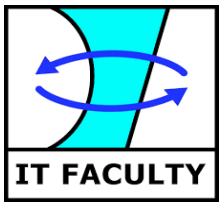
ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

-----📖-----



BÁO CÁO

DỰ ÁN HỆ ĐIỀU HÀNH & MẠNG MÁY TÍNH

Đề tài 402: Xây dựng chương trình định đường không thích nghi

SVTH :	**Phạm Quốc Tú**	**102200077**	**20TCLC_DT1**
:	**Phạm Chính Hiệu**	**102200088**	**20TCLC_DT2**
GVHD :	**Ths.Nguyễn Văn Nguyên**		

Đà Nẵng, 12/2022

MỤC LỤC

DANH SÁCH HÌNH ẢNH

LỜI MỞ ĐẦU

Các dịch vụ mạng ngày càng phát triển, mở rộng và hoàn thiện, tuy vẫn tồn tại nhiều khuyết điểm song không ít tiện lợi từ công nghệ đem lại cho xã hội loài người sự nhanh chóng và chính xác. Vì vậy, ứng dụng các kĩ thuật trong lập trình để tạo ra các sản phẩm cho người sử dụng là một việc cần thiết. Ngôn ngữ lập trình Java là một phần không thể thiếu trong việc xây dựng nên một thế giới công nghệ linh hoạt và mạnh mẽ. Hỗ trợ cho lập trình viên phát triển các ứng dụng mạng với kích thước nhẹ và mạnh mẽ trong xử lý. Với việc phát triển nhanh chóng của mạng, nhu cầu về truyền tải dữ liệu tốc độ cao càng lớn, việc tìm kiếm con đường để dữ liệu có thể đến được đích nhanh nhất cũng là vấn đề cần phải giải quyết. Chính vì thế, nhóm em sẽ dùng Java để hỗ trợ cho việc phát triển đề tài: **"Xây dựng chương trình định đường không thích nghi" (***Shortest Path Routing).*

Trong quá trình làm không tránh khỏi những thiếu sót, em mong nhận được lời nhận xét và giúp đỡ của các thầy cô giáo để chúng em hoàn thiện hơn. Đồng thời em xin được gửi lời cảm ơn chân thành đến Thầy Nguyễn Văn Nguyên đã nhiệt tình giúp đỡ em hoàn thành đồ án này.

Em xin chân thành cảm ơn!

1. Tóm tắt đề tài

Trong phần này, nhóm em sẽ xây dựng chương trình Client/ Server sử dụng giao thức TCP để Server giải quyết bài toán tìm đường đi được gửi lên từ Client với tập hợp các đỉnh và ma trận kề giữa chúng sau đó xử lí và trả kết quả về cho Client.

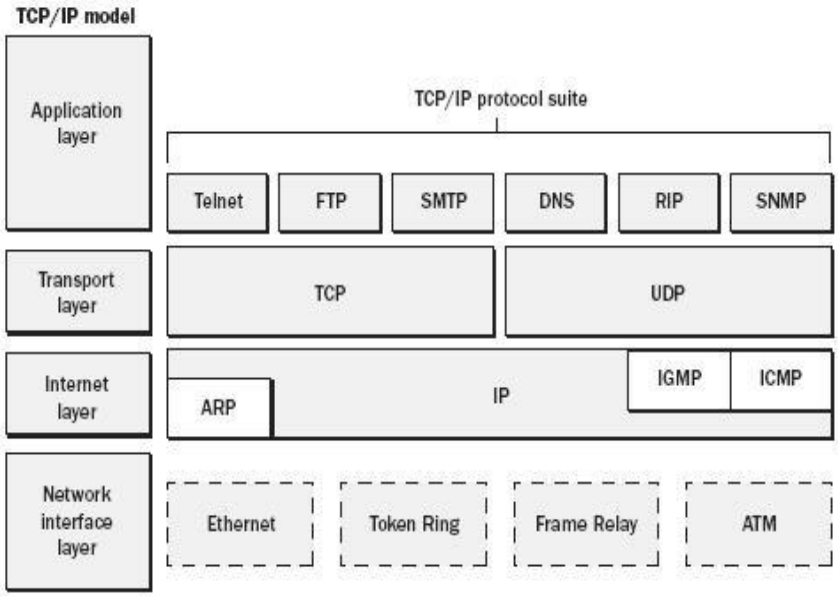
2. Cơ sở lý thuyết

2.1 Giao thức TCP/IP

TCP/IP là tên chung cho một tập hợp hơn 100 giao thức được sử dụng để kết nối các máy tính vào mạng, trong đó hai giao thức chính là TCP (Transmission Control Protocol) và IP (Internet Protocol).

Trong phạm vi Internet, thông tin không được truyền tải như một dòng riêng biệt từ máy tính này tới máy tính khác. Thay vào đó, dữ liệu được chia thành những gói nhỏ gọi là packet.

Các packet này được gửi trên mạng máy tính. Công việc của IP là chuyển chúng đến các máy tính ở xa. Tại trạm cuối, TCP nhận các packet và kiểm tra lỗi. Nếu một lỗi xuất hiện, TCP yêu cầu gói riêng biệt đó phải được gửi lại. Chỉ khi tất cả các packet đã nhận được là đúng, TCP sẽ sử dụng số thứ tự để tạo lại thông tin ban đầu.



Hình 1: Mô hình TCP/IP

2.1.1. Giao thức TCP

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.

Trong bộ giao thức TCP/IP, TCP là tầng trung gian giữa giao thức IP bên dưới và một ứng dụng bên trên. Các ứng dụng thường cần các kết nối đáng tin cậy kiểu đường ống để liên lạc với nhau, trong khi đó, giao thức IP không cung cấp những dòng kiểu đó, mà chỉ cung cấp dịch vụ chuyển gói tin không đáng tin cậy. TCP làm nhiệm vụ của tầng giao vận trong mô hình OSI đơn giản của các mạng máy tính.

Các ứng dụng gửi các dòng gồm các byte 8-bit tới TCP để chuyển qua mạng. TCP phân chia dòng byte này thành các đoạn (segment) có kích thước thích hợp (thường được quyết định dựa theo kích thước của đơn vị truyền dẫn tối đa (MTU) của tầng liên kết dữ liệu của mạng mà máy tính đang nằm trong đó). Sau đó, TCP chuyển các gói tin thu được tới giao thức IP để gửi nó qua một liên mạng tới mô đun TCP tại máy tính đích. TCP kiểm tra để đảm bảo không có gói tin nào bị thất lạc bằng cách gán cho mỗi gói tin một "số thứ tự" (sequence number). Số thứ tự này còn được sử dụng để đảm bảo dữ liệu được trao cho ứng dụng đích theo đúng thứ tự. Mô đun TCP tại đầu kia gửi lại "tin báo nhận" (acknowledgement) cho các gói tin đã nhận được thành công; một "đồng hồ" (timer) tại nơi gửi sẽ báo time-out nếu không nhận được tin báo nhận trong khoảng thời gian bằng một round-trip time (RTT), và dữ liệu (được coi là bị thất lạc) sẽ được gửi lại. TCP sử dụng checksum (giá trị kiểm tra) để xem có byte nào bị hỏng trong quá trình truyền hay không; giá trị này được tính toán cho mỗi khối dữ liệu tại nơi gửi trước khi nó được gửi, và được kiểm tra tại nơi nhận.

+	Bít 0 - 3	4 - 9	10 - 15	16 - 31
0	Source Port			Destination Port
32	Sequence Number			
64	Acknowledgement Number			
96	Data Offset	Reserved	Flags	Window
128	Checksum			Urgent Pointer
160	Options (optional)			
160/192+	Data			

Hình 2: Cấu trúc header của TCP

2.1.2. Giao thức IP

Là một giao thức hướng dữ liệu được sử dụng bởi các máy chủ nguồn và đích để truyền dữ liệu trong một liên mạng chuyển mạch gói.

Dữ liệu trong một liên mạng IP được gửi theo các khối được gọi là các gói (packet hoặc datagram). Cụ thể, IP không cần thiết lập các đường truyền trước khi một máy chủ gửi các gói tin cho một máy khác mà trước đó nó chưa từng liên lạc với.

