**Mục lục**

[**1.** **Tổng quan** 3](#_Toc45039744)

[**1.1.** **Tổng quan về đề tài** 3](#_Toc45039745)

[**1.2.** **Phân chia công việc** 3](#_Toc45039746)

[**2.** **Cơ sở lý thuyết** 3](#_Toc45039747)

[**2.1.** **Giao thức UDP** 3](#_Toc45039748)

[**2.2.** **Giao thức TCP** 4](#_Toc45039749)

[**2.3.** **Thư viện WinSock** 5](#_Toc45039750)

[**2.4.** **Các kỹ thuật vào ra trên WinSock** 6](#_Toc45039751)

[**3.** **Cấu trúc chương trình** 7](#_Toc45039752)

[**3.1.** **CLIENT** 7](#_Toc45039753)

[3.1.1. Header Files 7](#_Toc45039754)

[3.1.2. Resource Files 7](#_Toc45039755)

[3.1.3. Source Files 7](#_Toc45039756)

[**3.2.** **SERVER** 7](#_Toc45039757)

[3.2.1. Header Files 7](#_Toc45039758)

[3.2.2. Resource Files 7](#_Toc45039759)

[3.2.3. Source Files 8](#_Toc45039760)

[**4.** **Khuôn dạng gói tin, cách truyền tin** 8](#_Toc45039761)

[**4.1.** **Khuôn dạng gói tin** 8](#_Toc45039762)

[**4.2.** **Cách truyền tin** 9](#_Toc45039763)

[**5.** **Use case** 9](#_Toc45039764)

[**5.1.** **Biểu đồ use case** 9](#_Toc45039765)

[**5.2.** **Use case kết nối tới server** 10](#_Toc45039766)

[**5.3.** **Use case yêu cầu chuyển tiếp file** 10](#_Toc45039767)

[**5.4.** **Use case yêu cầu tìm kiếm file** 11](#_Toc45039768)

[**6.** **Biểu đồ hoạt động** 11](#_Toc45039769)

[**6.1.** **Kết nối tới server** 12](#_Toc45039770)

[**6.2.** **Chuyển tiếp file** 13](#_Toc45039771)

[**6.3.** **Tìm kiếm file** 14](#_Toc45039772)

[**7.** **Giao diện chương trình** 14](#_Toc45039773)

[**7.1.** **SERVER** 14](#_Toc45039774)

[**7.2.** **CLIENT** 14](#_Toc45039775)

[**Tài Liệu Tham Khảo** 17](#_Toc45039776)

# **Tổng quan**

* 1. **Tổng quan về đề tài**

1. **Tên đề tài**: đề tài 7, xây dựng ứng dụng chia sẻ file 1

* Server có các chức năng:
* Xử lý yêu cầu kết nối của client, trả về ID cho client đó
* Nhận yêu cầu tìm kiếm file theo tên của client A nào đó
* Gửi lệnh tìm kiếm tới các client khác đang kết nối
* Gửi lại cho client A danh sách các client có file mà client A tìm kiếm
* Client có các chức năng:
* Gửi yêu cầu tìm kiếm một file lên server
* Nhận danh sách các client có file mà client đó yêu cầu tìm kiếm
* Lựa chọn ra một client trong số đó để yêu cầu download file
* Yêu cầu server chuyển tiếp file đến client khác
* Thực hiện tìm kiếm file theo yêu cầu của client khác(sẽ tìm kiếm file trong thư mục Data nằm cùng thư mục với file .exe của client). Cụ thể, client sẽ tìm kiếm đệ quy vào thư mục Data(tìm kiếm tất cả các file nằm trong Data và các thư mục con của Data)

