***MENU***

[***1. Tổng quan*** 2](#_Toc494723668)

[***1.1 Topic File Transfer Between Computers và nhiệm vụ*** 2](#_Toc494723669)

[***1.2 Tiến độ thực hiện*** 2](#_Toc494723670)

[***1.3. Bảng phân công công việc*** 3](#_Toc494723671)

[***2. Nội dung chi tiết đồ án phát triển ứng dụng*** 5](#_Toc494723672)

[***2.1 Tìm hiểu về mạng máy tính*** 5](#_Toc494723673)

[***2.1.1 Định nghĩa mạng máy tính*** 5](#_Toc494723674)

[***2.1.2 Các mô hình hoạt động của mạng máy tính*** 5](#_Toc494723675)

[***2.1.3 Giao thức mạng*** 6](#_Toc494723676)

[***2.2 Tìm hiểu về Java cơ bản*** 7](#_Toc494723677)

[***2.2.1 Khái niệm về hướng đối tượng*** 7](#_Toc494723678)

[***2.2.2 Đa luồng*** 8](#_Toc494723679)

[***2.2.3 Socket trong Java*** 8](#_Toc494723680)

[***2.3 Xây dựng ứng dụng*** 9](#_Toc494723681)

[***2.3.1 Giới thiệu*** 9](#_Toc494723682)

[***2.3.2 Phân tích chương trình*** 10](#_Toc494723683)

[***2.3.3 Cơ chế hoạt động của chương trình*** 11](#_Toc494723684)

[***2.3.4 Nhận xét chung*** 11](#_Toc494723685)

[***3. Kết Luận*** 12](#_Toc494723686)

[***3.1 Những công việc đã hoàn thành*** 12](#_Toc494723687)

[***3.2 Những khó khăn đã và đang gặp phải*** 12](#_Toc494723688)

[***Tài liệu tham khảo:*** 13](#_Toc494723689)

[***Source Code:*** 13](#_Toc494723690)

# ***1. Tổng quan***

## ***1.1 Topic File Transfer Between Computers và nhiệm vụ***

1. Tìm hiểu về mạng máy tính là gì? Các mô hình hoạt động phổ biến của mạng máy tính và một số giao thức kết nối của chúng.

2. Nghiên cứu về lập trình hướng đối tượng, tìm hiểu về tiến trình và đa tiến trình cũng như tệp và cách thức làm việc với giao thức Socket UDP tất cả trên nền Java.

3. Viết chương trình kết nối đơn giãn và gửi một đoạn tin ngắn gửi qua lại giữa Client với Server.

4. Viết một chương trình đọc và ghi file đơn giãn.

5. Viết chương trình kết nối và gửi file từ Server tới Client.

## ***1.2 Tiến độ thực hiện***

|  |  |  |  |
| --- | --- | --- | --- |
| Tuần | Nội dung công việc | Tiến độ thực hiện | Khó khăn gặp phải |
| 01 | Nghiên cứu lại kiến thức Java cơ bản về phương pháp lập trình OOP và lập kế hoạch làm việc trong 10 tuần. | Đã hoàn thành. |  |
| 02 | Tìm hiểu về tiến trình và đa tiến trình cũng như cách làm việc với tệp và giao diện trong Java. | Đã hoàn thành. | Thiếu tài liệu trong quá trình nghiên cứu. |
| 03 | - Nghiên cứu về mô hình Client/Server và phương thức kết nối trên Java dựa trên giao thức Socket UDP.  - Viết một chương trình đơn giãn thực hiện kết nối dựa trên giao thức Socket UDP  (<http://www.cs.uic.edu/~troy/spring05/cs450/sockets/socket.html>) | - Đã nghiên cứu một cách cơ bản về giao thức Socket trong Java.  - Đã viết một chương trình đơn giản Client/Server kết nối bằng Socket UDP dựa trên tư liệu của thầy. | Phòng máy không có mạng để kết nối giữa 2 máy tính thực thi code. |
| 04 | Viết một chương trình đọc và ghi file đơn giãn. | Đã hoàn thành và có giao diện đơn giản để sử dụng. | Chưa khắc phục được lỗi mất nội dung khi send file(.txt) dung lượng 1kb. |
| 05 | Sử dụng Thread vào chương trình đọc và send file trong Java đã viết ở tuần 04. | Đã cập nhật chương trình đọc và ghi file đơn giãn có sử dụng thread. |  |
| 06 | Viết chương trình kết nối và gửi file từ Server tới Client. | Gần hoàn thành. | - Lỗi chưa thể dừng nhận dữ liệu gửi từ Server.  - Lỗi gửi file thiếu ổn định. Lúc nhận hết file lúc nhận thiếu. |

