Chuyển Socket Server từ Java sang Csharp và trả về thông điệp "Hello"

# 1. Socket Server bằng Csharp

Đây là một ứng dụng server đơn giản sử dụng C# để lắng nghe các kết nối TCP, nhận file từ client và lưu trữ vào một thư mục đã chỉ định. Server sẽ gửi phản hồi cho client sau khi nhận file thành công.

Mã nguồn dưới đây dử dụng .NET 8.0

## 1.1. Chi tiết mã nguồn (FileServer.cs)

### 1.1.1. Khai báo các biến và hàm khởi tạo

**Biến thành viên:**

* \_port: Số nguyên đại diện cho cổng mà server sẽ lắng nghe các kết nối.
* \_directoryPath: Chuỗi đại diện cho đường dẫn thư mục nơi các file sẽ được lưu trữ sau khi nhận từ client.

**Hàm khởi tạo:**

**Mục đích**: Khởi tạo một đối tượng FileServer với cổng và đường dẫn thư mục lưu trữ file.

**Tham số**:

* port: Cổng mà server sẽ lắng nghe.
* directoryPath: Đường dẫn tới thư mục lưu trữ file.

**Chức năng**:

* Gán giá trị của cổng và đường dẫn thư mục vào các biến thành viên \_port và \_directoryPath.
* Kiểm tra sự tồn tại của thư mục. Nếu thư mục không tồn tại, tạo mới thư mục và in ra thông báo.

| **private** readonly **int** \_port; **private** readonly string \_directoryPath;  **public** FileServer(**int** port, string directoryPath) {  \_port = port;  \_directoryPath = directoryPath;   *// Tạo thư mục nếu nó không tồn tại*  **if** (!Directory.Exists(directoryPath))  {  Directory.CreateDirectory(directoryPath);  Console.WriteLine($**"Thư mục {directoryPath} đã được tạo."**);  } } |
| --- |

### 1.1.2. Hàm Start

**Mục đích**: Khởi động server và lắng nghe các kết nối từ client.

**Chức năng**:

* **Khởi tạo TcpListener**: Tạo một đối tượng TcpListener để lắng nghe các kết nối trên cổng \_port.
* **Bắt đầu lắng nghe**: Gọi listener.Start() để bắt đầu lắng nghe các kết nối.
* **In thông tin server**: Lấy địa chỉ IP và cổng của server, in ra màn hình.
* **Vòng lặp chính**: Liên tục chấp nhận và xử lý các kết nối từ client.
  + **Chấp nhận kết nối**: Sử dụng listener.AcceptTcpClient() để chấp nhận một kết nối từ client.
  + **Nhận và lưu file**:
    - Lấy NetworkStream từ đối tượng TcpClient.
    - Sử dụng StreamReader để đọc tên file từ client.
    - Kiểm tra tính hợp lệ của tên file.
    - Tạo đường dẫn đầy đủ cho file và tạo file mới bằng FileStream.
    - Đọc dữ liệu từ stream và ghi vào file.
  + **Gửi phản hồi**: Sau khi nhận file thành công, gửi thông báo phản hồi cho client.

| **public** **void** Start() {  **try**  {  TcpListener listener = **new** TcpListener(IPAddress.Any, \_port);  listener.Start();   IPEndPoint localEndPoint = listener.LocalEndpoint as IPEndPoint;  **if** (localEndPoint != **null**)  {  IPAddress ipv4Address = GetLocalIPv4Address();  Console.WriteLine($**"Server đang lắng nghe trên IP: {ipv4Address} và cổng: {localEndPoint.Port}"**);  }   **while** (**true**)  {  using (TcpClient client = listener.AcceptTcpClient())  {  IPEndPoint remoteEndpoint = client.Client.RemoteEndPoint as IPEndPoint;  **if** (remoteEndpoint != **null**)  {  Console.WriteLine($**"Đã kết nối với client: {remoteEndpoint.Address}:{remoteEndpoint.Port}"**);  }   using (NetworkStream stream = client.GetStream())  using (StreamReader reader = **new** StreamReader(stream))  {  string fileName = reader.ReadLine();  **if** (string.IsNullOrEmpty(fileName))  {  Console.WriteLine(**"Tên file không hợp lệ."**);  **continue**;  }  string filePath = Path.Combine(\_directoryPath, fileName);   using (FileStream fileStream = File.Create(filePath))  {  **byte**[] buffer = **new** **byte**[4096];  **int** bytesRead;  **while** ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)  {  fileStream.Write(buffer, 0, bytesRead);  }  }   Console.WriteLine($**"File {fileName} đã được nhận và lưu vào thư mục {\_directoryPath}"**);   **byte**[] response = Encoding.UTF8.GetBytes(**"Hello, File received successfully!\n"**);  stream.Write(response, 0, response.Length);  stream.Flush();  }  }  }  }  **catch** (Exception ex)  {  Console.WriteLine(**"Lỗi khi khởi động server: "** + ex.ToString());  } } |
| --- |

