**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH**

****

**BÁO CÁO ĐỒ ÁN**

**Chuyên đề:**

**File transfer between computer**

Giáo viên hướng dẫn: Huỳnh Thái Học

Võ Công Minh

Sinh viên thực hiện: Phan Hữu Thiện 15066181

Đỗ Quang Vinh 15058091

**Nhận xét của giáo viên hướng dẫn:**

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………............

Contents

[**Chương 1: Giới thiệu** 4](#_Toc494745237)

[**1. File Transfer between computer là gì? Tại sao lại chọn đề tài ?** 4](#_Toc494745238)

[**2. Các giao thức dùng để truyền:** 4](#_Toc494745239)

[**POP3** 4](#_Toc494745240)

[**Chương 2: Phân Tích Lý thuyết** 6](#_Toc494745241)

[**1.** **Mạng máy tính:** 6](#_Toc494745242)

[**2.** **Ngôn ngữ lập trình Java** 7](#_Toc494745243)

[**3.Socket** 7](#_Toc494745244)

[3.1 Định nghĩa: 7](#_Toc494745245)

[3.2 Socket trong java: 8](#_Toc494745246)

[3.3 Một số lớp trong lập trình java socket 8](#_Toc494745247)

[**Chương 3:Mô hình** 10](#_Toc494745248)

[**1.** **Sơ đồ tương tác giữa Client và Server theo giao thức UDP như mô hình sau:** 10](#_Toc494745249)

[**2.Phân tích chương trình:** 10](#_Toc494745250)

[2.1. Mô đun phía server 10](#_Toc494745251)

[2.2. Mô đun phía client 11](#_Toc494745252)

[**3. Chương trình** 12](#_Toc494745253)

[3.1. Chương trình bên Client: 12](#_Toc494745254)

[3.2. Chương trình bên Server 17](#_Toc494745255)

[**6. Tài liệu tham khảo** 22](#_Toc494745256)

# **Chương 1: Giới thiệu**

## **1. File Transfer between computer là gì? Tại sao lại chọn đề tài ?**

File Transfer between computer là truyền một tập tin từ máy này sang máy khác bằng các giao thức và dựa trên các mô hình mạng.

Chọn đề tài vì nhóm chúng em muốn tìm hiểu về cách truyền và nhận 1 tập tin từ máy này sang máy khác, cách mà tập tin nó xử lý rồi gởi qua máy khác.

## **2. Các giao thức dùng để truyền:**

**TCP** ( Transmission Control Protocol ) giao thức ở tầng giao vận cũng giống UDP sử dụng giao thức IP trong Tầng mạng. Nhưng khác với UDP, TCP cung cấp dịch vụ liên kết tin cậy vì thế TCP hỗ trợ nhiều giao thức phổ biến nhất trên internet và các ứng dụng trong đó các dịch vụ web như WWW, thư điện tử email, SSH

**UDP** (User Datagram Protocol) là một trong những giao thức mạng máy tính của TCP/IP. UDP được sử dụng để gởi các datagream( gói tin) đến các máy tính trong hệ thống).

UDP gửi nhanh và hiệu quả hơn đối với các gói tin kích thước nhỏ và có yêu cầu khắt khe về thời gian .

UDP thường sử dụng trong các ứng dụng như : DNS ( Domain Name System), ứng dụng Streaming media , Voice Over IP , Trivial File Transfer Protocol (TFTP), game online.

**DHCP** ( Dynamic Host Configuration Protocol ) là giao thức được cài đặt và cấu hình trên cả Client và Server . Một máy chủ quản lý và cấp phát địa chỉ IP,được sử dụng để quản lý danh sách các địa chỉ IP , đồng thời nó luôn ở trạng thái lắng nghe các kết nối từ DHCP Client. Khi có một yêu cầu từ phía DHCP Client , nó sẽ kiểm tra trong cơ sở dữ liệu của nó để biết các địa chỉ IP nào chưa được cấp phát và cấp cho DHCP Client một địa chỉ .DHCP Client được setup trên máy client , nó có chức năng gửi yêu cầu đến DHCP Server và nhận địa chỉ được cấp về để sử dụng . DHCP được triền khai trên Windows Server (DHCP Server) và Linux Server (dhcpd.conf).

