

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



ASSIGNMENT 1
MÔN MẠNG MÁY TÍNH (CO3094)

Assignment report – version 0.1

NETWORK APPLICATION P2P
FILE SHARING

Giảng viên hướng dẫn:	Bùi Xuân Giang	
Sinh viên thực hiện:	Võ Tấn Hưng	2113623
	Cù Hoàng Nguyễn Sơn	2112185
	Nguyễn Đức An	2112737
	Nguyễn Trường Tuấn Anh	2112796

Tp. Hồ Chí Minh, Tháng 10/2023

Mục lục

1	Phân công nhiệm vụ	3
2	Đề bài	4
2.1	Objectives	4
2.2	Application descriptions	4
3	Mô tả hệ thống	5
3.1	Phân tích yêu cầu	5
3.1.1	Yêu cầu chức năng	5
3.1.2	Yêu cầu phi chức năng	6
3.2	Giao thức	7
3.2.1	Giữa Client-Server	7
3.2.2	Giữa Peer-to-Peer	9
3.3	Class diagram	11
3.4	Thiết kế kiến trúc	12
3.5	Các công nghệ được sử dụng	13
4	Hướng dẫn sử dụng	17
5	Kiểm tra và chạy thử ứng dụng	17
5.1	Chạy Tracker	17
5.2	Chạy Peer	18
5.3	Test tính năng start của Tracker	20
5.4	Kết nối 2 peer vào địa chỉ ip và port của tracker	20
5.5	Test tính năng liệt kê các peers "list_peers" của Tracker	21
5.6	Tính năng publish của Peer có port (62825) và ip (192.168.34.131)	21
5.7	Tính năng list_files của Tracker và Peer	22
5.8	Tính năng discover của Tracker	23
5.9	Tính năng list_chunkinfo của Tracker	23
5.10	Tính năng ping của Tracker	24



5.11 Tính năng fetch của Peer	24
-----------------------------------------	----

6 Tài liệu tham khảo	26
-----------------------------	-----------

1 Phân công nhiệm vụ

STT	Tên	MSSV	Phân chia công việc	Phần trăm hoàn thành
1	Võ Tấn Hưng	2113623	Hiện thực phần Back-end - Hiện thực Peer - Hiện thực Tracker - Hiện thực DownloadManager - Viết API cho Front-end - Kiểm thử ứng dụng - Viết báo cáo phần 3.2, 3.3, 3.4, 3.5	100%
2	Cù Hoàng Nguyễn Sơn	2112185	Hiện thực phần Front-end - Hiện thực giao diện Peer - Hiện thực giao diện Tracker - Kết nối API - Viết báo cáo phần 3.5 và 4	100%
3	Nguyễn Đức An	2112737	- Viết báo cáo - Viết báo cáo phần 2, 3.1, 5 - Kiểm thử ứng dụng	100%
4	Nguyễn Trường Tuấn Anh	2112796	- Viết báo cáo - Viết báo cáo phần 5, 6 - Kiểm thử ứng dụng	100%

Bảng 1: Danh sách công việc

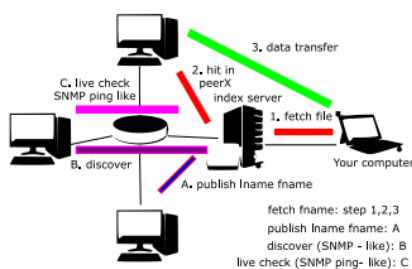
2 Đề bài

2.1 Objectives

- Build a simple file-sharing application with application protocols defined by each group, using the TCP/IP protocol stack.

2.2 Application descriptions

- A centralized server keeps track of which clients are connected and storing what files.
- A client informs the server as to what files are contained in its local repository but does not actually transmit file data to the server.
- When a client requires a file that does not belong to its repository, a request is sent to the server. The server identifies some other clients who store the requested file and sends their identities to the requesting client. The client will select an appropriate source node and the file is then directly fetched by the requesting client from the node that has a copy of the file without requiring any server intervention.
- Multiple clients could be downloading different files from a target client at a given point in time. This requires the client code to be multithreaded.
- The client has a simple command-shell interpreter that is used to accept two kinds of commands
 - **publish *lname fname***: a local file (which is stored in the client's file system at *lname*) is added to the client's repository as a file named *fname* and this information is conveyed to the server.
 - **fetch *fname***: fetch some copy of the target file and add it to the local repository
- The server has a simple command-shell interpreter
 - **discover *hostname***: discover the list of local files of the host named hostname
 - **ping *hostname***: live check the host named hostname
- It's important to note that the connecting infrastructure is not implied by the representation in Figure 1. All devices are interconnected through the Internet. Separating them is the logical point of view regarding the protocol's activities.



Hình 1: Class diagram của nhánh chức năng in tài liệu

3 Mô tả hệ thống

3.1 Phân tích yêu cầu

3.1.1 Yêu cầu chức năng

Chức năng cho Server:

- Bắt đầu server tại hostname của máy chạy server ("**start ip port**").
- Xem danh sách hostname các client đang tham gia ("**list_peers**"): Máy chủ (Server) phải có khả năng xem danh sách hostname của tất cả các client đang tham gia vào ứng dụng.
- Kiểm tra trạng thái của một client ("**ping ip port**"): Máy chủ (Server) sẽ được cung cấp chức năng kiểm tra trạng thái của một client còn tham gia hay đã rời đi.
- Xem thông tin các file đang chia sẻ của một client đang tham gia ("**discover ip port**"): Máy chủ (Server) được cung cấp khả năng xem thông tin về các file đang được chia sẻ bởi các client đang tham gia.
- Xem thông tin toàn bộ file đang chia sẻ của toàn bộ client ("**list_files**"): Máy chủ (Server) được cung cấp khả năng xem thông tin về toàn bộ các file đang được chia sẻ của toàn bộ client đang tham gia.
- Xem thông tin phân đoạn của toàn bộ các file đang được chia sẻ của tất cả các client ("**list_chunkinfo**"): Máy chủ (Server) được cung cấp khả năng quan sát thông tin phân đoạn của file đang được chia sẻ của toàn bộ các client đang tham gia.
- Kết thúc trạng thái hoạt động của Server ("**exit**").