#### 1.1.2.1. IPEndPoint localEndPoint = listener.LocalEndpoint as IPEndPoint;

**Ý nghĩa của dòng mã:**

* **TcpListener**: Đối tượng TcpListener được sử dụng để lắng nghe các kết nối TCP từ client. Nó hoạt động trên một cổng cụ thể trên máy chủ.
* **LocalEndpoint**: Thuộc tính LocalEndpoint của TcpListener trả về một EndPoint đại diện cho địa chỉ IP và cổng mà TcpListener đang lắng nghe.
* **as IPEndPoint**: Toán tử as trong C# được sử dụng để thực hiện chuyển đổi kiểu một cách an toàn. Nếu chuyển đổi không thành công, nó sẽ trả về null thay vì ném ra một ngoại lệ. Trong trường hợp này, chúng ta đang cố gắng chuyển đổi EndPoint thành IPEndPoint.

**Phân tích chi tiết:**

1. **TcpListener listener**:
   * Bạn đã tạo một đối tượng TcpListener để lắng nghe các kết nối từ client trên một cổng cụ thể.
2. **listener.LocalEndpoint**:
   * Thuộc tính LocalEndpoint của đối tượng TcpListener trả về một EndPoint đại diện cho địa chỉ IP và cổng mà server đang lắng nghe. EndPoint là lớp cơ sở trừu tượng cho các đối tượng chứa thông tin địa chỉ mạng.
3. **as IPEndPoint**:
   * EndPoint là một lớp cơ sở, và IPEndPoint là một lớp con của nó chứa thông tin cụ thể về địa chỉ IP và số cổng. Bằng cách sử dụng as IPEndPoint, bạn cố gắng chuyển đổi EndPoint thành IPEndPoint.
   * Nếu listener.LocalEndpoint thực sự là một đối tượng IPEndPoint, thì localEndPoint sẽ được gán đối tượng này. Nếu không, localEndPoint sẽ là null.

**Tại sao lại cần chuyển đổi sang IPEndPoint?**

* **Lấy thông tin cụ thể về IP và cổng**: IPEndPoint cung cấp thông tin chi tiết về địa chỉ IP và cổng, trong khi EndPoint không cung cấp các thông tin chi tiết này.
* **Sử dụng các thuộc tính cụ thể**: Khi bạn chuyển đổi thành IPEndPoint, bạn có thể truy cập các thuộc tính như Address và Port để biết chính xác địa chỉ IP và cổng mà server đang lắng nghe.

#### 1.1.2.2. IPEndPoint remoteEndpoint = client.Client.RemoteEndPoint as IPEndPoint;

Dòng mã IPEndPoint remoteEndpoint = client.Client.RemoteEndPoint as IPEndPoint; đóng vai trò quan trọng trong việc lấy thông tin chi tiết về client đang kết nối đến server. Hãy cùng phân tích chi tiết dòng mã này.

**Ý nghĩa của dòng mã:**

* **TcpClient**: Đối tượng TcpClient đại diện cho kết nối TCP tới server từ client.
* **Client**: Thuộc tính Client của TcpClient trả về một đối tượng Socket mà qua đó dữ liệu được truyền tải.
* **RemoteEndPoint**: Thuộc tính RemoteEndPoint của đối tượng Socket trả về một EndPoint đại diện cho địa chỉ IP và cổng của client đang kết nối đến server.
* **as IPEndPoint**: Toán tử as trong C# được sử dụng để thực hiện chuyển đổi kiểu một cách an toàn. Nếu chuyển đổi không thành công, nó sẽ trả về null thay vì ném ra một ngoại lệ.

