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

MẠNG MÁY TÍNH

Lớp TN01

BÀI TẬP LỚN 1

Giảng viên hướng dẫn: Bùi Xuân Giang

Sinh viên: Trang Sĩ Trọng - 2110621
Đặng Dương Minh Nhật - 2110416
Lã Nguyễn Gia Hy - 2110231
Trần Lê Quốc Khánh - 2111498

Thành phố Hồ Chí Minh, Ngày 19 tháng 12 năm 2023

Mục lục

1 Phân công công việc

STT	Họ và tên	MSSV	Công việc
1	Đặng Dương Minh Nhật	2110416	Hiện thực hệ thống (chính)
2	Trang Sĩ Trọng	2110621	Viết báo cáo (chính) + Đề xuất ý tưởng
3	Lã Nguyễn Gia Hy	2110231	Viết báo cáo (phụ) + Thiết kế giao thức
4	Trần Lê Quốc Khánh	2111498	Hiện thực hệ thống (phụ) + Thuyết trình

2 Bối cảnh

Trao đổi, chia sẻ tài liệu là một nhu cầu lớn và quan trọng trong thời đại công nghệ phát triển. Trong bối cảnh đất nước thực hiện chuyển đổi số, việc xây dựng các ứng dụng có khả năng tìm kiếm, trao đổi tài liệu một cách nhanh chóng, an toàn, tiện lợi sẽ giúp quá trình làm việc hiệu quả hơn. Một ứng dụng có khả năng tìm kiếm và chia sẻ file là rất đáng được đề cao và cần thiết.

Vì vậy, trong dự án này, nhóm sẽ xây dựng một ứng dụng chia sẽ file theo mô hình peer-to-peer phục vụ cho nhu cầu chia sẻ file cho các máy trong cùng một mạng Internet.

3 Phân tích yêu cầu

3.1 Yêu cầu chức năng

- **Thông báo về file:** Client có thể thông báo cho server về một file tên fname ở repo cục bộ lname thông qua lệnh publish fname lname.
- **Lấy file:** Client có thể lấy được bản sao của một file tên fname và lưu vào repo cục bộ thông qua lệnh fetch fname.
- **Chia sẻ file:** Client có thể gửi file từ repo cục bộ của mình cho một client khác.
- **Lưu thông tin client và danh sách file:** Server phải lưu thông tin về client và danh sách các file được client thông báo trên hệ thống.
- **Theo dõi danh sách file:** Server có thể kiểm tra các file nằm trong repo cục bộ của một client tên hostname thông qua lệnh discover hostname.
- **Kiểm tra tình trạng client:** Server có thể kiểm tra xem một client tên hostname có đang hoạt động hay không thông qua lệnh ping hostname.

3.2 Yêu cầu phi chức năng

- Đảm bảo bảo mật thông tin của client.
- Trong điều kiện mạng nhanh và tương đối ổn định, file được truyền đi đảm bảo thời gian tối đa là 2s đối với các file dưới 100MB.
- Tối đa 20 client có thể cùng tải một file từ thư mục cục bộ của một client tại một thời điểm.

- Thời gian phản hồi từ server ngắn, dưới 0.5s.
- Client có thể gửi nhận các file với những định dạng khác nhau.

4 Cơ sở lý thuyết

4.1 Kiến trúc peer-to-peer

Trong kiến trúc P2P, ứng dụng khai thác khả năng giao tiếp trực tiếp giữa các người dùng, được gọi là peer. Nhà cung cấp dịch vụ không sở hữu các peer, mà thay vào đó là các thiết bị đầu cuối điều khiển bởi người dùng. Kiến trúc này rất ít (hoặc không) phụ thuộc vào một máy chủ chuyên dụng.
Một trong những đặc điểm nổi bật của kiến trúc P2P là khả năng tự mở rộng của nó. Hơn nữa, kiến trúc P2P cũng có hiệu quả về chi phí nhờ tính không yêu cầu một kiến trúc hạ tầng đáng kể và băng thông cho server (trái ngược với mô hình client-server). Mặc dù vậy, các ứng dụng P2P cũng có những nhược điểm như các thách thức về độ bảo mật, hiệu suất và độ tin cậy do cấu trúc phân tán cao của kiến trúc.

