

ĐẠI HỌC QUỐC GIA TP.HCM
ĐẠI HỌC BÁCH KHOA TP.HCM
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



MẠNG MÁY TÍNH (CO3093)

Assignment 1

Develop a Network Application

Giảng viên hỗ trợ: Nguyễn Phương Duy
Sinh viên: Nguyễn Thành Đạt - 2111018
Tạ Nguyễn Tiến Dũng - 2110965
Nguyễn Duy Hà - 2011130
Nguyễn Việt Anh - 2112801

Thành phố Hồ Chí Minh, Tháng 12 năm 2020



Mục lục

1	Member list & Workload	2
2	Application specifications and requirements	2
2.1	Project objectives	2
2.2	Project specifications and description	2
3	System modeling and architecture design	4
3.1	General use-case diagram	4
3.2	Activity diagram main functions	5
3.2.1	publish <i>lname fname</i>	5
3.2.2	fetch <i>fname</i>	6
3.2.3	delete <i>fname</i>	7
3.2.4	discover <i>hostname</i> (Client side)	7
3.2.5	list	8
3.2.6	discover <i>hostname</i> (Server side)	9
3.2.7	ping <i>hostname</i>	10
4	Implement and validation	11
4.1	Manual document	11
4.1.1	System requirement	11
4.1.2	Client User Configuration Requirement	12
4.1.3	User Guide	12
4.2	Source code	13
4.3	Validation and evaluation	13
4.3.1	Thêm và xóa file từ file system của client vào dữ liệu của server	13
4.3.2	Truyền file từ một client sang client khác	15
4.3.3	Ping từ server tới một client	18
4.3.4	Đánh giá theo mô tả trong bài tập lớn 1	18



1 Member list & Workload

No.	Fullname	Student ID	Problems	Percentage of work
1	Nguyễn Thành Đạt	2111018	- Code chính - Testing	30%
2	Tạ Nguyễn Tiến Dũng	2110965	- Vẽ sơ đồ, thiết kế hệ thống	25%
2	Nguyễn Duy Hà	2011130	- Hỗ trợ code - Mô tả yêu cầu chương trình	20%
2	Nguyễn Việt Anh	2112801	- Viết báo cáo	25%

2 Application specifications and requirements

2.1 Project objectives

Mục đích của bài tập lớn môn Mạng Máy tính học kì 1 năm học 2023-2024 là xây dựng một chương trình chia sẻ file đơn giản, với giao thức chương trình do mỗi nhóm quyết định, dựa trên các giao thức TCP/IP đã học.

2.2 Project specifications and description

Mô tả chương trình trong đề bài bài tập lớn 1:

- Một máy chủ (server) trung tâm đóng vai trò theo dõi, quản lí các client đã kết nối và các file mà chúng giữ.
- Một client sẽ thông báo cho server biết có những file gì trong thư mục tại chỗ (local repository) nhưng sẽ không truyền những file đó cho server.
- Khi một client cần một file không nằm trong thư mục của nó, một yêu cầu sẽ được gửi đến server. Server có nhiệm vụ xác định những clients khác có chứa file được yêu cầu và gửi thông tin kết nối của chúng đến client yêu cầu. Client đó sẽ lựa chọn nguồn lấy phù hợp và tiến hành nhận file trực tiếp từ đó mà không qua can thiệp của server. (Loại mạng kết nối Peer-to-peer)
- Nhiều client có thể tải nhiều file khác nhau từ cùng một client gốc tại bất kì thời điểm nào
- Client có giao diện câu lệnh (Command line interface - CLI) đơn giản nhận hai câu lệnh:
 - **publish *lname fname***: một file trong file system của client đã được thêm vào local repository của client đó và thông tin này được truyền đến cho server.
 - **fetch *fname***: lấy một bản copy từ file mục tiêu và thêm nó vào local repository.
- Server cũng sẽ có CLI nhận hai câu lệnh:
 - **discover *hostname***: cho ra danh sách những file trong local repository của *hostname*.

- **ping *hostname***: kiểm tra trực tiếp *hostname*, tương tự như câu lệnh SNMP ping.

Bên cạnh đó, dựa trên ý kiến của các thành viên, nhóm có thêm các yêu cầu dựa trên các chức năng thêm và cách xử lý khi gặp các trường hợp đặc biệt:

- Với các câu lệnh bên phía client:
 - Câu lệnh **publish *lname fname*** sẽ tiến hành thêm file từ trong máy của client vào local repo và thông báo tới server. Nếu publish một file không tồn tại sẽ báo lỗi "File not exist!". Nếu file publish đã nằm trong local repo sẵn rồi báo "File already in local repo!".
 - Cách xử lý của câu lệnh **fetch *fname*** của client trong các trường hợp
 1. Nếu file đó không nằm trong hệ thống server quản lý thì báo lỗi "File not recognized!"
 2. Nếu file đó đã nằm trong repo của client thì thông báo đã file đã tồn tại trong local repository
 3. Nếu file đó không nằm trong repo thì thực hiện theo như đề bài yêu cầu
 4. Nếu client chứa file đó không kết nối được thì báo lỗi kết nối

Bên cạnh đó, việc fetch một file vào local repository cũng sẽ đồng thời thông báo với server, tức là file tên *fname* có mặt trên ít nhất hai client.