**Phân tích chi tiết:**

1. **client.Client**:
   * client là một đối tượng TcpClient đại diện cho kết nối giữa client và server.
   * client.Client trả về đối tượng Socket liên kết với kết nối này.
2. **client.Client.RemoteEndPoint**:
   * Thuộc tính RemoteEndPoint của đối tượng Socket trả về một EndPoint đại diện cho địa chỉ IP và cổng của client đang kết nối tới server.
3. **as IPEndPoint**:
   * EndPoint là một lớp cơ sở, và IPEndPoint là một lớp con của nó chứa thông tin cụ thể về địa chỉ IP và số cổng.
   * Bằng cách sử dụng as IPEndPoint, bạn cố gắng chuyển đổi EndPoint thành IPEndPoint.
   * Nếu client.Client.RemoteEndPoint thực sự là một đối tượng IPEndPoint, thì remoteEndpoint sẽ được gán đối tượng này. Nếu không, remoteEndpoint sẽ là null.

**Tại sao lại cần chuyển đổi sang IPEndPoint?**

* **Lấy thông tin chi tiết về client**: IPEndPoint cung cấp thông tin chi tiết về địa chỉ IP và cổng của client, trong khi EndPoint không cung cấp các thông tin chi tiết này.
* **Sử dụng các thuộc tính cụ thể**: Khi bạn chuyển đổi thành IPEndPoint, bạn có thể truy cập các thuộc tính như Address và Port để biết chính xác địa chỉ IP và cổng của client đang kết nối.

#### 1.1.2.3. NetworkStream stream = client.GetStream():

* **Mục đích**: Lấy một NetworkStream từ đối tượng TcpClient.
* **Chức năng**:
  + client.GetStream(): Trả về một NetworkStream dùng để đọc và ghi dữ liệu qua kết nối TCP.
  + NetworkStream cung cấp một cách để gửi và nhận dữ liệu qua kết nối TCP được thiết lập bởi TcpClient.

#### 1.1.2.4. StreamReader reader = new StreamReader(stream):

* **Mục đích**: Tạo một StreamReader để dễ dàng đọc dữ liệu từ NetworkStream.
* **Chức năng**:
  + StreamReader là một lớp tiện ích được xây dựng trên Stream (ở đây là NetworkStream), cung cấp các phương thức để đọc ký tự, chuỗi và các kiểu dữ liệu khác từ stream.
  + Nó giúp đơn giản hóa việc đọc dữ liệu dạng văn bản từ stream.

#### 1.1.2.5. Nhận dữ liệu file

**Tạo FileStream để ghi dữ liệu vào file**  
using (FileStream fileStream = File.Create(filePath))

* **File.Create(filePath)**: Tạo một file mới hoặc ghi đè lên file đã tồn tại với đường dẫn được cung cấp bởi filePath.
* **FileStream fileStream**: Đối tượng FileStream đại diện cho một luồng dữ liệu đến và từ file trên đĩa. Nó cho phép bạn ghi dữ liệu vào file.
* **using**: Đảm bảo rằng fileStream sẽ được đóng và giải phóng tài nguyên sau khi khối using kết thúc, ngay cả khi có ngoại lệ xảy ra.

**Tạo buffer để đọc dữ liệu**byte[] buffer = new byte[4096];

* **byte[] buffer = new byte[4096]**: Tạo một mảng byte với kích thước 4096 byte (4 KB). Buffer này sẽ được sử dụng để lưu trữ dữ liệu khi đọc từ NetworkStream và ghi vào FileStream.