**HTTP** ( Hyper Text Transfer Protocol ) giao thức truyền tin siêu văn bản, Được dùng để liên hệ thông tin giữ web Server và trình duyệt web ( Web Browser ). Trình duyệt web và Web Server liên hệ với nhau thông qua cổng mặc định 80

**POP3** ( Post Office Protocol Version 3 ): là một giao thức dùng để tải thư điện tử từ Mail Server về máy tính , thông qua kết nối TCP/IP. Trình duyệt mail Client sẽ kết nối đến Mail Server để nhận thư thông qua cổng mặc định 110 của giao thức POP3

**SMTP** ( Simple Mail Transfer Protocol ) giao thức truyền thư đơn giản , được sử dụng để truyền nội dung thư điện tử từ Mail Server này đến Mail Server khác . Nó thực hiện nhiệm vụ truyền thư giữa các Mail Server thông qua cổng mặc định 25.

**DCHP (**Dynamic Host Configuration Protocol**)** là một giao thức cho phép cấp phát địa chỉ IP một cách tự động cùng với các cấu hình liên quan khác như [subnet mark](https://vi.wikipedia.org/wiki/Subnet_mask" \o "Subnet mask) và [gateway](https://vi.wikipedia.org/wiki/Gateway_(truy%E1%BB%81n_th%C3%B4ng)) mặc định. Máy tính được cấu hình một cách tự động vì thế sẽ giảm việc can thiệp vào hệ thống mạng. Nó cung cấp một database trung tâm để theo dõi tất cả các máy tính trong hệ thống mạng. Mục đích quan trọng nhất là tránh trường hợp hai máy tính khác nhau lại có cùng địa chỉ IP.

# **Chương 2: Phân Tích Lý thuyết**

## **Mạng máy tính:**

* Mạng máy tính : là một tập hợp các máy tính được nối với nhau bởi đường truyền theo một cấu trúc nào đó và thông qua đó các máy tính trao đổi thông tin qua lại cho nhau.
* Nhu cầu phát triển: Mạng máy tính là một tập hợp các máy tính được nối với nhau bởi đường truyền theo một cấu trúc nào đó và thông qua đó các máy tính trao đổi thông tin qua lại cho nhau.
* Giao thức mạng: là một tập các quy tắc, quy ước để trao đổi thông tin giữa hai hệ thống máy tính hoặc hai thiết bị máy tính với nhau. Nói một cách hình thức thì giao thức mạng là một ngôn ngữ được các máy tính trong mạng sử dụng để trao đổi dữ liệu với nhau. Có nhiều loại giao thức được sử dụng trong mạng máy tính như: Apple Talk, DLC, NetBEUI,… nhưng hiện nay giao thức được sử dụng phổ biến nhất trong mạng máy tính là giao thức TCP/IP.
* Các mô hình hoạt động mạng:

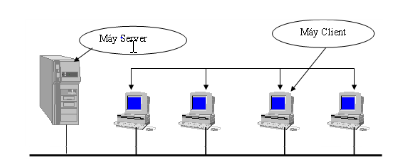
Mô hình peer to peer:

Không tồn tại bất kỳ máy chuyên dụng hoặc cấu trúc phân cấp giữa các máy tính. Mọi máy tính đều bình đẳng và có vai trò như nhau. Thông thường mỗi máy tính hoạt động với cả vai trò máy khách và máy phục vụ. Vì vậy không có máy nào được chỉ định quản lý toàn mạng. Người dùng ở từng máy tự quyết định dữ liệu nào trên máy của mình sẽ được chia sẻ để dùng chung trên mạng.