Giao thức IP cung cấp một dịch vụ gửi dữ liệu không đảm bảo (còn gọi là cố gắng cao nhất), nghĩa là nó hầu như không đảm bảo gì về gói dữ liệu. Gói dữ liệu có thể đến nơi mà không còn nguyên vẹn, nó có thể đến không theo thứ tự (so với các gói khác được gửi giữa hai máy nguồn và đích đó), nó có thể bị trùng lặp hoặc bị mất hoàn toàn. Nếu một phần mềm ứng dụng cần được bảo đảm, nó có thể được cung cấp từ nơi khác, thường từ các giao thức giao vận nằm phía trên IP.

Giao thức IP rất thông dụng trong mạng Internet công cộng ngày nay. Giao thức tầng mạng thông dụng nhất ngày nay là IPv4; đây là giao thức IP phiên bản 4. IPv6 được đề nghị sẽ kế tiếp IPv4: Internet đang hết dần địa chỉ IPv4, do IPv4 sử dụng 32 bit để đánh địa chỉ (tạo được khoảng 4 tỷ địa chỉ); IPv6 dùng địa chỉ 128 bit, cung cấp tối đa khoảng 3.4×1038 địa chỉ . Các phiên bản từ 0 đến 3 hoặc bị hạn chế, hoặc không được sử dụng. Phiên bản 5 được dùng làm giao thức dòng (stream) thử nghiệm. Còn có các phiên bản khác, nhưng chúng thường dành là các giao thức thử nghiệm và không được sử dụng rộng rãi.

Địa chỉ IP được chia thành 4 số giới hạn từ 0 - 255. Mỗi số được lưu bởi 1 byte - > IP có kích thước là 4byte, được chia thành các lớp địa chỉ. Có 3 lớp là A, B, và C. Nếu ở lớp A, ta sẽ có thể có 16 triệu địa chỉ, ở lớp B có 65536 địa chỉ. Ví dụ: Ở lớp B chúng ta có tất cả các địa chỉ từ 132.25.0.0 đến 132.25.255.255. Phần lớn các địa chỉ ở lớp A là sở hữu của các công ty hay của tổ chức. Một ISP thường sở hữu một vài địa chỉ lớp B hoặc C. Ví dụ: Nếu địa chỉ IP của bạn là 132.25.23.24 thì bạn có thể xác định ISP của bạn là ai. (có IP là 132.25.x.x)

+	Bit 0 - 3	4 - 7	8 - 9	10 - 15	16 - 31
0	Source address				
32	Destination address				
64	Zeros		Protocol		TCP length
96	Source Port				Destination Port
128	Sequence Number				
160	Acknowledgement Number				
192	Data Offset	Reserved		Flags	Window
225	Checksum				Urgent Pointer
257	Options (optional)				
257/289+	Data				

Hình 3: Cấu trúc Header của gói tin IP

2.2. Mô hình Client-Server

Mô hình được phổ biến nhất và được chấp nhận rộng rãi trong các hệ thống phân tán là mô hình client/server. Trong mô hình này sẽ có một tập các tiến trình mà mỗi tiến trình đóng vai trò như là một trình quản lý tài nguyên cho một tập hợp các tài nguyên cho trước và một tập hợp các tiến trình client trong đó mỗi tiến trình thực hiện một tác vụ nào đó cần truy xuất tới tài nguyên phần cứng hoặc phần mềm dùng chung. Bản thân các trình quản lý tài nguyên cần phải truy xuất tới các tài nguyên dùng chung được quản lý bởi một tiến trình khác, vì vậy một số tiến trình vừa là tiến trình client vừa là tiến trình server. Các tiến trình phát ra các yêu cầu tới các server bất kỳ khi nào chúng cần truy xuất tới một trong các tài nguyên của các server. Nếu yêu cầu là đúng đắn thì server sẽ thực hiện hành động được yêu cầu và gửi một đáp ứng trả lời tới tiến trình client.

Mô hình client/server cung cấp một cách tiếp cận tổng quát để chia sẻ tài nguyên trong các hệ thống phân tán. Mô hình này có thể được cài đặt bằng rất nhiều môi trường phần

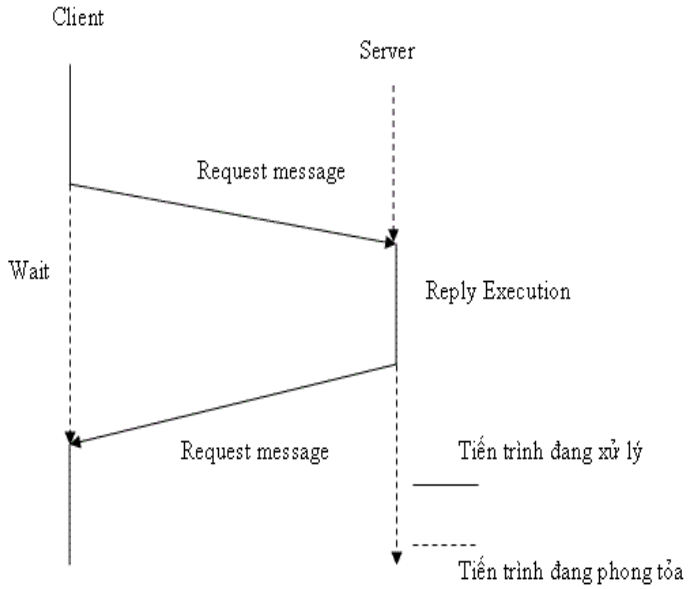
cứng và phần mềm khác nhau. Các máy tính được sử dụng để chạy các tiến trình client/server có nhiều kiểu khác nhau và không cần thiết phải phân biệt giữa chúng.

Mô hình truyền tin client/server hướng tới việc cung cấp dịch vụ. Quá trình trao đổi dữ liệu bao gồm:

1. Truyền một yêu cầu từ tiến trình client tới tiến trình server
2. Yêu cầu được server xử lý
3. Truyền đáp ứng cho client

Mô hình truyền tin này liên quan đến việc truyền hai thông điệp và một dạng đồng bộ hóa cụ thể giữa client và server. Tiến trình server phải nhận thức được thông điệp được yêu cầu ở bước một ngay khi nó đến và hành động phát ra yêu cầu trong client phải được tạm dừng (bị phong tỏa) và buộc tiến trình client ở trạng thái chờ cho tớ khi nó nhận được đáp ứng do server gửi về ở bước ba.

Mô hình client/server thường được cài đặt dựa trên các thao tác cơ bản là gửi (send) và nhận(receive).



Hình 4: Mô hình client/server

2.3. Socket trong Java

2.3.1. Khái quát về socket

Như chúng ta đã biết kết nối URLs và URL cung cấp cho chúng ta một cơ cấu để truy xuất vào các tài nguyên trên Internet ở một mức tương đối cao, nhưng đôi khi chương trình của chúng ta lại yêu cầu một giao tiếp ở tầng mạng mức thấp.Ví dụ khi chúng ta viết một ứng dụng client-server.

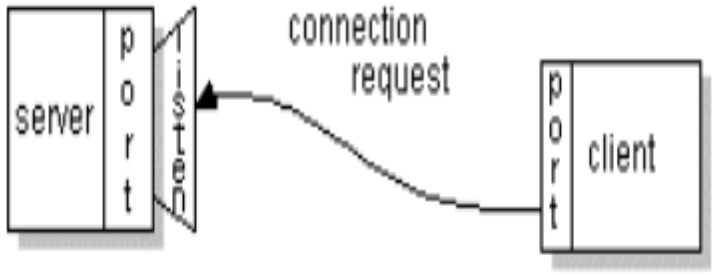
Trong một ứng dụng client-server thì phía server sẽ cung cấp một số dịch vụ, như: xử lí cơ sở dữ liệu, các yêu cầu bên phía client đưa ra, sau đó sẽ gửi lại cho phía client. Sự giao tiếp như vậy gọi là tin cậy bởi vì dữ liệu sẽ không bị mất mát, sai lệch trong quá trình truyền, server gửi cho client thông điệp gì thì phía client sẽ nhận được thông điệp nguyên như vậy. Giao thức TCP sẽ cung cấp cho chúng ta một cách thức truyền tin cậy. Để có thể nói chuyện được trên TCP thì chương trình client và chương trình server phải thiếp lập một đường truyền, và mỗi chương trình sẽ phải kết nối lại với socket là điểm cuối để kết nối, client và server muốn nói chuyện với nhau thì sẽ phải thông qua socket, mọi thông điệp sẽ phải đi qua socket. Chúng ta cứ mường tượng socket ở đây là một cái cửa mọi người muốn đi ra hay đi vào đều phải thông qua cái cửa này.

