Thành viên và phân chia công việc

Họ và tên	Nhiệm vụ	Mức độ hoàn thành
Nguyễn Hoàng Duy Tân		100%
Trần Nguyễn Minh Tuệ		100%
Nguyễn Minh Lộc		100%

Phân tích yêu cầu

Functional requirements

Xây dựng một ứng dụng chia sẻ file đơn giản với giao thức được định nghĩa sẵn, sử dụng những giao thức trong TCP/IP stack.

Client Functions

Basic functions

Đăng ký trong kho lưu trữ

- Máy khách có thể gửi yêu cầu đăng ký file có trong kho lưu trữ cho máy chủ.
- Thông điệp đắng ký file: "publish ". File trên máy khách sẽ được thêm vào kho lưu trữ dưới tên .
- Các file sau khi đăng ký sẽ được lưu trữ trong kho lưu trữ của tài khoản được liên kết với máy chủ.

Gửi yêu cầu tải file cho server

- Máy khách có thể gửi yêu cầu tải file không có sẵn trong kho lưu trữ của mình. Lúc này máy chủ sẽ phản hồi lại danh sách các máy khách khác có file được yêu cầu.

Tải file trực tiếp từ nguồn muốn chọn

- Máy khách sau khi nhận được phản hồi từ máy chủ danh sách máy khách có sẵn file được yêu cầu có thể chọn một nguồn thích hợp và gửi yêu cầu tải file tới đó.
- Các máy khách được cung cấp một danh sách yêu cầu tải file từ các máy khách khác, máy khách có thể chọn 1 file trong danh sách và gửi yêu cầu tải file tới máy khách đó.
- Thông điệp tải file: "fetch ". Trong đó fname là 1 trong những tên file muốn chọn sau khi server đã phản hồi.

Extended functions

Đăng ký tài khoản

- Người dùng đăng ký địa chỉ của máy vào hệ thống của máy chủ.

Đăng nhập tài khoản và xác thực

- Người dùng đăng nhập tài khoản để sử dụng các chức năng của hệ thống.

Liệt kê danh sách lưu trữ

- Máy khách có thể kiểm tra danh sach các file có trong kho lưu trữ của mình.

Tìm kiếm bằng từ khóa

- Server sẽ hỗ trợ người dùng tìm kiếm file theo từ khóa.

Server Functions

Basic functions

Kiểm tra trạng thái máy chủ

- Máy chủ có thể kiểm tra trạng thái của máy chủ khác thông qua lệnh "ping <hostname>"

Xem danh sách file của máy khách khác

- Máy chủ có thể xem danh sách file trong kho lưu trữ của các máy khách thông qua lệnh "discover <hostname>"

Gửi thông tin cần thiết sau khi nhận yêu cầu tải file từ clients

- Sau khi nhận được yêu cầu tìm file từ người dùng, server sẽ tiến hành theo dõi và tìm kiếm để trả về các thông tin nơi đang lưu trữ các file đó cho clients: ID peer, thời gian file được cập nhật.

Extended functions

Xem file log

- Máy chủ có thể xem file log của máy khách khác thông qua.

Non-functional requirements

Giao diện người dùng – Cung cấp giao diện người dùng dễ sử dụng cho máy khách để nhập các lệnh và theo dõi quá trình tải tệp. **Multi-threading** – Triển khai đa luồng trong máy khách để có thể xử lý nhiều tải xuống cùng lúc. **Hiệu năng và tích hợp** – Đảm bảo rằng hệ thống hoạt động hiệu quả và có khả năng tích hợp với các mạng internet và hệ thống người dùng khác nhau.

Phân tích kiến trúc

Kiến trúc Peer-to-Peer (P2P)

- Kiến trúc Peer-to-Peer (P2P) là một mô hình mạng máy tính trong đó các máy tính (được gọi là nút hoặc "peers") kết nối trực tiếp với nhau để chia sẻ tài nguyên và thông tin mà không cần sự tương tác trung tâm từ máy chủ. Trong kiến trúc P2P, mỗi máy tính có thể đồng thời hoạt động như máy khách và máy chủ, có nghĩa là chúng có khả năng yêu cầu tài nguyên từ các máy tính khác và chia sẻ tài nguyên với người khác.

Kiến trúc client-server

- Kiến trúc client-server (còn được gọi là mô hình client-server) là một kiến trúc máy tính phổ biến được sử dụng trong việc tổ chức và quản lý các dịch vụ và tài nguyên trên mạng. Nó dựa trên sự phân chia các vai trò chính trong hệ thống thành hai phần: máy khách (client) và máy chủ (server). Hai phần này tương tác với nhau để cung cấp các dịch vụ, ứng dụng, và tài nguyên cho người dùng.

Giới thiệu Protocol

HyperText Tràner Protocol

HyperText Transfer Protocol (HTTP) là một giao thức ở tầng ứng dụng trong mô hình OSI để gửi và nhận tài liệu, hình ảnh, văn bản như HTML document. Về cơ bản, giao thức HTTP xây dựng trên cơ chế request-response trong mô hình client-server. Trong mô hình này, client có thể là một process thuộc máy tính này, server có thể là process thuộc máy tính khác, hai process thuộc hai phần cứng khác nhau khi muốn giao tiếp với nhau thì sẽ thông qua HTTP để giao tiếp. Ai là người request thì đó là client, người nhận request để response sẽ là server.