## ***1.3. Bảng phân công công việc***

|  |  |  |
| --- | --- | --- |
| Tuần | Công việc | Phân công |
| 1 | Nghiên cứu lại kiến thức Java cơ bản về phương pháp lập trình OOP và lập kế hoạch làm việc trong 10 tuần. | Làm chung. |
| 2 | Tìm hiểu về tiến trình và đa tiến trình cũng như cách làm việc với tệp và giao diện trong Java. | Làm chung |
| 3 | Nghiên cứu về mô hình Client/Server. | Phạm Thế Anh |
| Tìm hiểu phương thức kết nối trên Java dựa trên giao thức Socket UDP. | Trần A Sừng |
| Viết một chương trình đơn giãn thực hiện kết nối dựa trên giao thức Socket UDP. | Nguyễn Thành Trung |
| 4 | Viết một chương trình đọc và ghi file đơn giãn. |  |
| Kiểm tra đánh giá và sửa lỗi. | Phạm Thế Anh |
| Trần A Sừng |
| 5 | Sử dụng Thread vào chương trình đọc và send file trong Java đã viết ở tuần 04. | Làm chung |
| 6 | Viết chương trình Server kết nối và gửi file tới Client. | Nguyễn Thành Trung |
| Viết chương trình Client kết nối và gửi file tới Server. | Phạm Thế Anh |
| Thực thi đánh giá kiểm lỗi. | Trần A Sừng |

# ***2. Nội dung chi tiết đồ án phát triển ứng dụng***

## ***2.1 Tìm hiểu về mạng máy tính***

### ***2.1.1 Định nghĩa mạng máy tính***

Mạng máy tính là một hệ thống gồm nhiều máy tính và các thiết bị được kết nối với nhau bởi đường truyền vật lý theo một kiến trúc (Network Architecture) nào đó nhằm thu thập, trao đổi dữ liệu và chia sẽ tài nguyên cho nhiều người sử dụng.

Các máy tính được kết nối với nhau có thể trong cùng một phòng, một tòa nhà, một thành phố hoặc trên phạm vi toàn cầu.

Mạng máy tính bao gồm ba thành phần chính:

-Các máy tính;

-Các thiết bị mạng đảm bảo kết nối các máy tính với nhau;

-Phần mềm cho phép thực hiện việc trao đổi thông tin giữa các máy tính.

### ***2.1.2 Các mô hình hoạt động của mạng máy tính***

#### **2.1.4.1 Mô hình peer to peer**

Trong mô hình này, tất cả các máy tính tham gia đều có vai trò giống nhau. Mỗi máy vừa có thể cung cấp trực tiếp tài nguyên của mình cho các máy khác, vừa có thể sử dụng trực tiếp tài nguyên của các máy khác trong mạng. Mô hình này chỉ thích hợp với mạng có quy mô nhỏ, tài nguyên được quản lý phân tán, chế độ bảo mật kém.

#### **2.1.5.2 Mô hình Clients/Server**

Trong mô hình này, một hoặc vài máy sẽ được chọn để đảm nhận việc quản lý và cung cấp tài nguyên (chương trình, dữ liệu, thiết bị,…) được gọi là máy chủ (Server), các máy khác sử dụng tài nguyên này được gọi là máy khách (Clients).

Máy chủ là máy tính đảm bảo việc phục vụ các máy khách bằng cách điều khiển việc phân phối tài nguyên nằm trong mạng với mục đích sử dụng chung. Máy khách là máy sử dụng tài nguyên do máy chủ cung cấp.

Mô hình khách – chủ có ưu điểm là dữ liệu được quản lý tập trung, bảo mật tốt, thích hợp với các mạng trung bình và lớn.

### ***2.1.3 Giao thức mạng***

Tập hợp tất cả các quy tắc, quy ước để đảm bảo cho các máy tính trên mạng có thể giao tiếp với nhau gọi là giao thức. Như vậy các máy trên mạng muốn giao tiếp với nhau thì phải có chung một giao thức.