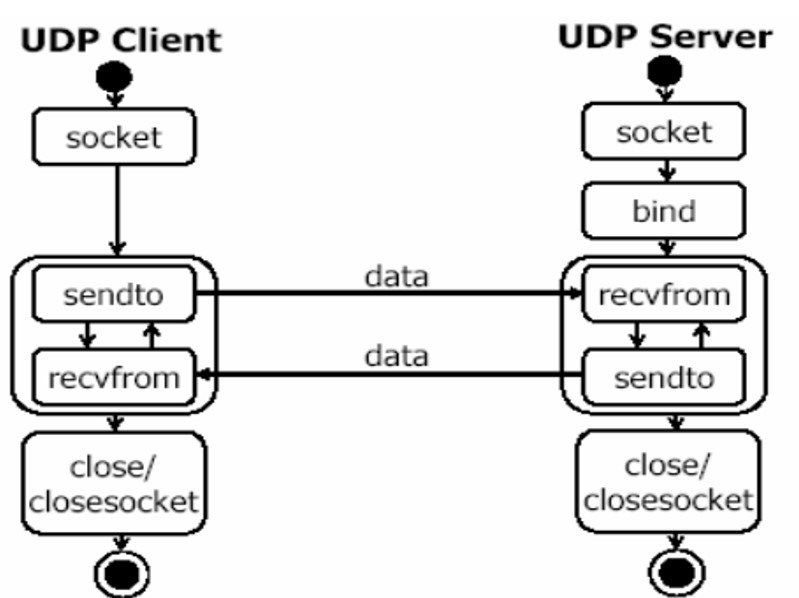
1. **Ngôn ngữ sử dụng**: C, C++
2. **Trình biên dịch sử dụng**: Visual Studio 2015
   1. **Phân chia công việc**

* Phạm Hồng Phúc
* Xây dựng khuôn dạng gói tin, cách truyền tin, cách truyền file và code các hàm liên quan
* Tìm hiểu cách tạo giao diện và code tạo giao diện
* Đỗ Viết Trí
* Tìm hiểu các hàm tương tác với file và code các hàm tương tác với file
* Tìm hiểu cách tạo giao diện

1. **Cơ sở lý thuyết**
   1. **Giao thức UDP**

**UDP** (***User Datagram Protocol***) là một trong những giao thức cốt lõi của bộ giao thức TCP/IP. UDP không cung cấp sự tin cậy, tuy nhiên UDP nhanh và hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian.

Do bản chất không trạng thái của nó nên nó hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu.



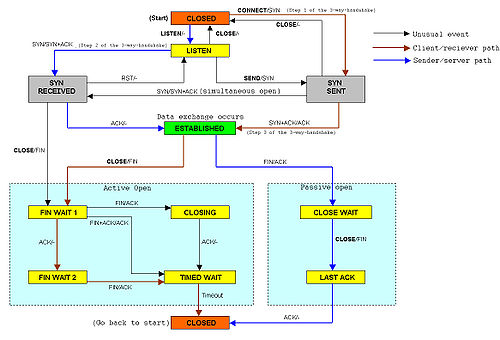
Hình 1: mô hình UDP

* 1. **Giao thức TCP**

**TCP** (***Transmission Control Protocol***) là một trong các giao thức cốt lõi của bộ giao thức TCP/IP. Sử dụng TCP, các ứng dụng trên các máy chủ 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. TCP còn phân biệt giữa dữ liệu của nhiều ứng dụng (chẳng hạn, dịch vụ Web và dịch vụ thư điện tử) đồng thời chạy trên cùng một máy chủ.

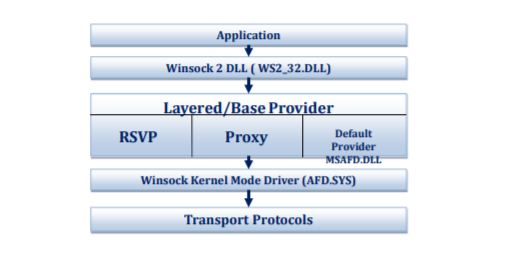
TCP hỗ trợ nhiều giao thức ứng dụng phổ biến nhất trên Internet và các ứng dụng kết quả, trong đó có WWW, thư điện tử và Secure Shell.