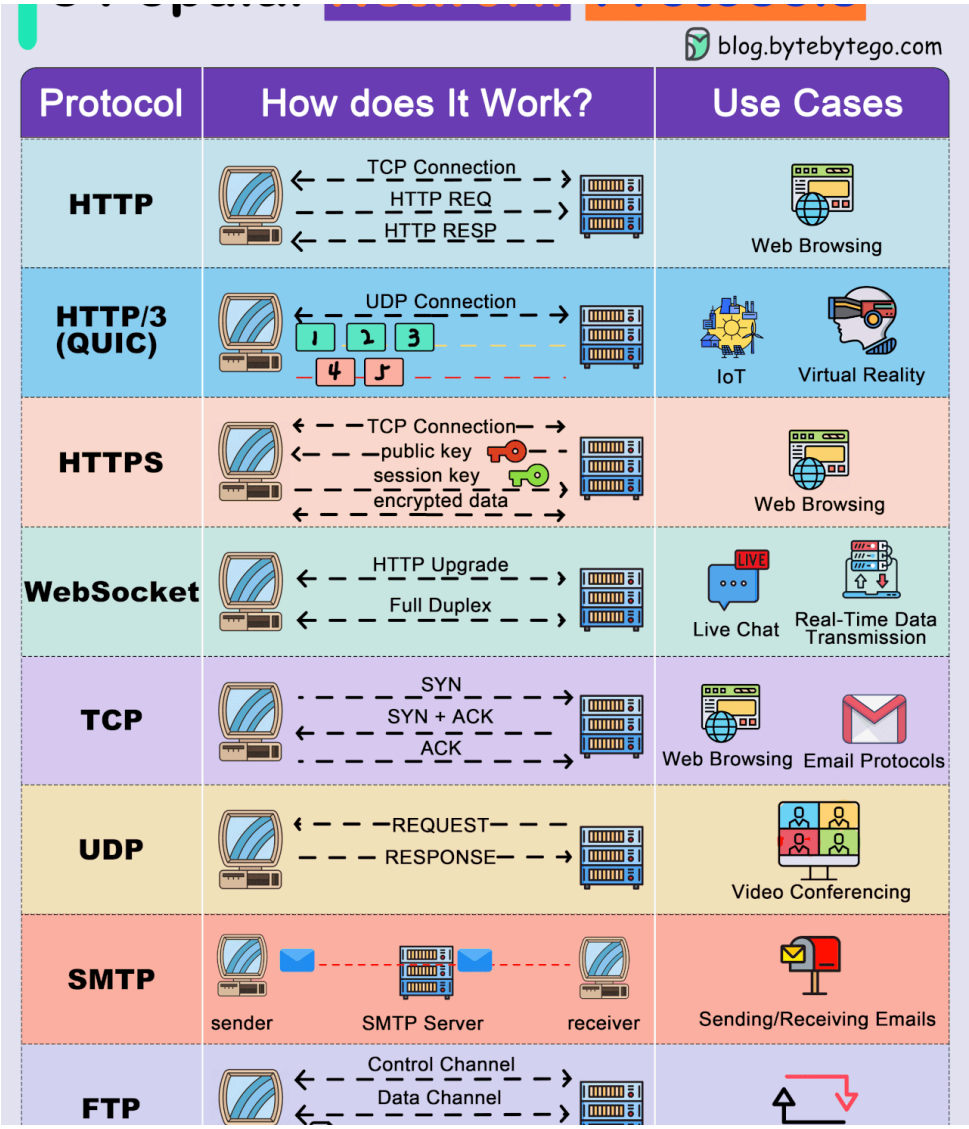


Figure 1: 8 giao thức mạng máy tính phổ biến (Nguồn Bytebytego)

Transmission Control Protocol

Transmission Control Protocol (TCP) là một trong các giao thức cốt lõi của bộ giao thức TCP/IP. Nhờ có TCP, các ứng dụng trên các host được nối mạng có thể tạo các kết nối với nhau, mà qua đó chúng có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự.

Cách đặc tính cơ bản của TCP:

- Point-to-point: Trong một giao thức TCP, chỉ có một sender và một server được kết nối với nhau bằng 3-way handshaking.
- Reliable, in-order bit stream: Hỗ trợ truyền tin cậy và đúng thứ tự.
- Pipelined: Truyền song song nhằm tăng hiệu quả gửi nhận
- Flow control: Receiver kiểm soát tốc độ gửi của sender để tránh làm quá tải receiver.
- Congestion control: Tự động điều chỉnh tốc độ gửi ở mức tối đa mà không làm tắc nghẽn hệ thống.
- Full-duplex connection: hỗ trợ truyền hai chiều trong cùng một thời điểm trong một kết nối.

Socket programming with Python

Socket là cánh cổng ngăn cách giữa Application Layer với Transport Layer.

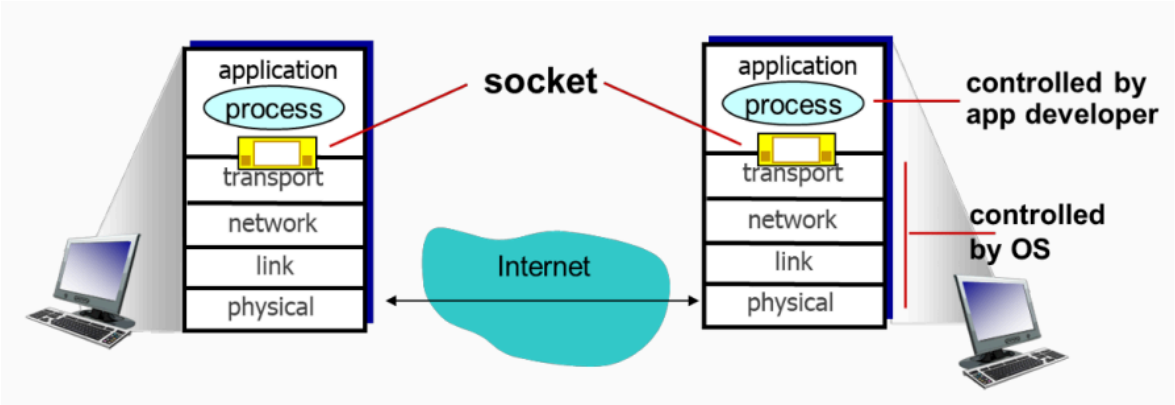


Figure 2: Mô hình socket

Thư viện socket của Python hỗ trợ một số API sau để thiết lập kết nối socket TCP/UDP:

- socket()
- bind()
- listen()
- accept()
- connect()
- connect_ex()
- send()
- recv()
- close()

Thư viện socket của Python hỗ trợ cả TCP socket lẫn UDP socket. Trong project này, nhóm dự định sử dụng TCP socket để hiện thực dự án.
Flow của một TCP socket connection có thể được thể hiện như hình dưới đây:

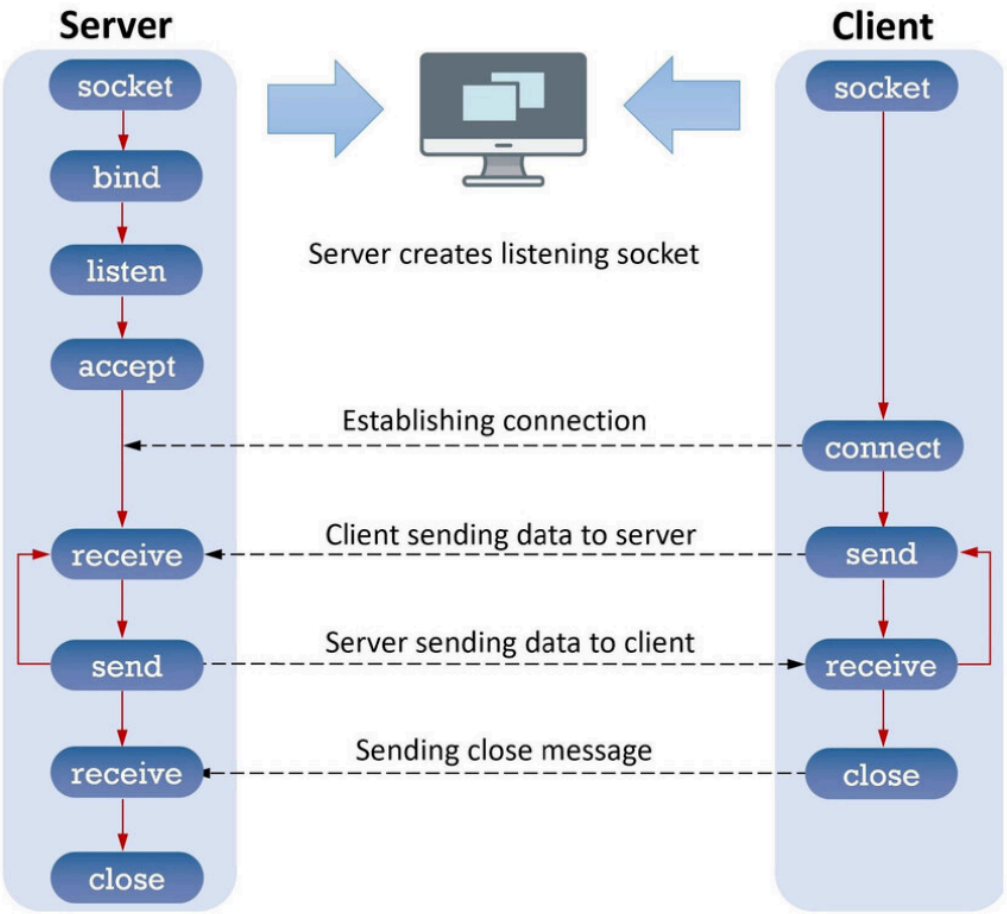
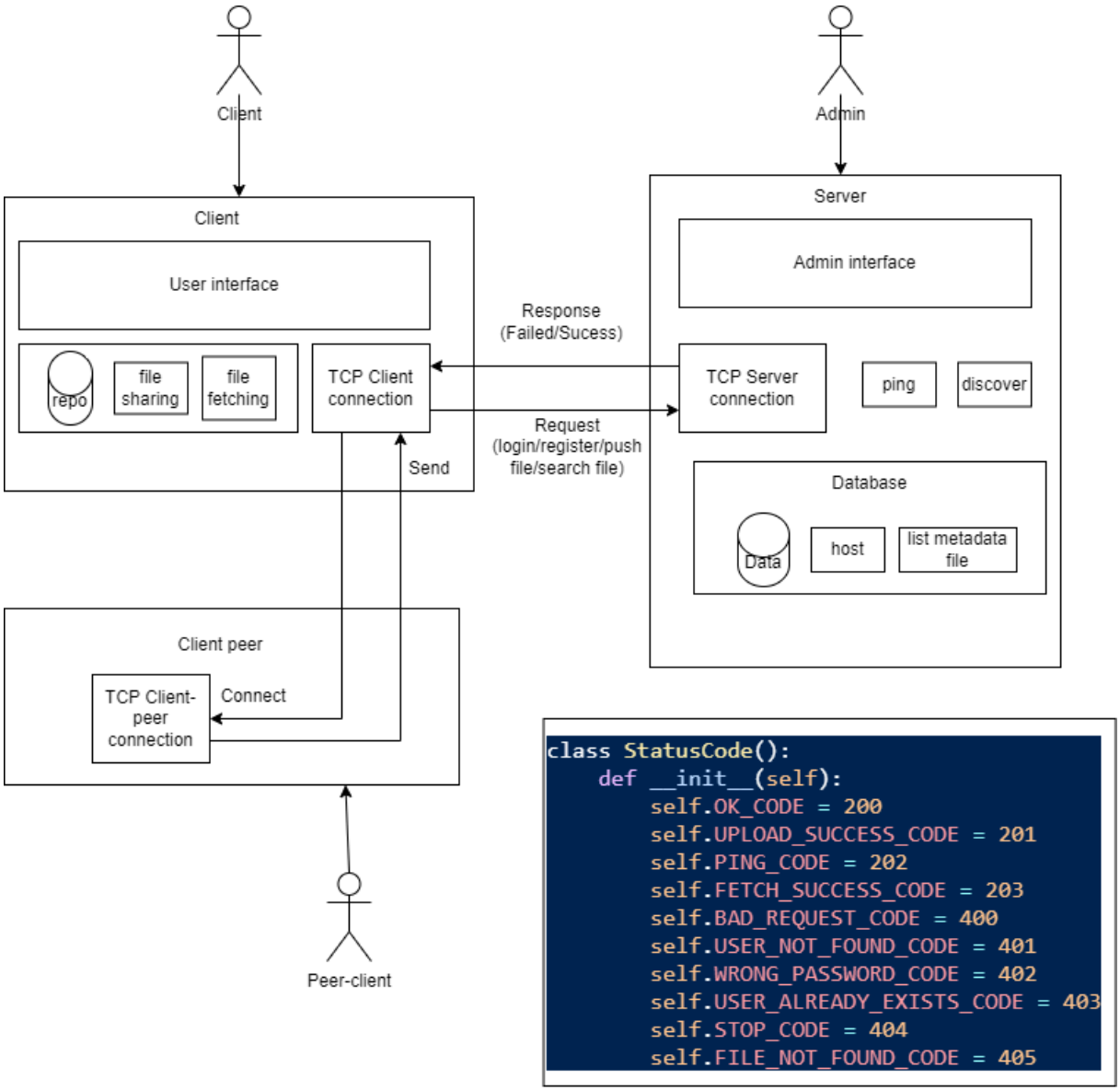