2.3.2. Cơ chế socket

Một socket là một điểm cuối của thông tin hai chiều liên kết giữa hai chương trình đang chạy trên mạng. Những lớp socket được dùng để đại diện cho kết nối giữa một chương trình client và một chương trình server. Trong Java gói Java.net cung cấp hai lớp Socket và ServerSocket để thực hiện kết nối giữa client và server.

Thông thường thì server sẽ chạy trên một máy đặc biệt và có một socket giới hạn trong 1 Portnumber đặc biệt.

Phía client: client được biết hostname của máy mà server đang chạy và port number mà server đang lắng nghe. Để tạo một yêu cầu kết nối client sẽ thử hẹn gặp server ở trên máy của server thông qua port number. Client cũng cần xác định chính nó với server thông qua local port number.



Hình 5: Client gửi yêu cầu kết nối tới Server

Nếu mọi thứ tốt đẹp thì server sẽ đồng ý kết nối. Khi đồng ý kết nối thì server sẽ tạo ra một socket mới để nói chuyện với client và cũng tạo ra một socket khác để tiếp tục lắng nghe.

Hình 6: Server đồng ý kết nối và tiếp tục lắng nghe.

2.3.3. Mô hình truyền tin socket

Có thể phân thành 4 giai đoạn như sau:

- Giai đoạn 1: Server tạo Socket, gán số hiệu cổng và lắng nghe yêu cầu nối kết. Server sẵn sàng phục vụ Client.socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
    - bind(): Server yêu cầu gán số hiệu cổng (port) cho socket.
    - listen(): Server lắng nghe các yêu cầu nối kết từ các client trên cổng đã được gán.

❖ Giai đoạn 2: Client tạo Socket, yêu cầu thiết lập một nối kết với Server.

- socket(): Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của Client.
- connect(): Client gởi yêu cầu nối kết đến server có địa chỉ IP và Port xác định.
- accept(): Server chấp nhận nối kết của client, khi đó một kênh giao tiếp ảo được hình thành, Client và server có thể trao đổi thông tin với nhau thông qua kênh ảo này.

❖ Giai đoạn 3: Trao đổi thông tin giữa Client và Server.

- Sau khi chấp nhận yêu cầu nối kết, thông thường server thực hiện lệnh read() và nghẽn cho đến khi có thông điệp yêu cầu (Request Message) từ client gởi đến.
- Server phân tích và thực thi yêu cầu. Kết quả sẽ được gởi về client bằng lệnh write().
- Sau khi gởi yêu cầu bằng lệnh write(), client chờ nhận thông điệp kết quả (ReplyMessage) từ server bằng lệnh read().

❖ Giai đoạn 4: Kết thúc phiên làm việc.

- Các câu lệnh read(), write() có thể được thực hiện nhiều lần (ký hiệu bằng hình ellipse).
- Kênh ảo sẽ bị xóa khi Server hoặc Client đóng socket bằng lệnh close().

3. Chương trình Client-Server

3.1. Phân tích bài toán

Bài toán đặt ra ở đây là người dùng (máy khách) gửi 1 list các đỉnh (router) và ma trận kề giữa các đỉnh (router) đó, máy chủ phải tiếp nhận và xử lí. Tức là máy chủ phải đọc dữ liệu và tìm ra đường đi tối ưu từ một đỉnh (router) đến các đỉnh (router) còn lại rồi trả kết quả về lại cho máy khách.

3.2. Thiết kế kiến trúc

Kiến trúc của hệ thống gồm có 3 phần:

• Máy chủ Server:

Máy chủ được tạo ra và quản lý bởi máy tính. Máy chủ sẽ tạo liên kết với các Client. Các tương tác của người dùng đều được báo cáo về máy chủ thông qua trình duyệt cmd trên máy tính. Máy chủ tạo ra 1 localhost. Các máy khác muốn kết nối với nhau thông qua máy chủ phải truy cập vào localhost này.

+ Lập trình cho máy chủ:

Bước 1: Tạo một đối tượng ServerSocket và mở cổng với port 9999

Bước 2: Tạo một đối tượng Socket bằng cách chấp nhận liên kết từ yêu cầu liên kết của client. Sau khi chấp nhận liên kết, phương thức accept () trả về đối tượng Socket thể hiện liên kết giữa Client và Server.

```
public void run() {
    try {
        serverSocket = new ServerSocket(port);
        System.out.println("server bat dau");
        while(!done)
        {
            Socket clientSocket = serverSocket.accept();
            System.out.println("CO 1 KET NOI MOI");
            ClientHandler client = new ClientHandler(clientSocket);
            clients.add(client);
            client.start();


        }

    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Hình 9: Kết nối đến Client

• Máy khách Client:

Người sử dụng sẽ tương tác trên máy khách để có thể thực hiện thao tác tìm đường đi.

+ Lập trình cho Client:

Ở client, ta cần xác định địa chỉ của server để truy cập. Sau khi định được địa chỉ, máy khách có thể truy cập máy chủ qua địa chỉ đấy.

```
public void connectToServer()
{
    try {
        socket = new Socket(InetAddress.getByName(host), serverPort);

        OutputStream outStream = socket.getOutputStream();
        objectOutputStream = new ObjectOutputStream(outStream);

        InputStream inputStream = socket.getInputStream();
        objectInputStream = new ObjectInputStream(inputStream);

    } catch (UnknownHostException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Hình 10: Kết nối đến Server

• Giao tiếp giữa máy chủ và máy khách:

+ Máy chủ và máy khách giao tiếp với nhau thông qua cơ chế Socket TCP.

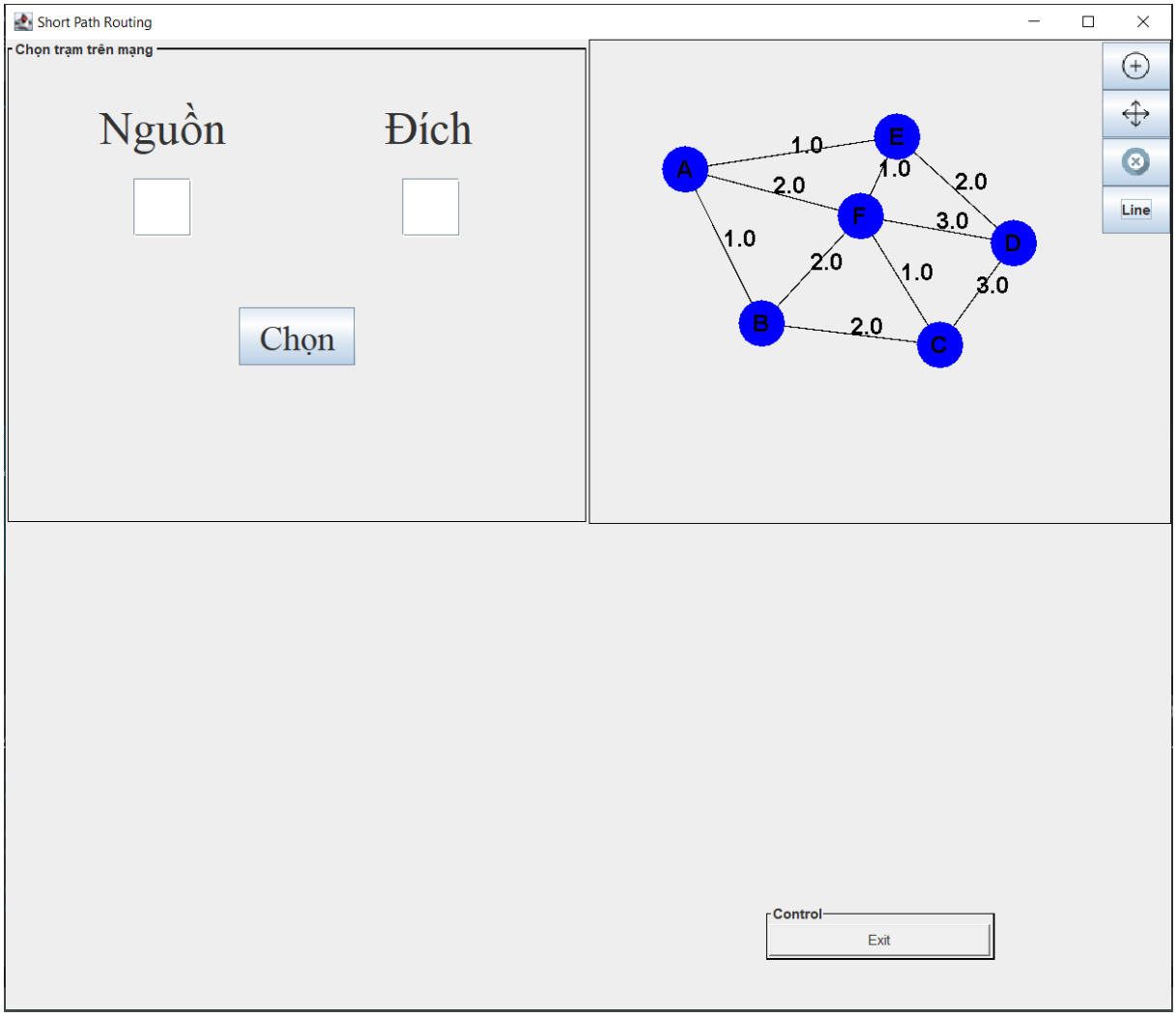
+ Nhóm em sử dụng thư viện Socket.io cùng với một số hàm đặc trưng trong thư viện này để thiết lập tương tác 2 chiều giữa máy chủ và máy khách. Với mỗi hành động như nhập đỉnh, nhập khoảng cách, xóa đỉnh… (Thực hiện trên máy khách) thì đều được máy chủ lắng nghe và thực hiện các lệnh để đáp những thông điệp đó.