Mô hình client/server:

Là một mô hình nổi tiếng trong [mạng máy tính](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_m%C3%A1y_t%C3%ADnh" \o "Mạng máy tính), được áp dụng rất rộng rãi và là mô hình của mọi trang web hiện có. Ý tưởng của mô hình này là máy con (đóng vài trò là máy khách) gửi một yêu cầu (request) để máy chủ (đóng vai trò người cung ứng dịch vụ), máy chủ sẽ xử lý và trả kết quả về cho máy khách.



## **Ngôn ngữ lập trình Java**

* 1. Java là một ngôn ngữ lập trình được Sun Microsystems giới thiệu vào tháng 6 năm 1995. Từ đó, nó đã trở thành một công cụ lập trình của các lập trình viên chuyên nghiệp. Java được xây dựng trên nền tảng của C và C++. Do vậy nó sử dụng các cú pháp của C và các đặc trưng hướng đối tượng của C++. Vào năm 1991, một nhóm các kỹ sư của Sun Microsystems có ý định thiết kế một ngôn ngữ lập trình để điều khiển các thiết bị điện tử như Tivi, máy giặt, lò nướng, … Mặc dù C và C++ có khả năng làm việc này nhưng trình biên dịch lại phụ thuộc vào từng loại CPU. Trình biên dịch thường phải tốn nhiều thời gian để xây dựng nên rất đắt. Vì vậy để mỗi loại CPU có một trình biên dịch riêng là rất tốn kém. Do đó nhu cầu thực tế đòi hỏi một ngôn ngữ chạy nhanh, gọn, hiệu quả và độc lập thiết bị tức là có thể chạy trên nhiều loại CPU khác nhau, dưới các môi trường khác nhau. “Oak” đã ra đời và vào năm 1995 được đổi tên thành Java. Mặc dù mục tiêu ban đầu không phải cho Internet nhưng do đặc trưng không phụ thuộc thiết bị nên Java đã trở thành ngôn ngữ lập trình cho Internet
  2. Một số tính chất của ngôn ngữ java:

Đơn giản

Hướng đối tượng

Độc lập phần cứng và hệ điều hành

Mạnh

Bảo mật

Phân tán

Đa luồng

Động

## **3.Socket**

### **3.1 Định nghĩa:**

Có nhiều định nghĩa khác nhau về socket tùy theo cách nhìn của người sử dụng.

- Một cách tổng quát nhất có thể định nghĩa : Một Socket là một điểm cuối trong một kết nối giữa hai chương trình đang chạy trên mạng.

- Nhìn trên quan điểm của người phát triển ứng dụng người ta có thể định nghĩa Socket là một phương pháp để thiết lập kết nối truyền thông giữa một chương trình yêu cầu dịch vụ (được gán nhãn là Client) và một chương trình cung cấp dịch vụ (được gán nhãn là server) trên mạng hoặc trên cùng một máy tính.

- Đối với người lập trình, họ nhìn nhận Socket như một giao diện nằm giữa tầng ứng dụng và tầng khác trong mô hình mạng OSI có nhiệm vụ thực hiện việc giao tiếp giữa chương trình ứng dụng với các tầng bên dưới của mạng.

### **3.2 Socket trong java:**

\*Xây dựng chương trình Client ở chế độ có nối kết

\*Các bước tổng quát:

- Mở một socket nối kết đến server đã biết địa chỉ IP(hay tên miền) và số hiệu cổng.

- Lấy InputStream và OutputStream gán với socket.

- Tham khảo protocol của dịch vụ để định dạng đúng dữ liệu trao đổi với server.

-Trao đổi dữ liệu với server nhờ vào các InputStream và OutputStream.

- Đóng socket trước khi kết thúc chương trình.

\*Lớp Java.net.Socket

Lớp socket hỗ trợ các phương thức cần thiết để xây dựng các chương trình client sử dụng socket ở chế độ có nối kết. Dưới đây là một số phương thức thường dùng để xây dựng clients.

Public Socket(String HostName, int PortNumber) throws IOException Phương thức này dùng để kết nối đến một server có tên là Hostname, cổng là PortNumber. Nếu nối kết thành công, một kênh ảo sẽ được hình thành giữa client và server

HostName: Địa chỉ IP hoặc tên logic theo dạng tên miền.