Hình 2: Giao thức TCP

* 1. **Thư viện WinSock**

WinSock(Window Socket) là bộ thư viện liên kết động của Microsoft, cung cấp các API để xây dựng các ứng dụng mạng hiệu năng cao



Hình 3: kiến trúc của WinSock

WinSock gồm ba tầng

* **Provider** do nhà sản xuất của các giao thức cung cấp. Tầng này bổ sung giao thức của các tầng mạng khác nhau cho WinSock như TCP/IP, IPX/SPX, AppleTalk, NetBIOS...tầng này vẫn chạy ở UserMode
* **WinSock Kernel Mode Driver (AFD.SYS)** là driver chạy ở KernelMode, nhận dữ liệu từ tầng trên, quản lý kết nối, bộ đệm, tài nguyên liên quan đến socket và giao tiếp với driver điều khiển thiết bị.
* **Transport Protocols** là các driver ở tầng thấp nhất, điều khiển trực tiếp thiết bị. Các driver này do nhà sản xuất phần cứng xây dựng, và giao tiếp với AFD.SYS thông qua giao diện TDI ( Transport Driver Interface)

Để xây dựng một chương trình sử dụng WinSock, ta cần phải chú ý những điều sau

* Thêm vào phần tiêu đề của chương trình: winsock2.h, ws2tcpip.h
* Khai báo tiền xử lý #pragma comment(lib, "Ws2\_32.lib")
  1. **Các kỹ thuật vào ra trên WinSock**

Trên WinSock có một số kỹ thuật vào ra nhưng sau

* Kỹ thuật đa luồng: đơn giản nhưng sử dụng tài nguyên không hiệu quả
* Kỹ thuật thăm dò(select()): đơn giản nhưng bị hạn chế do cấu trúc fd\_set chỉ quản lý được 1024 socket
* Kỹ thuật vào ra theo thông báo: đơn giản nhưng có nhược điểm là một cửa sổ trở thành nút thắt cổ chai trong ứng dụng nếu phải xử lý quá nhiều kết nối
* Kỹ thuật vào ra theo sự kiện: đơn giản, không yêu cầu phải có cửa sổ nhưng mỗi luồng chỉ quản lý được 64 bộ nghe sự kiện
* Kỹ thuật vào ra overlapped theo sự kiện: có hiệu năng cao, nhưng mỗi luồng chỉ quản lý được 64 bộ nghe sự kiện
* Kỹ thuật vào ra overlapped, xử lý bằng completon routine: : hiệu năng cao, không hạn chế số kết nối có thể xử lý nhưng completion routine không thực hiện được các tác vụ nặng
* Kỹ thuật vào ra overlapped theo completion port: : hiệu năng cao, không hạn chế số kết nối có thể xử lý. Là mô hình phù hợp nhất cho các ứng dụng cần xử lý số kết nối lớn nhưng là mô hình khó sử dụng

1. **Cấu trúc chương trình**

Chương trình xây dựng theo hướng cấu trúc, bao gồm hai project là CLIENT và SERVER

* 1. **CLIENT**

***Client là một empty project sử dụng win32(hàm WinMain) để tạo giao diện***

* + 1. Header Files

1. CONST.h: chứa những hằng số của chương trình client
2. Common.h: chứa những cấu trúc(struct) và tiêu đề hàm có chức năng chung của chương trình client
3. InteractFile.h: chứa những cấu trúc và tiêu đề hàm có chức năng tương tác với file(lấy tên file, tìm kiếm file, đọc file, kiểm tra sự tồn tại của file, …)
4. TCP\_SOCKET.h: chứa những cấu trúc và tiêu đề hàm có chức năng định nghĩa khuôn dạng gói tin, cách đóng gói và cách truyền tin trong chương trình
   * 1. Resource Files
5. Common.cpp: định nghĩa các hàm được khai báo trong Common.h
6. InteractFile.cpp: định nghĩa các hàm được khai báo trong InteractFile.h
7. TCP\_SOCKET.cpp: định nghĩa các hàm được khai báo trong TCP\_SOCKET.h
   * 1. Source Files
8. WinMain.cpp: chứa hàm WinMain(hàm chính) của chương trình client đồng thời có chức năng tạo giao diện, tương tác với người dùng và xử lý yêu cầu từ server
   1. **SERVER**