4.2 Các giao thức được sử dụng

4.2.1 Giao thức sẵn có

TCP/IP: Đây là sự kết hợp giữa giao thức TCP (ở tầng transport) có chức năng xác định các ứng dụng và tạo ra các kênh giao tiếp. TCP cũng có chức năng quản lý các thông tin khi được chia nhỏ để truyền tải qua internet. Giao thức này sẽ tập hợp các thông tin này theo đúng thứ tự, đảm bảo truyền tải thông tin chính xác tới địa chỉ đến. Trong khi đó giao thức IP (ở tầng mạng) sẽ đảm bảo thông tin được truyền đến đúng địa chỉ. IP sẽ gán các địa chỉ và định tuyến từng gói thông tin. Mỗi mạng sẽ có 1 địa chỉ IP để xác định được chính xác nơi chuyển/nhận thông tin, dữ liệu.

4.2.2 Giao thức tự định nghĩa

Để giao tiếp giữa server và các client một cách hiệu quả, cũng như cung cấp thêm khả năng giao tiếp giữa các client, nhóm đã xây dựng một giao thức trao đổi thông tin đơn giản, thuận tiện và dễ sử dụng, được gọi là P2PFS. Sau đây là định nghĩa của **Giao thức P2PFS:**

- Các loại thông điệp được trao đổi: REQUEST, LIST, GET, PING, DISCOVER, PUBLISH, FETCH, GET, DELETE, RESPONSE.

Thông điệp	Mục đích sử dụng
REQUEST	Sử dụng để client gửi yêu cầu kết nối đến server
LISTALL	Sử dụng để client gửi yêu cầu lấy danh sách các file được publish bởi tất cả các client
LISTMY	Sử dụng để client gửi yêu cầu lấy danh sách các file mà client đã publish
PING	Sử dụng cho lệnh **ping** của server
DISCOVER	Sử dụng cho lệnh **discover** của server
RESPONSE <code>	Phản hồi lại các thông điệp khác
PUBLISH	Sử dụng cho lệnh **publish** của client
FETCH	Sử dụng cho lệnh **fetch** của client
GET	Sử dụng để nhận client nhận file được gửi từ client khác
DELETE	Sử dụng để client xóa file đã được client **publish**

- Quy tắc (Rules): Mã trạng thái (code) sẽ được trả về cùng với thông điệp RESPONSE với những quy ước sau:
  - 200: Kết nối thành công.
  - 404: Kết nối thất bại.

5 Mô tả các hàm

5.1 Class Server

- __init__(self)
  - **Mô tả:** constructor cho class Server, tạo một socket để server nhận kết nối từ các client.
  - **Giao thức sử dụng:** TCP/IP
- server_run(self)
  - **Mô tả:** nhận kết nối từ các client và đưa từng kết nối vào mỗi thread để xử lý.
  - **Giao thức sử dụng:** Không
- handle_client(self, conn, addr)

  - **Mô tả:** nhận kết nối của một client và giao tiếp với client cho đến khi client đóng kết nối. Hàm này được từng thread trong server sử dụng để xử lý request.
  - **Giao thức sử dụng:** TCP/IP

- accept_connection(self, conn, addr)
  - **Mô tả:** nhận yêu cầu kết nối từ kết nối conn của một client và phản hồi lại client đó.
  - **Giao thức sử dụng:** P2PFS, TCP/IP

- receive_message(self, conn, addr)
  - **Mô tả:** nhận thông điệp giao thức từ client.
  - **Giao thức sử dụng:** P2PFS, TCP/IP

- publish(self, conn, addr)
  - **Mô tả:** xử lí và đáp lại thông điệp PUBLISH từ client.
  - **Giao thức sử dụng:** P2PFS, TCP/IP

- send_list_clients(self, conn, addr)
  - **Mô tả:** gửi cho client danh sách các client khác đã publish file mà client yêu cầu.
  - **Giao thức sử dụng:** P2PFS, TCP/IP

- ping(self, hostname)
  - **Mô tả:** kiểm tra xem client tên hostname có đang hoạt động hay không
  - **Giao thức sử dụng:** P2PFS, TCP/IP

- discover(self, hostname)
  - **Mô tả:** truy xuất danh sách các file đang nằm trong repo cục bộ của client tên hostname.
  - **Giao thức sử dụng:** P2PFS, TCP/IP

