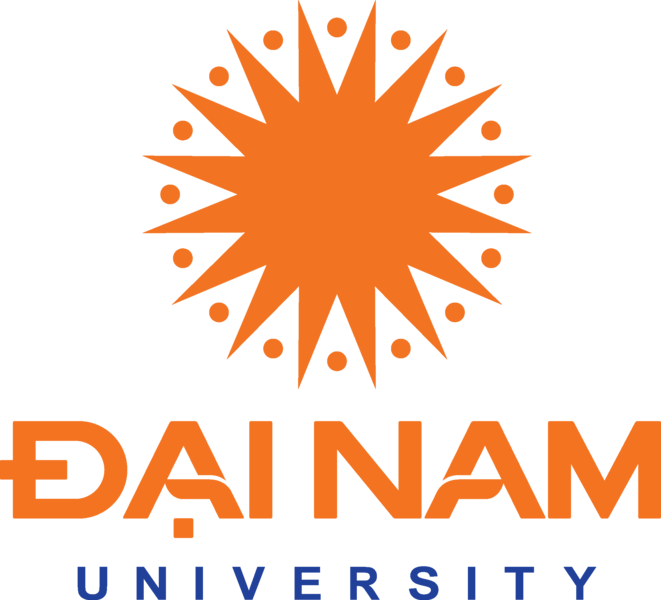
# **BỘ GIÁO DỤC & ĐẠO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

**-------------------🖎🕮✍-------------------**

****

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: LẬP TRÌNH MẠNG**

**ĐỂ TÀI:** *Viết chương trình gửi nhận file nhạc giữa client và server sử dụng UDP Socket. Sau đó file tự động bật lên phát nhạc.*

**Giáo viên hướng dẫn: ThS.Phạm Văn Tiệp  
Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Mã SV** | **Họ & Tên** | **Lớp** |
| 28 | 1571020168 | Lê Văn Mạnh | CNTT 15-01 |

Hà Nội, 08/2024

**BỘ GIÁO DỤC & ĐẠO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

**-------------------🖎🕮✍-------------------**

**A logo with a sun and text

Description automatically generated with medium confidence**

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: LẬP TRÌNH MẠNG**

**ĐỂ TÀI:** *Viết chương trình gửi nhận file nhạc giữa client và server sử dụng UDP Socket. Sau đó file tự động bật lên phát nhạc.*

**Giáo viên hướng dẫn: ThS.Phạm Văn Tiệp  
Sinh viên thực hiện:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã SV** | **Họ & Tên** | **Lớp** | **Điểm** | |
| Bằng số | Bằng chữ |
| 28 | 1571020168 | Lê Văn Mạnh | CNTT 15-01 |  |  |

Hà Nội, 08/2024

# **LỜI NÓI ĐẦU**

Trong thời đại kết nối mạng ngày nay, việc truyền nhận dữ liệu giữa các thiết bị thông qua mạng là một phần không thể thiếu. Trong đề tài này, chúng ta sẽ khám phá cách viết một chương trình gửi và nhận file nhạc giữa client và server sử dụng giao thức UDP (User Datagram Protocol). UDP là một giao thức truyền tải dữ liệu nhẹ và không yêu cầu việc thiết lập kết nối trước khi truyền dữ liệu, điều này có nghĩa là nó thích hợp cho các ứng dụng cần truyền tải dữ liệu nhanh chóng và không yêu cầu độ tin cậy cao như trò chơi trực tuyến, streaming media, và trong trường hợp này, truyền file nhạc.

Chương trình của chúng ta sẽ có hai phần chính: một phần sẽ là server, phụ trách việc nhận file từ client và chuyển tiếp nó tới client khác, và phần còn lại sẽ là client, phụ trách việc gửi file nhạc tới server và sau đó mở và phát nhạc. Việc sử dụng UDP sẽ giúp chúng ta tối ưu hóa tốc độ truyền tải dữ liệu, đặc biệt là với các loại dữ liệu như nhạc.

Chúng ta sẽ bắt đầu bằng việc hiểu rõ cách hoạt động của UDP, sau đó xây dựng server và client tương ứng. Tiếp theo, chúng ta sẽ thử nghiệm chương trình bằng cách gửi và nhận file nhạc, và cuối cùng là mở và phát nhạc tự động khi file được nhận.

Hãy bắt đầu khám phá và triển khai chương trình gửi nhận file nhạc qua UDP để trải nghiệm sự mạnh mẽ và linh hoạt của giao thức này trong việc truyền tải dữ liệu!

Sau đây là bài báo cáo sơ lược về quy trình xây dựng và phát triển mà em tùm hiểu được mong ***Thầy*** xem và cho ý kiến đánh giá để em nhận ra những khiêm skhuyeets của mình. Báo cáo đề tài gồm:

* Chương 1: Tổng quan đề tài.
* Chương 2: Giới thiệu bài toán.
* Chương 3: Kết luận.

Em xin trân thành cảm ơn ***ThS.Phạm Văn Tiệp*** đã tận tình hướng dẫn chỉ bảo em hoàn thành đề tài này.

# **LỜI CẢM ƠN**

Để hoàn thành được bài tiểu luận này, em xin chân thành cảm ơn Ban Giám hiệu, các khoa, phòng và quý thầy, cô của trường **Đại Học Đại Nam**, những người đã tận tình giúp đỡ và tạo điều kiện cho em trong quá trình học tập.

Đặc biệt, em muốn bày tỏ lòng biết ơn đặc biệt đến giảng viên ***ThS. Phạm Văn Tiệp*** - người đã trực tiếp chỉ bảo, hướng dẫn em trong quá trình hoàn thành bài tiểu luận này bằng tất cả lòng nhiệt tình và sự quan tâm sâu sắc. Những góp ý chân thành của ***Thầy*** đã giúp em làm sâu sắc hơn về vấn đề và cải thiện chất lượng của bài viết.

Trong quá trình thực hiện bài tiểu luận này, do hiểu biết còn nhiều hạn chế nên bài làm khó tránh khỏi những thiếu sót. Em rất kính mong nhận được ý kiến đóng góp của quý thầy cô để bài tiểu luận được hoàn thiện hơn.

Cuối cùng, em muốn gửi lời cảm ơn đặc biệt đến tất cả những người đã đọc và đánh giá bài báo cáo này. Sự quan tâm và sự phản hồi của ***Thầy*** không chỉ là nguồn động viên mà còn là động lực để em tiếp tục nghiên cứu và phát triển trong tương lai

Một lần nữa, em xin chân thành cảm ơn ***Thầy*** vì sự giúp đỡ và hỗ trợ trong việc hoàn thành bài tiểu luận này.