***Server là một empty project(hàm main) không có giao diện(console), sử dụng kỹ thuật vào ra theo thông báo kết hợp với đa luồng để quản lý các client kết nối đến***

* + 1. Header Files

1. CONST.h: chứa những hằng số của chương trình server
2. Common.h: chứa những cấu và tiêu đề hàm có chức năng chung của chương trình server
3. TCP\_SOCKET.h: chứa những cấu trúc và tiêu đề có chức năng định nghĩa khuôn dạng gói tin, cách đóng gói và cách truyền tim trong chương trình
   * 1. Resource Files
4. Common.cpp: định nghĩa các hàm được khai báo trong Common.h
5. TCP\_SOCKET.cpp: định nghĩa các hàm được khai báo trong TCP\_SOCKET.h
   * 1. Source Files
6. SERVER.cpp: chứa hàm main của chương trình phía server, có chức năng quản lý các phiên đăng nhập của các client và xử lý các yêu cầu từ client
7. **Khuôn dạng gói tin, cách truyền tin**
   1. **Khuôn dạng gói tin**

Khuôn dạng dược định nghĩa bằng một cấu trúc có 5 trường

struct Message {

int type;

char fileName[BUFF\_SIZE];

char ID[BUFF\_SIZE];

char data[BUFF\_SIZE + 1];

int dataLen;

};

(BUFF\_SIZE là 10240)

* Type: xác định chức năng mà client và server phải thực hiện

**SERVER**

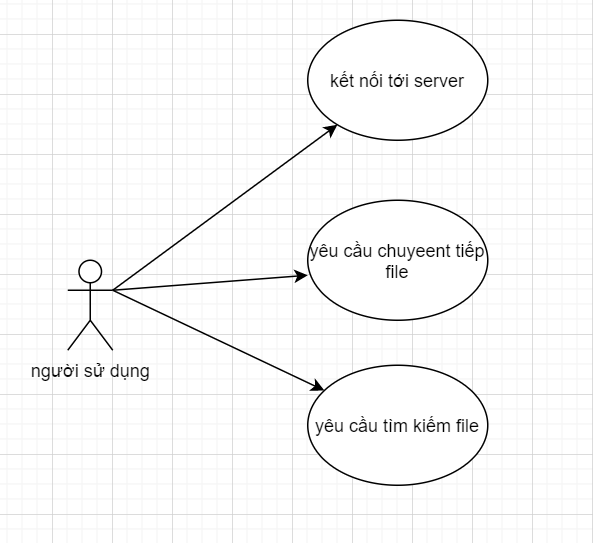
* 100: kết nối thành công, gửi ID lại cho client
* 110: gửi danh sách các client đang kết nối
* 111: gửi danh sách các client có file yêu cầu về client đang tìm kiếm file đấy
* 112: tải từng gói dữ liệu của file tìm kiếm về client đi kèm kích thức của gói dữ liệu đó
* 1120: có chức năng giống 112 nhưng là dấu hiệu của việc kết thúc truyền file về client
* 120: gửi yêu cầu tìm kiếm file từ một client đến các client còn lại
* 121: gửi yêu cầu tải file lên server đến client được chỉ định trong chức năng tìm kiếm file
* 200: yêu cầu chuyển tiếp file về client
* 201: tải từng gói dữ liệu của file chuyển tiếp về client đi kèm với kích thước của gói dữ liệu đó
* 2010: có chức năng giống 201 nhưng là dấu hiệu của việc kết thúc truyền file về client
* 202: ID do client gửi lên có thể kết nối và được phép chuyển tiếp
* 203: ID do client gửi lên không thể kết nối hoặc bị từ chối chuyển tiếp