3.3. Thiết kế ứng dụng

+ Thiết kế phần nhập dữ liệu

Nhập đồ thị:

Yêu cầu: Hệ thống yêu cầu người dùng nhập đỉnh bắt đầu và đỉnh kết thúc và vẽ ra các đỉnh của đồ thị và khoảng cách giữa các đỉnh đó. Nhập xong bấm nút Result. Nếu thành công, hiển thị ra kết quả bao gồm bảng chỉ dẫn khoảng cách ngắn nhất từ đỉnh đầu tới đỉnh cuối đã chọn ở trên và khoảng cách giữa chúng.



Hình 11: Giao diện nhập dữ liệu từ client

3.4. Giải thích thuật toán

3.4.1 Thuật toán Dijkstra

Thuật toán Dijkstra cho phép tìm đường đi ngắn nhất từ một đỉnh **S** đến các đỉnh còn lại của đồ thị và chiều dài (trọng số) tương ứng. Phương pháp của thuật toán là xác định tuần tự đỉnh có chiều dài đến **S** theo thứ tự tăng dần**.**

Thuật toán được xây dựng trên cơ sở gán cho mỗi đỉnh các nhãn tạm thời. Nhãn tạm thời của các đỉnh cho biết cận trên của chiều dài đường đi ngắn nhất từ **S** đến đỉnh đó. Nhãn của các đỉnh sẽ biến đổi trong các bước lặp, mà ở mỗi bước lặp sẽ có một nhãn tạm thời trở thành chính thức. Nếu nhãn của một đỉnh nào đó trở thành chính thức thì đó cũng chính là chiều dài ngắn nhất của đường đi từ **S** đến đỉnh đó.

Thuật toán của Dijkstra là một thuật toán Greedy (thuật toán tham lam). Điều này có nghĩa là chúng ta sẽ đi một con đường ngắn hơn từ đỉnh này đến đỉnh khác.

Thuật toán hoàn tất khi chúng ta truy cập tất cả các đỉnh của đồ thị. Tuy nhiên, đôi khi khi chúng ta tìm thấy một đỉnh mới, có thể có các đường đi ngắn hơn qua nó từ một đỉnh đã truy cập đến một đỉnh đã được truy cập khác.

3.4.2 Các bước thực hiện:

+ Thêm vào tất cả các đỉnh của đồ thị.

Tạo ra class Vert chứa thông tin của mỗi đỉnh gồm các thuộc tính: visited (đánh dấu), name (tên đỉnh), List<Edge> (Edge là class chứa thông tin về Vert liền kề với nhau), dist (Trọng số), pr (đỉnh kề trước nó).

```
for (int i = 0; i < numberOfVert ; i++){
        Vert vert = new Vert(nodes.get(i).getName());
        verts.add(vert);
        if (vert.getName().equals(startVertName))
              beginVert = vert;
}
```

+ Thêm vào trọng số của từng đỉnh đến các đỉnh còn lại (nếu có).

Từ hình vẽ trên ứng dụng ta lấy ra các đường (Line) trong đồ thị

Dùng phương thức addNeighbour để truyền Edge vào cho các Vert.

```
for (Line line : lines){
        String startVertName = line.getStartNode().getName();
        String endVertName = line.getEndNode().getName();
        Vert startVert = findVertWithName(startVertName);
        Vert endVert = findVertWithName(endVertName);
        //ADD NEIGHTBOUR CHO CAC VERT
        startVert.addNeighbour(new Edge(line.getWeight(), startVert, endVert));
        endVert.addNeighbour(new Edge(line.getWeight(), endVert, startVert));
}
```

+ Chọn đỉnh bắt đầu và kết thúc.

Dùng phương thức ShortestP để gán đỉnh bắt đầu vào.

```
PathFinder shortestPath = new PathFinder();
shortestPath.ShortestP(beginVert);
```

+ Tạo 1 PriorityQuery để đánh dấu các đỉnh đã được đi qua.

Ở đây, chúng ta đã tạo ra một Priority Queue không có đối số. Trong trường hợp này, phần tử đầu của Priority Queue là phần tử nhỏ nhất của queue. Và các phần tử được loại bỏ khỏi queue theo thứ tự tăng dần.

```
PriorityQueue<Vert> priorityQueue = new PriorityQueue<>();
```

+ Tìm khoảng cách lần lượt từ các đỉnh trong PriorityQuery tới đỉnh kề với nó và đánh dấu đỉnh nó vừa đi qua. Và duyệt cho đến khi hết các phần tử trong Priority Query thì dừng lại.

Lấy từng đỉnh (phần tử) trong Priority Query ra, tính khoảng cách từ đỉnh đó đến các đỉnh kề với nó có nhỏ hơn trọng số mà nó ghi lại khi xét với đỉnh trước đó

hay không(nếu đỉnh đầu tiên thì trọng số là +∞). Nếu nhỏ hơn thì thiết lập lại trọng số cho đỉnh đó bằng phương thức setDist() và thiết lập đỉnh kề trước nó bằng phương thức setPr(), còn nếu khoảng cách nó lớn hơn khoảng cách hiện tại từ đỉnh khác đến đỉnh đó thì bỏ qua. Lặp vòng for cho đến khi hết đỉnh kề cho nó thì chuyển qua đỉnh (phần tử) tiếp theo trong Priority Query cho đến hết.