**Đọc dữ liệu từ NetworkStream và ghi vào FileStream**int bytesRead;

while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)

{

fileStream.Write(buffer, 0, bytesRead);

}

* **int bytesRead**: Biến để lưu trữ số byte thực tế được đọc từ NetworkStream.
* **stream.Read(buffer, 0, buffer.Length)**: Đọc dữ liệu từ NetworkStream vào buffer. Phương thức Read trả về số byte thực tế được đọc và lưu vào bytesRead. Nó sẽ đọc tối đa số byte bằng kích thước của buffer (4096 byte trong trường hợp này).
* **while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)**: Vòng lặp sẽ tiếp tục đọc dữ liệu từ NetworkStream cho đến khi không còn dữ liệu để đọc (Read trả về 0, nghĩa là đã đọc đến cuối luồng).
* **fileStream.Write(buffer, 0, bytesRead)**: Ghi dữ liệu từ buffer vào FileStream. Số byte ghi vào file là số byte thực sự được đọc từ NetworkStream (được lưu trong bytesRead).

**Tổng quan**

* **Mục đích**: Đoạn mã này nhận dữ liệu từ một kết nối mạng và ghi dữ liệu đó vào một file trên đĩa.
* **Các bước**:
  + Tạo hoặc mở file bằng FileStream.
  + Sử dụng một buffer để đọc dữ liệu từ NetworkStream theo từng khối.
  + Ghi các khối dữ liệu vào file.
  + Lặp lại quá trình cho đến khi tất cả dữ liệu được đọc và ghi.

#### 1.1.2.6. Gửi phản hồi lại cho client

**Tạo phản hồi dưới dạng byte array**  
byte[] response = Encoding.UTF8.GetBytes("Hello, File received successfully!\n");

* **Encoding.UTF8.GetBytes("Hello, File received successfully!\n")**: Chuyển đổi chuỗi văn bản "Hello, File received successfully!\n" thành mảng byte (byte[]).
* **Encoding.UTF8**: Sử dụng mã hóa UTF-8 để chuyển đổi chuỗi ký tự thành byte. UTF-8 là một phương thức mã hóa phổ biến cho văn bản vì nó có thể biểu diễn tất cả các ký tự trong Unicode.
* **byte[] response**: Mảng byte chứa dữ liệu của phản hồi mà server sẽ gửi đến client.

**Ghi phản hồi vào NetworkStream**

stream.Write(response, 0, response.Length);

* **stream.Write(response, 0, response.Length)**: Ghi dữ liệu từ mảng byte response vào NetworkStream.
* **response**: Mảng byte chứa dữ liệu cần gửi.
* **0**: Vị trí bắt đầu trong mảng byte từ đó bắt đầu ghi dữ liệu (ở đây là từ đầu mảng).
* **response.Length**: Số byte cần ghi từ mảng byte. Ở đây, ghi toàn bộ mảng byte.

**Đảm bảo tất cả dữ liệu đã được gửi đi**stream.Flush();

* **stream.Flush()**: Đảm bảo rằng tất cả dữ liệu đã được gửi từ bộ đệm của NetworkStream đến client.
* **Flush**: Phương thức này sẽ đẩy bất kỳ dữ liệu nào còn lại trong bộ đệm của NetworkStream và gửi nó ngay lập tức đến client. Đây là một bước quan trọng để đảm bảo rằng không có dữ liệu nào bị bỏ sót hoặc trì hoãn gửi.

**Tổng quan**

* **Mục đích**: Đoạn mã này gửi một thông báo đến client để xác nhận rằng file đã được nhận thành công và lưu trữ. Phản hồi này có thể được sử dụng để thông báo cho client rằng thao tác gửi file đã hoàn tất thành công.
* **Các bước**:
  + Chuyển đổi chuỗi văn bản thành mảng byte.
  + Ghi mảng byte vào NetworkStream.
  + Đảm bảo rằng tất cả dữ liệu đã được gửi đến client bằng cách gọi Flush.

### 1.1.3. Hàm GetLocalIPv4Address

**Mục đích**: Lấy địa chỉ IPv4 của máy chủ.

**Chức năng**:

* Lấy danh sách các địa chỉ IP của máy chủ.
* Lặp qua danh sách này và trả về địa chỉ IPv4 đầu tiên tìm thấy.
* Nếu không tìm thấy địa chỉ IPv4 nào, trả về IPAddress.None.

| private IPAddress GetLocalIPv4Address()  {  foreach (var ip in Dns.GetHostEntry(Dns.GetHostName()).AddressList)  {  if (ip.AddressFamily == AddressFamily.InterNetwork)  {  return ip;  }  }  return IPAddress.None;  } |
| --- |