Em xin chân thành cảm ơn!

**MỤC LỤC**

[**LỜI NÓI ĐẦU** 3](#_Toc169012756)

[**LỜI CẢM ƠN** 4](#_Toc169012757)

[**GIỚI THIỆU.** 8](#_Toc169012758)

[**Chương I: MỞ ĐẦU.** 9](#_Toc169012759)

[1.1. Tính cấp thiết của đề tài. 9](#_Toc169012760)

[1.2. Tổng quan về đề tài. 10](#_Toc169012761)

[1.3. Mục Đích nghiên cứu. 10](#_Toc169012762)

[1.4. Đối tượng và phạm vi nhiên cứu. 11](#_Toc169012763)

[1.5. Phương pháp nghiên cứu. 12](#_Toc169012764)

[**Chương II: CƠ SỞ LÝ THUYẾT.** 14](#_Toc169012765)

[2.1. Giao thức UDP. 14](#_Toc169012766)

[2.2. Giới thiệu về mô hình Client – Server. 16](#_Toc169012767)

[2.3. Giới thiệu về phầm mềm sử dụng. 17](#_Toc169012768)

[2.4. Giới thiệu ngôn ngữ sử dụng. 17](#_Toc169012769)

[2.5. Giới thiệu về các thư viện sử dụng. 18](#_Toc169012773)

[**Chương III: TRIỂN KHAI XÂY DỰNG CHƯƠNG TRÌNH.** 21](#_Toc169012774)

[3.1. Giới thiệu về đề tài. 21](#_Toc169012775)

[3.2. Cài đặt thuật toán. 21](#_Toc169012776)

[3.3. Phác thảo giao diện. 22](#_Toc169012777)

[3.4. Cài đặt thuật toán. 23](#_Toc169012782)

[3.4.1. Chương trình gửi nhận giữa Client và Server UDP Socket. 23](#_Toc169012783)

[3.4.2. Chương trình giao diện Java Swing. 27](#_Toc169012784)

[**Chương IV: KẾT LUẬN.** 31](#_Toc169012785)

[4.1. Kết luận. 31](#_Toc169012786)

[4.1.1. Kết quả đạt được. 31](#_Toc169012787)

[4.1.2. Nội dung đạt được. 31](#_Toc169012789)

[**TÀI LIỆU THAM KHẢO.** 33](#_Toc169012790)

# **GIỚI THIỆU.**

Lập trình mạng là một lĩnh vực quan trọng trong công nghệ thông tin, tập trung vào việc phát triển và quản lý các ứng dụng, dịch vụ hoặc hệ thống kết nối mạng. Đây là quá trình xây dựng và tối ưu hóa các phần mềm để truyền thông và trao đổi dữ liệu qua mạng, từ các thiết bị như máy tính, điện thoại thông minh, đến các hệ thống phức tạp như Internet.

Kỹ sư lập trình mạng phải hiểu sâu về các giao thức mạng, các nguyên tắc cơ bản của việc truyền thông dữ liệu, và các công nghệ liên quan như TCP/IP, HTTP, DNS, và nhiều giao thức khác. Họ cũng cần có kiến thức vững về lập trình, bao gồm các ngôn ngữ như Python, Java, C++, và Ruby, để xây dựng các ứng dụng và công cụ quản lý mạng.

Lập trình mạng cũng liên quan chặt chẽ đến bảo mật mạng, với mục tiêu đảm bảo rằng dữ liệu được truyền tải qua mạng an toàn và bảo mật. Kỹ sư lập trình mạng thường phải làm việc với các công nghệ bảo mật như mã hóa dữ liệu (encryption), chứng thực (authentication), và kiểm soát truy cập (access control).

Ngoài ra, lập trình mạng cũng liên quan đến việc phát triển các ứng dụng và dịch vụ mạng như các ứng dụng web, ứng dụng di động, và các hệ thống phân phối nội dung (content delivery systems). Điều này đòi hỏi kỹ sư lập trình mạng phải có khả năng thiết kế và triển khai các hệ thống có khả năng mở rộng và đáng tin cậy, đồng thời đảm bảo hiệu suất và khả năng mở rộng của hệ thống.

Lập trình mạng chính là một lĩnh vực đa dạng và phức tạp, đòi hỏi kiến thức sâu rộng về cả lập trình và mạng máy tính. Những kỹ sư thành công trong lĩnh vực này thường là những người có khả năng tư duy logic tốt, có kiến thức chuyên sâu về các công nghệ mạng, và có khả năng làm việc hiệu quả trong môi trường đa dạng và thay đổi nhanh chóng.

1. **Chương I: MỞ ĐẦU.**
   1. **Tính cấp thiết của đề tài.**

**Đề tài:** *Viết chương trình gửi nhận file nhạc giữa client và server sử dụng UDP Socket. Sau đó file tự động bật lên phát nhạc.*

Trong thời đại số hóa hiện nay, truyền tải và chia sẻ dữ liệu âm nhạc qua mạng Internet đang trở thành một phần không thể thiếu của cuộc sống hàng ngày. Người dùng không chỉ muốn thưởng thức âm nhạc trên thiết bị cá nhân của họ mà còn muốn chia sẻ và truyền tải nó qua mạng tới các thiết bị khác, tạo điều kiện thuận lợi cho việc tiếp cận âm nhạc từ bất kỳ nơi nào trên thế giới.

Giao thức UDP (User Datagram Protocol) được lựa chọn cho đề tài này vì tính nhẹ nhàng và tốc độ truyền tải cao của nó. Trong trường hợp truyền tải dữ liệu âm nhạc, việc đảm bảo dữ liệu được truyền tải một cách nhanh chóng và mượt mà là rất quan trọng. UDP giúp tối ưu hóa hiệu suất truyền tải, đặc biệt là đối với các loại dữ liệu như nhạc, giúp đảm bảo trải nghiệm người dùng được tốt nhất.

Người dùng hiện đại yêu cầu không chỉ việc truyền tải dữ liệu âm nhạc mà còn muốn trải nghiệm một cách thuận tiện và tự động. Tính năng tự động mở và phát nhạc khi file được nhận là một tính năng hấp dẫn và tiện ích mà người dùng đánh giá cao. Điều này giúp tăng cường trải nghiệm người dùng và tạo ra một quy trình truyền tải và thưởng thức âm nhạc mượt mà và không gián đoạn.