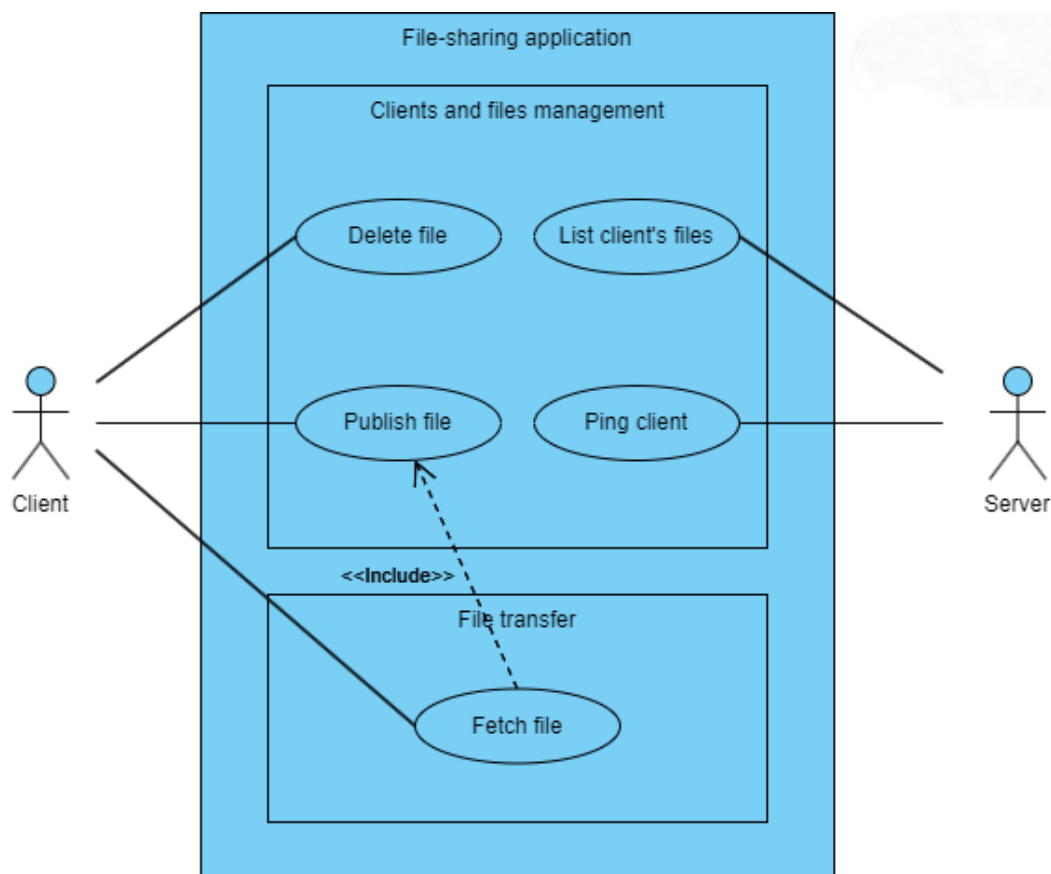
- Thêm câu lệnh **delete *filename***: Xóa một file từ local repo ra khỏi hệ thống i.e server sẽ không còn lưu thông tin về file này trên client nữa. Báo lỗi "Cannot delete! File not in local repo!" nếu file đó không nằm trong local repo của client yêu cầu. Tức là một client chỉ có thể xóa một file trong local repo của bản thân.
- Thêm câu lệnh **discover *hostname***: In ra danh sách file mà *hostname* đã publish trên server. Nếu một *hostname* chưa từng kết nối với server từ trước thì báo lỗi "Host not recognize!". Nếu *hostname* đã kết nối nhưng chưa publish bất kì file thì báo "*hostname* has not published any files!".
- Thêm câu lệnh **list**: In ra danh sách các client đã từng kết nối với server. Khi đó một client kết hợp câu lệnh này với câu lệnh **discover *hostname*** để tìm được file mình muốn chuyển.
- Với câu lệnh bên phía server:
 - Lưu ý của câu lệnh **discover *hostname***: Chức năng và lưu ý tương tự như với câu lệnh ở bên client.
 - **ping *hostname***: Tương tự như câu lệnh ping trong command prompt, gửi vài packet/message nhỏ đến *hostname* rồi chờ phản hồi.
- Client sẽ biết trước thông tin kết nối của server.
- Client cần có cơ chế tạo, quản lý nhiều cổng kết nối sử dụng đa luồng để nhiều client có thể lấy file từ một client cùng lúc.
- Các client không cần liên tục kết nối với server để thực hiện việc truyền file, miễn là vẫn nằm trong cùng một network.
- Server lưu thông tin về các client đã kết nối cũng như các file họ đã publish trong hệ dữ liệu của chính nó (tức là nếu tắt server đi thì dữ liệu này cũng sẽ biến mất).

- Server cũng cần có cơ chế tạo, quản lý nhiều cổng kết nối cùng một lúc.
- Có thể có nhiều bản copy của một file trên nhiều client khác nhau. Tức là một tên file có thể tồn tại ở nhiều client khác nhau
- Hệ thống không nhận diện các file trùng lặp trong local repo của client. Tức là kể cả khi client có nhiều file trùng lặp thì hệ thống ghi nhận chỉ một file trong local repo. (**publish *lname fname*** không cho phép file trùng lặp được publish lên server)

3 System modeling and architecture design

3.1 General use-case diagram

Hệ thống có 2 actor, bao gồm server và client. Bên cạnh đó, hệ thống cũng có hai module xử lý hai dịch vụ khác nhau của chương trình: quản lý file và client (clients and files management), truyền file (file transfer). Với mỗi module đó là các use-case ứng với mỗi câu lệnh và chức năng của từng actor.