**CLIENT**

* 300**:**  yêu cầu kết nối và yêu cầu server gửi ID
* 310: gửi yêu cầu tìm kiếm file
* 311: tải từng gói dữ liệu file từ client được chỉ định bởi client khác lên server đi kèm với kích thức của gói dữ liệu đó
* 3110: có chức năng giống 311 nhưng là dấu hiệu của việc kết thúc truyền file lên server
* 312: chỉ định một client để tải file đang tìm kiếm
* 320: không tìm thấy tên file mà server yêu cầu
* 321: tìm thấy tên file mà server yêu cầu
* 400: gửi ID và fileName để yêu cầu chuyển tiếp file đến client tương ứng với ID được gửi
* 401: gửi lần lượt từng gói dữ liệu file chuyển tiếp lên server đi kèm với kích thước của gói dữ liệu đó
* 4010: có chức năng giống 401 nhưng là dấu hiệu của việc kết thúc truyền file lên server
* 410: không cho chuyển tiếp file về client
* 411: cho phép chuyển tiếp file về client
* fileName: chứa thông tin tên file
* ID: chứa thông tin ID của client(ID của client cần chuyển tiếp hoặc yếu cầu chuyển tiếp, hoặc của client thực hiện tìm kiếm file hoặc client được yêu cầu tải file tìm kiếm lên server)
* data: chứa dữ liệu của file
* dataLen: kích thức của trường data
  1. **Cách truyền tin**

Sử dụng tcp socket và kỹ thuật truyền dòng để truyền tin giữa client và server

1. **Use case**
   1. **Biểu đồ use case**



Hình 4: biểu đồ use case

* 1. **Use case kết nối tới server**

1. Đối tượng sử dụng use case: người dùng chương trình
2. tóm tắt: người dùng sử dụng use case này để kết nối tới server và nhận ID từ server
3. Dòng sự kiện

* Hệ thống hiển thị giao diện ban đầu
* Người dùng thực hiện kết nối bằng việc nhập thông tin IP và số hiệu cổng của server, sau đó bấm Connect
* Nếu kết nối thành công , server sẽ gửi trả ID của người dùng
* Nếu không thành công hệ thống sẽ báo lỗi, yêu cầu kết nối lại
  1. **Use case yêu cầu chuyển tiếp file**

1. Đối tượng sử dụng use case: người dùng chương trình
2. Tóm tắt: người dùng sử dụng use case để chuyển tiếp file đến client khác
3. Dòng sự kiện

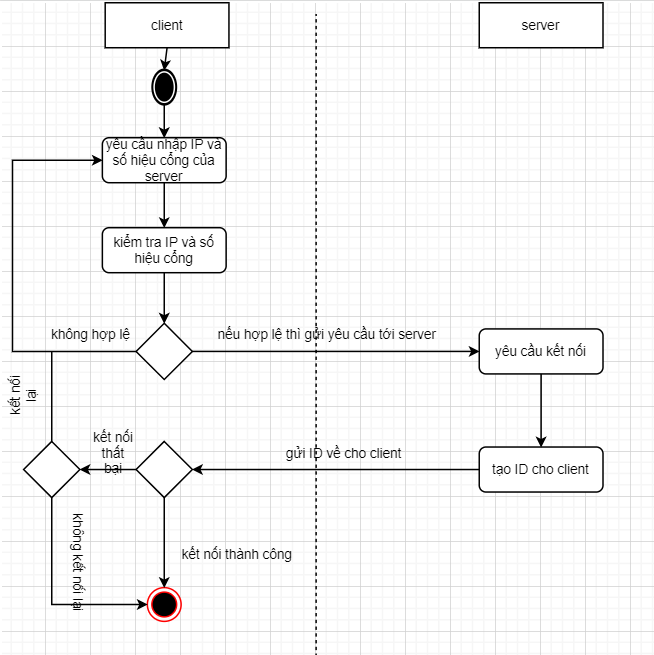
* Use case bắt đầu khi người dùng bấm browse chọn file cần chuyển tiếp và bấm forward để chuyển tiếp file
* Hệ thống sẽ yêu cầu nhập ID của người muốn chuyển tiếp
* Sau khi nhập ID, bấm forward một lần nữa
* Server sẽ tìm kiếm client được client kia yêu cầu, nếu tìm thấy server sẽ yêu cầu được chuyển tiếp file về client đó, nếu không tìm thấy thông báo về cho client yêu cầu chuyển tiếp, kết thúc use case
* Nếu yêu cầu chuyển tiếp được chấp nhận, bắt đầu chuyển tiếp file về client đó, nếu không chấp nhận thì thông báo về cho client yêu cầu chuyển tiếp, kết thúc use case