- send_list(self, conn, addr)
  - **Mô tả:** gửi cho client danh sách tất cả các file đã được publish bởi tất cả các client.
  - **Giao thức sử dụng:** P2PFS, TCP/IP

- send_my_file(self, conn, addr)

  - **Mô tả:** gửi cho client danh sách tất cả các file đã được publish bởi client đó.
  - **Giao thức sử dụng:** P2PFS, TCP/IP

- delete_file(self, conn, addr)
  - **Mô tả:** xử lí và đáp lại thông điệp DELETE từ client.
  - **Giao thức sử dụng:** P2PFS, TCP/IP

5.2 Class Client

- __init__(self)
  - **Mô tả:** constructor cho class Client, tạo một socket để kết nối với server và một socket để nhận kết nối từ các client khác.
  - **Giao thức sử dụng:** TCP/IP
- init_connection(self)
  - **Mô tả:** gửi yêu cầu kết nối tới server.
  - **Giao thức sử dụng:** P2PFS, TCP/IP
- client_run(self)
  - **Mô tả:** nhận kết nối từ các client và đưa từng kết nối vào mỗi thread để xử lý.
  - **Giao thức sử dụng:** Không
- handle_connect(self, conn, addr)
  - **Mô tả:** nhận kết nối từ một client khác hoặc server. Hàm này được từng thread trong server sử dụng để xử lý request.
  - **Giao thức sử dụng:** TCP/IP
- send_message(self, message)
  - **Mô tả:** gửi thông điệp giao thức tới server.
  - **Giao thức sử dụng:** P2PFS, TCP/IP
- receive_message(self)
  - **Mô tả:** nhận thông điệp giao thức từ server.

  - **Giao thức sử dụng:** P2PFS, TCP/IP

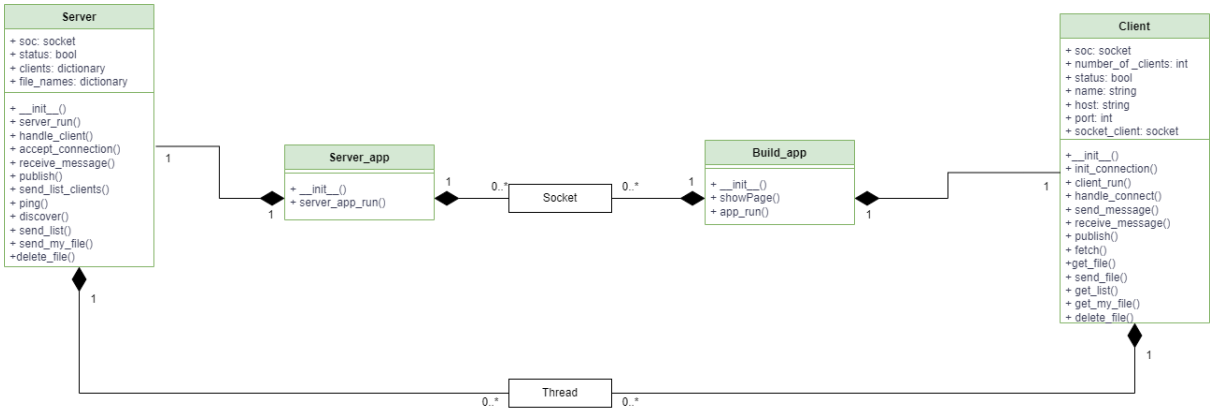
- publish(self, file_name, local_name)
  - **Mô tả:** thông báo cho server về file có tên file_name ở repo cục bộ local_name.
  - **Giao thức sử dụng:** P2PFS, TCP/IP
- request_file(self, file_name)
  - **Mô tả:** gửi yêu cầu file có tên file_name đến server.
  - **Giao thức sử dụng:** P2PFS, TCP/IP
- fetch(self, file_name)
  - **Mô tả:** nhận file tên file_name nếu repo cục bộ của client chưa có file này.
  - **Giao thức sử dụng:** Không
- get_file(self, addr, local_file)
  - **Mô tả:** gửi yêu cầu nhận file và nhận file đã yêu cầu từ client khác và lưu về repo cục bộ local_name.
  - **Giao thức sử dụng:** P2PFS, TCP/IP
- send_file(self, conn, addr)
  - **Mô tả:** gửi cho client khác file mà client đó yêu cầu.
  - **Giao thức sử dụng:** P2PFS, TCP/IP
- get_list(self)
  - **Mô tả:** gửi yêu cầu lấy danh sách tất cả các file đã được publish bởi tất cả các client đến server.
  - **Giao thức sử dụng:** P2PFS
- get_my_file(self)
  - **Mô tả:** gửi yêu cầu lấy danh sách tất cả các file mà client này đã publish đến server.
  - **Giao thức sử dụng:** P2PFS
- delete_file(self, file_name)

– **Mô tả:** gửi yêu cầu xóa file tên file_name trong danh sách các file mà client này đã publish đến server.