Một máy tính tương tác với thế giới thông qua một hoặc nhiều ứng dụng. Những ứng dụng này thực hiện các nhiệm vụ cụ thể và quản lý dữ liệu ra và vào. Nếu máy tính đó là một phần của hệ thống mạng, thì một trong số các ứng dụng trên sẽ có thể giao tiếp với các ứng dụng trên các máy tính khác thuộc cùng hệ thống mạng. Bộ giao thức mạng là một hệ thống các quy định chung giúp xác định quá trình truyền dữ liệu phức tạp. Dữ liệu đi từ ứng dụng trên máy này, qua phần cứng mạng của máy, tới bộ phận trung gian và đến nơi nhận, thông qua phần cứng của máy tính đích rồi tới ứng dụng.

#### **2.1.3.1 Giao Thức TCP**

Giao thức TCP (Transmission Control Protocol – “Giao thức điều khiển truyền vận“) 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.

#### **2.1.3.2 Giao thức UDP**

UDP (User Datagram Protocol) là một trong những giao thức cốt lõi của giao thức TCP/IP. Dùng UDP, chương trình trên mạng máy tính có thể gửi những dữ liệu ngắn được gọi là datagram tới máy khác. UDP không cung cấp sự tin cậy và thứ tự truyền nhận mà TCP làm; các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo. 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.

Những ứng dụng phổ biến sử dụng UDP như DNS (Domain Name System), ứng dụng streaming media, Voice over IP, Trivial File Transfer Protocol (TFTP), và game trực tuyến.

## ***2.2 Tìm hiểu về Java cơ bản***

### ***2.2.1 Khái niệm về hướng đối tượng***

**Object** (đối tượng) nghĩa là một thực thể trong thế giới thực, chẳng hạn như bàn, quả bóng, con bò, … Lập trình hướng đối tượng là một phương pháp để thiết kế một chương trình bởi sử dụng các lớp và các đối tượng. Nó làm đơn giản hóa việc duy trì và phát triển phần mềm bằng việc cung cấp một số khái niệm:

**Đối tượng**: Một thực thể có trạng thái và hành vi. Ví dụ như xe đạp, bàn, ghế, … Nó có thể mang tính vật lý hoặc logic.

**Lớp**: Một tập hợp các đối tượng. Nó là một thực thể logic.

**Tính kế thừa**: Khi một đối tượng đạt được các thuộc tính và các hành vi của đối tượng cha, thì đó là tính kế thừa. Điều này làm tăng tính tái sử dụng cho code. Nó được sử dụng để đạt được tính đa hình tại runtime.

**Tính đa hình**: Khi một tác vụ được thực hiện theo nhiều cách khác nhau được gọi là tính đa hình. Ví dụ: như vẽ hình chữ nhật hoặc hình tam giác, … Trong Java, chúng ta sử dụng nạp chồng phương thức (method overloading) và ghi đè phương thức (method overriding) để có tính đa hình. Một ví dụ khác: con mèo kêu meooo, còn chú chó thì sủa goooo.

**Tính trừu tượng**: Đó là ẩn các chi tiết nội tại và hiển thị tính năng. Ví dụ, với cuộc gọi điện thoại, chúng ta không biết tiến trình xử lý nội tại là như thế nào. Trong Java, chúng là sử dụng lớp abstract và abstract interface để có tính trừu tượng.

**Tính bao đóng**: Đó là gắn kết code và dữ liệu cùng với nhau vào trong một đơn vị unit đơn. Ví dụ: có thể bạn đã biết đến viên thuốc con nhộng (hay đơn giản hơn là gói bột giặt), các viên thuốc (hạt bột giặt) khác nhau được đóng gói.

Một lớp Java là một ví dụ về tính bao đóng. Java Bean là lớp bao đóng thực sự, vì tất cả thành viên dữ liệu là private.

### ***2.2.2 Đa luồng***

**Thread** là đơn vị nhỏ nhất của mã thực thi mà đoạn mã đó thực hiện một nhiệm vụ cụ thể.

Một ứng dụng có thể được chia nhỏ thành nhiều nhiệm vụ và mỗi nhiệm vụ có thể được giao cho một thread.

Nhiều thread cùng thực hiện đồng thời được gọi là đa luồng (multithread). Các quá trình đang chạy dường như là đồng thời, nhưng thực ra nó không phải là như vậy.