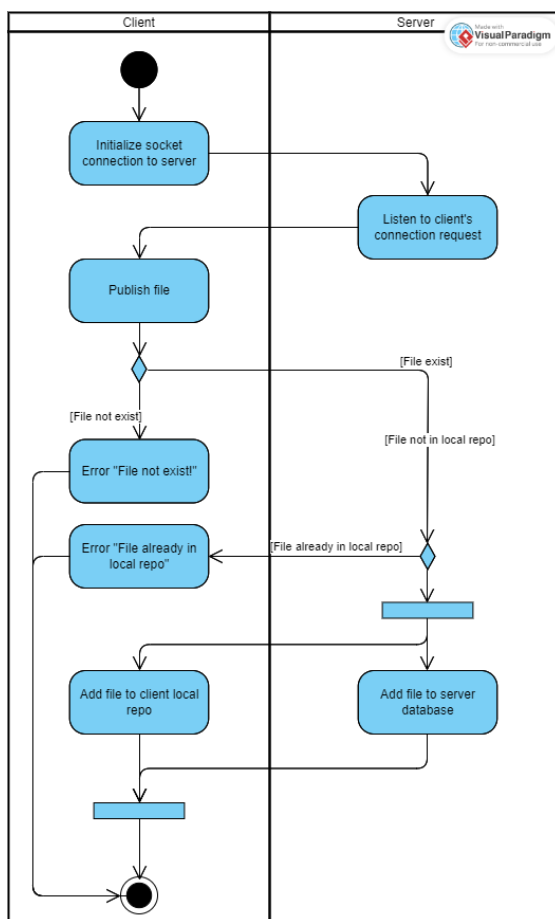
Hình 1: Lược đồ use-case tổng quát

3.2 Activity diagram main functions

Ứng với mỗi câu lệnh (use-case) chính của từng actor, ta có được các lược đồ hoạt động (activity diagram) dưới đây. Ở đây lưu ý, khi lược đồ đi đến final node thì cũng đồng thời đóng mọi kết nối socket đã thiết lập trước đó, nhằm tránh việc tạo quá nhiều kết nối không cần thiết.

3.2.1 publish *lname fname*

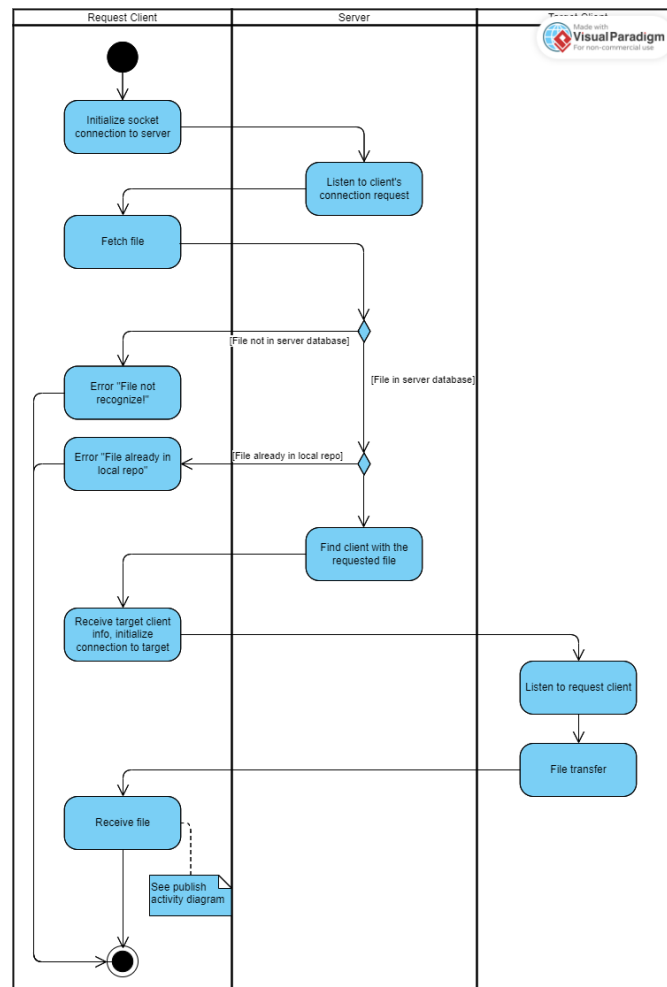
Client sẽ đầu tiên gửi yêu cầu kết nối với server, server sẽ lắng nghe yêu cầu từ một socket đang chạy của mình, thiết lập kết nối giữa client và server. Sau đó, client thực hiện việc publish file lên dữ liệu của server. Kiểm tra tại client file đó có tồn tại không. Nếu có thì tiếp tục use-case, không thì báo lỗi và kết thúc. Sau đó, server sẽ kiểm tra xem client này đã publish file này vào local repo trước đó chưa. Nếu có rồi thì báo lỗi và kết thúc, nếu không thì tiếp tục use-case. Server sẽ thêm file đó vào dữ liệu của mình và client thì thêm file đó vào local repo trên máy. Cuối cùng là đóng kết nối và kết thúc use-case.



Hình 2: Lược đồ hoạt động của câu lệnh **publish *lname fname***

3.2.2 fetch *fname*

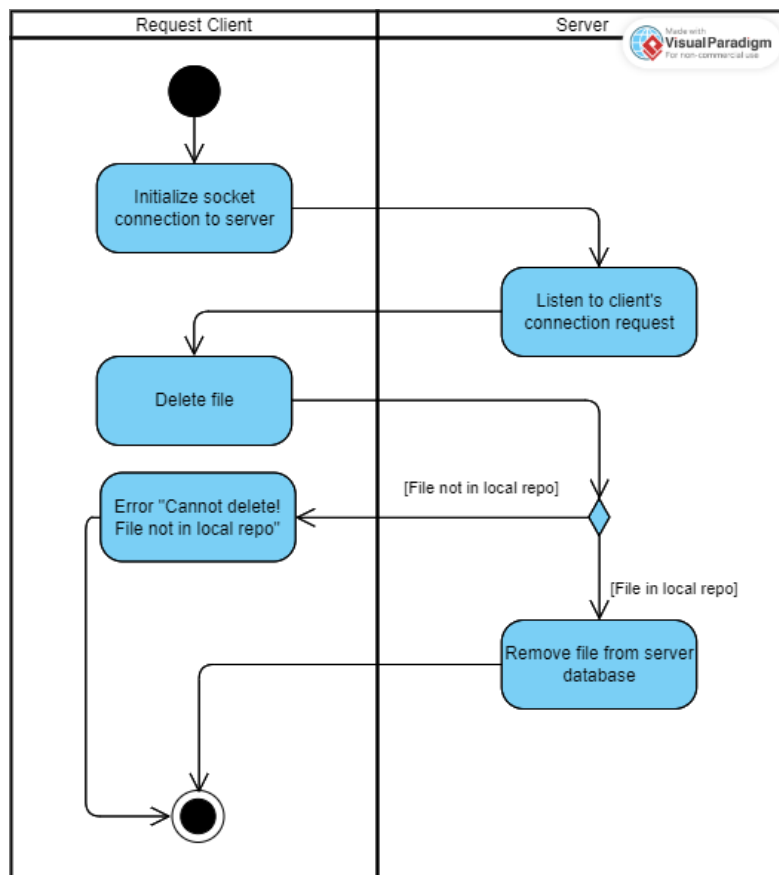
Thiết lập kết nối tương tự như **publish**. Tuy nhiên, trong lúc implement thực tế chương trình, nhóm có sửa đổi thứ tự hoạt động khác lược đồ một chút để thuận tiện cho việc code. Trong đó, sau khi client gửi yêu cầu fetch file, server sẽ kiểm tra file đó đã có trong local repo chưa. Nếu có rồi thì báo lỗi và kết thúc, nếu không thì tiếp tục use-case. Sau đó, server tiến hành tìm client có file được yêu cầu. Nếu không tìm được i.e file không nằm trong dữ liệu server thì báo lỗi và kết thúc, còn nếu tìm được thì gửi thông tin kết nối của target client cho client yêu cầu. Sau đó client yêu cầu thiết lập kết nối với target client và tiến hành truyền file. Cuối cùng, đóng mọi socket đã thiết lập (cả với server và target client).