– **Giao thức sử dụng:** P2PFS

5.2.1 Class Diagram

Đây là class diagram của hệ thống



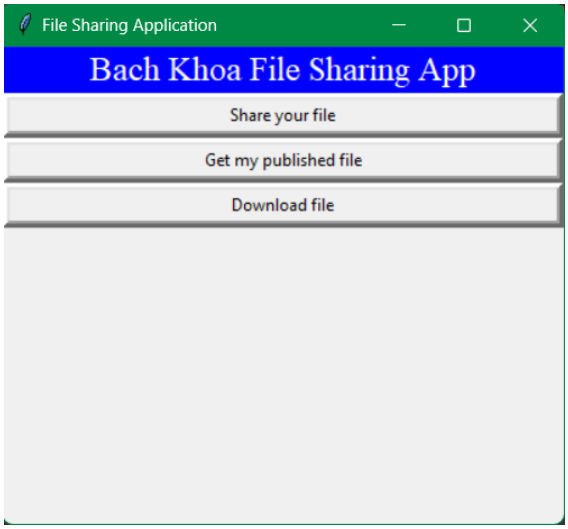
Trang chính của client khi khởi động ứng dụng

6 Đánh giá chung

6.1 Kết quả đạt được

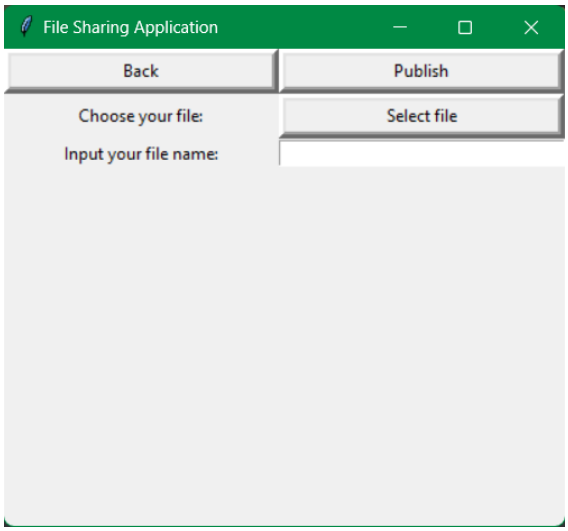
6.1.1 Kiểm tra và chạy thử ứng dụng

Hệ thống chia sẻ file của nhóm đã đạt được các yêu cầu đề ra và ngoài ra còn đáp ứng thêm một số yêu cầu khác. Dưới đây là kết quả chạy ứng dụng:
Sau khi khởi động, màn hình bắt đầu ứng dụng bên phía client như hình.



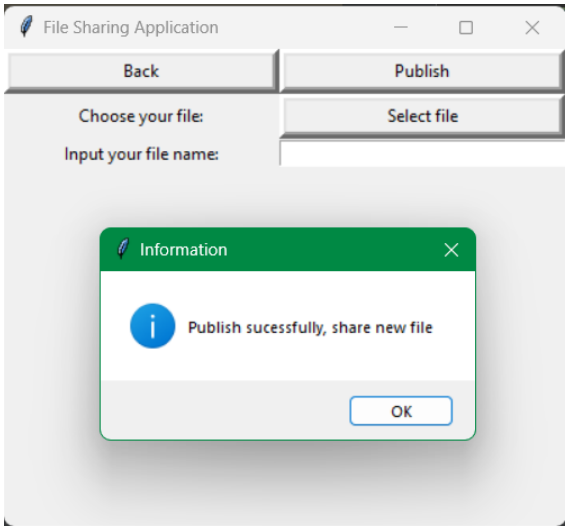
Trang chính của client khi khởi động ứng dụng

Khi client bấm vào nút "Share your file", giao diện để client thực hiện tính năng publish file sẽ hiện lên.