Figure 3: Quy trình một kết nối socket để gửi nhận dữ liệu (Nguồn ResearchGate)

Application design

Architecture design




```
class StatusCode():
    def __init__(self):
        self.OK_CODE = 200
        self.UPLOAD_SUCCESS_CODE = 201
        self.PING_CODE = 202
        self.FETCH_SUCCESS_CODE = 203
        self.BAD_REQUEST_CODE = 400
        self.USER_NOT_FOUND_CODE = 401
        self.WRONG_PASSWORD_CODE = 402
        self.USER_ALREADY_EXISTS_CODE = 403
        self.STOP_CODE = 404
        self.FILE_NOT_FOUND_CODE = 405
```

Figure 4: Kiến trúc tổng quan của hệ thống

Flow protocol design

Register

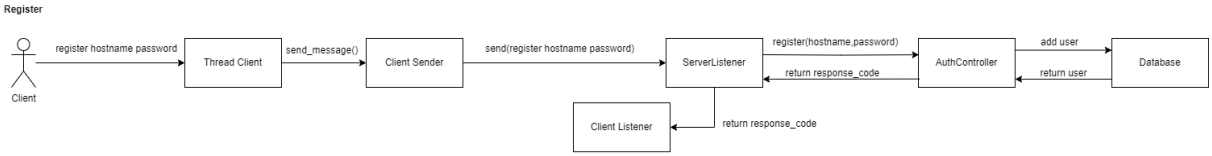


Figure 5: Sơ đồ luồng của chức năng đăng ký

Login

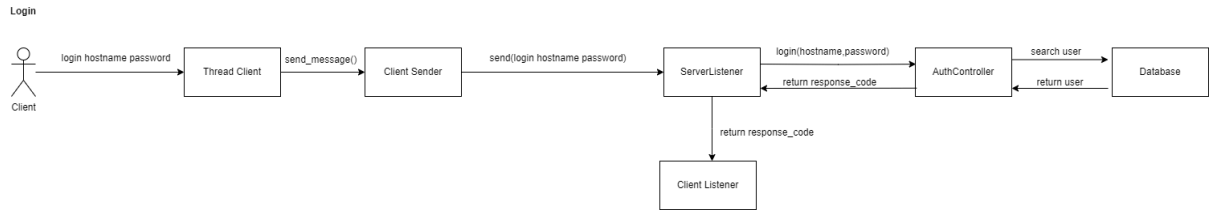