Chức năng cho Client:

- Kết nối với một server đang online tại hostname ("**connect ip port**").
- Xem thông tin toàn bộ file đang chia sẻ của toàn bộ client ("**list_files**"): Máy client được cung cấp khả năng xem thông tin về toàn bộ các file đang được chia sẻ của toàn bộ client đang tham gia bao gồm chính máy client.
- Xem thông tin các file đang chia sẻ của một client đang tham gia ("**discover ip port**"): Máy client được cung cấp khả năng xem thông tin về các file đang được chia sẻ bởi các client đang tham gia bao gồm chính máy client.
- Chia sẻ thông tin (không chứa dữ liệu) của 1 file trong máy chạy client lên Server ("**publish lname fname**"): Client có khả năng chia sẻ file từ máy của mình trong ứng dụng và sau khi hoàn thành, thông báo sẽ được gửi đến máy chủ (Server) và máy client cũng được nhận thông tin này.
- Tải file từ client khác ("**fetch fname destination peerIP peerPort**"): Client phải có khả năng tải file từ client khác về repository của họ. Để có thể tải file thì client sẽ gửi yêu cầu tìm thông tin phân đoạn

về client nào đang nắm giữ file đó đến máy chủ (Server), máy chủ sẽ gửi lại cho client thông tin của một client đang nắm giữ file cần tải, sau đó hai client sẽ giao tiếp với nhau để bắt đầu tải file mà được yêu cầu.

- Kết thúc trạng thái hoạt động của client ("exit").

3.1.2 Yêu cầu phi chức năng

a. Cho toàn hệ thống:

Hiệu suất

- Đảm bảo khả năng tải file thành công 90%: Hệ thống phải có khả năng đảm bảo rằng tối thiểu 90% các tải file được thực hiện thành công mà không gặp lỗi.
- Tỷ lệ mất gói tin dưới 10%: Hệ thống phải đảm bảo rằng tỷ lệ mất gói tin trong quá trình truyền dữ liệu không vượt quá 10%.

Tính khả dụng

- Giao diện của hệ thống phải hiển thị đầy đủ các tính năng chính: publish, fetch, discover, ping...
- Ứng dụng dễ sử dụng: Người dùng mới phải có khả năng sử dụng cơ bản ứng dụng một cách dễ dàng trong vòng 2 phút sau khi làm quen với ứng dụng.

Tính bảo mật

- Đảm bảo các file được chia sẻ trong hệ thống không bị rò rỉ ra bên ngoài.

b. Cho chức năng:

Chức năng publish lname fname:

- Thời gian server nhận được thông báo tính từ khi client publish file lên repository hoàn tất sẽ không quá 2s.
- Tên fname của file trong repository của client phải là khác với các tên file đã được chia sẻ có trong hệ thống, tuy nhiên có thể trùng tên file với file đã chia sẻ trên hệ thống của một client khác.
- Client không thể publish file có đường dẫn trùng với một file mà client này đã chia sẻ trước đó.

Chức năng fetch fname destination peerIP peerPort:

- Multithread: Hệ thống cần hỗ trợ chạy nhiều luồng để cho phép nhiều client cùng tải file đồng thời đến một client đích.

- Thời gian từ khi client yêu cầu một file đến khi nhận được file từ client khác hoặc nhận thông báo yêu cầu thất bại từ server khi không có file sẽ không quá 2s.
- Client không thể tải một file mà mình đã publish lên Server.
- Ngoài ra client cũng không thể tải một file của client khác mà có đường dẫn destination trùng với một file trong máy của client này.
- Thời gian tải một file sẽ đảm bảo dưới 2s với những file có kích thước nhỏ hơn 1GB, với những file có kích thước lên đến 5GB thì việc tải này vẫn có thể thực hiện nhưng có thể lên đến 1-2 phút.

Chức năng discover ip port:

- Khi được gọi, chức năng "discover" cần chấp nhận thông tin cụ thể về máy khách, chẳng hạn như địa chỉ IP và port của máy.
- Tính năng "discover" cần trả về danh sách các tập tin cục bộ đã được chia sẻ lên Server trên máy khách cụ thể, cung cấp thông tin về tên tập tin, kích thước, tổng đoạn chia, đường dẫn, địa chỉ máy khách.

Chức năng ping ip port:

- Khi Server sử dụng chức năng này thì không thể tự ping chính địa chỉ của mình.
- Sau khi thực hiện kiểm tra, tính năng "ping" cần trả về kết quả về thời gian từ lúc thực hiện lệnh cho đến khi nhận được thông báo.

3.2 Giao thức

3.2.1 Giữa Client-Server

Xác định giao thức dùng cho các chức năng liên quan đến mô hình Client-Server:

- Chức năng kết nối ("connect ip port")

- Giao thức sử dụng là : **Message Passing Protocol**
- Giao tiếp giữa Client và Server.
- Nội dung giao tiếp

Loại Message	Các fields	Gửi	Nhận
REQUEST_REGISTER	- type : REQUEST_REGISTER - address : địa chỉ của client gửi	Client	Server
REPLY_REGISTER	- type : REPLY_REGISTER	Server	Client

- Chức năng publish ("publish lname fname")

- Giao thức sử dụng là : **Message Passing Protocol**
- Giao tiếp giữa Client và Server.
- Nội dung giao tiếp

Loại Message	Các fields	Gửi	Nhận
REQUEST_PUBLISH	- type : REQUEST_PUBLISH - primary_key : "{fname}_{ip}_{port}" - fname : Tên file remote - size : Kích thước file - total_chunknum : Tổng số đoạn chia file - author_address : Địa chỉ client - local_file : Đường dẫn của file	Client	Server
REPLY_PUBLISH	- type : REPLY_REGISTER - filename : Tên file vừa publish - result : true hoặc false	Server	Client

- Chức năng liệt kê toàn bộ file đã chia sẻ ("list_files")

- Giao thức sử dụng là : **Message Passing Protocol**
- Giao tiếp giữa Client và Server.
- Nội dung giao tiếp

Loại Message	Các fields	Gửi	Nhận
REQUEST_FILE_LIST	- type : REQUEST_FILE_LIST	Client	Server
REPLY_FILE_LIST	- type : REPLY_FILE_LIST - file_list : Danh sách các file	Server	Client

- Chức năng liệt kê file chia sẻ của một Client cụ thể ("discover ip port")

- Giao thức sử dụng là : **Message Passing Protocol**
- Giao tiếp giữa Client và Server.
- Nội dung giao tiếp