Hình 3: Lược đồ hoạt động của câu lệnh **fetch *fname***

3.2.3 delete *fname*

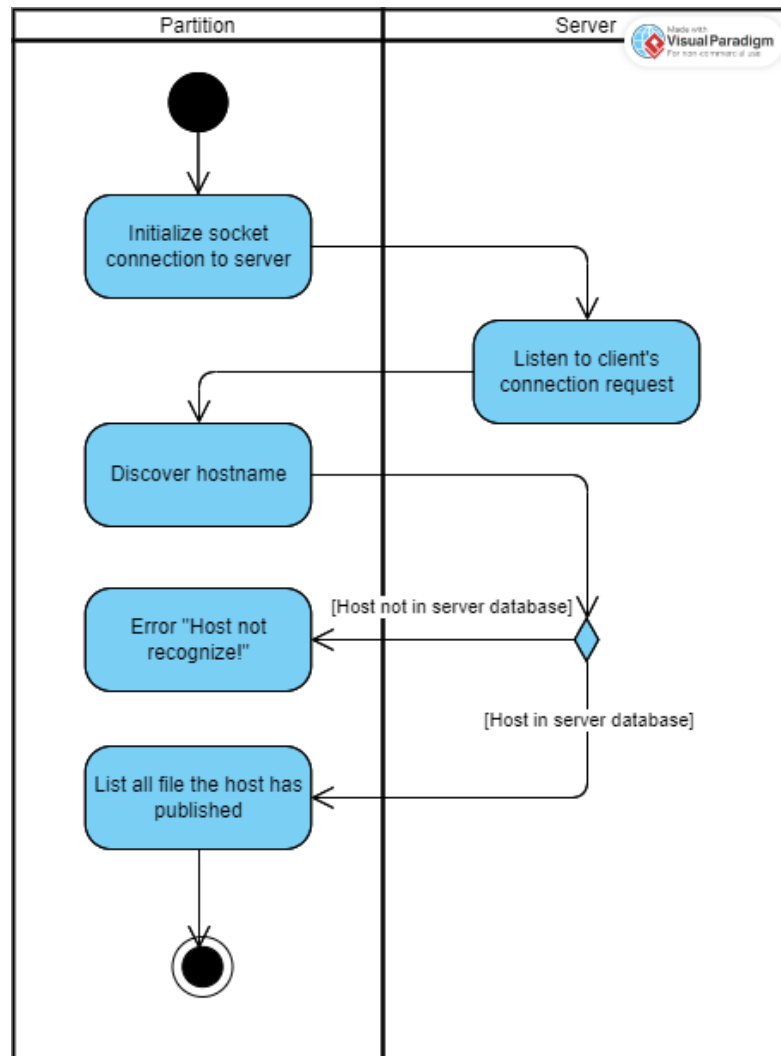
Thiết lập kết nối tương tự hai câu lệnh trước. Server sẽ tiến hành kiểm tra xem file đó có trong local repo của client hay không. Trường hợp có thì tiến hành xóa khỏi dữ liệu server, trường hợp không thì đóng kết nối và kết thúc.



Hình 4: Lược đồ hoạt động của câu lệnh **delete *fname***

3.2.4 discover *hostname* (Client side)

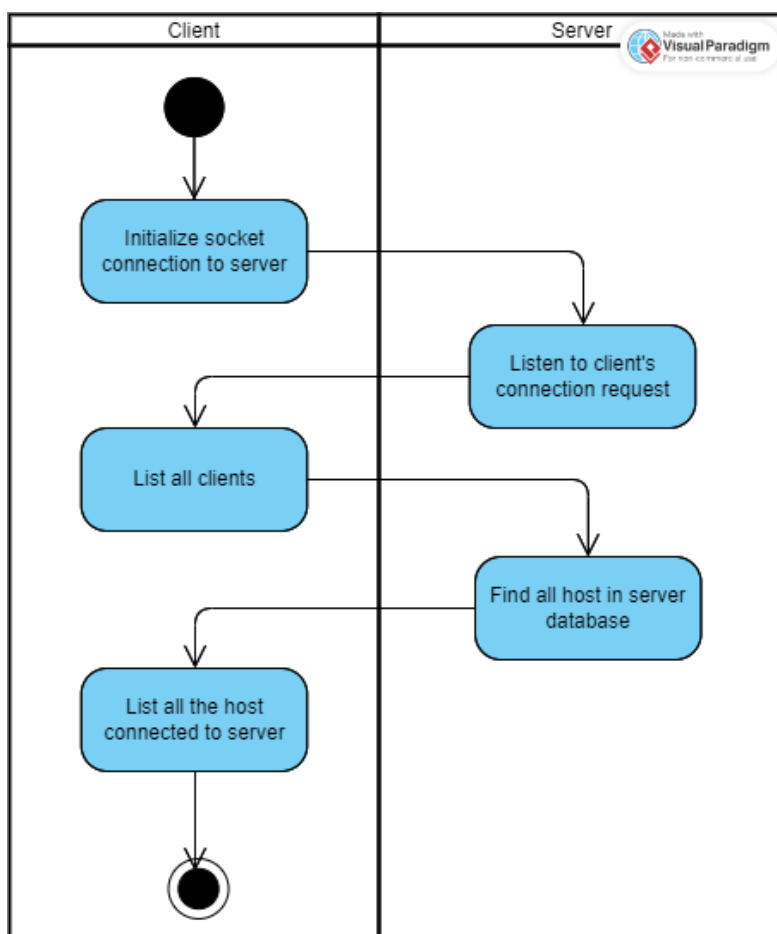
Client gửi dữ liệu tới server về *hostname*. Server sẽ kiểm tra xem nếu *hostname* đã từng kết nối từ trước chưa. Nếu chưa thì thông báo lỗi và kết thúc, còn không thì gửi về client danh sách các file mà *hostname* đã publish. Cách hoạt động và lưu ý khi xử lý lỗi tương tự với câu lệnh **discover *hostname*** bên server.