1. Luồng ngoại lệ: nếu client được yêu cầu chuyển tiếp chính là client yêu cầu chuyển tiếp thì báo lỗi cho người dùng, kết thúc use case
   1. **Use case yêu cầu tìm kiếm file**
2. Đối tượng sử dụng use case: người sử dụng chương trình
3. Tóm tắt: người dùng sử dụng use case để tìm kiếm file ở các client khác
4. Dòng sự kiện

* Use case bắt đầu khi người dùng nhập tên file cần chuyển tiếp và bấm nút search
* Server sẽ gửi yêu cầu tìm kiếm này tới các client khác
* Nếu tìm thấy, server sẽ gửi trả client yêu cầu tìm kiếm danh sách các client có thể tải file cần tìm kiếm, client sẽ chọn một chient trong số đó để yêu cầu tải file
* Nếu không tìm thấy, server sẽ thông báo cho client, client sẽ hiện thông báo cho người dùng

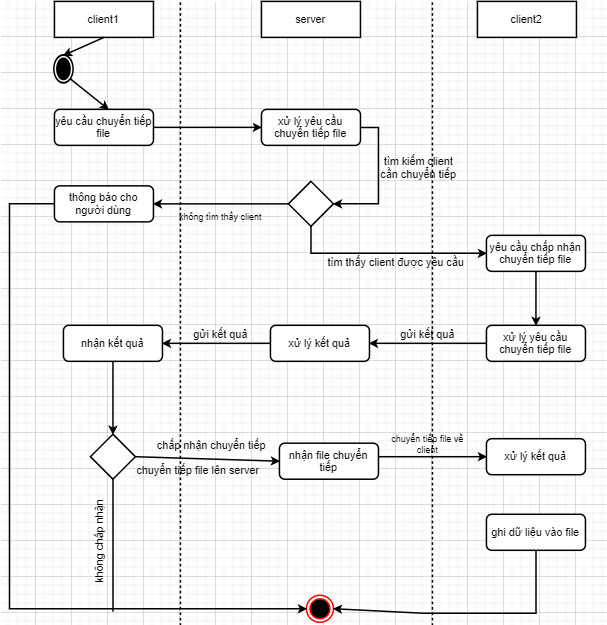
1. **Biểu đồ hoạt động**

## **Kết nối tới server**



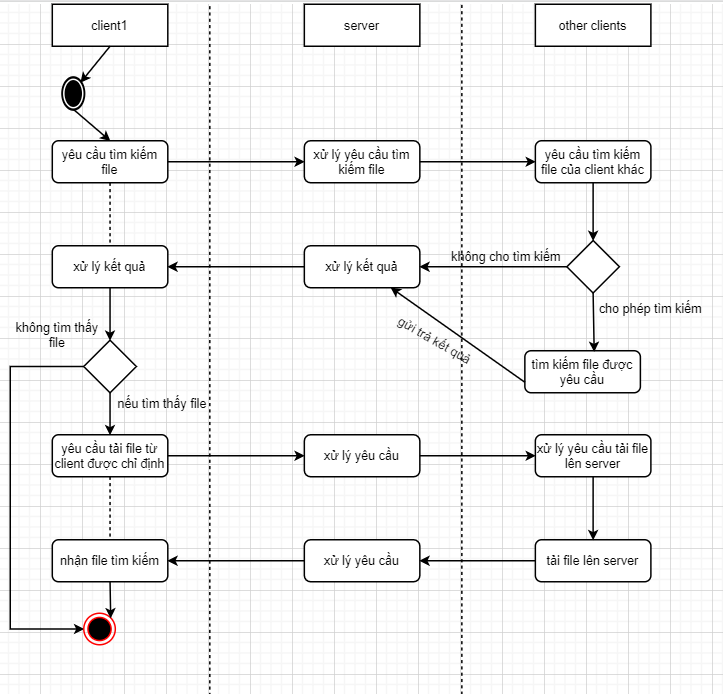
Hình 5: kết nối đến server

* 1. **Chuyển tiếp file**



Hình 6: chuyển tiếp file

* 1. **Tìm kiếm file**

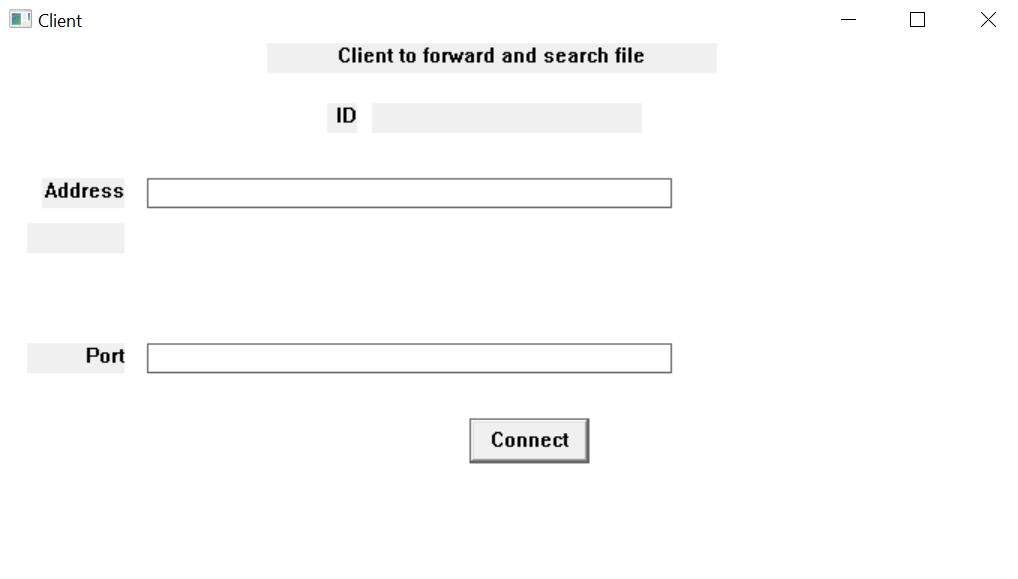


Hình 7: tìm kiếm file

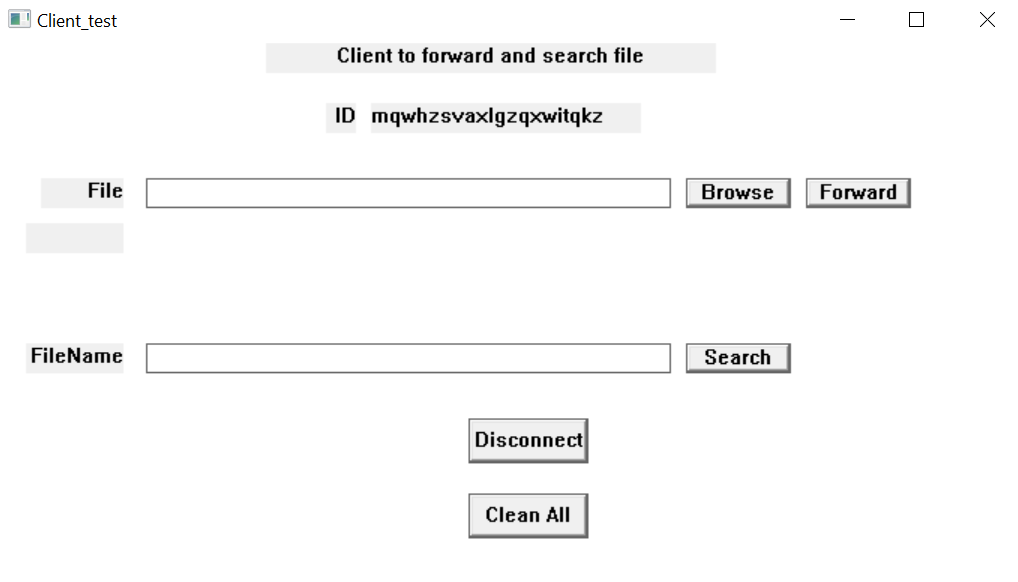
1. **Giao diện chương trình**
   1. **SERVER**