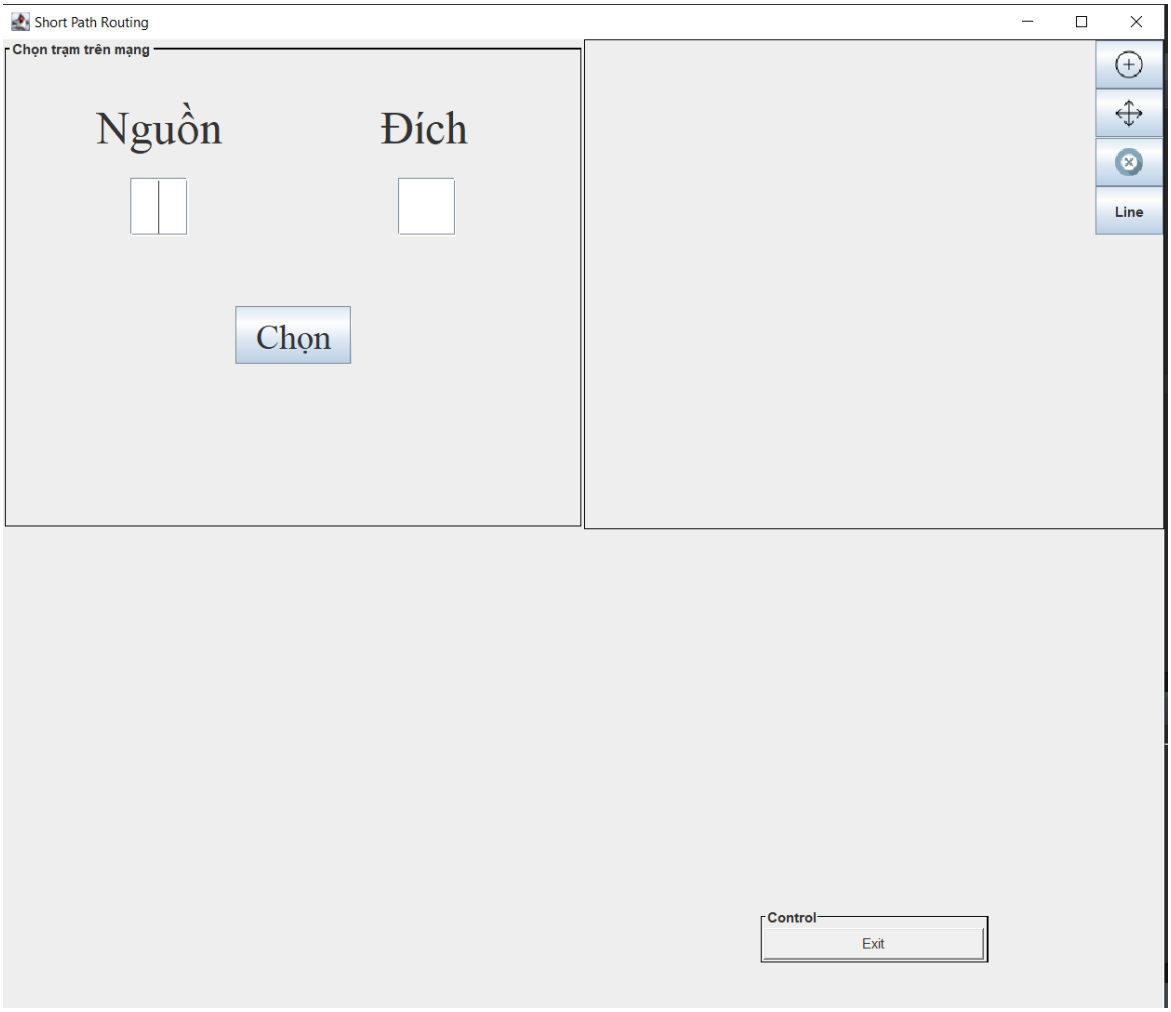
```
while (!priorityQueue.isEmpty())
{
	Vert actualVertex = priorityQueue.poll();
	for (Edge edge : actualVertex.getList())
	{
		Vert v = edge.getTargetVert();
		if (!v.Visited())
		{
			double newDistance = actualVertex.getDist() +
			edge.getWeight();
			if (newDistance < v.getDist())
			{
			    priorityQueue.remove(v);
			    v.setDist(newDistance);
			    v.setPr(actualVertex);
			    priorityQueue.add(v);
			}
		}
	}
	actualVertex.setVisited(true);
}
```

+ Lúc này đã đã có được khoảng cách từ đỉnh xuất phát tới tất cả các đỉnh trong đồ thị. Ta sẽ tìm được khoảng cách ngắn nhất từ đỉnh xuất phát tới đỉnh kết thúc mà ta đã lựa chọn.