## 1.2. Mã nguồn đầy đủ

### 1.2.1. FileServer.cs

| using System.Net; using System.Net.Sockets; using System.Text;  namespace FileTransferServerCsharp.Services {  **public** **class** FileServer  {  **private** readonly **int** \_port;  **private** readonly string \_directoryPath;   **public** FileServer(**int** port, string directoryPath)  {  \_port = port;  \_directoryPath = directoryPath;   *// Tạo thư mục nếu nó không tồn tại*  **if** (!Directory.Exists(directoryPath))  {  Directory.CreateDirectory(directoryPath);  Console.WriteLine($**"Thư mục {directoryPath} đã được tạo."**);  }  }   **public** **void** Start()  {  **try**  {  *// Chỉ lắng nghe trên IPv4*  TcpListener listener = **new** TcpListener(IPAddress.Any, \_port);  listener.Start();   *// In ra địa chỉ IP và cổng của server*  IPEndPoint localEndPoint = listener.LocalEndpoint as IPEndPoint;  **if** (localEndPoint != **null**)  {  IPAddress ipv4Address = GetLocalIPv4Address();  Console.WriteLine($**"Server đang lắng nghe trên IP: {ipv4Address} và cổng: {localEndPoint.Port}"**);  }   **while** (**true**)  {  *// Chấp nhận kết nối từ client*  using (TcpClient client = listener.AcceptTcpClient())  {  IPEndPoint remoteEndpoint = client.Client.RemoteEndPoint as IPEndPoint;  **if** (remoteEndpoint != **null**)  {  Console.WriteLine($**"Đã kết nối với client: {remoteEndpoint.Address}:{remoteEndpoint.Port}"**);  }   using (NetworkStream stream = client.GetStream())  using (StreamReader reader = **new** StreamReader(stream))  {  *// Nhận tên file từ Client*  string fileName = reader.ReadLine();  **if** (string.IsNullOrEmpty(fileName))  {  Console.WriteLine(**"Tên file không hợp lệ."**);  **continue**;  }  string filePath = Path.Combine(\_directoryPath, fileName);   *// Nhận dữ liệu file*  using (FileStream fileStream = File.Create(filePath))  {  **byte**[] buffer = **new** **byte**[4096];  **int** bytesRead;  **while** ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)  {  fileStream.Write(buffer, 0, bytesRead);  }  }   Console.WriteLine($**"File {fileName} đã được nhận và lưu vào thư mục {\_directoryPath}"**);   *// Gửi phản hồi lại cho client*  **byte**[] response = Encoding.UTF8.GetBytes(**"Hello, File received successfully!\n"**);  stream.Write(response, 0, response.Length);  stream.Flush(); *// Đảm bảo tất cả dữ liệu đã được gửi đi*  }  }  }  }  **catch** (Exception ex)  {  Console.WriteLine(**"Lỗi khi khởi động server: "** + ex.ToString());  }  }   **private** IPAddress GetLocalIPv4Address()  {  foreach (var ip in Dns.GetHostEntry(Dns.GetHostName()).AddressList)  {  **if** (ip.AddressFamily == AddressFamily.InterNetwork)  {  **return** ip;  }  }  **return** IPAddress.None;  }  } } |
| --- |

### 1.2.2. Program.cs

| using FileTransferServerCsharp.Services; using System.Text;  Console.OutputEncoding = Encoding.UTF8;  string ROOT\_DIRECTORY = Directory.GetParent(AppContext.BaseDirectory).Parent.Parent.Parent.FullName;  **int** port = 100; string directoryPath = Path.Combine(ROOT\_DIRECTORY, **"FileReceived"**);  FileServer server = **new** FileServer(port, directoryPath); server.Start(); |
| --- |

# 2. Sửa lại FileTransferTask phía client (Android)

FileTransferTask là một lớp kế thừa từ AsyncTask trong Android, được thiết kế để gửi một file từ thiết bị Android đến một máy chủ qua một kết nối TCP/IP. Lớp này sử dụng các luồng dữ liệu để truyền file và nhận phản hồi từ máy chủ.