Figure 6: Sơ đồ luồng của chức năng đăng nhập

Ping

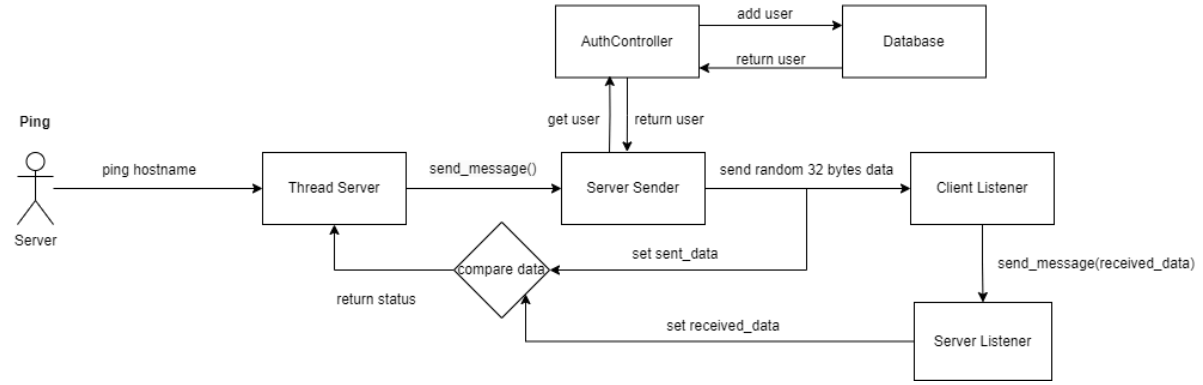


Figure 7: Sơ đồ luồng của chức năng ping

Discover

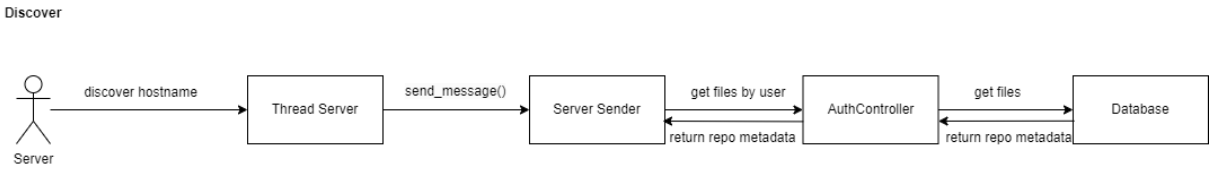


Figure 8: Sơ đồ luồng của chức năng discover

Publish

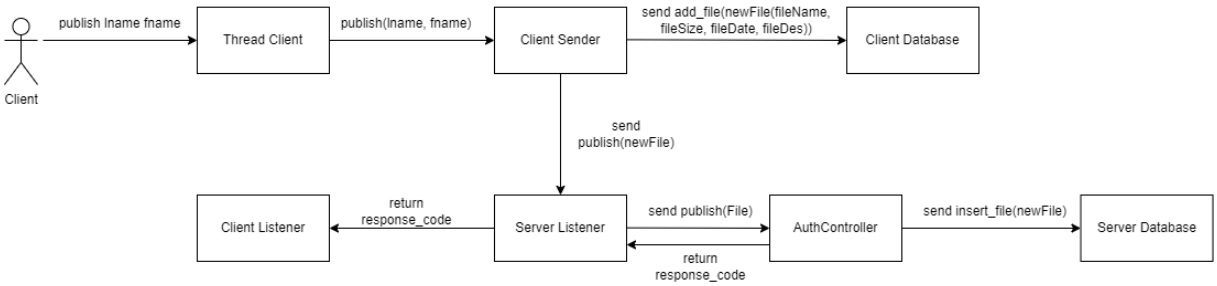


Figure 9: Sơ đồ luồng của chức năng publish

Fetch