Loại Message	Các fields	Gửi	Nhận
REQUEST_DISCOVER	- type : REQUEST_DISCOVER - address : Địa chỉ của client cần xem	Client	Server

REPLY_DISCOVER	<ul style="list-style-type: none"> - type: REPLY_DISCOVER - file_list: Danh sách các file của client cần xem 	Server	Client
----------------	----------------------------------------------------------------------------------------------------------------------------------------------------	--------	--------

- Chức năng kiểm tra trạng thái Client ("ping ip port")

- Giao thức sử dụng là : **Message Passing Protocol**
- Giao tiếp giữa Client và Server.
- Nội dung giao tiếp

Loại Message	Các fields	Gửi	Nhận
REQUEST_PING_PONG	<ul style="list-style-type: none"> - type: REQUEST_PING_PONG - peer_address: Địa chỉ của client cần xem 	Server	Client
REPLY_PING_PONG	<ul style="list-style-type: none"> - type: REPLY_PING_PONG - response: true hoặc false 	Client	Server

3.2.2 Giữa Peer-to-Peer

Xác định giao thức dùng cho chức năng liên quan đến mô hình Peer-to-Peer:

- Tải file từ client khác ("fetch fname destination peerIP peerPort")

- Giao thức sử dụng là: **TCP**.
 - Cơ sở lý thuyết: Sử dụng TCP protocol trong việc truyền dữ liệu giữa các peers. TCP cho phép tạo các kết nối trực tiếp giữa các máy client, thực hiện việc truyền tải dữ liệu dưới dạng các luồng dữ liệu (data streams). Điều này giúp tối ưu hóa trải nghiệm tải file nhanh chóng và hiệu quả.
 - Cụ thể cách peer nhận được file:
 - Việc fetch sẽ có đối tượng **DownloadManager** quản lí.
 - Thuộc tính **to_download** là một danh sách hàng đợi của **chunknum** (các đoạn dữ liệu) được sắp xếp dựa trên độ hiếm có của chunknum (số lượng peer sở hữu chunknum càng nhiều thì sẽ sắp ở vị trí ưu tiên càng cao của hàng đợi), sắp xếp lại mỗi lần thông qua **update_chunk** (cập nhật vị trí sắp xếp các đoạn dữ liệu).
 - Thuộc tính **file_chunk_info** chứa thông tin về cái phần tử nào thuộc sở hữu của các peers.
 - **pending_chunknum** là một tập hợp các yêu cầu đoạn mà chúng ta đã gửi nhưng chưa nhận dữ liệu phần tử, điều này cần thiết cho việc phục hồi gửi lại yêu cầu khi có vấn đề mất gói xảy ra.
1. Gửi ra các gói yêu cầu ban đầu với số lượng cửa sổ (**window_size = 30**) từ **to_download** (lấy ra số thứ tự)

2. Đặt cặp (chunknum, peer) vào tập hợp pending_chunknum.
3. Mỗi khi dữ liệu phần tử đã đến, loại bỏ chunknum từ pending_chunknum và file_chunk_info.
4. Nếu có bất kỳ peer nào mất kết nối, loại bỏ nó khỏi thuộc tính `_peers`, kiểm tra xem chúng ta đã gửi yêu cầu đến nó chưa (sử dụng `pending_chunk_num`), gửi cùng một yêu cầu đến các peers thay thế.
5. Nếu có một số chunknum mà không có peer nào sở hữu, đưa ra ngoại lệ.

- Peer muốn xem thông tin toàn bộ đoạn dữ liệu về một file của Peer nào đó thông qua Server (Tracker)

- Giao thức sử dụng là : **Message Passing Protocol**
- Giao tiếp giữa Client và Server.
- Nội dung giao tiếp

Loại Message	Các fields	Gửi	Nhận
REQUEST_FILE_LOCATION	- type : REQUEST_FILE_LOCATION - primary_key : Khóa của file	Client	Server
REPLY_FILE_LOCATION	- type : REPLY_FILE_LOCATION - fileinfo : Thông tin chi tiết của file - chunkinfo : Thông tin đoạn dữ liệu của file	Server	Client

- Peer muốn lấy dữ liệu về một đoạn trong số các đoạn dữ liệu của một file từ một peer nào đó đang có dữ liệu file này

- Giao thức sử dụng là : **Message Passing Protocol**
- Giao tiếp giữa Client và Client.
- Nội dung giao tiếp