Hình 5: Lược đồ hoạt động của câu lệnh **discover *hostname*** của client

3.2.5 list

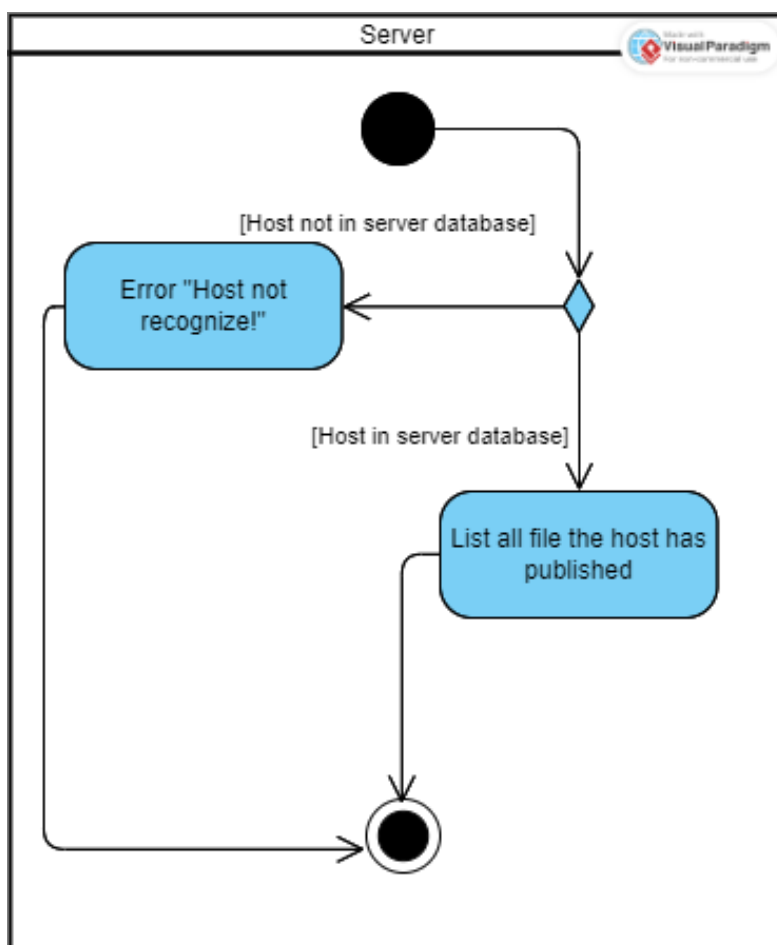
Client gửi yêu cầu tới server để biết về toàn bộ các client đã từng kết nối với server trước đây.



Hình 6: Lược đồ hoạt động của câu lệnh **list**

3.2.6 discover *hostname* (Server side)

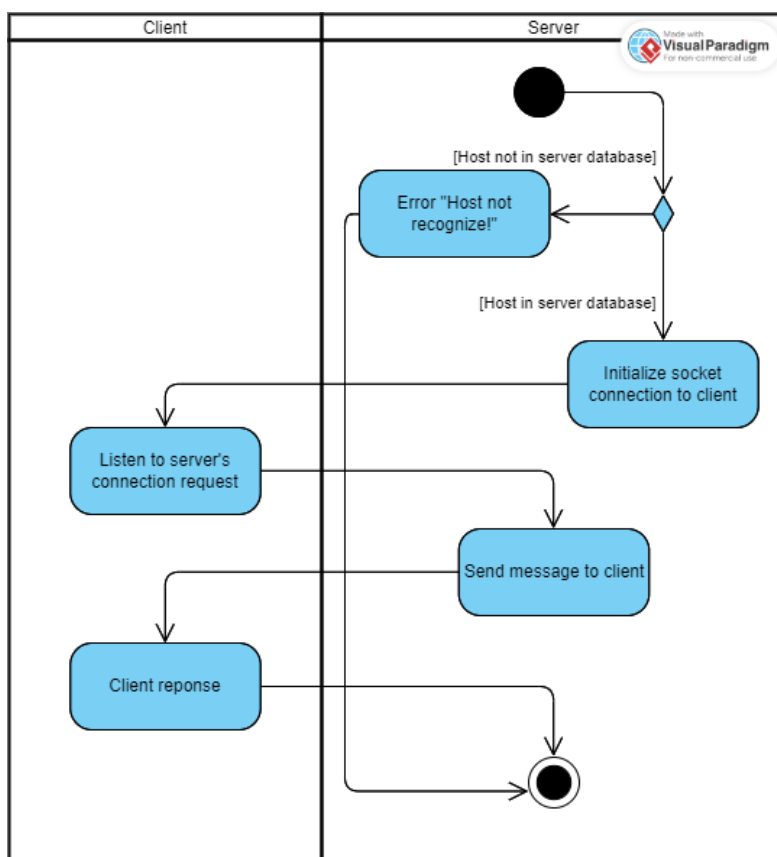
Từ dữ liệu server đã nhận được từ việc các client publish file của mình lên. Server kiểm tra xem *hostname* đã từng kết nối chưa. Trường hợp có rồi và đã publish file thì cho ra danh sách các file trong local repo, còn nếu chưa publish file thì cũng thông báo ra màn hình. Trường hợp chưa từng kết nối thì báo lỗi và kết thúc. Câu lệnh này chỉ cần duy nhất server thực hiện, không cần kết nối với client nào.