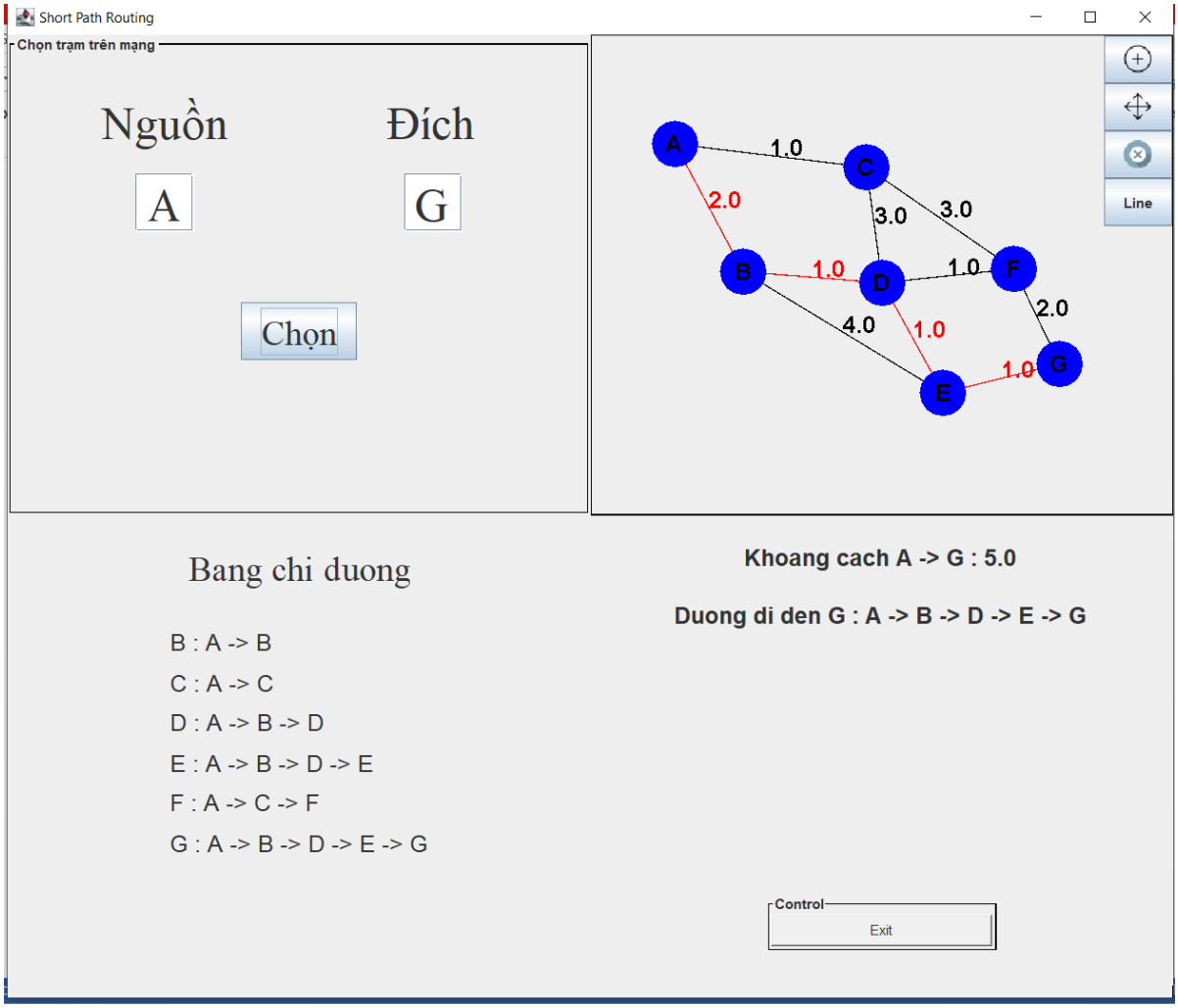
4. Triển khai và đánh giá kết quả

4.1. Triển khai

4.1.1. Giao diện phía Client



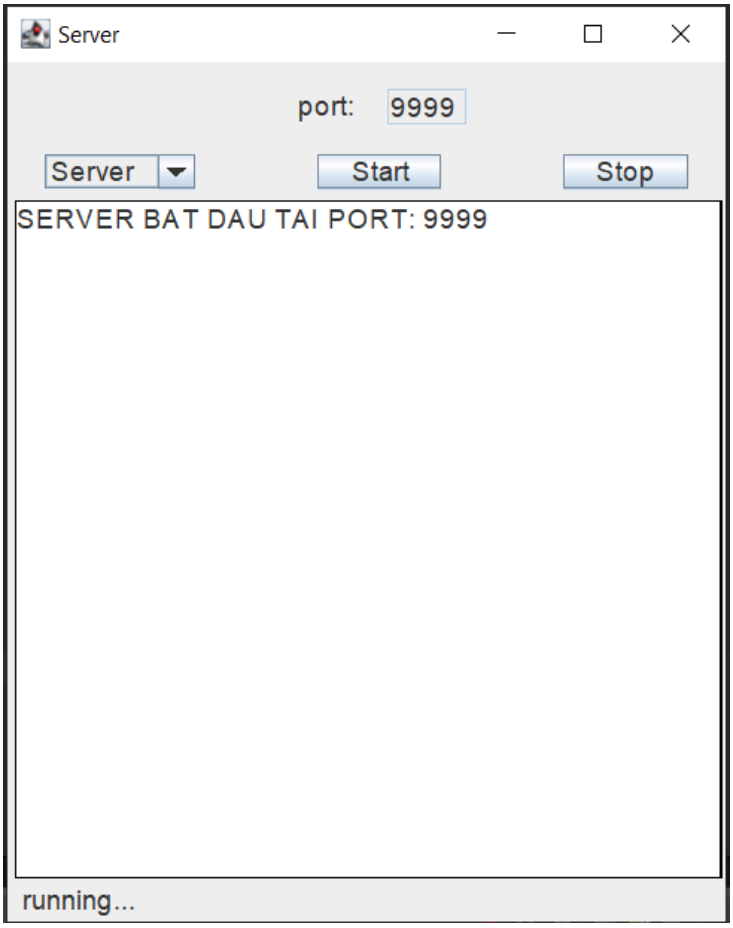
Hình 12: Giao diện chính của client



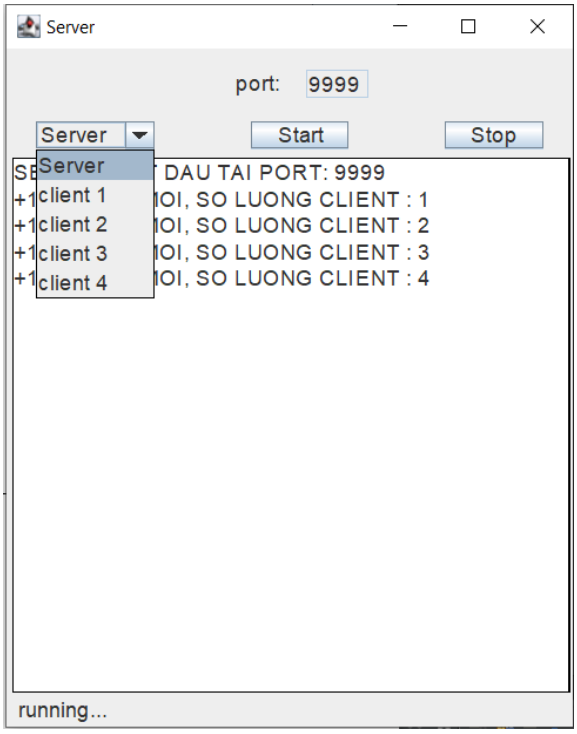
Hình 13: Kết quả tìm đường đi

4.1.2. Giao diện phía Server:

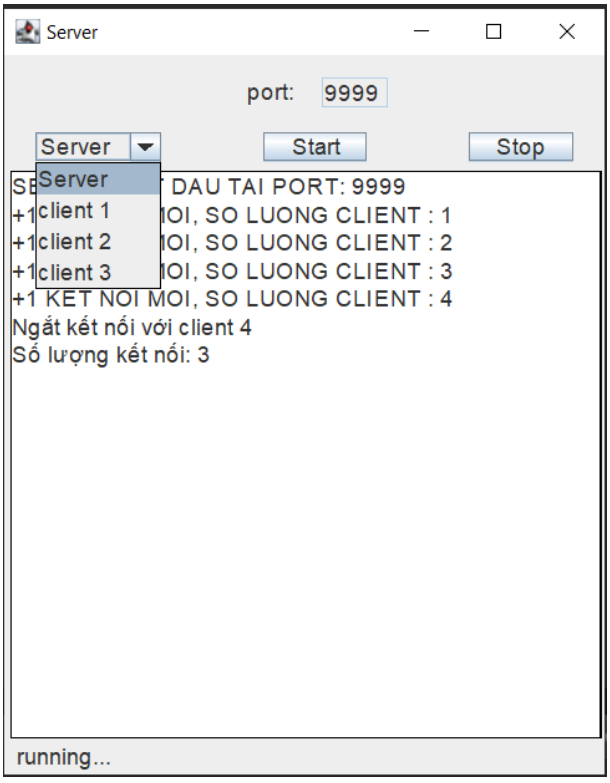
Sau khi khởi chạy, server sẽ tự động chạy tại port mặc định là 9999, lắng nghe kết nối từ người dùng.



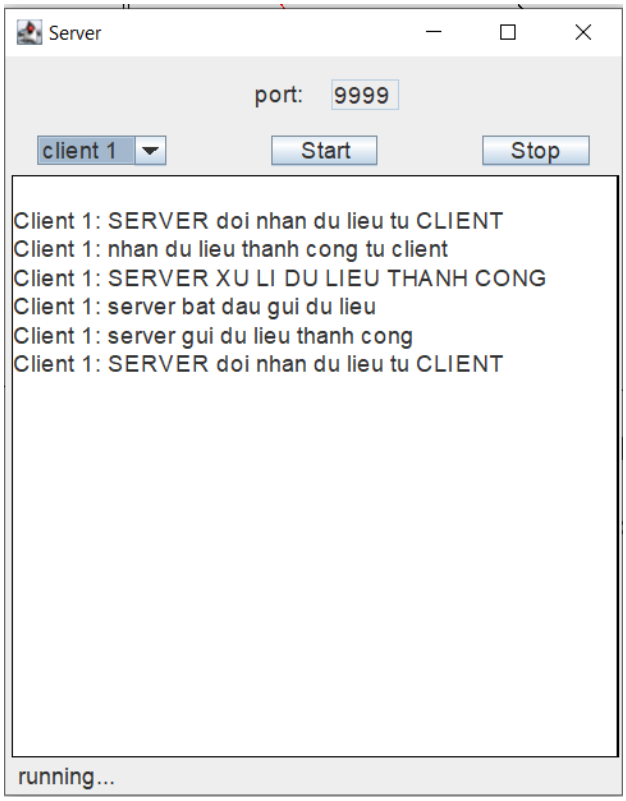
Hình 14: Giao diện server lúc bắt đầu



Hình 15: Server sau khi thiết lập kết nối với client



Hình 16: Server ngắt kết nối với client



Hình 17: Server hiển thị quá trình xử lí dữ liệu nhận được từ client

4.2. Đánh giá

4.2.1. Đánh giá thiết kế kiến trúc

Xây dựng kiến trúc đơn giản dựa trên mô hình Client- Server với 3 phần:

+ Server, Client, Giao tiếp giữa máy chủ và máy khách.

Mô hình client server hỗ trợ, giúp chúng ta có thể làm việc trên bất kì một máy tính nào có hỗ trợ giao thức truyền thông. Client server đảm bảo được sự toàn vẹn dữ liệu khi có sự cố xảy ra.

Dễ dàng mở rộng, xây dựng hệ thống mạng.

Chỉ cần chung định dạng giao tiếp mà không cần chung nền tảng là có thể hoạt động được.

Có thể có nhiều server cùng làm một dịch vụ, chúng có thể nằm trên nhiều máy tính hoặc một máy tính.

4.2.2. Đánh giá thiết kế giao diện

Giao diện tương đối thân thiện với người dùng, các phần chức năng được tách biệt với nhau để tránh người dùng nhầm lẫn. Tuy nhiên, kiến thức lập trình còn hạn chế nên giao diện còn khá đơn giản.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

• Kết luận:

- Qua đồ án này, em đã tìm hiểu được rất nhiều kiến thức, thực hiện hóa được các kiến thức liên quan đến môn học Nguyên lý hệ điều hành và Lập trình mạng.