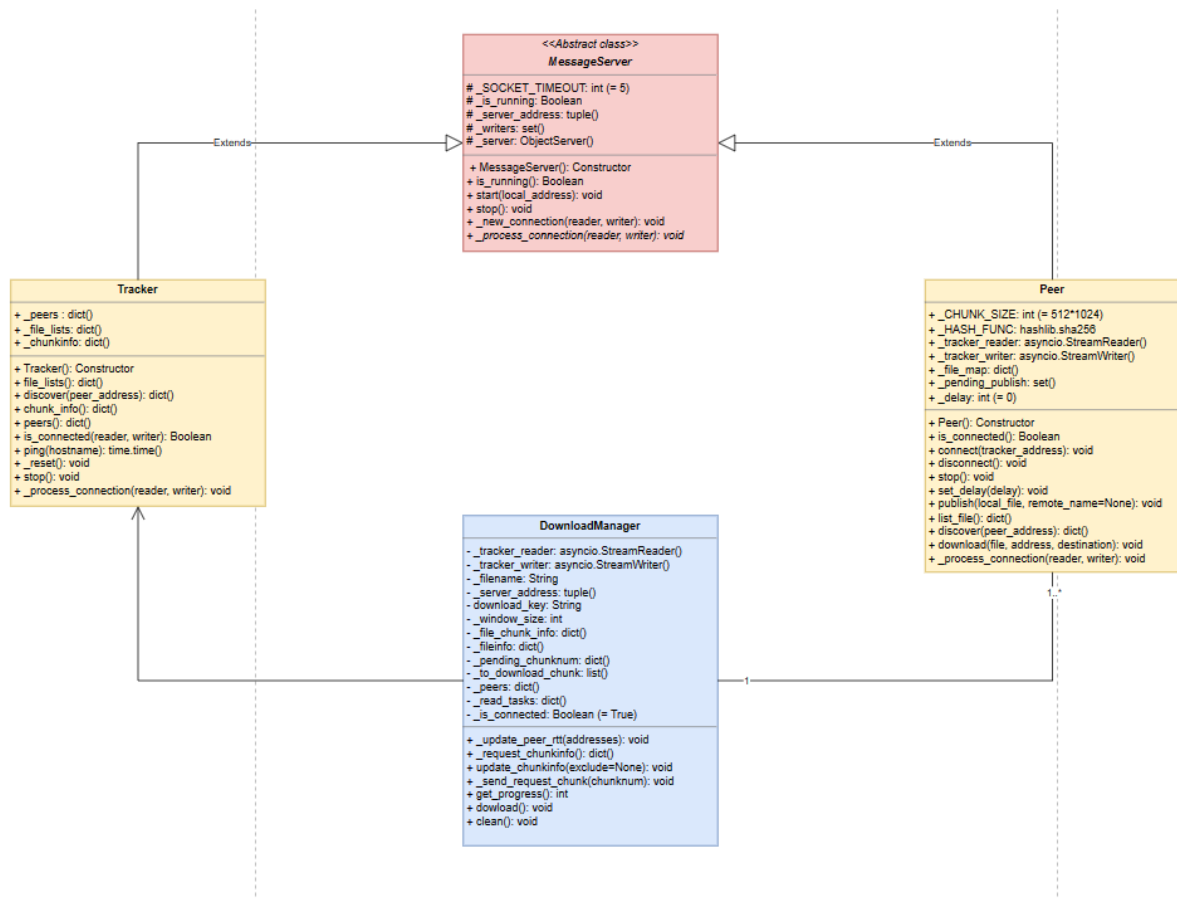
Loại Message	Các fields	Gửi	Nhận
PEER_REQUEST_CHUNK	- type : PEER_REQUEST_CHUNK - filename : Tên của file - chunknum : Vị trí đoạn dữ liệu của file	Client	Client
PEER_REPLY_CHUNK	- type : PEER_REPLY_CHUNK - filename : Tên của file - chunknum : Vị trí đoạn của file - data : Dữ liệu của đoạn được yêu cầu - digest : Thông tin để xác nhận dữ liệu có bị mất gói	Client	Client

- Peer muốn đăng ký thông tin file đã tải xong từ một peer khác lên Server (Tracker) để phục vụ cho các Peer khác muốn tải file này

- Giao thức sử dụng là : **Message Passing Protocol**
- Giao tiếp giữa Client và Server.
- Nội dung giao tiếp

Loại Message	Các fields	Gửi	Nhận
REQUEST_CHUNK_REGISTER	<ul style="list-style-type: none"> - type: REQUEST_CHUNK_REGISTER - primary_key: Khóa của file - chunknum: Vị trí đoạn muốn gửi 	Client	Server

3.3 Class diagram

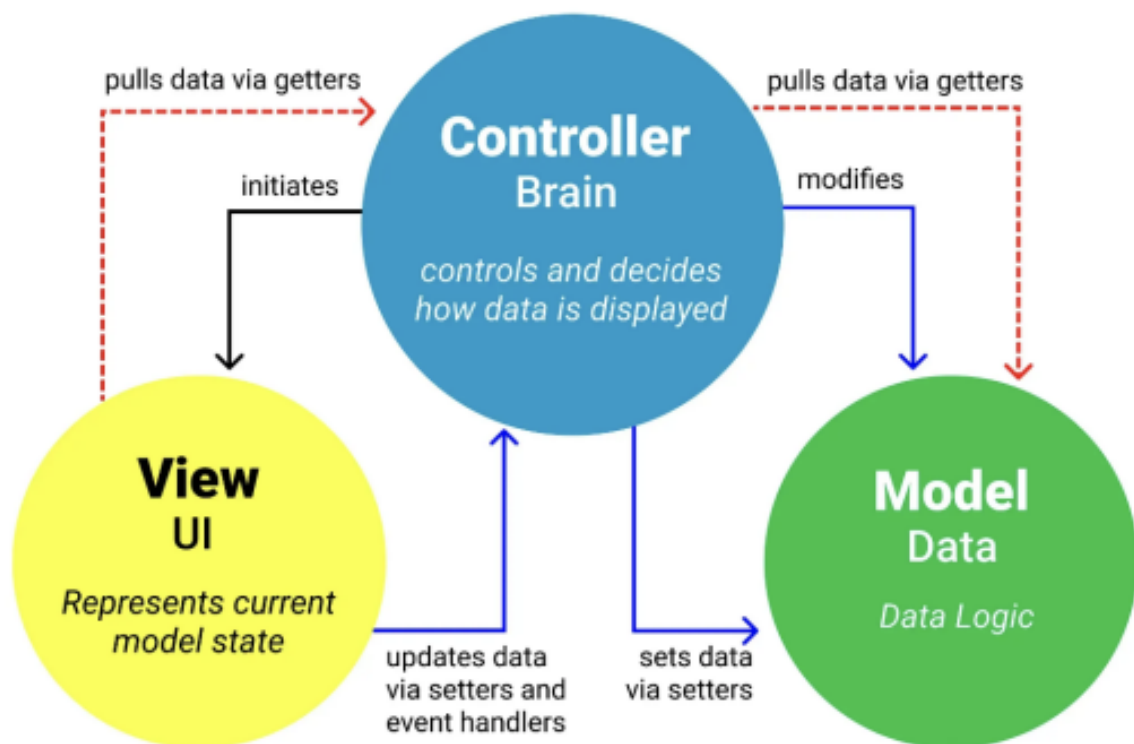


Hình 2: Class Diagram cho toàn bộ hệ thống

- DownloadManager đảm nhiệm chức năng quản lý và thực hiện quá trình tải file từ một Peer khác thông qua Tracker.
- Tracker có chức năng xử lý xác thực, kiểm tra hoạt động của các peer và gửi hostname của peer khác cho peer muốn tải file.