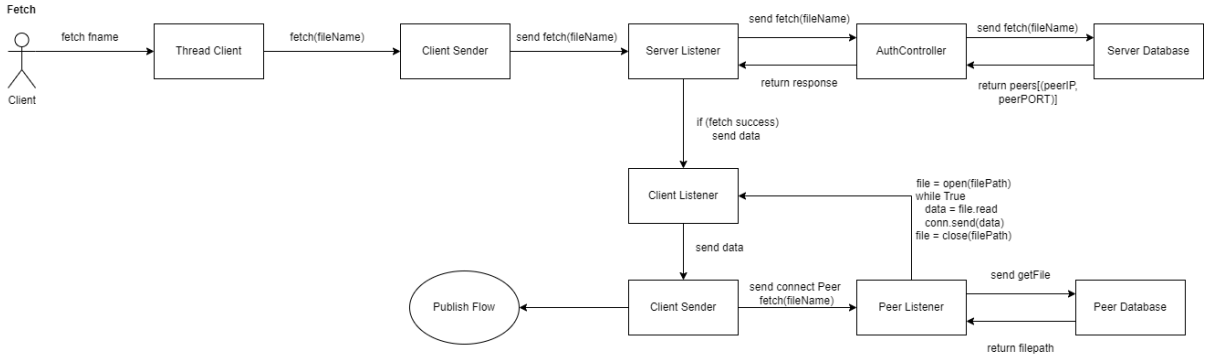


Figure 10: Sơ đồ luồng của chức năng fetch

Activity diagram

Register

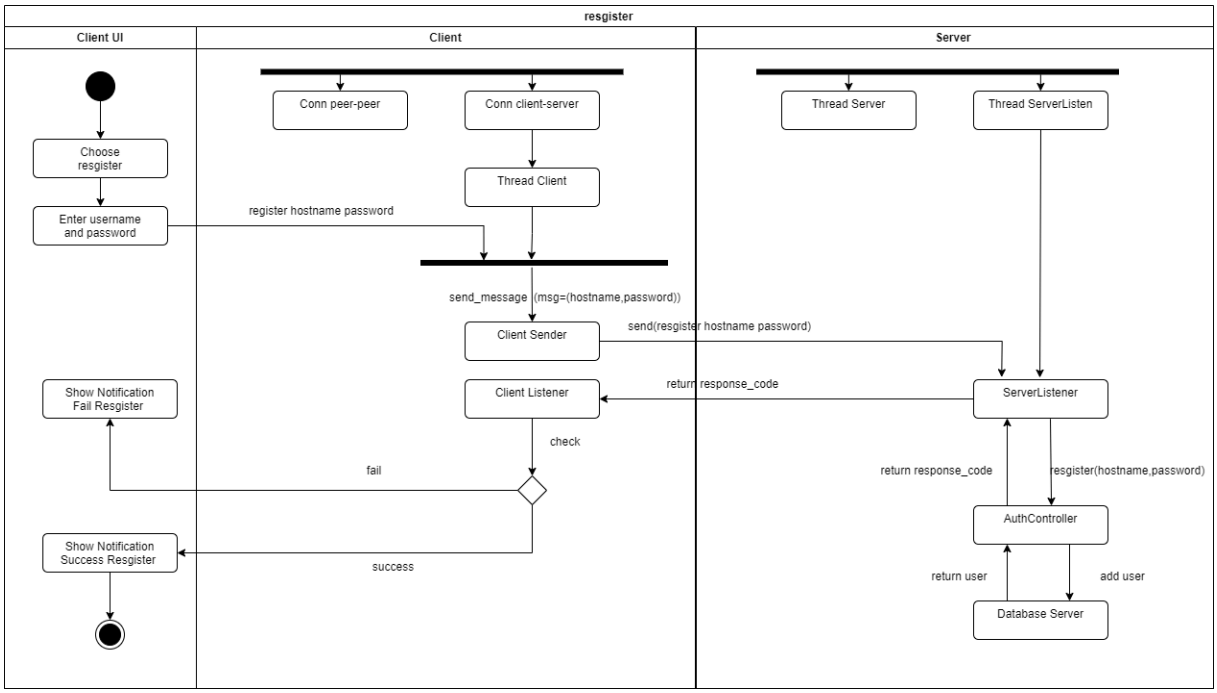


Figure 11: Sơ đồ hoạt động của chức năng đăng ký

Login

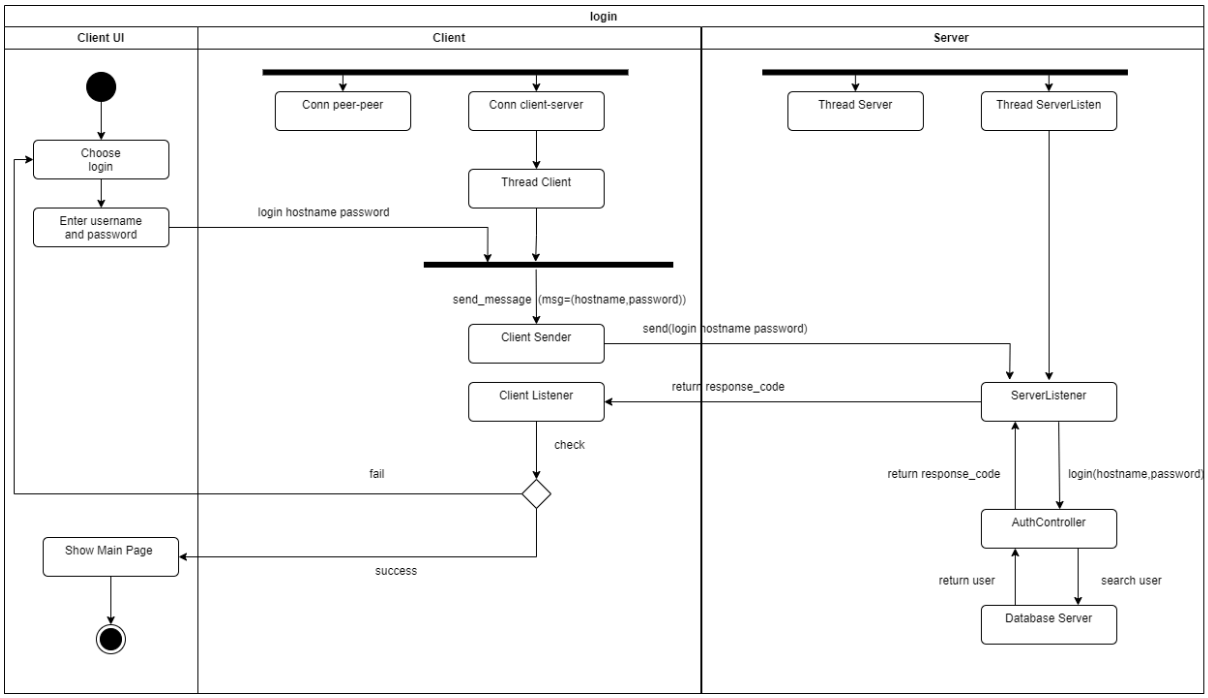


Figure 12: Sơ đồ hoạt động của chức năng đăng nhập

Ping

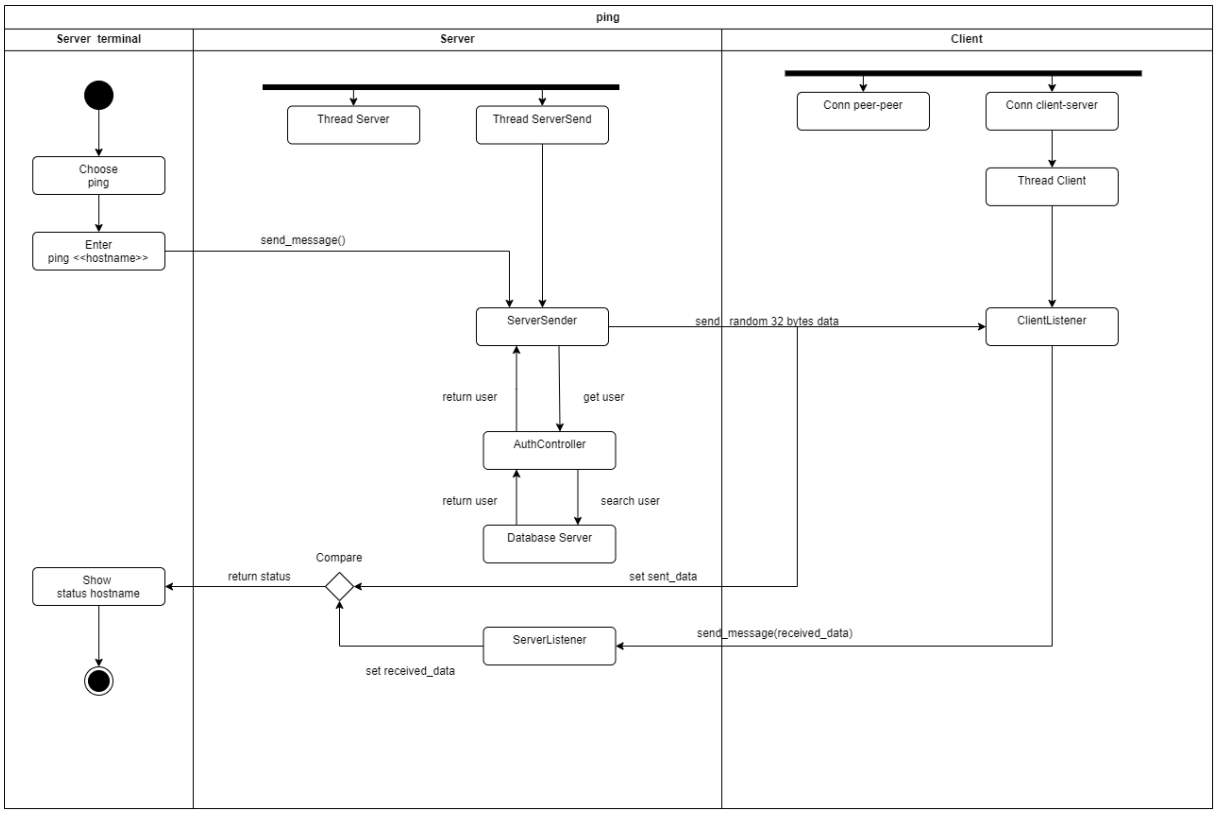