Hình 7: Lược đồ hoạt động của câu lệnh **discover hostname** của server

3.2.7 ping hostname

Dùng để ping tới một *hostname* đã kết nối với server từ trước. Nếu đã kết nối từ trước thì thực hiện việc trao đổi message xác nhận cho nhau. Còn nếu chưa thì báo lỗi và kết thúc.



Hình 8: Lược đồ hoạt động của câu lệnh **ping hostname**

4 Implement and validation

4.1 Manual document

Đây là phần hướng dẫn sử dụng ứng dụng chia sẻ file của Bài tập lớn 1 môn Mạng Máy Tính

4.1.1 System requirement

Yêu cầu về hệ thống để ứng dụng hoạt động:

- Hệ điều hành Windows có hỗ trợ Python (đã test trên Win 11, có test trên máy ảo Ubuntu 22.04). File `Client_for_Linux` chỉ dùng để test trên máy ảo sử dụng Linux. Trong phạm vi bài tập lớn này, ứng dụng sẽ chạy hoàn toàn trên các máy sử dụng hệ điều hành Windows.
- Cả server và client có chung mạng cục bộ (LAN - Local Area Network) với nhau, đảm bảo ping nhau qua lại được trên terminal của máy.
- Có cài đặt Python, tốt nhất là Python 3 từ bản 3.11 trở xuống (Bản 3.12 khi test đã không thể tạo được thread mới, đây là lỗi thuộc về Python Interpreter 3.12 nên nó nằm ngoài

phạm vi xử lý của bài tập lớn). Đồng thời có cài đặt module "Typer" cho ứng dụng CLI.

4.1.2 Client User Configuration Requirement

Yêu cầu cấu hình từ người dùng client để ứng dụng hoạt động được bình thường:

1. Trong phạm vi của bài tập lớn này, server là một máy tính cá nhân của thành viên trong nhóm, do đó địa chỉ IP của server có thể thay đổi, cần phải sửa biến global `SERVER_HOST` trên từng client.
2. Qua module socket, Python có thể lấy được địa chỉ IP của máy client. Tuy nhiên, vì thực tế cấu hình mỗi máy có nhiều địa chỉ IP, mà địa chỉ IP thực sự dùng để kết nối lại tùy thuộc vào loại kết nối mà máy đang sử dụng (Wifi, Ethernet, ...). Do đó cần người dùng thay đổi cách lấy địa chỉ IP trong file `Client.py` để chương trình có thể hoạt động:
 - Tìm các địa chỉ IP của máy bằng việc gõ câu lệnh `ipconfig` vào terminal. Trên terminal sẽ có thể hiện nhiều địa chỉ IP, trong đó địa chỉ IP có trường **Default Gateway** không bị bỏ trống chính là địa chỉ IP thực sự dùng để kết nối mạng.
 - Địa chỉ IP của máy được tìm thông qua đoạn code ở phần đầu file `Client.py`: `CLIENT_HOST = socket.gethostbyname_ex(socket.gethostname())[2][X]`. Trong đó, **X** là thứ tự của địa chỉ IP có trường **Default Gateway** tìm thấy ở trên trong số các địa chỉ IP được terminal hiện ra.
Một máy Windows có thể có nhiều địa chỉ IP bao gồm cho kết nối Ethernet, Wireless (như Wifi), WSL (Windows Subsystem for Linux),...

4.1.3 User Guide

Dành cho Server user:

1. Bật server bằng việc cho chạy file `"Server.py"`
2. Từ đây có thể bắt đầu sử dụng CLI của server. Chuyển terminal đến vị trí folder chứa file `"Server_app.py"`. Gõ câu lệnh (command) theo cú pháp: `py Server_app.py` + câu lệnh
3. CLI của server có những câu lệnh sau:
 - **discover hostname**: In ra danh sách file mà `hostname` có trong local repo. Lưu ý, `hostname` ở đây là địa chỉ IP.
 - **ping hostname**: Ping đến `hostname` bằng việc gửi các gói tin và chờ phản hồi, tương tự câu lệnh ping trong terminal

Dành cho Client user:

1. Bật client bằng việc cho chạy file `"Client.py"`
2. Từ đây có thể bắt đầu sử dụng CLI của client. Chuyển terminal đến vị trí folder chứa file `"Client_app.py"`. Gõ câu lệnh (command) theo cú pháp: `py Client_app.py` + câu lệnh
3. CLI của client có những câu lệnh sau:
 - **publish lname fname**: Thêm file tên `fname` từ trong máy của client tại địa chỉ `lname` vào local repo và thông báo tới server.
 - **fetch fname**: Lấy file tên `fname` từ một trong các client đã kết nối tới server

- **delete *filename***: Xóa file tên *fname* từ local repo ra khỏi local repo của client trong server
 - **discover *hostname***: In ra danh sách file mà *hostname* đã publish trên server.
 - **list**: In ra danh sách các client đã từng kết nối với server. Một client có thể kết hợp câu lệnh này với câu lệnh **discover *hostname*** để tìm được file mình muốn chuyển.
4. **Lưu ý khi sử dụng CLI**: với các trường hợp tên file hoặc tên đường dẫn có chứa khoảng trống, khi nhập vào terminal phải đóng trong dấu ngoặc kép. Khi nhập tên file cần nhập luôn cả file extension (.txt, .doc, .png, ...). Đường dẫn file sẽ là đường dẫn tuyệt đối, chỉ tới thư mục chứa file đó.