Việc xây dựng một ứng dụng client-server cho phép người dùng truyền tải dữ liệu âm nhạc giữa các thiết bị một cách dễ dàng và linh hoạt. Đồng thời, việc tổ chức dữ liệu theo mô hình client-server cũng giúp quản lý và kiểm soát dữ liệu một cách hiệu quả, tạo điều kiện thuận lợi cho việc quản lý và phân phối dữ liệu âm nhạc trên mạng.

Tính năng tự động mở và phát nhạc khi file được nhận không chỉ làm tăng tính tiện ích mà còn tạo ra một trải nghiệm người dùng tốt hơn. Người dùng sẽ không cần phải thực hiện bất kỳ hành động nào sau khi nhận được file âm nhạc mà chỉ cần ngồi và thưởng thức, tạo điều kiện thuận lợi cho việc sử dụng và trải nghiệm ứng dụng truyền tải âm nhạc.

Đề tài này không chỉ tạo ra một ứng dụng đơn giản để truyền tải dữ liệu âm nhạc mà còn mở ra nhiều khả năng mở rộng và cải tiến. Việc phát triển các tính năng bổ sung như tương tác người dùng, quản lý file, hoặc tích hợp công nghệ âm thanh tiên tiến sẽ làm cho ứng dụng trở nên phong phú và hấp dẫn hơn, tạo điều kiện thuận lợi cho việc phát triển và mở rộng ứng dụng trong tương lai.

* 1. **Tổng quan về đề tài.**

Việc truyền tải và chia sẻ dữ liệu âm nhạc giữa các thiết bị thông qua mạng Internet là một nhu cầu ngày càng phổ biến trong xã hội số hóa hiện nay. Đề tài này tập trung vào việc phát triển một ứng dụng cho phép truyền tải file nhạc giữa client và server sử dụng giao thức UDP Socket. Mục tiêu chính của đề tài là xây dựng một hệ thống cho phép người dùng gửi và nhận file nhạc một cách đơn giản và nhanh chóng.

Giao thức UDP (User Datagram Protocol) được chọn lựa vì tính nhẹ nhàng và tốc độ truyền tải cao. Trong trường hợp truyền tải dữ liệu âm nhạc, việc đảm bảo dữ liệu được truyền tải một cách nhanh chóng và mượt mà là rất quan trọng. UDP giúp tối ưu hóa hiệu suất truyền tải, đặc biệt là trong việc truyền tải dữ liệu âm nhạc.

Một tính năng đặc biệt của ứng dụng là khả năng tự động mở và phát nhạc khi file được nhận. Điều này giúp tăng cường trải nghiệm người dùng và tạo ra một quy trình truyền tải và thưởng thức âm nhạc mượt mà và không gián đoạn.

Đề tài sử dụng mô hình client-server để quản lý việc truyền tải dữ liệu âm nhạc. Mô hình này cho phép người dùng truyền tải dữ liệu giữa các thiết bị một cách linh hoạt và hiệu quả. Đồng thời, việc tổ chức dữ liệu theo mô hình client-server cũng giúp quản lý và kiểm soát dữ liệu một cách hiệu quả.Và không chỉ giải quyết một vấn đề cụ thể mà còn mở ra nhiều khả năng mở rộng và cải tiến trong tương lai. Việc phát triển các tính năng bổ sung như tương tác người dùng, quản lý file, hoặc tích hợp công nghệ âm thanh tiên tiến sẽ làm cho ứng dụng trở nên phong phú và hấp dẫn hơn, tạo điều kiện thuận lợi cho việc phát triển và mở rộng ứng dụng trong tương lai.

* 1. **Mục Đích nghiên cứu.**

Mục đích chính của nghiên cứu này là xây dựng và triển khai một ứng dụng cho phép truyền tải và nhận file nhạc giữa các thiết bị thông qua mạng sử dụng giao thức UDP Socket. Nghiên cứu này đặt ra một số mục tiêu cụ thể sau:

* **Phát triển ứng dụng truyền tải dữ liệu âm nhạc:**
* Xây dựng một ứng dụng client-server cho phép người dùng gửi và nhận file nhạc qua mạng.
* Tối ưu hóa giao thức truyền tải để đảm bảo tốc độ và hiệu suất cao nhất trong việc truyền tải dữ liệu âm nhạc.
* **Tạo ra tính năng tự động mở và phát nhạc:**
* Phát triển tính năng tự động mở và phát nhạc khi file được nhận, tạo ra một trải nghiệm người dùng tiện lợi và thú vị.
* **Nghiên cứu và áp dụng giao thức UDP Socket:**
* Nghiên cứu cách thức hoạt động và tính năng của giao thức UDP Socket.
* Áp dụng kiến thức và kỹ thuật của giao thức này vào việc truyền tải dữ liệu âm nhạc một cách hiệu quả và tin cậy.
* **Kiểm thử và đánh giá hiệu suất:**
* Thực hiện các bài kiểm thử để đánh giá hiệu suất của ứng dụng trong việc truyền tải dữ liệu âm nhạc.
* Đánh giá và so sánh kết quả kiểm thử để tối ưu hóa và cải thiện hiệu suất của ứng dụng.
* **Ứng dụng thực tiễn:**
* Xây dựng một ứng dụng thực tế và áp dụng vào các tình huống truyền tải dữ liệu âm nhạc trong cuộc sống hàng ngày.
* Đưa ra các khuyến nghị và phản hồi từ người dùng để phát triển và cải thiện ứng dụng trong tương lai.
  1. **Đối tượng và phạm vi nhiên cứu.**
* **Đối tượng nghiên cứu:**
* Đối tượng của nghiên cứu này là mọi người dùng có nhu cầu truyền tải và nhận file nhạc qua mạng Internet. Đối tượng này bao gồm cả người dùng cá nhân và tổ chức hoặc doanh nghiệp có nhu cầu chia sẻ dữ liệu âm nhạc giữa các thiết bị trong mạng nội bộ hoặc qua Internet.
* **Phạm vi nghiên cứu:**
* *Xây dựng ứng dụng client-server:* Phát triển một ứng dụng client-server cho phép người dùng gửi và nhận file nhạc qua mạng. Phạm vi này bao gồm việc thiết kế giao diện người dùng, triển khai các chức năng gửi và nhận file, và xây dựng các module liên quan đến việc tự động mở và phát nhạc.
* *Tối ưu hóa giao thức UDP Socket:* Nghiên cứu và áp dụng các kỹ thuật tối ưu hóa để cải thiện hiệu suất truyền tải dữ liệu âm nhạc qua giao thức UDP Socket. Phạm vi này bao gồm việc đánh giá và điều chỉnh các tham số liên quan đến giao thức UDP để đảm bảo tốc độ và độ tin cậy của việc truyền tải dữ liệu.
* *Thử nghiệm và đánh giá hiệu suất:* Thực hiện các bài kiểm thử để đánh giá hiệu suất của ứng dụng trong việc truyền tải dữ liệu âm nhạc. Phạm vi này bao gồm việc sử dụng các kịch bản thử nghiệm khác nhau để đo lường và so sánh tốc độ và độ ổn định của việc truyền tải dữ liệu.
* *Áp dụng thực tế*: Ứng dụng ứng dụng vào các tình huống thực tiễn để đánh giá hiệu quả và tính ứng dụng của nó trong việc truyền tải và nhận file nhạc. Phạm vi này bao gồm việc thu thập phản hồi từ người dùng và đề xuất các cải tiến và điều chỉnh để cải thiện trải nghiệm người dùng và hiệu suất của ứng dụng.
  1. **Phương pháp nghiên cứu.**
* **Phân tích yêu cầu và thiết kế:**
* Xác định yêu cầu chức năng và phi chức năng của ứng dụng dựa trên nhu cầu của người dùng.
* Thiết kế kiến trúc ứng dụng bằng cách sử dụng mô hình phần mềm, đảm bảo tính linh hoạt và mở rộng.
* **Triển khai và phát triển:**
* Sử dụng ngôn ngữ lập trình Java và thư viện socket, jplayer để triển khai ứng dụng client-server.
* Phát triển các tính năng gửi, nhận và tự động mở file nhạc trên cả hai phía client và server.
* **Thử nghiệm và đánh giá:**
* Tạo ra các kịch bản thử nghiệm đa dạng để đánh giá hiệu suất và độ tin cậy của ứng dụng.
* Đo lường tốc độ truyền tải, độ trễ và độ ổn định của việc truyền tải dữ liệu âm nhạc qua mạng.
* **Thu thập phản hồi và phân tích:**
* Tiến hành cuộc khảo sát người dùng để thu thập phản hồi về trải nghiệm sử dụng ứng dụng.
* Phân tích phản hồi từ người dùng để xác định các vấn đề và cải thiện tiềm năng của ứng dụng.
* **Đề xuất và triển khai cải tiến:**
* Dựa trên phản hồi từ người dùng và kết quả thử nghiệm, đề xuất các cải tiến và điều chỉnh để tối ưu hóa hiệu suất và trải nghiệm người dùng.
* Triển khai các cải tiến vào ứng dụng và tiến hành các bài kiểm thử để đảm bảo tính ổn định và hiệu suất của chúng.
* **Tổng hợp kết quả và báo cáo:**
* Tổng kết kết quả của nghiên cứu và phân tích kết quả để rút ra các kết luận và hướng dẫn cho tương lai.
* Viết báo cáo nghiên cứu với cấu trúc rõ ràng, bao gồm mục đích, phương pháp, kết quả và đề xuất cải tiến.

1. **Chương II: CƠ SỞ LÝ THUYẾT.**
   1. **Giao thức UDP.**

**Giao thức UDP (User Datagram Protocol)** là một giao thức cốt lõi của giao thức TCP/IP được sử dụng để thiết lập các kết nối có độ trễ thấp và giảm mất mát giữa các ứng dụng trên Internet. Giao thức UDP cung cấp hai dịch vụ không được cung cấp bởi lớp IP, đó là:

* Cung cấp các [port](https://bkhost.vn/blog/port-la-gi/) number để giúp phân biệt các yêu cầu khác nhau từ người dùng.
* Đồng thời sử dụng một thuật toán checksum để xem dữ liệu có được toàn vẹn hay không.

UDP là một giải pháp được dùng thay cho giao thức TCP. Cả hai giao thức này đều hoạt động trên lớp IP và có lúc được gọi là UDP/IP hoặc TCP/IP. Mặc dù đều hoạt động trên giao thức IP, nhưng UDP và TCP có những khác biệt rất quan trọng. Cụ thể:

Giao thức TCP hỗ trợ giao tiếp máy chủ tới máy chủ (host-to-host). Được coi là phương thức vận chuyển bảo mật đáng tin cậy nhưng có tốc độ truyền chậm hơn. TCP chia nhỏ các tập [dữ liệu lớn](https://bkhost.vn/blog/big-data-du-lieu-lon/) thành các gói dữ liệu riêng lẻ, đảm bảo lưu lượng và kiểm soát lỗi, thực hiện gửi lại các tập tin bị mất và gửi theo đúng thứ tự.

Giao thức UDP cho phép giao tiếp quy trình tới quy trình (process-to-process). UDP chỉ gửi gói tin được gọi là datagram, các gói tin này được truyền đi theo các đường dẫn khác nhau giữa người gửi và người nhận. Nghĩa là UDP có khả năng làm mất dữ liệu hoặc truyền dữ liệu không theo thứ tự, tuy nhiên lại sử dụng chi phí [băng thông](https://bkhost.vn/blog/bandwidth-bang-thong/) và độ trễ thấp hơn.

**Các tính năng của giao thức UDP:**

Giao thức UDP có đặc tính nổi bật là có thể sử dụng trên các ứng dụng chấp nhận khả năng bị mất dữ liệu. Những tính năng mà UDP mang lại ấn tượng nhất bao gồm:

Cho phép các gói bị loại bỏ và nhận không theo thứ tự truyền đi ban đầu, điều này làm cho UDP phù hợp với các ứng dụng chạy thời gian thực, trong đó ưu tiên tốc độ và độ trễ thấp.

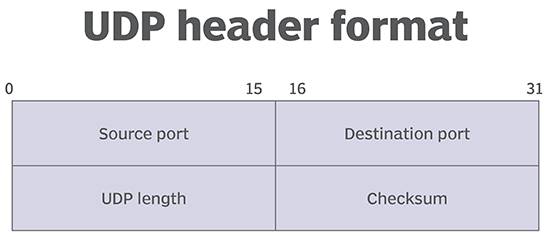
Nó có thể được sử dụng cho các giao thức dựa trên giao dịch.

UDP có thể hữu ích khi có nhiều người truy cập, kết nối và khi không có nhu cầu điều chỉnh lỗi thời gian thực.