Figure 13: Sơ đồ hoạt động của chức năng ping

Discover

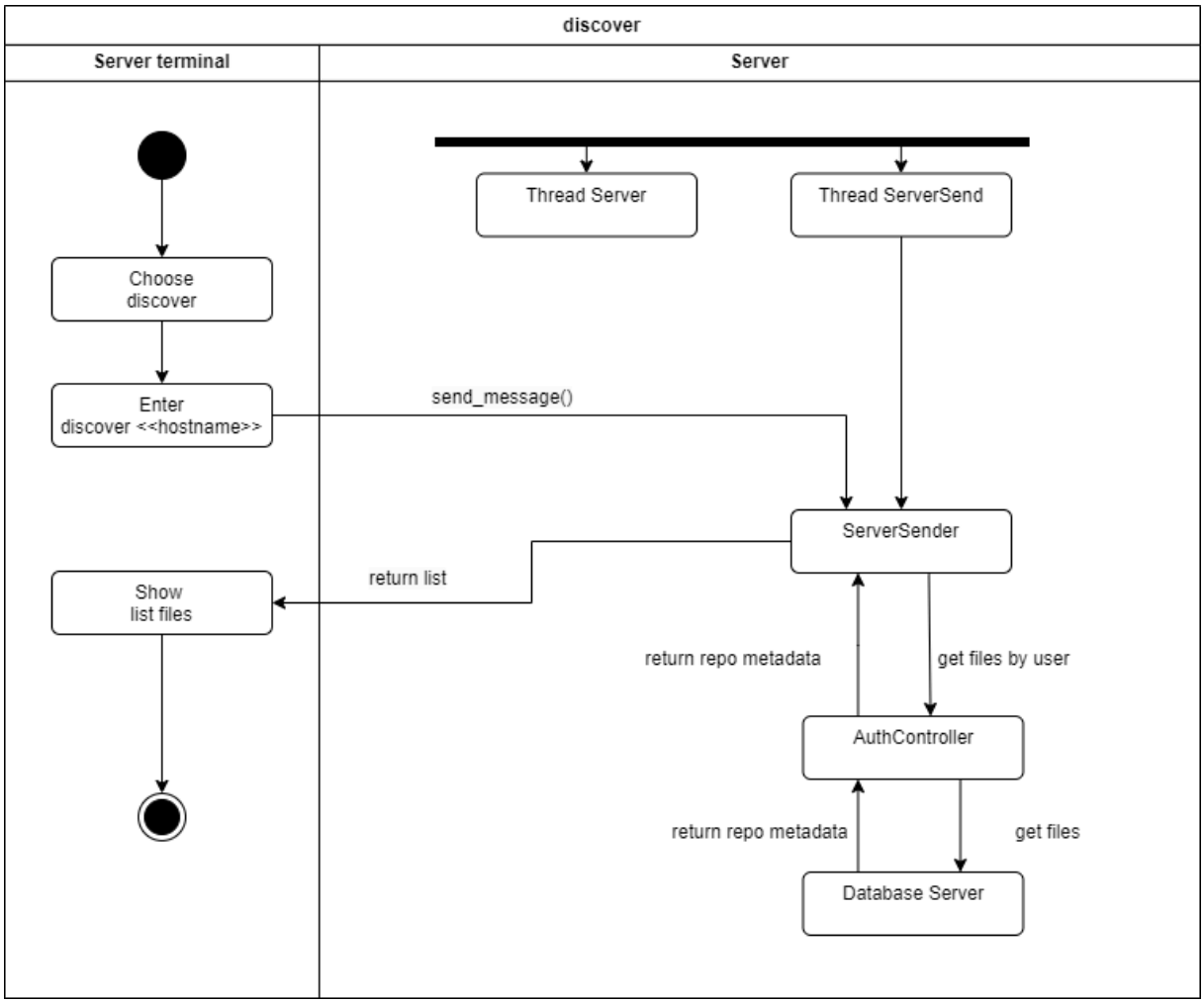


Figure 14: Sơ đồ hoạt động của chức năng discover

Publish

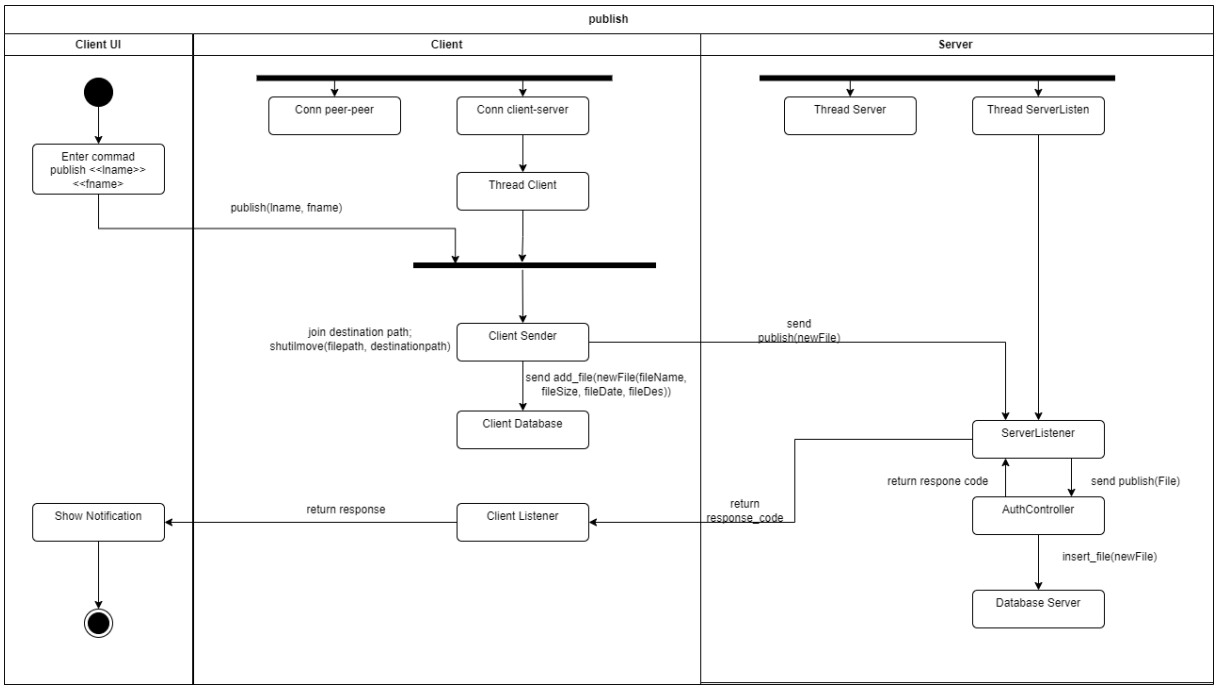


Figure 15: Sơ đồ hoạt động của chức năng publish

Fetch

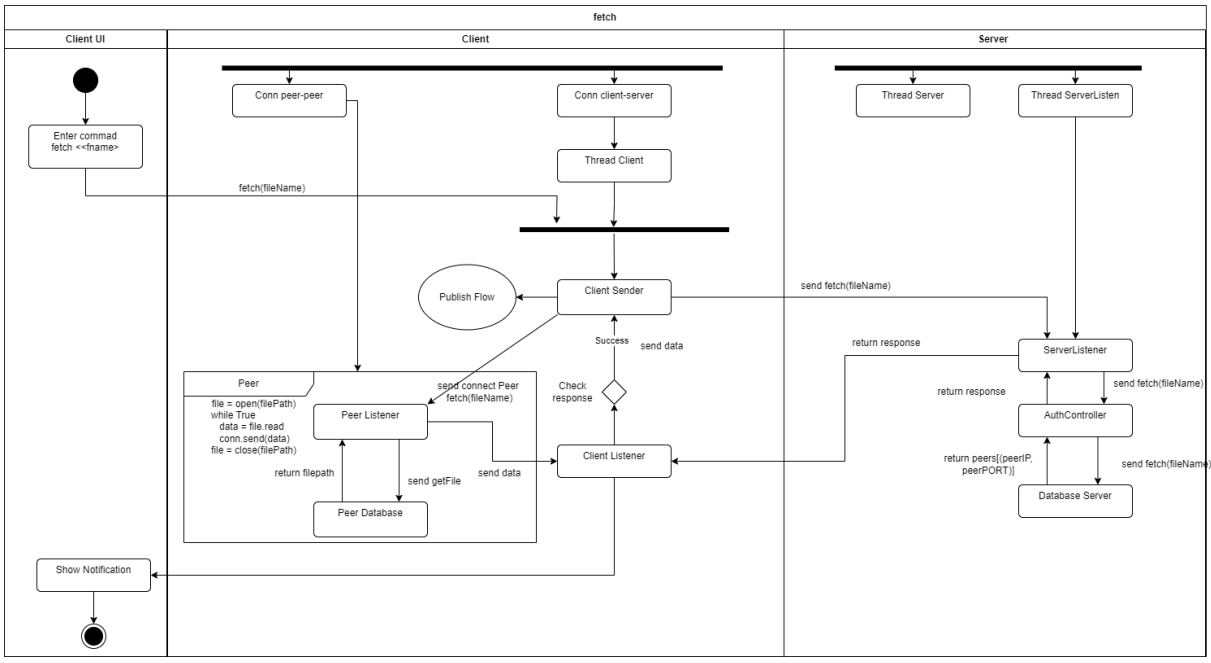


Figure 16: Sơ đồ hoạt động của chức năng fetch

Design UI

Tutorial - Guides