3.4 Thiết kế kiến trúc

MVC Architecture Pattern



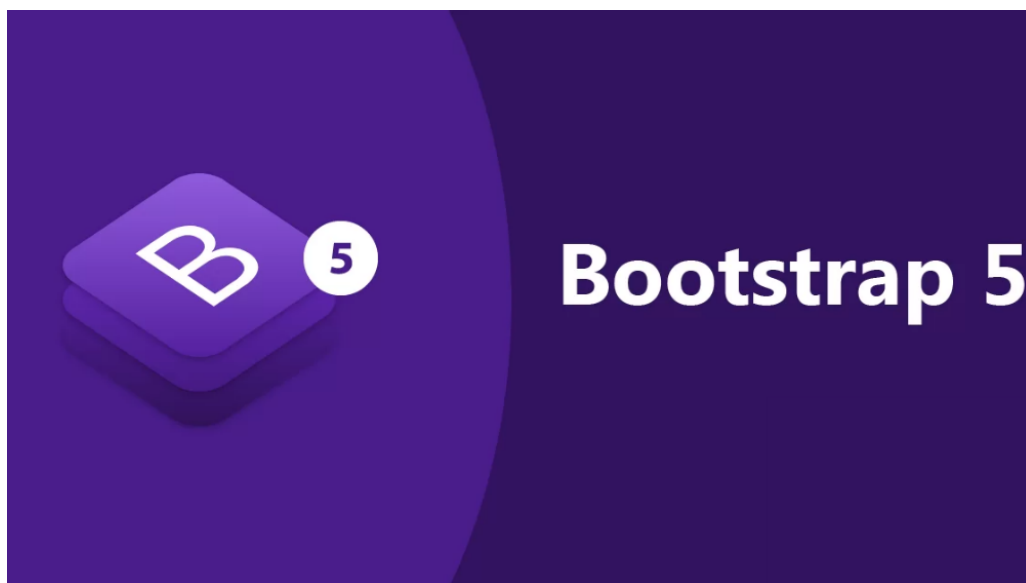
Hình 3: Kiến trúc MVC

- **Model**: Là bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng và là cầu nối giữa 2 thành phần bên dưới là View và Controller. Một model là dữ liệu được sử dụng bởi chương trình. Đây có thể là cơ sở dữ liệu, hoặc file XML bình thường hay một đối tượng đơn giản. Chẳng hạn như biểu tượng hay là một nhân vật trong game.
- **View**: Đây là phần giao diện (theme) dành cho người sử dụng. View là phương tiện hiển thị các đối tượng trong một ứng dụng. Chẳng hạn như hiển thị một cửa sổ, nút hay văn bản trong một cửa sổ khác. Nó bao gồm bất cứ thứ gì mà người dùng có thể nhìn thấy được.

- **Controller:** Là bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua View. Một controller bao gồm cả Model lẫn View. Nó nhận input và thực hiện các update tương ứng.

3.5 Các công nghệ được sử dụng

Bootstrap:



Bootstrap là một framework front-end mã nguồn mở, được phát triển bởi Twitter. Nó cung cấp một bộ công cụ chất lượng cao để phát triển giao diện người dùng web hiện đại và responsive nhanh chóng. Bootstrap sử dụng HTML, CSS, và JavaScript để giúp tạo ra các trang web và ứng dụng web đẹp và dễ sử dụng trên nhiều thiết bị và kích thước màn hình.

- Bootstrap được xây dựng với việc ưu tiên hàng đầu là tạo ra các trang web responsive. Các thành phần và lớp của Bootstrap tự động thích ứng với kích thước màn hình khác nhau, giúp đảm bảo trải nghiệm người dùng tốt trên cả điện thoại di động, máy tính bảng và máy tính.
- Sử dụng một hệ thống grid linh hoạt, giúp bạn dễ dàng xây dựng cột và hàng để tổ chức nội dung trên trang web của mình. Điều này giúp tạo ra các trang web có cấu trúc gọn gàng và dễ bảo trì.
- Cung cấp nhiều lớp CSS và các plugin JavaScript được tích hợp sẵn, giúp giảm thiểu công việc lập trình và tăng tốc quá trình phát triển. Điều này giúp tăng hiệu suất và giảm khả năng xảy ra lỗi.
- Cung cấp nhiều thành phần UI mô-đun như nút, biểu đồ, biểu mẫu và nhiều hơn nữa, giúp tiết kiệm thời gian và công sức khi phát triển.
- Bootstrap được kiểm thử kỹ lưỡng và hỗ trợ trên nhiều trình duyệt phổ biến như Chrome, Firefox, Safari, và Internet Explorer.

- Có một cộng đồng lớn của những người phát triển sử dụng Bootstrap, điều này có nghĩa là bạn có thể dễ dàng tìm thấy hỗ trợ và tài liệu trực tuyến khi bạn gặp vấn đề.

ExpressJS:



ExpressJS là một framework ứng dụng web có mã nguồn mở và miễn phí được xây dựng trên nền tảng Node.js. ExpressJS được sử dụng để thiết kế và phát triển các ứng dụng web một cách nhanh chóng. Để hiểu ExpressJS, người dùng chỉ cần phải biết JavaScript, do đó nên việc xây dựng các ứng dụng web và API trở nên đơn giản hơn đối với các lập trình viên và nhà phát triển đã thành thạo JavaScript trước đó.

- Giảm độ phức tạp: Express.js giúp giảm độ phức tạp khi phát triển ứng dụng web bằng cách cung cấp một cấu trúc mô-đun và linh hoạt. Nó không buộc bạn phải tuân thủ một cấu trúc cụ thể, giúp bạn linh hoạt trong việc tổ chức mã nguồn của mình.
- Routing: Express cung cấp một hệ thống định tuyến mạnh mẽ, giúp bạn dễ dàng xác định cách xử lý các yêu cầu HTTP đến các đường dẫn cụ thể.
- Middleware: Middleware là một tính năng quan trọng của Express, cho phép bạn chạy mã trước hoặc sau khi xử lý yêu cầu. Điều này rất hữu ích để thực hiện các chức năng như xác thực, quản lý phiên, ghi log, và nhiều hơn nữa.
- Hỗ trợ Template Engine: Express hỗ trợ nhiều template engine như EJS, Pug, và Handlebars, giúp bạn dễ dàng tạo ra trang web động với dữ liệu động.
- Hỗ trợ cho RESTful API: Express được sử dụng rộng rãi để xây dựng các RESTful API do tích hợp các chức năng như xử lý yêu cầu HTTP, xác thực, và quản lý route dễ dàng.
- Linh hoạt và Mở rộng: Express.js là một framework mở rộng, có nghĩa là bạn có thể tích hợp nhiều thư viện và middleware khác nhau để mở rộng chức năng của ứng dụng mình.
- Active Community: Express có một cộng đồng lớn và hoạt động, điều này đồng nghĩa với việc có nhiều tài liệu, hướng dẫn, và hỗ trợ từ cộng đồng khi bạn gặp vấn đề.