PortNumber: Có giá trị từ 0....65535

Public InputStream getInputStream()

Phương thức này trả về InputStream nối với Socket. Chương trình clients dùng InputStream này để nhận dữ liệu từ server gửi về

Public OutputStream getOutputStream()

Phương thức này trả về OutputStream nối với socket. Chương trình client dùng OutputStream này để gửi dữ liệu cho server.

Public close() Phương thức này sẽ đóng socket lại, giải phóng kênh ảo, xóa nối kết giữa clients và server.

### **3.3 Một số lớp trong lập trình java socket**

Java hỗ trợ lập trình mạng thông qua các lớp trong gói java.net. Một số lớp tiêu biểu cần dùng cho lập trình clients/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 chươnng trình clients ở chế độ có kết nối.

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

DatagramSocket: Hỗ trợ các phương thức liên quan đến socket ở chế độ không kết nối cho cả clients và server.

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

# **Chương 3:Mô hình**

## **Sơ đồ tương tác giữa Client và Server theo giao thức UDP như mô hình sau:**

Create Socket

Create Socket

Tạo Socket để giao tiếp Tạo Socket lắng nge

Gắn vào địa chỉ IP và port

Bind

Send to

Reveice from

Gửi dữ liệu đến server

tham số cần truyền vào Nhận dữ liệu từ Client

là địa chỉ IP và port mà

Server đang lắng nge.

Gởi dữ liệu đến Client tham

Reveive from

Send to

Nhận dữ liệu từ Server số cần truyền vào là địa chỉ

IP và port mà Client đang sử

dụng.

Close

Close

## **2.Phân tích chương trình:**

### **2.1. Mô đun phía server**

Tạo server socket và lắng nghe kết nối từ client

Chấp nhận kết nối từ client

**sai**

Báo lỗi

**đúng**

Chọn fille cần gửi cho client

END

Gửi file cho client

Gửi file mới

đúng sai

### **2.2. Mô đun phía client**

Client tạo Socket kết nối đến Socket Server

Kết nối thành công

Sai

Báo lỗi

đúng

END

Chọn file cần gửi cho server

Gửi file cho server

Gửi file mới

đúng sai

## **3. Chương trình**

Chương trình chúng em là gửi 1 file từ 1 file từ client lên sever và lấy file từ server về client