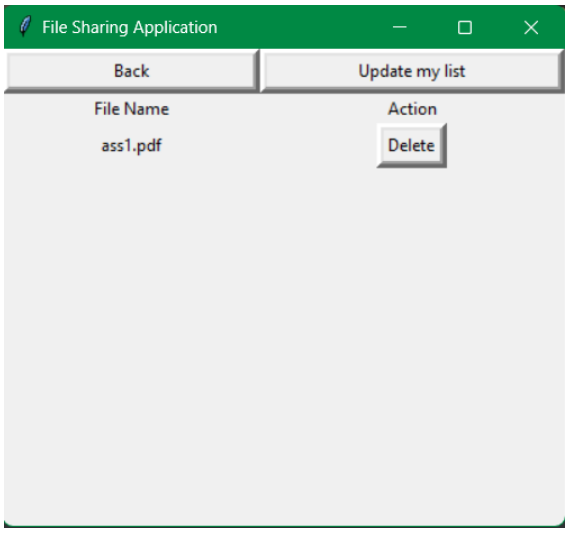
Giao diện publish file của client

Client bấm vào nút "Select file" để chọn file từ repo cục bộ để tải lên và nhập tên file để publish lên hệ thống. Tên file được publish lên không nhất thiết phải giống với tên file ở cục bộ nhưng phải cùng định dạng. Đây là thông báo khi client publish file thành công.



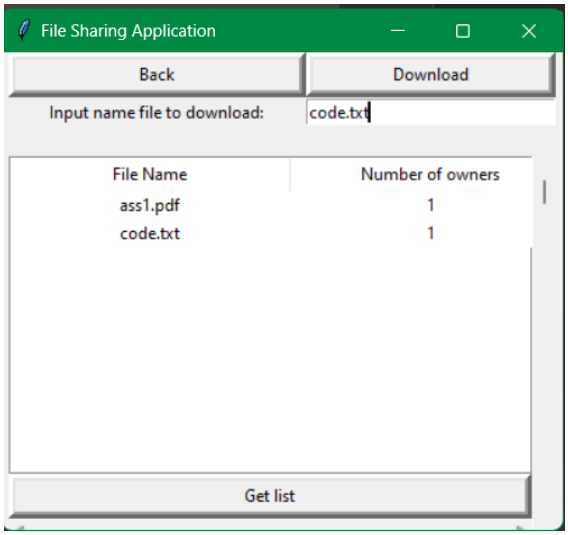
Publish thành công

Ở trang chính, client có thể bấm vào nút "Get my published file" để xem danh sách mà bản thân đã publish lên hệ thống. Ở trang này, client có thể bấm nút "Delete" một file để khi không muốn chia sẻ file này nữa.



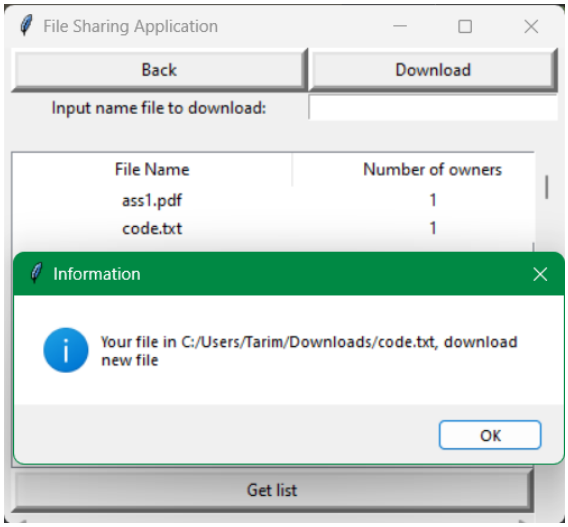
Giao diện liệt kê các file đã được upload của client

Ở trang chính, client có thể bấm vào nút "Download file" để tải file.