- Performance: Express được tối ưu hóa để hoạt động hiệu quả trên Node.js, với khả năng xử lý đồng thời cao và overhead thấp.

Flask:



- Mục tiêu chính: Flask là một micro-framework web, điều này có nghĩa là nó cung cấp một lõi nhỏ và linh hoạt mà bạn có thể mở rộng bằng cách sử dụng các extension theo nhu cầu.
- Đơn giản và linh hoạt: Flask được thiết kế để đơn giản và linh hoạt, cho phép bạn tự do chọn lựa các thư viện và công nghệ khác nhau để tích hợp vào dự án của mình.
- Jinja2 Template Engine: Flask sử dụng Jinja2 làm engine template, giúp tạo ra các trang web động và linh hoạt.
- Không tích hợp sẵn asyncio: Flask không tích hợp sẵn hỗ trợ cho asyncio, nó thường được sử dụng trong môi trường đồng bộ. Tuy nhiên vẫn có thể sử dụng bản hỗ trợ cho asyncio thông qua phiên bản `flask[async]`.

aiohttp:

- aiohttp được xây dựng với mục tiêu chính là hỗ trợ lập trình bất đồng bộ. Điều này giúp xử lý hàng loạt yêu cầu IO mà không chặn (blocking) quá trình chính của ứng dụng, điều này quan trọng để đảm bảo hiệu suất và mở rộng tốt trong ứng dụng có nhiều người dùng.
- Asynchronous I/O: aiohttp cung cấp cả Client API và Server API. Client API cho phép bạn thực hiện các yêu cầu HTTP từ phía client một cách không đồng bộ, trong khi Server API cho phép bạn xây dựng các ứng dụng web server với khả năng xử lý không đồng bộ.



aiohttp

- Nó hỗ trợ WebSockets, một giao thức giúp thiết lập kết nối hai chiều giữa client và server, thích hợp cho các ứng dụng real-time như trò chơi trực tuyến, chat, và cập nhật trạng thái real-time.
- Hỗ trợ HTTP/2, một phiên bản nâng cấp của giao thức HTTP giúp tối ưu hóa hiệu suất và giảm độ trễ trong việc tải trang web.
- Middleware cho phép bạn thêm các chức năng xử lý yêu cầu hoặc phản hồi trước hoặc sau khi chúng được xử lý, giúp tăng cường chức năng của ứng dụng.
- Hỗ trợ routing để xác định cách xử lý các yêu cầu HTTP đến các đường dẫn cụ thể.
- aiohttp tích hợp chặt chẽ với asyncio, giúp bạn xây dựng các ứng dụng không đồng bộ một cách dễ dàng hơn và tận dụng được các lợi ích của lập trình không đồng bộ.
- aiohttp có tài liệu chi tiết và hướng dẫn sử dụng rất tốt, giúp người phát triển nhanh chóng làm quen và triển khai các dự án.

4 Hướng dẫn sử dụng

Link code github: [Tại đây](#)

Source code chính trên nhánh **P2P-WEB**.

Hoặc có thể clone về:

- `git clone -b P2P-WEB https://github.com/hungvo2003vn/File-Sharing-App.git`

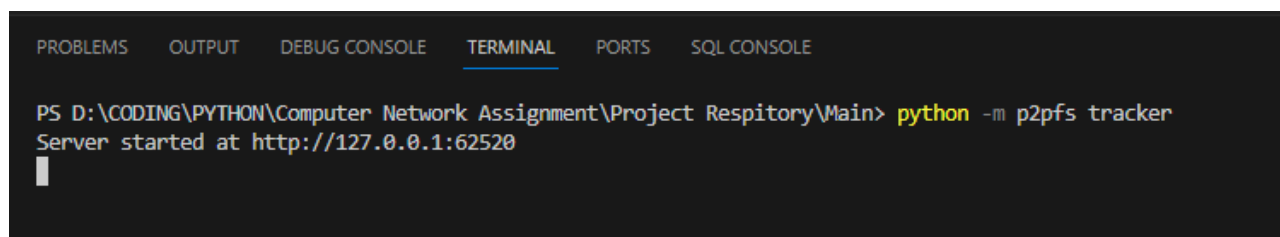
Tải trước các module cần thiết:

- Phía Backend: `python setup.py install`.
- Phía Frontend: `cd p2pfe`, sau đó `npm install`

5 Kiểm tra và chạy thử ứng dụng

5.1 Chạy Tracker

Back-end của Tracker

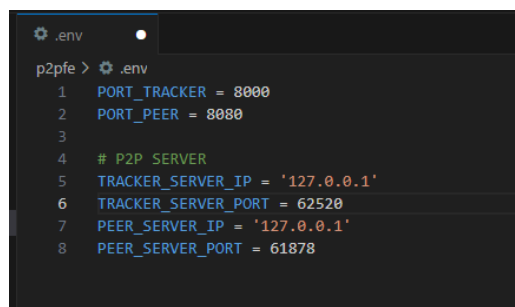


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE

PS D:\CODING\PYTHON\Computer Network Assignment\Project Respiratory\Main> python -m p2pfs tracker
Server started at http://127.0.0.1:62520
```

Front-end của Tracker

- Đổi thông tin **TRACKER_SERVER_PORT** trong file `.env` thành **PORT** mà Back-end của Tracker đang sử dụng như hình (62520) để có thể sử dụng API của Back-end.