**Cấu trúc header của giao thức UDP:**

Cấu trúc header của UDP chứa một tập hợp các tham số gọi là các trường được xác định bởi các thông số kỹ thuật của giao thức. Có 4 trường trong cấu trúc header của UDP, mỗi trường bằng 2 byte và các trường này gồm:

* Source Port Number là số của người gửi.
* Destination Port Number là số địa chỉ mà diagram được gửi đến.
* UDP Length là độ dài tính bằng byte của UDP header hoặc bất kỳ gói dữ liệu nào.
* Checksum là thuật toán kiểm tra lỗi đảm bảo tính toàn vẹn của dữ liệu được sử dụng trong [IPv6](https://bkhost.vn/blog/ipv6/) hoặc [IPv4](https://bkhost.vn/blog/ipv4/).



Hình : Định dạng cấu trúc header của giao thức UDP.

**Giao thức UDP hoạt động như thế nào?**

Giao thức UDP sử dụng IP để lấy datagram từ máy tính này sang máy tính khác. UDP hoạt động bằng cách thu thập dữ liệu trong gói UDP và thêm thông tin tiêu đề của riêng mình vào gói. Dữ liệu này bao gồm các port nguồn và đích để giao tiếp, độ dài gói và thuật toán kiểm tra checksum. Sau khi các gói UDP được gói gọn trong gói IP, chúng sẽ gửi đến các điểm đích được chỉ định.

Giao thức UDP sử dụng một mô hình truyền đơn giản không bao gồm các cuộc đối thoại bắt tay (handshaking dialogues) để cung cấp độ tin cậy, sắp đặt hoặc tính toàn vẹn dữ liệu. Do đó, dịch vụ của UDP không đáng tin cậy. Các gói có thể không xuất hiện, dữ liệu có thể thêm bản sao hoặc bị mất mát mà không có cảnh báo.

Mặc dù phương pháp truyền dữ liệu của UDP không đảm bảo rằng dữ liệu được gửi sẽ đến đích, nhưng nó có chi phí thấp và phổ biến cho các dịch vụ, ứng dụng chấp nhận việc mất mát về dữ liệu.

**Các ứng dụng của giao thức UDP:**

UDP là một giao thức lý tưởng cho các ứng dụng mạng trong đó độ trễ thấp rất quan trọng, chẳng hạn như trong trò chơi, truyền thông thoại và video. Dữ liệu có thể bị mất mà không ảnh hưởng xấu đến chất lượng nhận thức.

UDP có thể được sử dụng trong các ứng dụng yêu cầu bảo đảm tính toàn vẹn của dữ liệu. Chẳng hạn một ứng dụng được cấu hình để quản lý quá trình truyền lại các gói bị mất và sắp xếp chính xác các gói nhận được. Cách tiếp cận này có thể giúp cải thiện tốc độ truyền dữ liệu của các tệp lớn so với TCP.

UDP cũng được sử dụng để phát đa dạng vì nó hỗ trợ chuyển đổi gói. Ngoài ra, UDP được sử dụng cho một số giao thức cập nhật định tuyến, chẳng hạn như giao thức thông tin định tuyến (RIP-Routing Information Protocol).

UDP có thể được sử dụng trong các ứng dụng trong đó tốc độ được ưu tiên hơn độ tin cậy. Chẳng hạn, có thể thận trọng khi sử dụng UDP trong một ứng dụng gửi dữ liệu từ việc thu thập nhanh, trong quá trình sử dụng việc mất một số điểm dữ liệu vẫn được chấp nhận.

* 1. **Giới thiệu về mô hình Client – Server.**

Mô hình client server mô hình [mạng máy tính](https://vietnix.vn/mang-may-tinh/) gồm có 2 thành phần chính là client và server (tức là máy khách – máy chủ). Client sẽ là bên yêu cầu dịch vụ cài đặt cũng như lưu trữ tài nguyên từ phía server

Khi máy client gửi yêu cầu dữ liệu đến server thông qua Internet, server sẽ chấp nhận quy trình được yêu cầu. Sau đó gửi các gói dữ liệu được yêu cầu trở lại client. Client không chia sẻ bất kỳ tài nguyên nào của họ.

**Mô hình Client Server là gì?**

Đây là một cấu trúc ứng dụng phân tán, nó phân vùng các nhiệm vụ hay workload giữa các nhà cung cấp tài nguyên hoặc dịch vụ, gọi là server, và người yêu cầu dịch vụ (Client).

Vậy mô hình Client Server ứng dụng như thế nào? Hiện nay mô hình này có một số ứng dụng phổ biến trong email và [World Wide Web](https://vietnix.vn/www-la-gi/),…

**Nguyên tắc hoạt động của mô hình Client Server:**

*Client:*

Khi nói đến Client (khách hàng), thì nó có nghĩa là một người hay một tổ chức sử dụng một dịch vụ cụ thể nào đó. Trong thế giới kỹ thuật số cũng tương tự như vậy. Client là một máy tính (Host), tức là có khả năng nhận thông tin hoặc sử dụng một dịch vụ cụ thể từ các nhà cung cấp dịch vụ (Server).

*Server:*

Tương tự như vậy, khi nói đến Serverthì nó có nghĩa là một máy chủ hay một phương tiện phục vụ các dịch vụ nào đó. Trong lĩnh vực công nghệ thì Server là một máy tính từ xa. Nó cung cấp các thông tin (dữ liệu) hoặc quyền truy cập vào các dịch vụ cụ thể.

Vì vậy, về cơ bản thì trong mô hình Client và Server, Client là đối tượng yêu cầu một thứ gì đó. Server thì phục vụ nó, miễn là nó có mặt trong cơ sở dữ liệu.

* 1. **Giới thiệu về phầm mềm sử dụng.**

**Eclipse là gì?**

Eclipse là 1 công cụ hỗ trợ lập trình mã nguồn mở được phát triển bởi IBM.

Eclipse như một môi trường phát triển Java tích hợp (IDE), với Eclipse chúng ta có thể mở rộng hơn mã nguồn bằng cách chèn thêm các plugins cho project (PDE- Plug-in Development Environment). Mặc dù Eclipse được viết bằng ngôn ngữ lập trình Java, nhưng việc sử dụng nó không hạn chế chỉ cho ngôn ngữ Java. Ví dụ, Eclipse hỗ trợ sẵn hoặc có thể cài thêm các plugins để hỗ trợ cho các ngôn ngữ lập trình như C/C + + và COBOL. Ngoài ra, còn rất nhiều ngôn ngữ khác như PHP, Groovy, ...

Eclipse còn hỗ trợ cho lập trình viên code theo các mô hình phát triển như MVC, tạo thêm các lib hỗ trợ phát triển phần mềm.