### ***2.2.3 Socket trong Java***

Java hỗ trợ lập trình mạng thông qua các lớp trong gói java.net. Một số lớp tiêu biểu được dùng cho lập trình Client-Server sử dụng socket làm phương tiện giao tiếp như:

InetAddress: Lớp này quản lý địa chỉ Internet bao gồm địa chỉ IP và tên máy tính.

Socket: Hỗ trợ các phương thức liên quan đến Socket cho chương trình Client ở chế độ có nối kết.

ServerSocket: Hỗ trợ các phương thức liên quan đến Socket cho chương trình Server ở chế độ có nối kết.

DatagramSocket: Hỗ trợ các phương thức liên quan đến Socket ở chế độ không nối kết cho cả Client và Server.

DatagramPacket: Lớp cài đặt gói tin dạng thư tín người dùng (Datagram Packet) trong giao tiếp giữa Client và Server ở chế độ không nối kết.

## ***2.3 Xây dựng ứng dụng***

### ***2.3.1 Giới thiệu***

Truyền tải file giữa hai hoặc nhiều máy tính thực chất là truyền dẫn dữ liệu giữa các máy tính dựa trên mô hình kết nối Clients/Server. Một chương trình là Client và một chương trình là Server. Có nhiều cách để xây dựng chương trình ứng dụng mạng nhưng phổ biến hơn hết là sử dụng socket. Và chương trình được xây dựng dưới đây sử dụng giao thức socket UDP dựa trên ngôn ngữ Java.

### ***2.3.2 Phân tích chương trình***

- Client:

Client mở cổng port

tạo socket kết nối đến Server

Không

Báo lỗi

Kết nối thành công

Có

Đợi Server chấp nhận

kết nối và nhận file gửi về

Kết thúc

- Server:

Server tạo socket

đợi kết nối đến từ Client

Không

Thông báo

Chấp nhận kết nối từ Client

Có

Tìm file mà client muốn nhận và tiến hành thao tác gửi

Kết thúc

### ***2.3.3 Cơ chế hoạt động của chương trình***

Như đã nói ở phần giới thiệu thì chương trình này được viết trên ngôn ngữ Java và sử dụng giao thức socket UDP để tạo kết nối giữa Client với Server.

Đầu tiên ta cho chương trình Client chạy mở cổng port tạo socket kết nối tới Server và đợi Server chấp nhận. Tiếp đến cho chạy chương trình Server tạo socket với cổng port mà chương trình Client đã mở sau đó chấp nhận kết nối nếu hợp lệ. Sau khi kết nối thành công đến Client lúc này chương trình Server bắt đầu tìm file muốn gửi sang Client. Nếu file không tồn tại thì kết thúc chương trình và đóng kết nối, nếu file tồn tại thì tiến hành gửi dữ liệu sang Client (dữ liệu ở đây là chương trình Server đọc file theo từng kilobyte và gửi đi). Lúc này ở chương trình Client bắt đầu nhận dữ liệu và ghi vào ổ cứng (dữ liệu theo từng kilobyte nhận được từ Server) sau khi nhận xong dữ liệu từ Server thì chương trình Client tiến hành đóng các kết nối và kết thúc chương trình. Ở phía Server cũng vậy, sau khi gửi file thành công tiến hành đóng các kết nối và kết thúc chương trình.

### ***2.3.4 Nhận xét chung***

Nhìn chung thì nhóm làm việc tương đối đúng tiến độ, chương trình kết nối tốt, phía chương trình Server đã hoàn thành nhiệm vụ gửi file. Nhưng khi đến chương trình Client thì lại mắc phải lỗi không thể xác nhận thời điểm đã nhận file hoàn tất để tiến hành đóng kết nối và kết thúc chương trình. Thêm vào đó mặc dù kết nối giữa hai bên tốt nhưng khi gửi file thì Client nhận file khá là thiếu ổn định, đôi lúc nhận hoàn toàn file còn đôi lúc lại nhận thiếu dữ liệu.

# ***3. Kết Luận***

## ***3.1 Những công việc đã hoàn thành***

1. Hoàn thành tìm hiểu về mạng máy tính là gì? Các mô hình hoạt động phổ biến của mạng máy tính và một số giao thức kết nối của chúng.