```
.env
p2pfe > .env
1 PORT_TRACKER = 8000
2 PORT_PEER = 8080
3
4 # P2P SERVER
5 TRACKER_SERVER_IP = '127.0.0.1'
6 TRACKER_SERVER_PORT = 62520
7 PEER_SERVER_IP = '127.0.0.1'
8 PEER_SERVER_PORT = 61878
```

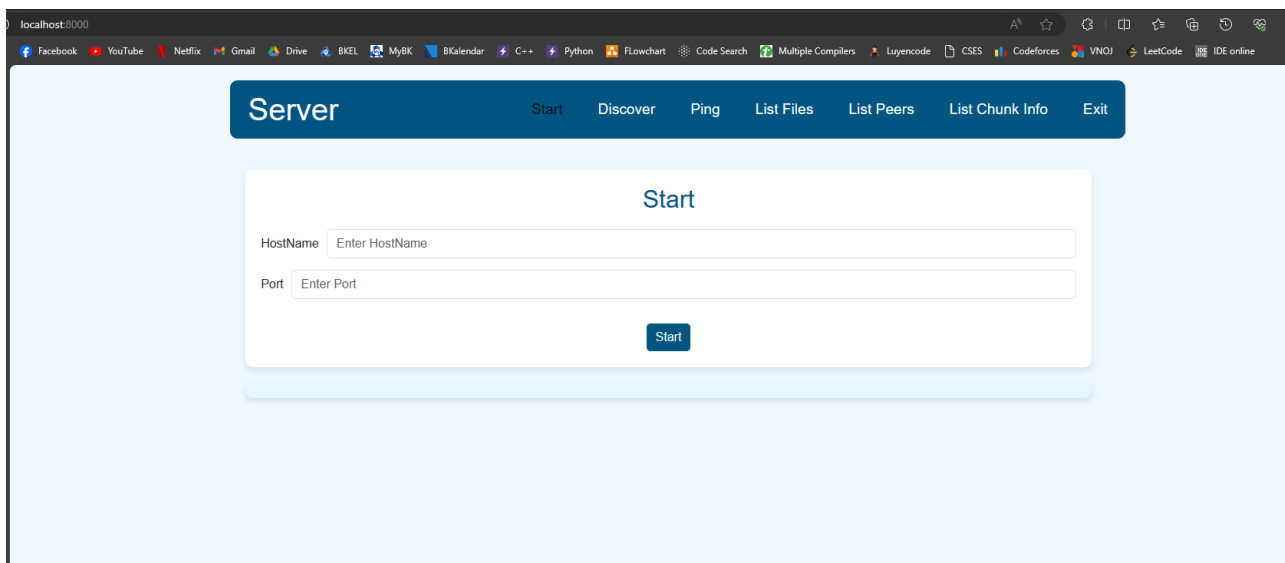
- Chạy Front-end của Tracker

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE

PS D:\CODING\PYTHON\Computer Network Assignment\Project Respiratory\Main\p2pfe> npm run tracker

> tracker
> node ./apps/AppSer.js

Server started at http://localhost:8000
```



5.2 Chạy Peer

Back-end của Peer

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE

PS D:\CODING\PYTHON\Computer Network Assignment\Project Respiratory\Main> python -m p2pfs peer
Peer Address: ('192.168.34.131', 62825)
Server started at http://127.0.0.1:62826
```

Front-end của Peer

- Đổi thông tin **PEER_SERVER_PORT** trong file .env thành PORT mà Back-end của Peer đang sử dụng như hình (62826) để có thể sử dụng API của Back-end.

```
.env
p2pfe > .env
1  PORT_TRACKER = 8000
2  PORT_PEER = 8080
3
4  # P2P SERVER
5  TRACKER_SERVER_IP = '127.0.0.1'
6  TRACKER_SERVER_PORT = 62520
7  PEER_SERVER_IP = '127.0.0.1'
8  PEER_SERVER_PORT = 62820
```

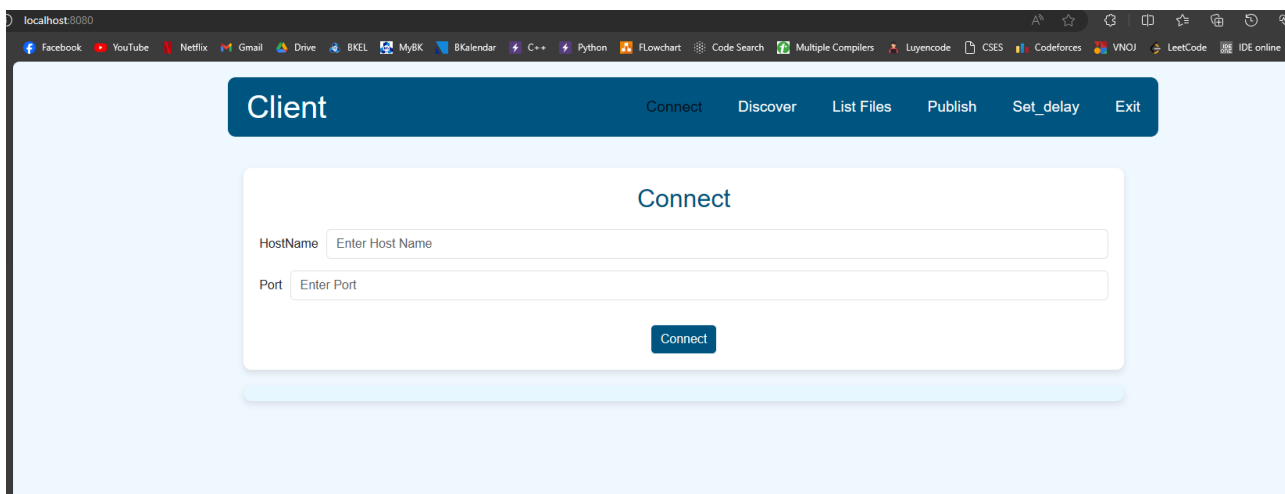
- Chạy Front-end của Peer

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE

PS D:\CODING\PYTHON\Computer Network Assignment\Project Respiratory\Main> cd p2pfe
PS D:\CODING\PYTHON\Computer Network Assignment\Project Respiratory\Main\p2pfe> npm run peer

> peer
> node ./apps/AppCli.js

Server started at http://localhost:8080
```



- Tương tự ta có thể chạy thêm 1 peer nữa để test thử các tính năng

5.3 Test tính năng start của Tracker

Server

StartDiscoverPingList FilesList PeersList Chunk InfoExit

Start

HostName192.168.34.131

Port3000

Start

Status	Message
success	Tracker started listening on ('192.168.34.131', 3000)

5.4 Kết nối 2 peer vào địa chỉ ip và port của tracker

Client

ConnectDiscoverList FilesPublishSet_delayExit

Connect

HostName192.168.34.131

Port3000

Connect

Status	Message
success	Successfully connected!