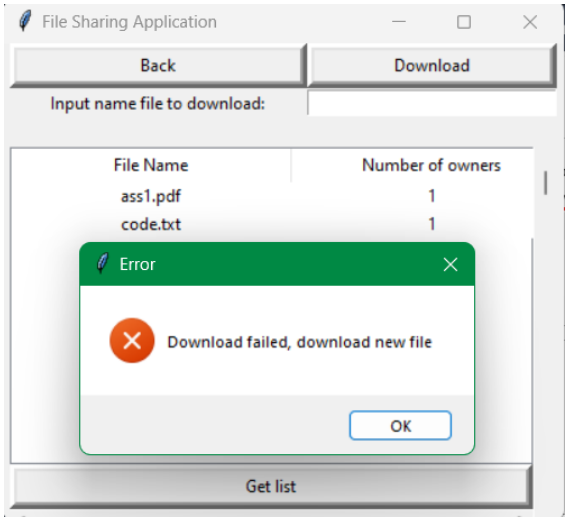
Giao diện trang tải file

Client có thể bấm nút "Get list" để xem danh sách các file đã được publish lên hệ thống bởi tất cả các client. Client nhập tên file cần tải và bấm "Download" để tải file về. Đây là thông báo khi client tải file thành công.



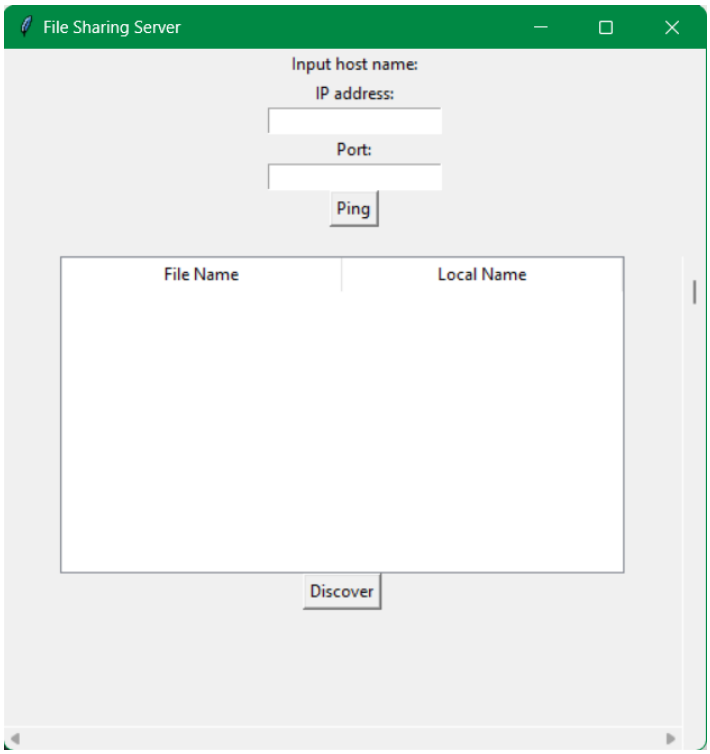
Tải file thành công

Đây là thông báo khi client tải file thất bại.



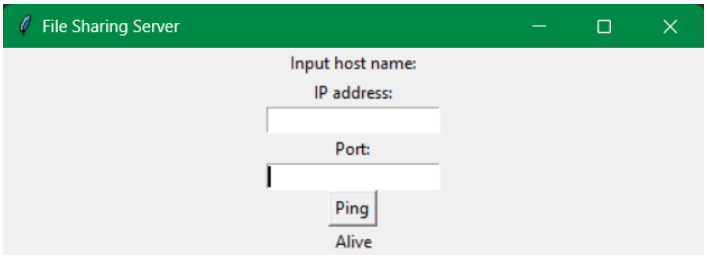
Tải file thất bại

Sau khi khởi động, màn hình bắt đầu ứng dụng bên phía server như hình.

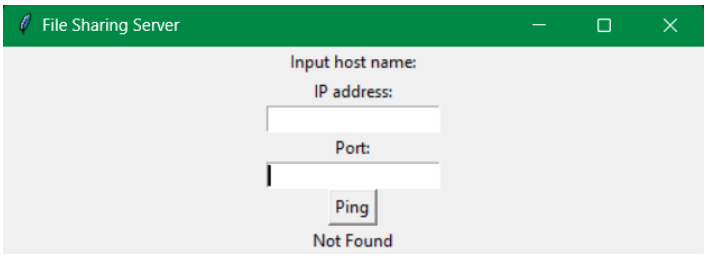


Trang chính của server khi khởi động ứng dụng

Server nhập vào IP và Port và bấm nút "Ping" để ping xem client còn hoạt động không.