### **3.1. Chương trình bên Client:**

3.1.1. Class FileInfo

Import java.io.Serializable

|  |
| --- |
|  |
|  |  |
|  | public class FileInfo implements Serializable { |
|  | private static final long serialVersionUID = 1L; |
|  |  |
|  | private String destinationDirectory; |
|  | private String sourceDirectory; |
|  | private String filename; |
|  | private long fileSize; |
|  | private int piecesOfFile; |
|  | private int lastByteLength; |
|  | private String status; |
|  |  |
|  | public FileInfo() { |
|  | } |
|  |  |
|  | public String getDestinationDirectory() { |
|  | return destinationDirectory; |
|  | } |
|  |  |
|  | public void setDestinationDirectory(String destinationDirectory) { |
|  | this.destinationDirectory = destinationDirectory; |
|  | } |
|  |  |
|  | public String getSourceDirectory() { |
|  | return sourceDirectory; |
|  | } |
|  |  |
|  | public void setSourceDirectory(String sourceDirectory) { |
|  | this.sourceDirectory = sourceDirectory; |
|  | } |
|  |  |
|  | public String getFilename() { |
|  | return filename; |
|  | } |
|  |  |
|  | public void setFilename(String filename) { |
|  | this.filename = filename; |
|  | } |
|  |  |
|  | public long getFileSize() { |
|  | return fileSize; |
|  | } |
|  |  |
|  | public void setFileSize(long fileSize) { |
|  | this.fileSize = fileSize; |
|  | } |
|  |  |
|  | public int getPiecesOfFile() { |
|  | return piecesOfFile; |
|  | } |
|  |  |
|  | public void setPiecesOfFile(int piecesOfFile) { |
|  | this.piecesOfFile = piecesOfFile; |
|  | } |
|  |  |
|  | public int getLastByteLength() { |
|  | return lastByteLength; |
|  | } |
|  |  |
|  | public void setLastByteLength(int lastByteLength) { |
|  | this.lastByteLength = lastByteLength; |
|  | } |
|  |  |
|  | public String getStatus() { |
|  | return status; |
|  | } |
|  |  |
|  | public void setStatus(String status) { |
|  | this.status = status; |
|  | } |
|  | } |
|  |  |
| 3.1.2.Class Client |  |
|  | import java.io.BufferedInputStream; |
|  | import java.io.ByteArrayOutputStream; |
|  | import java.io.File; |
|  | import java.io.FileInputStream; |
|  | import java.io.IOException; |
|  | import java.io.InputStream; |
|  | import java.io.ObjectOutputStream; |
|  | import java.net.DatagramPacket; |
|  | import java.net.DatagramSocket; |
|  | import java.net.InetAddress; |
|  | import java.net.SocketException; |
|  | import java.net.UnknownHostException; |
|  | import FileInfo; |
|  | public class UDPClient { |
|  | private static final int PIECES\_OF\_FILE\_SIZE = 1024 \*32; |
|  | private DatagramSocket clientSocket; |
|  |  |
|  | private int serverPort = 6677; |
|  | private String serverHost = "localhost"; |
|  |  |
|  | public static void main(String[] args) { |
|  | String sourcePath = "G:\\client\\New Folder.rar"; |
|  | String destinationDir = "G:\\server\\"; |
|  | UDPClient udpClient = new UDPClient(); |
|  | udpClient.connectServer(); |
|  | udpClient.sendFile(sourcePath, destinationDir); |
|  | } |
|  |  |