4.2 Source code


Protocol trao đổi Message qua lại giữa các client và server trong việc thực hiện các câu lệnh tuân theo cú pháp: (địa chỉ IP)*(Tên câu lệnh)*(Danh sách input của câu lệnh). Về source code của chương trình, xem các file `Client.py`, `Client_for_Linux.py`, `Client_app.py`, `Server.py`, `Server_app.py` trong hai thư mục *Client*, *Server*

4.3 Validation and evaluation

Tiến hành việc chạy thử và kiểm tra chương trình. Thực hiện theo các bước như trong phần 4.1, tiến hành chạy các câu lệnh

4.3.1 Thêm và xóa file từ file system của client vào dữ liệu của server

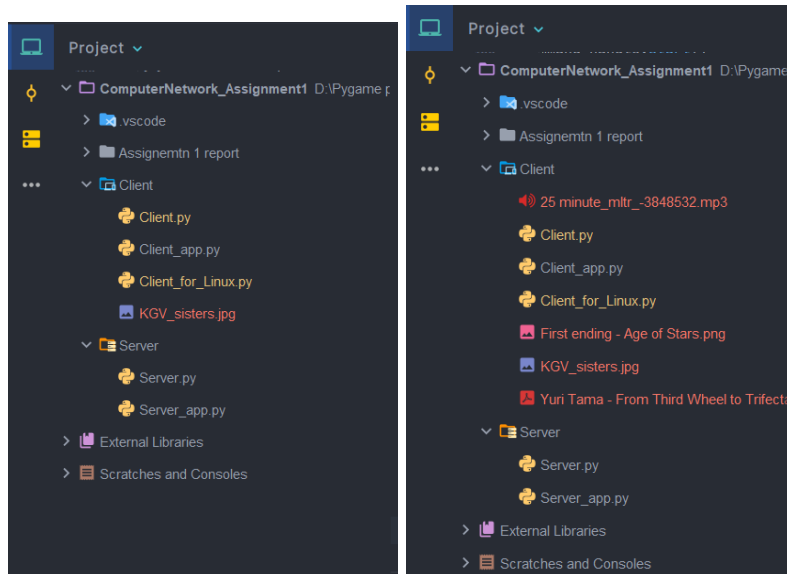
Tiến hành publish ba file từ client lên server.



```
Terminal Local (2) x + v*
py
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(base) PS D:\Pygame projects\ComputerNetwork_Assignment1\Client> py Client_app.py publish "C:\Users\Admin\OneDrive\Pictures\Elden Ring" "First ending - Age of Stars.png"
File publish from system to local repo successfully!
(base) PS D:\Pygame projects\ComputerNetwork_Assignment1\Client> py Client_app.py publish "D:\Ca nhac\cac cua dat 4" "25 minute_nltr_3848532.ap3"
File publish from system to local repo successfully!
(base) PS D:\Pygame projects\ComputerNetwork_Assignment1\Client> py Client_app.py publish "E:\Download LM\Yuri Tama" "Yuri Tama - From Third Wheel to Trifecta vol 1 Premium.pdf"
File publish from system to local repo successfully!
(base) PS D:\Pygame projects\ComputerNetwork_Assignment1\Client>
```

Hình 9: Câu lệnh **publish** từ client



(a) Trước khi publish

(b) Sau khi publish

Hình 10: Client trước và sau khi publish

Kiểm tra file trên server.



Hình 11: Câu lệnh **discover** từ server

Sau đó, delete một file ra khỏi dữ liệu của server.

```
(base) PS D:\Pygame projects\ComputerNetwork_Assignment1\Client> py Client_app.py publish "E:\Download LM\Yuri Tama" "Yuri Tama.mp3"
File publish from system to local repo successfully!
(base) PS D:\Pygame projects\ComputerNetwork_Assignment1\Client> py Client_app.py delete "25 minute_mltr_-3848532.mp3"
Delete successfully
(base) PS D:\Pygame projects\ComputerNetwork_Assignment1\Client>
ComputerNetwork_Assignment1 > Server > Server.py CRLF UTF-8
```

Hình 12: Câu lệnh **delete** từ client

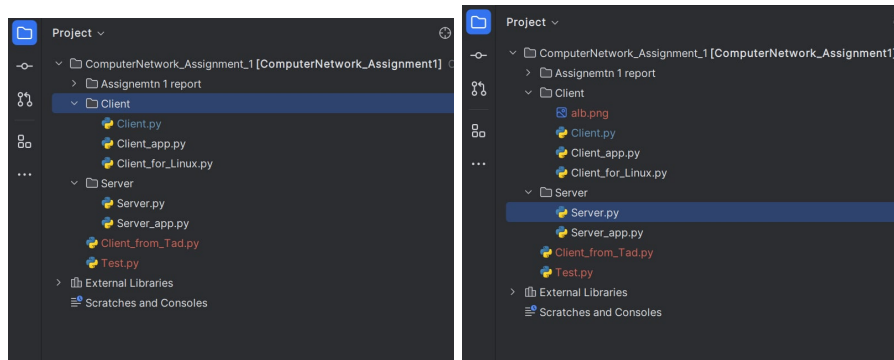
Kiểm tra lại trên server.

```
(base) PS D:\Pygame projects\ComputerNetwork_Assignment1\Server> py Server_app.py discover 192.168.31.48
Host not recognize
(base) PS D:\Pygame projects\ComputerNetwork_Assignment1\Server> py Server_app.py discover 192.168.1.28
['First ending - Age of Stars.png', 'Yuri Tama - From Third Wheel to Trifecta vol 1 Premium.pdf']
(base) PS D:\Pygame projects\ComputerNetwork_Assignment1\Server>
ComputerNetwork_Assignment1 > Server > Server.py
```

Hình 13: Câu lệnh **publish** từ client

4.3.2 Truyền file từ một client sang client khác

Tiếp tục từ trường hợp phần trước, lần này có thêm một client mới (client 2) cũng publish một file của mình lên server.



