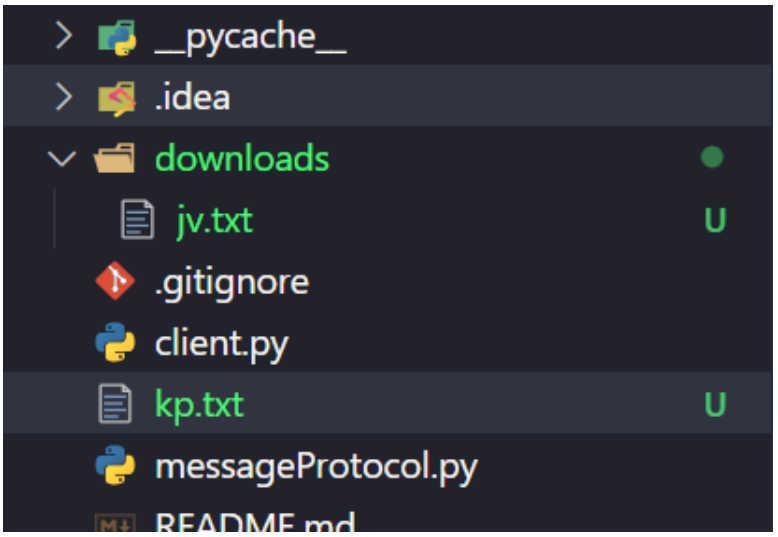
- Để thực hiện publish, gõ "publish" và lần lượt cung cấp các thông tin lName, fName

```
Enter your request:publish
lName=kp.mp3
fName=kp.mp3
Succesfull to send a PUBLISH message to server
Enter your request:Succesfull to send a DISCOVER m
essage to server
```

- Để thực hiện fetch, gõ lệnh "fetch"

```
Enter your request:fetch
Type file name that you want:jv.txt
Succesfull to send a FETCH message to server
['192.168.7.37']
Choose a host to retrieve: 192.168.7.37
Succesfull to send a RETRIEVE message to 192.168.7
.37
Enter your request:
```

- Sau đó lần lượt cung cấp các thông tin cho “Type file name that you want: “ và “Choose a host to retrieve: ”. File nhận được sẽ trả về ở thư mục downloads. Ở đây nhận được file đã yêu cầu là jv.txt



V. Extend

VI. Source Code

Truy cập source code tại: https://github.com/vanbuile/NetworkApp