|  | private void connectServer() { |
|  | try { |
|  | clientSocket = new DatagramSocket(); |
|  | } catch (SocketException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | private void sendFile(String sourcePath, String destinationDir) { |
|  | InetAddress inetAddress; |
|  | DatagramPacket sendPacket; |
|  |  |
|  | try { |
|  | File fileSend = new File(sourcePath); |
|  | InputStream inputStream = new FileInputStream(fileSend); |
|  | BufferedInputStream bis = new BufferedInputStream(inputStream); |
|  | inetAddress = InetAddress.getByName(serverHost); |
|  | byte[] bytePart = new byte[PIECES\_OF\_FILE\_SIZE]; |
|  |  |
|  | // get file size |
|  | long fileLength = fileSend.length(); |
|  | int piecesOfFile = (int) (fileLength / PIECES\_OF\_FILE\_SIZE); |
|  | int lastByteLength = (int) (fileLength % PIECES\_OF\_FILE\_SIZE); |
|  |  |
|  | // check last bytes of file |
|  | if (lastByteLength > 0) { |
|  | piecesOfFile++; |
|  | } |
|  |  |
|  | // split file into pieces and assign to fileBytess |
|  | byte[][] fileBytess = new byte[piecesOfFile][PIECES\_OF\_FILE\_SIZE]; |
|  | int count = 0; |
|  | while (bis.read(bytePart, 0, PIECES\_OF\_FILE\_SIZE) > 0) { |
|  | fileBytess[count++] = bytePart; |
|  | bytePart = new byte[PIECES\_OF\_FILE\_SIZE]; |
|  | } |
|  |  |
|  | // read file info |
|  | FileInfo fileInfo = new FileInfo(); |
|  | fileInfo.setFilename(fileSend.getName()); |
|  | fileInfo.setFileSize(fileSend.length()); |
|  | fileInfo.setPiecesOfFile(piecesOfFile); |
|  | fileInfo.setLastByteLength(lastByteLength); |
|  | fileInfo.setDestinationDirectory(destinationDir); |
|  |  |
|  | // send file info |
|  | ByteArrayOutputStream baos = new ByteArrayOutputStream(); |
|  | ObjectOutputStream oos = new ObjectOutputStream(baos); |
|  | oos.writeObject(fileInfo); |
|  | sendPacket = new DatagramPacket(baos.toByteArray(), baos.toByteArray().length, |
|  | inetAddress, serverPort); |
|  | clientSocket.send(sendPacket); |
|  |  |
|  | // send file content |
|  | System.out.println("Sending file..."); |
|  | // send pieces of file |
|  | for (int i = 0; i < (count - 1); i++) { |
|  | sendPacket = new DatagramPacket(fileBytess[i], PIECES\_OF\_FILE\_SIZE, |
|  | inetAddress, serverPort); |
|  | clientSocket.send(sendPacket); |
|  | waitMillisecond(40); |
|  | } |
|  | // send last bytes of file |
|  | sendPacket = new DatagramPacket(fileBytess[count - 1], PIECES\_OF\_FILE\_SIZE, |
|  | inetAddress, serverPort); |
|  | clientSocket.send(sendPacket); |
|  | waitMillisecond(40); |
|  |  |
|  | // close stream |
|  | bis.close(); |
|  | } catch (UnknownHostException e) { |
|  | e.printStackTrace(); |
|  | } catch (IOException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | System.out.println("Sent."); |
|  | } |
|  |  |
|  |  |
|  | public void waitMillisecond(long millisecond) { |
|  | try { |
|  | Thread.sleep(millisecond); |
|  | } catch (InterruptedException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  | } |
|  |  |