**Ưu điểm:**

* Tạo thuận lợi cho tích hợp liền mạch các công cụ bên trong mỗi một và xuyên qua nhiều kiểu nội dung và các nhà cung cấp công cụ khác nhau.
* Hỗ trợ việc xây dựng nhiều công cụ.
* Hỗ trợ một tập hợp không hạn chế các nhà cung cấp công cụ, bao gồm cả các nhà cung cấp phần mềm độc lập (ISV).
* Hỗ trợ các công cụ thao tác các kiểu nội dung bất kỳ (bao gồm cả HTML, Java, C, JSP, EJB, XML, và GIF).
* Hỗ trợ cả môi trường phát triển ứng dụng GUI lẫn không dựa trên GUI.
* Chạy trên nhiều hệ điều hành, bao gồm cả Windows và Linux.
* Lợi dụng tính phổ biến của ngôn ngữ Java để viết các công cụ.
* Do sử dụng SWT/JFace nên có lẽ load nhanh hơn.

**Nhược điểm:**

* Cài đặt phức tạp.
* Tốn phần cứng máy Eclipse cũng khá tốn bộ nhớ máy để cài đặt nên hiện nay eclipse đã khắc phục bằng phiên bản Eclipse Luna, không cần cái đặt.
* Tốn bộ nhớ máy: chạy eclipse cũng khá tốn bộ nhớ máy tính nên để sử dụng nó máy của bạn cũng cần có cấu hình tương đối cao.
* Thiếu tính nhất quán vì có nhiều plugins quá.
  1. **Giới thiệu ngôn ngữ sử dụng.**

Java là một ngôn ngữ lập trình và nền tảng tính toán đa mục đích được phát triển bởi Sun Microsystems, hiện nay là một phần của Oracle Corporation. Ngôn ngữ này được ra mắt lần đầu vào năm 1995 và nhanh chóng trở thành một trong những công cụ phát triển phần mềm phổ biến nhất trên thế giới.

Ưu điểm chính của Java bao gồm tính đa nền tảng cao, cho phép mã nguồn Java được viết một lần và chạy trên nhiều hệ điều hành và thiết bị khác nhau mà không cần sửa đổi. Điều này đảm bảo tính di động và linh hoạt của ứng dụng, giúp tiết kiệm thời gian và công sức cho các nhà phát triển. Java cũng có một cộng đồng lớn và năng động, cùng với một hệ sinh thái phong phú của các thư viện và công cụ hỗ trợ, giúp cho việc phát triển ứng dụng trở nên thuận tiện và hiệu quả.

Tuy nhiên, Java cũng có một số nhược điểm. Việc quản lý bộ nhớ tự động của Java có thể dẫn đến hiệu suất thấp hơn so với một số ngôn ngữ khác như C++ trong một số trường hợp. Ngoài ra, một số lập trình viên cho rằng cú pháp của Java có thể phức tạp hơn so với một số ngôn ngữ khác, đặc biệt là khi phải làm việc với các đối tượng phức tạp và kiến trúc phần mềm lớn.

* 1. **Giới thiệu về các thư viện sử dụng.**

**Javazoom** là một thư viện Java phổ biến được phát triển bởi javazoom.net, cung cấp các công cụ và thư viện cho việc xử lý âm thanh trong ứng dụng Java. Trong thư viện này, JLayer là một thành phần nổi bật, được sử dụng để phát và điều khiển các tệp âm thanh MPEG 1/2/2.5 Layer 3 (MP3). Đây là một giải pháp tiện lợi và mạnh mẽ cho việc tích hợp chức năng phát nhạc vào ứng dụng của bạn.

**Ưu điểm của Javazoom và JLayer bao gồm:**

* Dễ sử dụng: Javazoom cung cấp một API đơn giản và dễ hiểu, giúp cho việc phát và điều khiển âm thanh trong ứng dụng Java trở nên dễ dàng hơn.
* Hỗ trợ đa nền tảng: Thư viện này hoạt động trên nhiều nền tảng khác nhau, bao gồm cả Windows, Linux và macOS, đảm bảo tính linh hoạt và di động cho ứng dụng của bạn.
* Hiệu suất cao: Javazoom được tối ưu hóa để đạt hiệu suất tốt, giúp ứng dụng của bạn chạy mượt mà và phản hồi nhanh chóng trong việc phát nhạc.

**Tuy nhiên, cũng có một số nhược điểm cần xem xét:**

* Hạn chế định dạng: Mặc dù JLayer hỗ trợ phát MP3, nhưng nó không hỗ trợ các định dạng âm thanh khác như FLAC, AAC, và OGG, điều này có thể là một hạn chế đối với một số ứng dụng cần đa dạng hơn.
* Phát triển không được hỗ trợ tích cực: Javazoom đã không còn được cập nhật hoặc phát triển tích cực trong một thời gian dài, điều này có thể gây ra một số vấn đề liên quan đến sửa lỗi và cải thiện tính năng.

**Socket:** Đây chính là điểm cuối end-point tại liên kết truyền thông 2 chiều (two-way communication) và biểu diễn kết nối giữa Server - Client.

**Datagram Packet:**

* Lớp này cho phép truyền thông với giao thức UDP.
* Kế thừa trực tiếp từ lớp Object.
* Gói tin chứa 4 thành phần quan trọng:
* Địa chỉ: getAddress()
* Dữ liệu truyền: getData()
* Kích thước gói tin: getLength()
* Số hiệu cổng chứa gói tin: getPort()

**Datagram Socket:**

* Datagram Socket còn được biết tới với một tên gọi khác đó chính là một Socket không hướng kết nối. Theo đó đây là Socket hoạt động qua giao thức User Datagram Protocol (UDP).
* Nó hoàn toàn có thể hoạt động tốt nhất ngay cả khi không có thiết lập kết nối của hai máy với nhau. Như vậy nói cách khác thì đây cũng là định nghĩa dành cho câu hỏi là: “Giao thức UDP là gì?”

**Ưu điểm:**

* Quá trình truyền tải cũng như kết nối thông tin vô cùng đơn giản và không cần phải thực hiện quá nhiều thao tác.
* Thời gian dữ liệu truyền tải vô cùng nhanh chóng

**Nhược điểm:**

* Nhược điểm của Datagram Socket mà người dùng cần phải lưu ý hiện tại quá trình truyền tải thông tin sẽ không đảm bảo độ tin cậy và thông tin hoàn toàn có thể bị lặp hoặc truyền sai thứ tự.