(a) Client 2 trước khi publish

(b) Client 2 sau khi publish

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

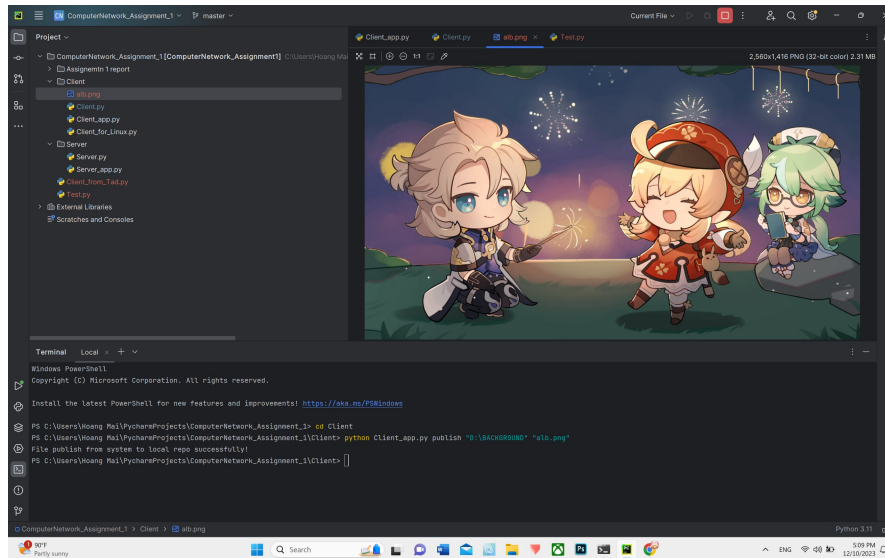
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Hoang Mai\PycharmProjects\ComputerNetwork_Assignment_1> cd Client
PS C:\Users\Hoang Mai\PycharmProjects\ComputerNetwork_Assignment_1\Client> python Client_app.py publish "D:\BACKGROUND" "alb.png"
File publish from system to local repo successfully!
PS C:\Users\Hoang Mai\PycharmProjects\ComputerNetwork_Assignment_1\Client>
```

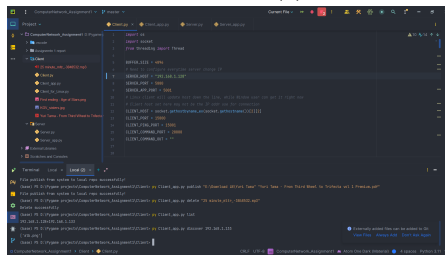
(c) Câu lệnh **publish** của client 2

Hình 14: Client 2 thực thi câu lệnh **publish**

Client 1 list tìm ra các client đã kết nối và discover client 2 để tìm file cần fetch. Client 1 tiến hành fetch một file từ client 2.



(a) File dùng để fetch của client 2



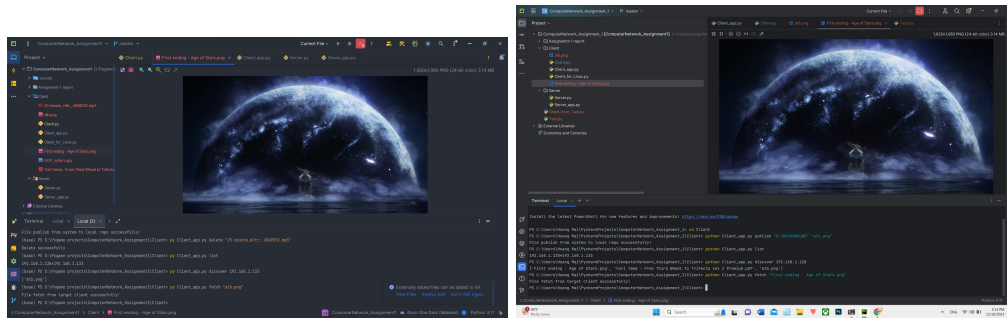
(b) Câu lệnh **list** và **discover** của client 1



(c) Câu lệnh **fetch** của client 1

Hình 15: Client 1 thực thi câu lệnh **list**, **discover**, **fetch**

Tương tự, client 2 cũng list và discover client 1 để tìm file cần fetch. Có thể thấy sau khi fetch file, client 1 cũng đồng thời publish file đó lên cho server cùng biết. Sau đó, client 2 tiến hành fetch một file từ client 1.



(a) File dùng để fetch của client 1

(b) Câu lệnh **fetch** của client 1

Hình 16: Client 2 thực thi câu lệnh **fetch**

4.3.3 Ping từ server tới một client

Ping từ server tới một *hostname*. Có thể thấy, trong trường hợp *hostname* đó chưa từng kết nối với server thì server sẽ không nhận ra được *hostname* đó.

```
(base) PS D:\Pygame projects\ComputerNetwork_Assignment1\Server> py Server_app.py ping 192.168.1.28
Ping to 192.168.1.28. Acknowledge received!
py
Ping to 192.168.1.28. Acknowledge received!
Ping to 192.168.1.28. Acknowledge received!
Ping to 192.168.1.28. Acknowledge received!
(base) PS D:\Pygame projects\ComputerNetwork_Assignment1\Server> py Server_app.py ping 192.168.31.48
Host not recognize
Host not recognize
Host not recognize
Host not recognize
(base) PS D:\Pygame projects\ComputerNetwork_Assignment1\Server>
```

Hình 17: Câu lệnh **ping** từ server

4.3.4 Đánh giá theo mô tả trong bài tập lớn 1

1. Chương trình đã xây dựng được một ứng dụng theo kiểu mạng P2P.
2. Các chức năng chính được nêu trong phần mô tả bài tập lớn đã được hiện thực và hoạt động bình thường.
3. Các client có thể gửi file tới nhiều client khác cùng một lúc.
4. Xây dựng được một command line interpreter với câu lệnh tương ứng cho bên phía server lẫn client.