- Tạo điều kiện cho em rèn luyện các kỹ năng:

- Tổng hợp các kiến thức đã học trên mọi phương diện,

- Tính tự chủ và tinh thần trách nhiệm trong công việc,

- Khả năng thiết kế và lập trình

- Khả năng báo cáo bằng luận văn

- Khả năng thuyết trình và bảo vệ đồ án

• Hướng phát triển:

- Giao diện đơn giản với người dùng hơn.

- Tích hợp các tính năng mới.

TÀI LIỆU THAM KHẢO

1. https://vi.wikipedia.org/wiki/K%C3%ADph%C3%A1pBaLan
2. Giáo trình Lập trình mạng của Thầy Mai Văn Hà
3. Giáo trình Nguyên lý Hệ Điều Hành của Cô Trần Hồ Thủy Tiên.
4. https://vi.wikipedia.org/wiki/TCP/IP
5. https://www.youtube.com/watch?v=qKMRBMhJpXM
6. https://cafedev.vn/tu-hoc-java-priorityqueue-trong-java/
7. https://gpcoder.com/3679-xay-dung-ung-dung-client-server-voi-socket-trong-java/
8. Các tài liệu khác trên mạng Internet.

PHỤ LỤC

Server:

```
public class Server extends JFrame implements Runnable, ActionListener
{
    ServerSocket serverSocket;
    ArrayList<ClientHandler> clients;
    private int port = 9999;
    public boolean done;
    Thread serverThread;
    /*
     * Setup GUI
     */
    private int screenHeight = 500;
    private int screenWidth = 400;
    private ArrayList<JTextArea> listTextArea;
    private JComboBox cbbClients;
    private JButton btnStart, btnStop;
    private JLabel lblPort, lblAppStatus;
    private JTextField txtPort;
    private JLayeredPane pnlMain;
    private JTextArea txtServerArea;
    Font Arial_Plain_15 = new Font("Arial", Font.PLAIN, 15);
    public Server()
    {
        clients = new ArrayList<>();
        done = false;
        serverThread = new Thread(this);
        this.listTextArea = new ArrayList<JTextArea>();
```

```
            GUI();
    }
    private void GUI() {
            int x, y, w, h;
            pnlMain = new JLayeredPane();
            this.btnStart = new JButton("Start");
            this.btnStart.setFont(Arial_Plain_15);
            this.btnStart.addActionListener(this);
            w = (int) (getBoundOfText(btnStart.getText(), Arial_Plain_15).getWidth()*2.1);
            h = (int) getBoundOfText(btnStart.getText(), Arial_Plain_15).getHeight() + 2;
            x = (screenWidth - w)/2;
            y = (int) (0.1*screenHeight);
            this.btnStart.setBounds(x, y, w, h);
            this.btnStop = new JButton("Stop");
            this.btnStop.addActionListener(this);
            this.btnStop.setFont(Arial_Plain_15);
            w = (int) (getBoundOfText(btnStop.getText(), Arial_Plain_15).getWidth()*2.2);
            h = (int) getBoundOfText(btnStop.getText(), Arial_Plain_15).getHeight() + 2;
            x =  screenWidth/3*2 + (screenWidth/3 - w)/2;
            y = (int) (0.1*screenHeight);
            this.btnStop.setBounds(x, y, w, h);
            this.cbbClients = new JComboBox();
            this.cbbClients.addActionListener(this);
            this.cbbClients.addItem("Server");
            this.cbbClients.setFont(Arial_Plain_15);
            w = (int) (getBoundOfText("client xx", Arial_Plain_15).getWidth()*1.5);
            h = (int) getBoundOfText("client xx", Arial_Plain_15).getHeight() + 2;
            x = (int) (0.05*screenWidth);
```

```
y = (int) (0.1*screenHeight);
this.cbbClients.setBounds(x, y, w, h);
//          Using a template JTEXTAREA
this.txtServerArea = new JTextArea();
settingJTextArea(txtServerArea,0);
txtServerArea.setVisible(true);
this.lblPort = new JLabel("port:");
this.lblPort.setFont(Arial_Plain_15);
w = (int) (getBoundOfText(lblPort.getText(), Arial_Plain_15).getWidth() + 2);
h = (int) getBoundOfText(lblPort.getText(), Arial_Plain_15).getHeight() + 2;
double percentWidth = w*1.0/screenWidth;
x =  (int) ((screenWidth/2*(1-percentWidth/2)-w) - 0.01*screenWidth);
y = (int) (0.03*screenHeight);
this.lblPort.setBounds(x, y, w, h);
this.txtPort = new JTextField();
this.txtPort.setFont(Arial_Plain_15);
this.txtPort.setEditable(false);
txtPort.setText(Integer.toString(port));
w = (int) (getBoundOfText("11111", Arial_Plain_15).getWidth() +2);
h = (int) getBoundOfText("11111", Arial_Plain_15).getHeight() + 2;
x =  (int) (screenWidth/2 + 0.01*screenWidth);
y = (int) (0.03*screenHeight);
this.txtPort.setBounds(x, y, w, h);
this.lblAppStatus = new JLabel("running...");
this.lblAppStatus.setFont(Arial_Plain_15);
w = (int) (getBoundOfText(lblAppStatus.getText(), Arial_Plain_15).getWidth()*2.2);
h = (int) getBoundOfText(lblAppStatus.getText(), Arial_Plain_15).getHeight() + 2;
x =  (int) (0.02 * screenWidth);
```

```
            y = (int) (0.885*screenHeight);
            this.lblAppStatus.setBounds(x, y, w, h);
            pnlMain.add(lblPort);
            pnlMain.add(txtPort);
            pnlMain.add(btnStart);
            pnlMain.add(btnStop);
            pnlMain.add(cbbClients);
            pnlMain.add(lblAppStatus);
            this.add(pnlMain);
            this.pack();
            this.setVisible(true);
            this.setSize(new Dimension(screenWidth, screenHeight));
            this.setTitle("Server");
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            this.setLocationRelativeTo(null);
    }
    private Rectangle2D getBoundOfText(String text, Font font)
    {
            AffineTransform affinetransform = new AffineTransform();
            FontRenderContext frc = new FontRenderContext(affinetransform,true,true);
            return font.getStringBounds(text, frc);
    }
    public void settingJTextArea(JTextArea jTextArea, int index)
    {
            if (listTextArea.size() == index)
                    this.listTextArea.add(jTextArea);
            else if (listTextArea.size() > index)
            {
```

```
                this.listTextArea.add(index, jTextArea);
            }
            jTextArea.setFont(Arial_Plain_15);
            int w = (int) (screenWidth*0.95) ;
            int h = (int) (screenHeight*0.73);
            int x = (int) (screenWidth*0.01);
            int y = (int) (screenHeight*0.15);
            jTextArea.setBounds(x, y, w, h);
            jTextArea.setEditable(false);
            jTextArea.setVisible(false);
            Border tempBorder = BorderFactory.createLineBorder(Color.black);
            jTextArea.setBorder(tempBorder);
            pnlMain.add(jTextArea);
        }

        @Override
        public void run()
        {
            done = true;
            try {
                serverSocket = new ServerSocket(port);
                done = false;
            } catch (IOException e1) {
                txtServerArea.setText("Port " + port + " đã được sử dụng, vui lòng chọn port
khác!");
                System.out.println("hehe");
                done = true;
                return;
```

```
            }
            try
            {
                String mess = "SERVER BAT DAU TAI PORT: " + port;
                String temp = txtServerArea.getText();
                txtServerArea.setText(mess);
                while(!done)
                {
                    Socket clientSocket = serverSocket.accept();
                    ClientHandler client = new ClientHandler(clientSocket);
                    clients.add(client);
                    client.start();
                    mess = "+1 KET NOI MOI, SO LUONG CLIENT : " + clients.size();
                    txtServerArea.setText(txtServerArea.getText() + "\n" + mess);
                }
            } catch (IOException e)
            {}
        }
        class ClientHandler extends Thread
        {
            Socket clientSocket;
            ObjectOutputStream objectOutputStream;
            ObjectInputStream objectInputStream;
            ArrayList<Node> nodes;
            ArrayList<Line> lines;
            String startVertName, endVertName;
            //OBJECT CHUA DU LIEU SE DUOC GUI DEN CLIENT SAU KHI XU LI
            ArrayList<String> distances;
```

```
ArrayList<String> paths;
ArrayList<Vert> verts;
boolean done;
int sttClient;
JTextArea clientStatus;
public ClientHandler(Socket clientSocket)
{
        this.clientSocket = clientSocket;
        verts = new ArrayList<>();
        distances = new ArrayList<String>();
        paths = new ArrayList<String>();
        nodes = new ArrayList<Node>();
        lines = new ArrayList<Line>();
        done = false;
        clientStatus = new JTextArea();
}

@Override
public void run()
{
        try {
                OutputStream outStream = clientSocket.getOutputStream();
                objectOutputStream = new ObjectOutputStream(outStream);
                InputStream inputStream = clientSocket.getInputStream();
                objectInputStream = new ObjectInputStream(inputStream);


                if (cbbClients.getItemCount() > 1)