### **3.2. Chương trình bên Server**

3.2.1. Class FileInfor

Import java.io.Serializable

|  |
| --- |
|  |
|  |
|  |  |
|  | public class FileInfo implements Serializable { |
|  | private static final long serialVersionUID = 1L; |
|  |  |
|  | private String destinationDirectory; |
|  | private String sourceDirectory; |
|  | private String filename; |
|  | private long fileSize; |
|  | private int piecesOfFile; |
|  | private int lastByteLength; |
|  | private String status; |
|  |  |
|  | public FileInfo() { |
|  | } |
|  |  |
|  | public String getDestinationDirectory() { |
|  | return destinationDirectory; |
|  | } |
|  |  |
|  | public void setDestinationDirectory(String destinationDirectory) { |
|  | this.destinationDirectory = destinationDirectory; |
|  | } |
|  |  |
|  | public String getSourceDirectory() { |
|  | return sourceDirectory; |
|  | } |
|  |  |
|  | public void setSourceDirectory(String sourceDirectory) { |
|  | this.sourceDirectory = sourceDirectory; |
|  | } |
|  |  |
|  | public String getFilename() { |
|  | return filename; |
|  | } |
|  |  |
|  | public void setFilename(String filename) { |
|  | this.filename = filename; |
|  | } |
|  |  |
|  | public long getFileSize() { |
|  | return fileSize; |
|  | } |
|  |  |
|  | public void setFileSize(long fileSize) { |
|  | this.fileSize = fileSize; |
|  | } |
|  |  |
|  | public int getPiecesOfFile() { |
|  | return piecesOfFile; |
|  | } |
|  |  |
|  | public void setPiecesOfFile(int piecesOfFile) { |
|  | this.piecesOfFile = piecesOfFile; |
|  | } |
|  |  |
|  | public int getLastByteLength() { |
|  | return lastByteLength; |
|  | } |
|  |  |
|  | public void setLastByteLength(int lastByteLength) { |
|  | this.lastByteLength = lastByteLength; |
|  | } |
|  |  |
|  | public String getStatus() { |
|  | return status; |
|  | } |
|  |  |
|  | public void setStatus(String status) { |
|  | this.status = status; |
|  | } |
|  | } |
|  |  |
| 3.2.2.Server  . | import java.io.BufferedOutputStream; |
|  | import java.io.ByteArrayInputStream; |
|  | import java.io.File; |
|  | import java.io.FileOutputStream; |
|  | import java.io.IOException; |
|  | import java.io.ObjectInputStream; |
|  | import java.net.DatagramPacket; |
|  | import java.net.DatagramSocket; |
|  | import java.net.InetAddress; |
|  | import java.net.SocketException; |
|  |  |
|  | import FileInfo; |
|  |  |
|  | public class UDPServer { |
|  | private static final int PIECES\_OF\_FILE\_SIZE = 1024 \* 32; |
|  | private DatagramSocket serverSocket; |
|  | private int port = 6677; |
|  |  |
|  |  |
|  | public static void main(String[] args) { |
|  | UDPServer udpServer = new UDPServer(); |
|  | udpServer.openServer(); |
|  | } |
|  |  |
|  |  |
|  | private void openServer() { |
|  | try { |
|  | serverSocket = new DatagramSocket(port); |
|  | System.out.println("Server is opened on port " + port); |
|  | listening(); |
|  | } catch (SocketException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  |  |
|  | private void listening() { |
|  | while (true) { |
|  | receiveFile(); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | public void receiveFile() { |
|  | byte[] receiveData = new byte[PIECES\_OF\_FILE\_SIZE]; |
|  | DatagramPacket receivePacket; |
|  |  |
|  | try { |
|  | // get file info |
|  | receivePacket = new DatagramPacket(receiveData, receiveData.length); |
|  | serverSocket.receive(receivePacket); |
|  | InetAddress inetAddress = receivePacket.getAddress(); |
|  | ByteArrayInputStream bais = new ByteArrayInputStream(receivePacket.getData()); |
|  | ObjectInputStream ois = new ObjectInputStream(bais); |
|  | FileInfo fileInfo = (FileInfo) ois.readObject(); |
|  | // show file info |
|  | if (fileInfo != null) { |
|  | System.out.println("File name: " + fileInfo.getFilename()); |
|  | System.out.println("File size: " + fileInfo.getFileSize()); |
|  | System.out.println("Pieces of file: " + fileInfo.getPiecesOfFile()); |
|  | System.out.println("Last bytes length: " + fileInfo.getLastByteLength()); |
|  | } |
|  | // get file content |
|  | System.out.println("Receiving file..."); |
|  | File fileReceive = new File(fileInfo.getDestinationDirectory() |
|  | + fileInfo.getFilename()); |
|  | BufferedOutputStream bos = new BufferedOutputStream( |
|  | new FileOutputStream(fileReceive)); |
|  |  |
|  | for (int i = 0; i < (fileInfo.getPiecesOfFile() - 1); i++) { |
|  | receivePacket = new DatagramPacket(receiveData, receiveData.length, |
|  | inetAddress, port); |
|  | serverSocket.receive(receivePacket); |
|  | bos.write(receiveData, 0, PIECES\_OF\_FILE\_SIZE); |
|  | } |
|  |  |
|  | receivePacket = new DatagramPacket(receiveData, receiveData.length, |
|  | inetAddress, port); |
|  | serverSocket.receive(receivePacket); |
|  | bos.write(receiveData, 0, fileInfo.getLastByteLength()); |
|  | bos.flush(); |
|  | System.out.println("Done!"); |
|  |  |
|  |  |
|  | bos.close(); |
|  | } catch (IOException e) { |
|  | e.printStackTrace(); |
|  | } catch (ClassNotFoundException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  | } |
|  |  |

**4. Tiến độ sản phẩm**

Hoàn thành sản phẩm: 60%

**5. Kế hoạch tiếp theo**

Nhóm em sẽ tiếp tục viết code giao diện,sửa lỗi,hoàn thiện chương trình và nộp

sản phẩm đúng thời hạn.

# **6. Tài liệu tham khảo**

1. [http://luanvan.net.vn/luan-van/do-an-tim-hieu-lap-trinh-socket-tcp-trong- java-va-ung-dung-truyen-file-qua-mang-46006/](http://luanvan.net.vn/luan-van/do-an-tim-hieu-lap-trinh-socket-tcp-trong-%20%20%20java-va-ung-dung-truyen-file-qua-mang-46006/)

2. <https://stackoverflow.com/questions/23177351/udp-client-server-file-transfer>

3.Tài khoản github: <https://github.com/phanhuuthien/DoAn>