**Mục đích**

Lớp này thực hiện các nhiệm vụ sau:

* Kết nối đến máy chủ thông qua địa chỉ IP và cổng đã định.
* Đọc file từ thiết bị Android.
* Gửi tên file và dữ liệu file đến máy chủ.
* Nhận và xử lý phản hồi từ máy chủ.
* Hiển thị kết quả (thành công hoặc thất bại) trên giao diện người dùng.

## 2.1. Chi tiết mã nguồn

### 2.1.1. Các Tham số

**Constructor**

public FileTransferTask(Context context)

* **context**: Context của ứng dụng Android. Dùng để truy cập các dịch vụ của hệ thống, chẳng hạn như ContentResolver để đọc file từ URI.

### 2.1.2. Các Phương thức

#### 2.1.2.1. doInBackground(Uri... uris)

Phương thức chính của AsyncTask, chạy trên một luồng nền để thực hiện các tác vụ dài hạn.

Tham số:

* uris: Một mảng Uri đại diện cho các địa chỉ của file cần gửi. Phương thức chỉ xử lý file từ URI đầu tiên (uris[0]).

Quá trình thực hiện:

1. Mở kết nối đến máy chủ: Kết nối đến máy chủ qua IP và cổng được chỉ định.
2. Mở luồng dữ liệu:

* inputStream từ ContentResolver của context để đọc dữ liệu file.
* outputStream để gửi dữ liệu đến máy chủ.

1. Gửi dữ liệu:

* Lấy tên file từ URI và gửi tên file trước dữ liệu.
* Đọc dữ liệu từ file và gửi qua kết nối.

1. Nhận phản hồi từ máy chủ: Đọc phản hồi từ máy chủ để xử lý kết quả gửi file.
2. Xử lý lỗi: Nếu có lỗi trong quá trình gửi file, trả về thông báo lỗi.

Trả về: Chuỗi phản hồi từ máy chủ hoặc thông báo lỗi nếu có lỗi xảy ra.

| @Override **protected** String doInBackground(Uri... uris) {  Uri fileUri = uris[0];  String responseMessage = **""**;  **try** (  Socket socket = **new** Socket(SERVER\_IP, SERVER\_PORT);  InputStream inputStream = context.getContentResolver().openInputStream(fileUri);  BufferedInputStream bis = **new** BufferedInputStream(inputStream);  OutputStream os = socket.getOutputStream();  BufferedOutputStream bos = **new** BufferedOutputStream(os);  BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(socket.getInputStream(), **"UTF-8"**));  ) {   **if** (inputStream == **null**) {  Log.e(**"FileTransferTask"**, **"InputStream is null."**);  **return** **"Luồng dữ liệu không hợp lệ!"**;  }   *// Lấy tên file từ URI*  String fileName = getFileName(fileUri);   *// Gửi tên file trước dữ liệu*  bos.write((fileName + **"\n"**).getBytes());  bos.flush();   *// Gửi dữ liệu file*  **byte**[] buffer = **new** **byte**[4096];  **int** bytesRead;  **while** ((bytesRead = bis.read(buffer)) != -1) {  bos.write(buffer, 0, bytesRead);  }  bos.flush();  socket.shutdownOutput();   *// Nhận thông điệp từ server*  responseMessage = reader.readLine();   **return** responseMessage;  } **catch** (Exception e) {  e.printStackTrace();  responseMessage = **"Gửi file thất bại: "** + e.getMessage();  **return** responseMessage;  } } |
| --- |

**Kết nối đến máy chủ** qua Socket.

**Đọc dữ liệu từ file** thông qua InputStream và BufferedInputStream.

**Gửi dữ liệu** đến máy chủ thông qua OutputStream và BufferedOutputStream.

**Nhận phản hồi từ máy chủ** bằng BufferedReader.

#### 2.1.2.2. onPostExecute(String result)

Phương thức này được gọi khi doInBackground hoàn tất.

Tham số:

* result: Chuỗi kết quả trả về từ máy chủ hoặc thông báo lỗi.

Quá trình thực hiện:

* Hiển thị thông báo Toast với kết quả hoặc thông báo lỗi trên giao diện người dùng.