5.5 Test tính năng liệt kê các peers "list_peers" của Tracker

Server

StartDiscoverPingList FilesList PeersList Chunk InfoExit

List Peers

List Peers

Peer IP	Peer Port
192.168.34.131	62825
192.168.34.131	62938

5.6 Tính năng publish của Peer có port (62825) và ip (192.168.34.131)

Client

ConnectDiscoverList FilesPublishSet_delayExit

Publish

Lnamep2pfs/repo/requirements.txt

Fnamereq1.txt

Publish

Status	Message
success	File req1.txt successfully published on tracker.

5.7 Tính năng list_files của Tracker và Peer

Server

StartDiscoverPingList FilesList PeersList Chunk InfoExit

List Files

List Files


Filename	Author IP	Author Port
req1.txt	192.168.34.131	62825

Client

ConnectDiscoverList FilesPublishSet_delayExit

List Files

List Files

Filename	Author IP	Author Port	Fetch
req1.txt	192.168.34.131	62825	

5.8 Tính năng discover của Tracker

Server

StartDiscoverPingList FilesList PeersList Chunk InfoExit

Discover

Address192.168.34.131

Port62825

Discover

Status	Filename
Success	req1.txt

5.9 Tính năng list_chunkinfo của Tracker

Server

StartDiscoverPingList FilesList PeersList Chunk InfoExit

List Chunk Info

List Chunk Info

Filename	Address	Chunk Info
req1.txt	["192.168.34.131", 62825]	0

5.10 Tính năng ping của Tracker

Server

StartDiscoverPingList FilesList PeersList Chunk InfoExit

Ping

HostName192.168.34.131

Port62825

Ping

Status	Message
success	Received packet from ('192.168.34.131', 62825) in 1.0ms


5.11 Tính năng fetch của Peer

Client

ConnectDiscoverList FilesPublishSet_delayExit

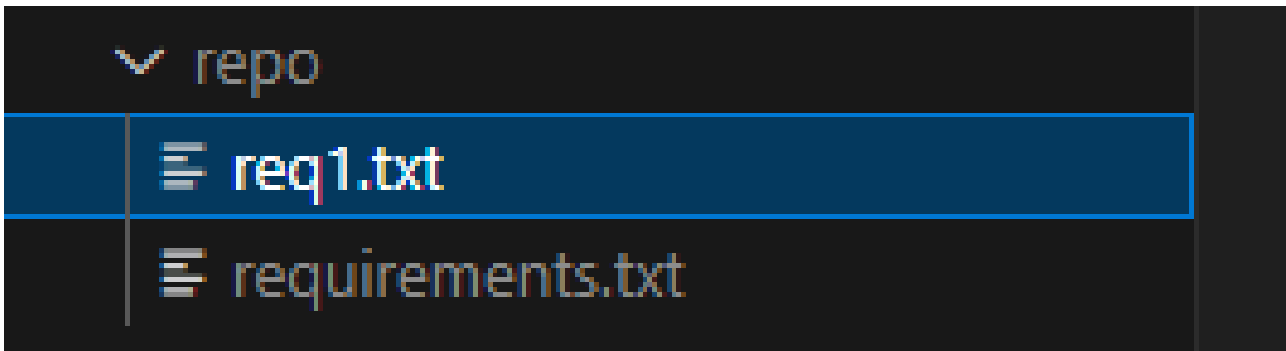
List Files

List Files

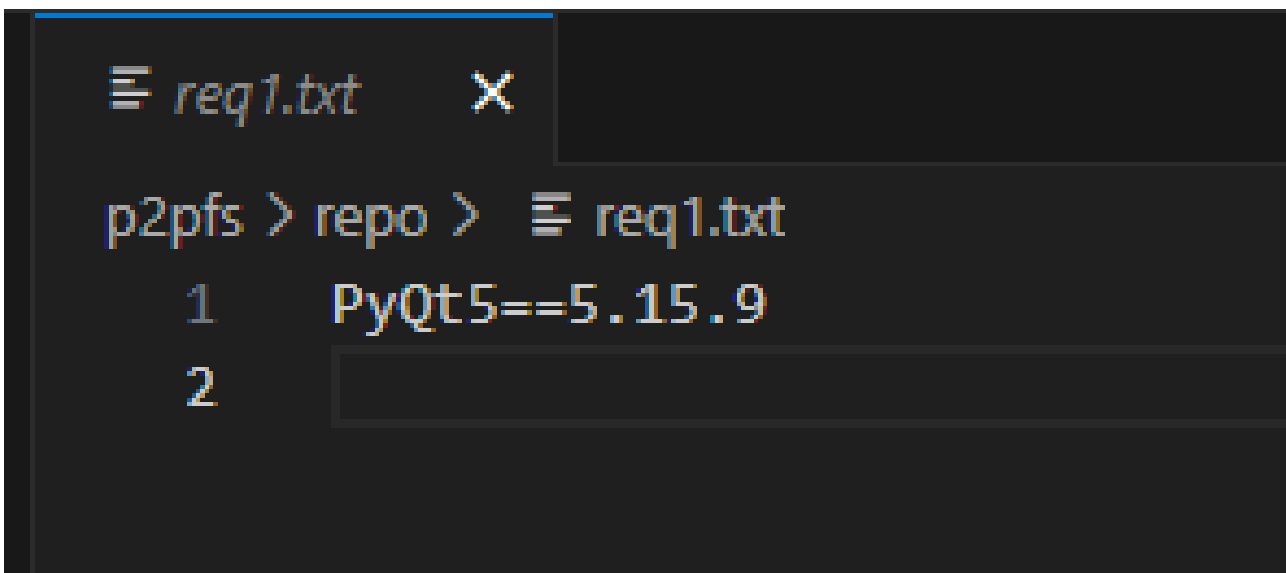
Filename	Author IP	Author Port	Fetch
req1.txt	192.168.34.131	62825	

Success: File req1.txt successfully downloaded to p2pfs//repo//req1.txt. X

- Kiểm tra repo của peer vừa tải xuống



- Kiểm tra dữ liệu file mà peer vừa tải xuống



6 Tài liệu tham khảo

Tài liệu

- [1] Client-Server and Peer-to-Peer Network: <https://www.geeksforgeeks.org/difference-between-client-server-and-peer-to-peer-network/>
- [2] P2P (Peer To Peer) File Sharing: <https://www.geeksforgeeks.org/p2p-peer-to-peer-file-sharing/>
- [3] asyncio — Asynchronous I/O <https://docs.python.org/3/library/asyncio.html>
- [4] Flask documentation <https://flask.palletsprojects.com/en/3.0.x/>