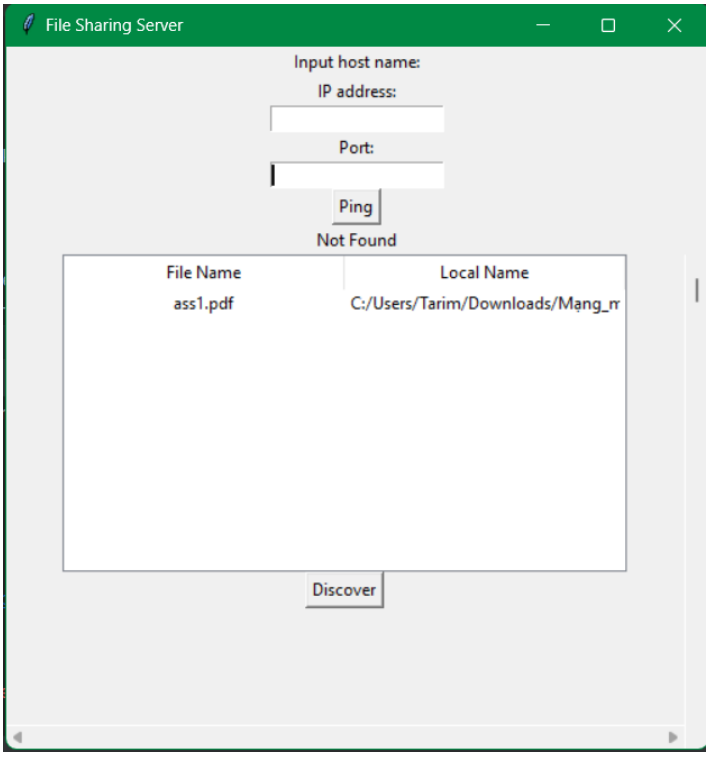


Ping một client còn hoạt động



Ping một client không còn hoạt động

Server nhập vào IP và Port và bấm nút "Discover" để discover các file đã được publish bởi client.



Discover các file của một client

Đây là đường dẫn đến link Github cho sản phẩm của nhóm: Github

Về lập trình, thông qua bài tập lớn hiện thực hệ thống chia sẻ file peer-to-peer này, nhóm đã có thể:

- Vận dụng các kiến thức đã học về TCP/IP và hiện thực một ứng dụng mạng bằng ngôn ngữ Python sử dụng thư viện socket.
- Thiết kế một giao thức và cấu trúc giao tiếp giữa client và server.
- Thiết kế GUI cho người dùng để có thể thực hiện các lệnh cơ bản thông qua thư viện tkinter của Python.
- Vận dụng kiến thức về lập trình song song để xây dựng một ứng dụng đa luồng thông qua việc sử dụng thư viện threading .

6.2 Những hạn chế và định hướng nâng cấp trong tương lai

6.2.1 Hạn chế

- Thiết kế giao diện còn đơn giản.

- Chưa giải quyết được trường hợp nhiều file có nội dung khác nhau nhưng được publish lên hệ thống với cùng một tên, khi đó client không biết được file nào là file mình thực sự cần.
- Khi một client tắt hoạt động và sau đó tham gia lại vào mạng thì server sẽ nhìn nhận client đó như một client mới hoàn toàn, thông tin về các file đã được client này publish trước đó không còn được lưu trên server nữa.

6.2.2 Định hướng nâng cấp

- Thiết kế lại giao diện cho đặc sắc hơn.
- Thêm nút tải file để thuận tiện cho việc tải file hơn so với khi phải nhập chính xác tên file.
- Lưu thêm thông tin về địa chỉ MAC của client để giữ thông tin về các file đã được client publish trong trường hợp client tắt hoạt động và sau đó tham gia lại vào mạng.

7 Tài liệu tham khảo

Tài liệu

[1] Kurose, J. R., & Ross, K. (2021b). Computer Networking Global Edition.

[2] Python. "Socket — Low-Level Networking Interface — Python 3.12.1 Documentation." Python.org, 2023, https://docs.python.org/3/library/socket.html.

[3] Python. "Threading — Thread-Based Parallelism — Python 3.12.1 Documentation." Python.org, 2023 https://docs.python.org/3/library/threading.html.

[4] Python Software Foundation. "Tkinter — Python Interface to Tcl/Tk — Python 3.12.1 Documentation." Python.org, 2023, https://docs.python.org/3/library/tkinter.html.