```

```
{
        boolean isAdded = false;
        for (int index = 1; index < cbbClients.getItemCount(); index++)
        {
                if
(cbbClients.getItemAt(index).toString().endsWith(Integer.toString(index)))
                {
                        continue;
                }
                sttClient = index;
                String temp = "client " + sttClient;
                cbbClients.insertItemAt(temp, sttClient);
                isAdded = true;
                break;
        }

        if (!isAdded)
        {
                sttClient = clients.indexOf(this) + 1;
                String temp = "client " + sttClient;
                cbbClients.addItem(temp);
        }
}
else
{
        sttClient = clients.indexOf(this) + 1;
        String temp = "client " + sttClient;
        cbbClients.addItem(temp);
```

```
                }

                settingJTextArea(clientStatus,sttClient);

                while (!done)
                {
                        String status;
                        status = "Client " + sttClient + ": " + "SERVER doi nhan du lieu
tu CLIENT";
                        clientStatus.setText(clientStatus.getText() + "\n" + status);
                        receiveData();
                        status = "Client " + sttClient + ": " + "nhan du lieu thanh cong tu
client";
                        clientStatus.setText(clientStatus.getText() + "\n" + status);
                        xuLiDuLieu();
                        status = "Client " + sttClient + ": " + "SERVER XU LI DU LIEU
THANH CONG";
                        clientStatus.setText(clientStatus.getText() + "\n" + status);
                        status = "Client " + sttClient + ": " + "server bat dau gui du lieu";
                        clientStatus.setText(clientStatus.getText() + "\n" + status);
                        sendDataToClient();
                        status = "Client " + sttClient + ": " + "server gui du lieu thanh
cong";
                        clientStatus.setText(clientStatus.getText() + "\n" + status);
                }

        } catch (Exception e) {
                /*
                 * SOCKET CLOSE
                 * DELETE CLIENT OUT OF DATA
```

```
                                */
                                System.out.println("delete client " + sttClient);
                                listTextArea.remove(clientStatus);
                                cbbClients.removeItem("client " + sttClient);
                                clients.remove(this);
                                txtServerArea.setText(txtServerArea.getText() + "\nNgắt kết nối với client
" + sttClient + "\nSố lượng kết nối: " + clients.size());

                        }
                }

                private void receiveData() throws ClassNotFoundException, IOException
                {

                        //THU TU GUI OBJECT : NODES, LINES, STARTNODENAME,
ENDNODENAME
                        nodes = (ArrayList<Node>) objectInputStream.readObject();
                        System.out.println("Client " + sttClient + ": " + "bat dau nhan du lieu");
                        lines = (ArrayList<Line>) objectInputStream.readObject();
                        startVertName = objectInputStream.readUTF();
                        endVertName = objectInputStream.readUTF();
                }

                private void xuLiDuLieu()
                {
                        System.out.println("Client " + sttClient + ": " + "SERVER BAT DAU XU LI DU
LIEU");
                        int numberOfVert = nodes.size();
                        Vert beginVert = null;
```

```
verts = null;
verts = new ArrayList<Vert>();

for (int i = 0; i < numberOfVert ; i++)
{
        Vert vert = new Vert(nodes.get(i).getName());

        verts.add(vert);

        if (vert.getName().equals(startVertName))
                beginVert = vert;
}

for (Line line : lines)
{
        String startVertName = line.getStartNode().getName();
        String endVertName = line.getEndNode().getName();
        Vert startVert = findVertWithName(startVertName);
        Vert endVert = findVertWithName(endVertName);

        //XU LI KHI STARTVERT OR ENDVERT NULL (KHONG TIM THAY)

        //ADD NEIGHTBOUR CHO CAC VERT
        startVert.addNeighbour(new Edge(line.getWeight(), startVert, endVert));
        endVert.addNeighbour(new Edge(line.getWeight(), endVert, startVert));
}
```

```
PathFinder shortestPath = new PathFinder();
shortestPath.ShortestP(beginVert);

distances = null;
distances = new ArrayList<String>();
paths = null;
paths =  new ArrayList<String>();

String distance, path;

for (int i = 0; i< numberOfVert; i++)
{
    if ( ! verts.get(i).getName().equals(beginVert.getName()))
    {
        distance = beginVert.getName() + " -> " + verts.get(i).getName()
+ " : " + verts.get(i).getDist();

        String pathTemp = 
shortestPath.getShortestP(verts.get(i)).toString();
        pathTemp = pathTemp.substring(1, pathTemp.length()-1);
        path = verts.get(i).getName() + " : " + pathTemp;

        //DU LIEU DUOC GUI DI DEN CLIENT
        distances.add(distance);
        path = path.replaceAll(", ", " -> ");
        System.out.println("path: " + path);
        paths.add(path);
    }
```

```
                }
        }

        private Vert findVertWithName(String name) {
                for (Vert vert : verts)
                {
                        if (vert.getName().equals(name))
                                return vert;
                }
                return null;
        }

        private void sendDataToClient() throws IOException
        {
                objectOutputStream.writeObject(distances);
                objectOutputStream.flush();
                objectOutputStream.writeObject(paths);
                objectOutputStream.flush();

                //PHAI CO DONG NAY DE RESET SOCKET, NEU KO THI OBJECT GUI DI SE BI GHI DE BOI OBJECT TRUOC, KO TAO MOI DOI TUONG DC
                objectOutputStream.reset();
        }
}

public static void main(String[] args)
{
        Server server = new Server();
```

```
            server.run();
        }

        @Override
        public void actionPerformed(ActionEvent e)
        {
            if (e.getSource() == btnStart)
            {
                if (done == false)
                {
                    JOptionPane.showMessageDialog(this ,"Để khởi chạy cổng mới bạn phải dừng server lại trước tiên!");
                    return;
                }
                /*
                 * Reset all
                 */
                /*
                 * restart server
                 */
                try
                {
                    port = Integer.parseInt(txtPort.getText());
                    if (port >= 10000) throw new NumberFormatException();
                    done = false;
                    lblAppStatus.setText("Running...");
                    serverThread.stop();
                    serverThread = new Thread(this);
```

```
                    serverThread.start();
                }
                catch (NumberFormatException ex)
                {
                    JOptionPane.showMessageDialog(this ,"Port phải là số nguyên dương và nhỏ hơn 10.000");
                }
                catch(Exception e2)
                {
                    done = true;
                }
            }
            else if(e.getSource() == btnStop)
            {
                if (done == true)
                    return;

                try {
                    serverSocket.close();
                    serverSocket = null;
                    serverThread.stop();
                    serverThread = new Thread(this);
                    done = true;
                    lblAppStatus.setText("Stoped");
                    txtServerArea.setText(txtServerArea.getText() + "\nServer Kết thúc!");
                } catch (IOException e1) {
                    System.out.println("khong the close server");
                }
```

```
                }
                else if(e.getSource() == cbbClients)
                {
                        int index = cbbClients.getSelectedIndex();
                        setVisibleTextArea(index);
                }
        }

        private void setVisibleTextArea(int index)
        {
                for (int i = 0; i < listTextArea.size(); i++)
                {
                        if (index == i) listTextArea.get(i).setVisible(true);
                        else
                                listTextArea.get(i).setVisible(false);
                }

        }
}
```