#### 2.1.2.3. getFileName(Uri uri)

Phương thức phụ trợ để lấy tên của file từ URI.

Tham số:

* uri: Uri đại diện cho file từ đó cần lấy tên.

Quá trình thực hiện:

1. Sử dụng ContentResolver để truy vấn tên file từ URI.
2. Nếu không lấy được tên file, trả về giá trị mặc định "unknown.jpg".

Trả về: Tên của file hoặc "unknown.jpg" nếu không thể xác định tên file.

### 2.1.3. Xử lý Ngoại lệ

* **Lỗi Kết nối**: Nếu có lỗi khi kết nối đến máy chủ hoặc khi gửi dữ liệu, thông báo lỗi sẽ được trả về.
* **Lỗi Đọc/Viết File**: Nếu không thể đọc file từ URI hoặc gặp lỗi khi ghi dữ liệu, thông báo lỗi sẽ được trả về.

## 2.2. Mã nguồn đầy đủ

| **package** work.ngangiang.camera.Services;  **import** android.content.ContentResolver; **import** android.content.Context; **import** android.database.Cursor; **import** android.net.Uri; **import** android.os.AsyncTask; **import** android.util.Log; **import** android.widget.Toast;  **import** java.io.BufferedInputStream; **import** java.io.BufferedOutputStream; **import** java.io.BufferedReader; **import** java.io.BufferedWriter; **import** java.io.InputStream; **import** java.io.InputStreamReader; **import** java.io.OutputStream; **import** java.io.OutputStreamWriter; **import** java.net.Socket;  **public** **class** FileTransferTask **extends** AsyncTask<Uri, Void, String> {   **private** **static** **final** String SERVER\_IP = **"192.168.1.41"**; *// Địa chỉ IP của server*  **private** **static** **final** **int** SERVER\_PORT = 100; *// Cổng của server*  **private** Context context;   *// Constructor để nhận Context*  **public** FileTransferTask(Context context) {  **this**.context = context;  }   @Override  **protected** String doInBackground(Uri... uris) {  Uri fileUri = uris[0];  String responseMessage = **""**;  **try** (  Socket socket = **new** Socket(SERVER\_IP, SERVER\_PORT);  InputStream inputStream = context.getContentResolver().openInputStream(fileUri);  BufferedInputStream bis = **new** BufferedInputStream(inputStream);  OutputStream os = socket.getOutputStream();  BufferedOutputStream bos = **new** BufferedOutputStream(os);  BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(socket.getInputStream(), **"UTF-8"**));  ) {   **if** (inputStream == **null**) {  Log.e(**"FileTransferTask"**, **"InputStream is null."**);  **return** **"Luồng dữ liệu không hợp lệ!"**;  }   *// Lấy tên file từ URI*  String fileName = getFileName(fileUri);   *// Gửi tên file trước dữ liệu*  bos.write((fileName + **"\n"**).getBytes());  bos.flush();   *// Gửi dữ liệu file*  **byte**[] buffer = **new** **byte**[4096];  **int** bytesRead;  **while** ((bytesRead = bis.read(buffer)) != -1) {  bos.write(buffer, 0, bytesRead);  }  bos.flush();  socket.shutdownOutput();   *// Nhận thông điệp từ server*  responseMessage = reader.readLine();   **return** responseMessage;  } **catch** (Exception e) {  e.printStackTrace();  responseMessage = **"Gửi file thất bại: "** + e.getMessage();  **return** responseMessage;  }  }   @Override  **protected** **void** onPostExecute(String result) {  Toast.makeText(context, result, Toast.LENGTH\_SHORT).show();  }   **private** String getFileName(Uri uri) {  ContentResolver contentResolver = context.getContentResolver();  String[] projection = {android.provider.MediaStore.Images.Media.DISPLAY\_NAME};  **try** (Cursor cursor = contentResolver.query(uri, projection, **null**, **null**, **null**)) {  **if** (cursor != **null** && cursor.moveToFirst()) {  **int** nameIndex = cursor.getColumnIndexOrThrow(android.provider.MediaStore.Images.Media.DISPLAY\_NAME);  **return** cursor.getString(nameIndex);  }  }  **return** **"unknown.jpg"**;  } } |
| --- |