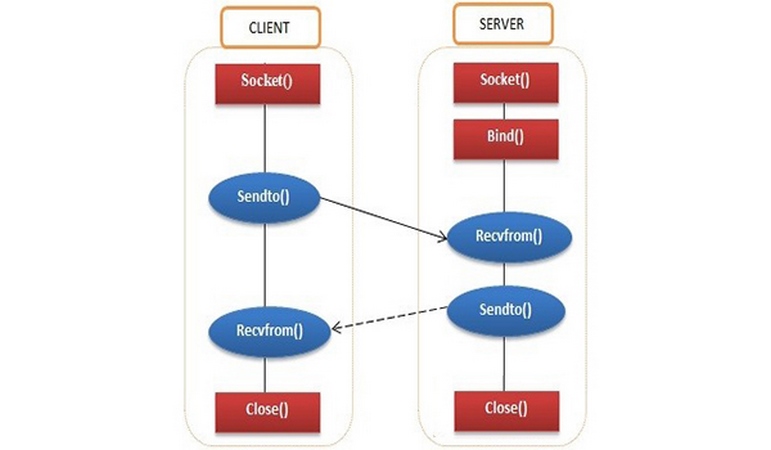
Việc sử dụng UDP thường được ưa chuộng trong các ứng dụng cần truyền tải dữ liệu nhanh và không quá quan trọng về việc mất mát dữ liệu, ví dụ như truyền tải video, audio, hoặc game online. Tuy nhiên, do UDP không đảm bảo tính toàn vẹn của dữ liệu và không đảm bảo thứ tự của dữ liệu, nên trong một số trường hợp cần sự đảm bảo này, TCP có thể là sự lựa chọn phù hợp hơn.

1. **Chương III: TRIỂN KHAI XÂY DỰNG CHƯƠNG TRÌNH.**
   1. **Giới thiệu về đề tài.**

Bài toán *"Viết chương trình gửi nhận file nhạc giữa client và server sử dụng UDP Socket và tự động phát nhạc"* là một ví dụ điển hình về việc sử dụng mạng để truyền tải dữ liệu giữa các thiết bị, trong trường hợp này là giữa một client và một server qua giao thức UDP (User Datagram Protocol). Bài toán này bao gồm ba phần chính:

* **Gửi và Nhận File Nhạc:**
* Server gửi một file nhạc đến Client thông qua giao thức UDP.
* Server nhận file nhạc từ client và lưu lại trên hệ thống của mình.
* **Phát Nhạc Tự Động:**
* Sau khi Client nhận được file nhạc từ Server, nó sẽ mở và phát nhạc tự động.
* **Giao Tiếp Mạng qua UDP:**
* Sử dụng UDP Socket để thiết lập kết nối giữa client và server.

UDP được sử dụng thay vì TCP (Transmission Control Protocol) vì nó có thể gửi dữ liệu mà không cần thiết lập kết nối trước đó, giúp giảm độ trễ và tăng tốc độ truyền tải.



Hình : Giao thức UDP.

* 1. **Cài đặt thuật toán.**
* **Tạo Server:**
* Tạo một lớp UDPServer để thực hiện vai trò của server.
* Sử dụng lớp DatagramSocket để tạo socket để nhận dữ liệu từ client.
* Tạo một mảng byte để nhận dữ liệu từ client.
* Nhận dữ liệu từ client và lưu vào một file âm nhạc.
* Sau khi nhận xong, chạy file nhạc sử dụng lớp Runtime.
* **Tạo Client:**
* Tạo một lớp UDPClient để thực hiện vai trò của client.
* Tạo một lớp DatagramSocket để tạo socket để gửi dữ liệu đến server.
* Đọc file nhạc từ ổ đĩa và chuyển đổi thành mảng byte.
* Gửi mảng byte này đến server sử dụng lớp DatagramPacket.
* **Cài đặt thư viện:**
  + Sử dụng thư viện Jplayer: **jPlayer** là một thư viện mã nguồn mở viết bằng JavaScript, hoàn toàn miễn phí và được sử dụng như một plugin của jQuery (hoặc Zepto). Nó cho phép bạn dễ dàng tích hợp âm thanh và video đa nền tảng vào trang web của bạn.
  + Sử dụng Java Swing: **Java Swing** là một bộ công cụ giao diện người dùng (GUI) cho Java. Nó là một phần của **Java Foundation Classes (JFC)**, một API cung cấp giao diện người dùng đồ họa cho các chương trình Java.
  1. **Phác thảo giao diện.**

**A screenshot of a computer

Description automatically generated**

**Hình: Giao diện Server.**

**A white rectangular object with black text

Description automatically generated**

**Hình : Giao diện Client.**

* 1. **Cài đặt thuật toán.**
     1. **Chương trình gửi nhận giữa Client và Server UDP Socket.**
* **Server:**

1. private void browseFile() {

2. JFileChooser fileChooser = new JFileChooser();

3. int result = fileChooser.showOpenDialog(this);

4. if (result == JFileChooser.APPROVE\_OPTION) {

5. File selectedFile = fileChooser.getSelectedFile();

6. chatTextArea.append("Selected file: " + selectedFile.getAbsolutePath() + "\n");

7. }

8. }

9.

10. private void sendFile() {

11. try {

12. // Trước khi gửi file, chờ yêu cầu kết nối từ máy khách

13. byte[] receiveData = new byte[1024];

14. DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

15. socket.receive(receivePacket);

16. // Lấy địa chỉ IP và cổng của máy khách

17. clientIP = receivePacket.getAddress();

18. clientPort = receivePacket.getPort();

19. File file = new File("D:\\App\\MusicUDP\_VanManh\\Server\\musicserver.mp3");

20. FileInputStream fis = new FileInputStream(file);

21. byte[] sendData = new byte[1024];

22. while (true) {

23. int bytesRead = fis.read(sendData);

24. if (bytesRead == -1) {

25. break;

26. }

27. DatagramPacket sendPacket = new DatagramPacket(sendData, bytesRead, clientIP, clientPort);

28. socket.send(sendPacket);

29. Thread.sleep(5); // Thời gian chờ giữa các gói

30. }

31. fis.close();

32. chatTextArea.append("File sent successfully.\n");

33. } catch (IOException | InterruptedException e) {

34. e.printStackTrace();

35. }

36. }

37.

38. public static void main(String[] args) {

39. EventQueue.invokeLater(new Runnable() {

40. public void run() {

41. Server server = new Server();

42. server.setVisible(true);

43. }

44. });

45. }

46.

* **Client:**

1. private void receiveFile() {

2. try {

3. byte[] receiveData = new byte[1024];

4.