2. Hoàn thành nghiên cứu về lập trình hướng đối tượng, tìm hiểu về tiến trình và đa tiến trình cũng như tệp và cách thức làm việc với giao thức Socket UDP tất cả trên nền Java.

3. Viết thành công chương trình kết nối đơn giãn và gửi một đoạn tin ngắn gửi qua lại giữa Client với Server.

4. Viết thành công chương trình đọc và ghi file đơn giãn.

5. Viết gần hoàn thành chương trình kết nối và gửi file từ Server tới Client.

## ***3.2 Những khó khăn đã và đang gặp phải***

1. Như đã nói ở trên phần nhận xét chung thì lỗi không thể dừng nhận dữ liệu từ Server của chương trình Client vẫn chưa thể khắc phục (phỏng đoán có thể do sai điều kiện dừng ở vòng lặp nhận dữ liệu)

2. Cũng đã nói ở trên. Việc nhận file của Client khá là không ổn định. Có lúc gửi file video nhận thành công và chạy thành công nhưng có lúc lại không thể chạy được cũng như dung lượng nhỏ hơn file gốc,...

3. Phòng máy khi thực hành không có kết nối mạng nên bất tiện cho việc kiểm thử và sửa lỗi.

4. Nếu như chưa thể khắc phục được vấn đề trên thì có thể tuần kế tiếp sẽ bị chậm tiến độ công việc.

Nhóm vẫn đang tích cực giải quyết lỗi và sẽ hỏi giảng viên hướng dẫn ở tuần tiếp để khắc phục thành công và hoàn chỉnh được chương trình.

# ***Tài liệu tham khảo:***

[***http://www.oktot.com/mang-may-tinh-la-gi/***](http://www.oktot.com/mang-may-tinh-la-gi/)

[***http://it.die.vn/t/tcp-transmission-control-protocol/***](http://it.die.vn/t/tcp-transmission-control-protocol/)

[***http://it.die.vn/u/udp-user-datagram-protocol/***](http://it.die.vn/u/udp-user-datagram-protocol/)

[***http://vietjack.com/java/khai\_niem\_huong\_doi\_tuong\_trong\_java.jsp***](http://vietjack.com/java/khai_niem_huong_doi_tuong_trong_java.jsp)

[***http://voer.edu.vn/c/socket-duoi-ngon-ngu-java/761b0302/4f8ba8e9***](http://voer.edu.vn/c/socket-duoi-ngon-ngu-java/761b0302/4f8ba8e9)

# ***Source Code:***

- Client:

**package** fileTransfer;

**import** java.io.\*;

**import** java.net.\*;

**public** **class** MyClient {

**public** **static** **void** main(String[] args) **throws** Exception {

String t = **new** String();

DatagramSocket clientSocket = **new** DatagramSocket(7777);

**byte**[] arr = **new** **byte**[1024];

File destFile = **new** File("G:\\file\\input.mp4");

FileOutputStream fos = **new** FileOutputStream(destFile, **true**);

**long** i = 0;

DatagramPacket receivePacket;

System.***out***.println("Begin");

**try** {

**do** {

receivePacket = **new** DatagramPacket(arr, 0, arr.length);

clientSocket.receive(receivePacket);

fos.write(arr);

i++;

System.***out***.println("# "+i);

} **while** (arr.length == 1024);

System.***out***.println("finish");

fos.close();

clientSocket.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

- Server:

**package** fileTransfer;

**import** java.io.\*;

**import** java.net.\*;

**public** **class** Server {

**public** **static** **void** main(String[] args) **throws** Exception {

String serverHostname = **new** String ("localhost");

InetAddress IPAddress = InetAddress.*getByName*(serverHostname);

DatagramSocket serverSocket = **new** DatagramSocket();

**byte**[] arr = **new** **byte**[1024];

DatagramPacket sendPacket;

File sourceFile = **new** File("G:\\file\\output.mp4");

FileInputStream fis = **new** FileInputStream(sourceFile);

System.***out***.println("Begin");

**long** i = 0;

**if** (sourceFile.exists()) {

**try** {

**while** (fis.read(arr) != -1) {

sendPacket = **new** DatagramPacket(arr, arr.length, IPAddress, 7777);

serverSocket.send(sendPacket);

i++;

System.***out***.println("# "+i);

}

fis.close();

serverSocket.close();

System.***out***.println("finish");

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

}