Server là console, không có giao diện

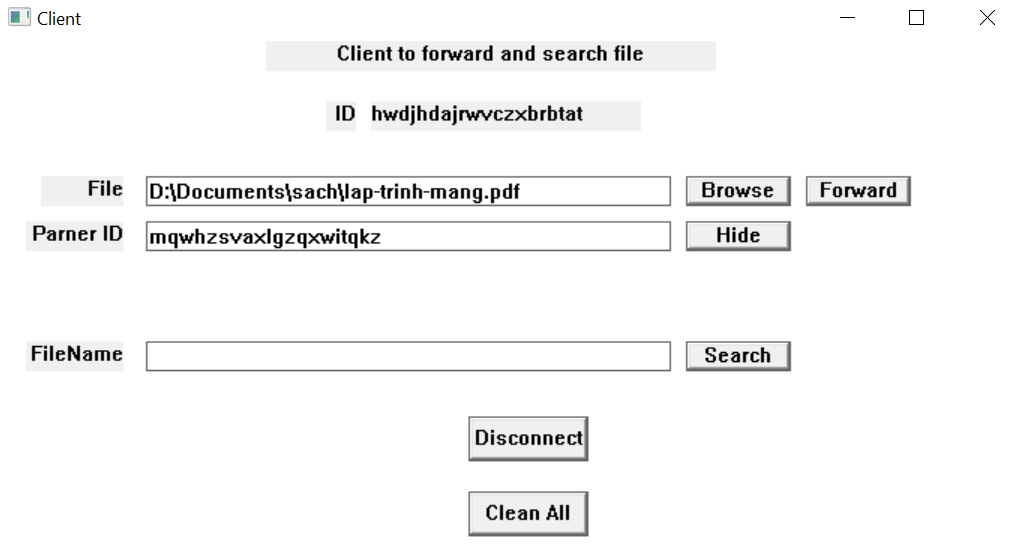
* 1. **CLIENT**



Hình 8: giao diện khởi tạo



Hình 9: giao diện sau khi đăng nhập thành công



Hình 10: giao diện chuyển tiếp file



Hình 11: giao diện danh sách ID client có file tìm kiếm

# **Tài Liệu Tham Khảo**

* <https://docs.microsoft.com/en-us/windows/win32/controls/user-controls-intro>
* <https://stackoverflow.com/questions/7598067/how-to-create-a-windows-style-textbox-in-a-c-win32-application>
* <https://stackoverflow.com/questions/8520560/get-a-file-name-from-a-path>
* <https://stackoverflow.com/questions/2886609/how-to-make-multiple-windows-using-win32-api#:~:text=5%20Answers&text=You%20can%20hit%20CreateWindow(),1st%20one%20if%20you%20want>.
* <https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-messagebox>
* <https://stackoverflow.com/questions/42046666/test-project-exe-has-triggered-a-breakpoint>
* <https://stackoverflow.com/questions/42438135/c-winapi-listbox-getting-selected-item-using-lb-getsel-lb-getcursel>
* <https://docs.microsoft.com/en-us/windows/win32/learnwin32/closing-the-window>
* <https://docs.microsoft.com/en-us/windows/win32/winsock/creating-a-basic-winsock-application>
* <https://users.soict.hust.edu.vn/tungbt/it4060/>