5. InetAddress serverIP = InetAddress.getLocalHost();

6. int serverPort = 9876;

7. // Gửi yêu cầu kết nối đến server

8. DatagramPacket sendPacket = new DatagramPacket(new byte[1024], 0, serverIP, serverPort);

9. clientSocket.send(sendPacket);

10. // Nhận tên file mới từ server

11. DatagramPacket fileNamePacket = new DatagramPacket(receiveData, receiveData.length);

12. clientSocket.receive(fileNamePacket);

13. String newFileName = new String(fileNamePacket.getData()).trim();

14. chatTextArea.append("Received file name: " + newFileName + "\n");

15. // Tạo file mới để nhận dữ liệu

16. FileOutputStream fos = new FileOutputStream("D:\\App\\MusicUDP\_VanManh\\Client\\Recevie\_Music.mp3");

17. // Nhận dữ liệu nhạc từ server và ghi vào file

18. while (true) {

19. DatagramPacket receivePacket = new DatagramPacket(receiveData,receiveData.length);

20. clientSocket.receive(receivePacket);

21. fos.write(receivePacket.getData(), 0, receivePacket.getLength());

22. // Kiểm tra xem đã nhận đủ dữ liệu chưa

23. if (receivePacket.getLength() < 1024) {

24. break;

25. }

26. }

27. fos.close();

28.

29. chatTextArea.append("Bài hát " + newFileName + " đã phát thành công.\n");

30. // Phát nhạc từ tệp tin vừa nhận được

31. playAudio("D:\\App\\MusicUDP\_VanManh\\Client\\Recevie\_Music.mp3");

32. } catch (IOException e) {

33. e.printStackTrace();

34. }

35. }

36. private void playAudio(String filePath) {

37. try {

38. FileInputStream fis = new FileInputStream(filePath);

39. AdvancedPlayer player = new AdvancedPlayer(fis);

40. player.play();

41. } catch (JavaLayerException | FileNotFoundException e) {

42. e.printStackTrace();

43. }

44. }

45. public static void main(String[] args) {

46. EventQueue.invokeLater(new Runnable() {

47. public void run() {

48. Client client = new Client();

49. client.setVisible(true);

50. }

51. });

52. }

53.

* + 1. **Chương trình giao diện Java Swing.**
* **Server:**

1. public Server() {

2. setTitle("Server");

3. setSize(400, 300);

4. setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

5.

6. JPanel panel = new JPanel();

7. panel.setLayout(new BorderLayout());

8.

9. browseButton = new JButton("Browse");

10. sendButton = new JButton("Send");

11. chatTextArea = new JTextArea();

12.

13. JPanel buttonPanel = new JPanel(new GridLayout(1, 2));

14. buttonPanel.add(browseButton);

15. buttonPanel.add(sendButton);

16.

17. panel.add(buttonPanel, BorderLayout.NORTH);

18. panel.add(new JScrollPane(chatTextArea), BorderLayout.CENTER);

19. add(panel);

20. browseButton.addActionListener(new ActionListener() {

21. public void actionPerformed(ActionEvent e) {

22. browseFile();

23. }

24. });

25. sendButton.addActionListener(new ActionListener() {

26. public void actionPerformed(ActionEvent e) {

27. sendFile();

28. }

29. });

30.

31. try {

32. socket = new DatagramSocket(9876);

33. } catch (SocketException ex) {

34. ex.printStackTrace();

35. }

36. }

37.

* **Client:**

1. public class Client extends JFrame {

2. private JButton receiveButton;

3. private JTextArea chatTextArea;

4. private DatagramSocket clientSocket;

5.

6. public Client() {

7. setTitle("Client");

8. setSize(400, 300);

9. setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

10.

11. JPanel panel = new JPanel();

12. panel.setLayout(new BorderLayout());

13.

14. receiveButton = new JButton("Receive");

15. chatTextArea = new JTextArea();

16.

17. // Đặt phông chữ cho JTextArea thành Arial

18. Font font = new Font("Arial", Font.PLAIN, 12); // Thay đỏi kích thước phù hợp

19. chatTextArea.setFont(font);

20. panel.add(receiveButton, BorderLayout.NORTH);

21. panel.add(new JScrollPane(chatTextArea), BorderLayout.CENTER);

22. add(panel);

23. receiveButton.addActionListener(new ActionListener() {

24. public void actionPerformed(ActionEvent e) {

25. receiveFile();

26. }

27. });

28. try {

29. clientSocket = new DatagramSocket();

30. } catch (SocketException ex) {

31. ex.printStackTrace();

32. }

33. }

34.

1. **Chương IV: KẾT LUẬN.**
   1. **Kết luận.**
      1. **Kết quả đạt được.**

**A screenshot of a computer

Description automatically generated**

* + 1. **Nội dung đạt được.**
* **Mô tả chương trình:**

Chương trình này sẽ cho phép một server gửi một file nhạc đến client thông qua giao thức UDP. Sau đó, client sẽ nhận file và tự động phát nhạc đó.

* **Các bước thực hiện:**
  + *Đối với server:*
* Server nhập đường dẫn đến file nhạc cần gửi.
* Gửi dữ liệu qua UDP tới client.
  + *Đối với client:*
* Nhận dữ liệu từ server thông qua giao thức UDP.
* Lưu file nhận được vào thư mục cục bộ.
* Phát file nhạc đã nhận.
* **Phân tích cụ thể:**
  + *Server:*
* Yêu cầu người dùng nhập đường dẫn của file nhạc.
* Tạo socket UDP để gửi dữ liệu.
* Đọc file nhạc và gửi dữ liệu qua socket UDP tới client.
  + *Client:*
* Tạo socket UDP để nhận dữ liệu từ server.
* Nhận dữ liệu và lưu vào một file trong thư mục cục bộ.
* Sử dụng thư viện âm thanh để phát nhạc từ file vừa nhận.

**TÀI LIỆU THAM KHẢO.**

**[1].** Hướng dẫn thêm thư viện Java vào các dự án sử dụng IDE Eclipse(2020), Nguyễn Nghĩa.

**[2].** Truyền đối tượng bằng UDP (2024), Phạm Văn Tiệp.

**[3].** UDP Socket (2024), Phạm Văn Tiệp.

**[4].** Lập trình với giao thức UDP (2024), Phạm Văn Tiệp. **[5].** Eclipse (2015), Nguyễn Thị Hồng Nhung.

**[6].** Mô hình Client – Server (2021), Hưng Nguyễn.

**[7].** Giao thức UDP (2022), Trịnh Duy Thanh.