HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA ĐÀO TẠO SAU ĐẠI HỌC

-----

# BÀI TẬP TIỂU LUẬN

# CÁC HỆ THỐNG PHÂN TÁN

## ĐỀ TÀI: HỆ THỐNG CHIA SẺ FILE NGANG HÀNG P2P


| | | |
|---|---|---|
| **Giảng viên** | **:** | **Kim Ngọc Bách** |
| **Chuyên ngành** | **:** | **Hệ thống thông tin** |
| **Lớp** | **:** | **M25CQHT01-B** |
| **Nhóm** | **:** | **08** |
| **Thành viên** | **:** | **Bùi Đăng Tân - B25CHHT051** |
| | | **Phạm Hồng Thái - B25CHHT052** |
| | | **Trần Anh Đức - B25CHHT012** |


**Hà Nội, 12-2025**

# MỤC LỤC

# DANH MỤC HÌNH ẢNH

# LỜI MỞ ĐẦU

Trong bối cảnh các hệ thống phân tán ngày càng phát triển mạnh mẽ, mô hình Peer-to-Peer (P2P) nổi bật như một kiến trúc quan trọng, giúp giảm tải đáng kể cho server trung tâm, tăng khả năng mở rộng hệ thống và nâng cao độ chịu lỗi. Khác với mô hình client-server truyền thống – nơi mọi dữ liệu đều phải đi qua một server duy nhất, dễ dẫn đến điểm nghẽn và rủi ro tập trung – P2P cho phép các nút mạng (peer) kết nối trực tiếp với nhau, chia sẻ tài nguyên một cách ngang hàng.

WebRTC (Web Real-Time Communication) là công nghệ hiện đại lý tưởng để triển khai P2P trên nền web, cho phép thiết lập kết nối trực tiếp giữa các trình duyệt mà không cần plugin, hỗ trợ truyền dữ liệu thời gian thực với độ bảo mật cao nhờ mã hóa DTLS/SRTP tích hợp.

Đề tài **"File Transfer P2P - WebRTC"** nhằm mục đích hiểu sâu cơ chế hoạt động của WebRTC, bao gồm Signaling, ICE (STUN/TURN), và Data Channel; đồng thời triển khai thực tế một ứng dụng chia sẻ file phân tán hoàn chỉnh từ đầu, áp dụng kiến thức về hệ thống phân tán như kết nối ngang hàng, quản lý peer và truyền dữ liệu không qua server trung tâm.

Mục tiêu chính của đề tài là xây dựng một ứng dụng web cho phép hai hoặc nhiều người dùng chia sẻ file trực tiếp qua kết nối P2P, chỉ sử dụng server tối thiểu (signaling server) để trao đổi thông tin kết nối ban đầu, trong khi toàn bộ dữ liệu file được truyền trực tiếp giữa các peer. Ứng dụng đảm bảo các tính năng cơ bản như kết nối dễ dàng bằng Peer ID, hỗ trợ drag & drop file, hiển thị progress realtime và tự động tải file nhận được.

# I. GIỚI THIỆU CHUNG

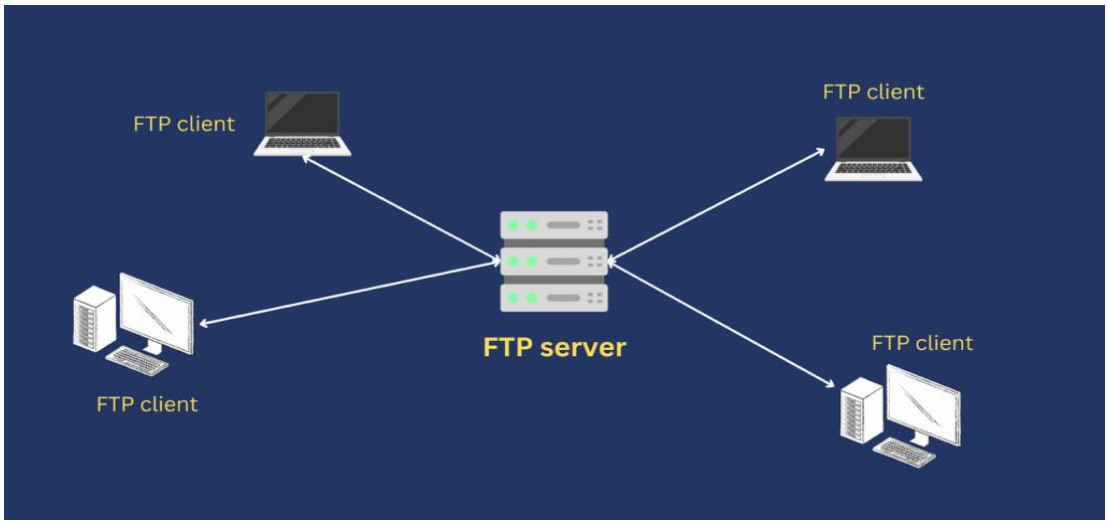
## 1. Mô hình chia sẻ tệp tin truyền thống Client-Server

Mô hình Client-Server đã thống trị kiến trúc mạng trong nhiều thập kỷ qua. Trong mô hình này, một máy chủ trung tâm (Server) đóng vai trò là kho lưu trữ và điều phối dữ liệu duy nhất. Khi một người dùng muốn chia sẻ file, quy trình này bị chia làm hai giai đoạn tách biệt:

• Người gửi (client) upload file lên server trung tâm.

• Người nhận (client khác) download file từ server đó.

Sự phụ thuộc vào máy chủ trung tâm tạo ra một rào cản về mặt vật lý và logic. Dữ liệu không được truyền đi dựa trên con đường ngắn nhất giữa hai người dùng, mà phải đi qua "trạm trung chuyển" là server trung tâm, ngay cả khi hai người dùng đang ngồi cạnh nhau trong cùng một mạng LAN. Điều này dẫn đến sự lãng phí tài nguyên mạng đáng kể và kéo dài thời gian trễ (latency).




*Hình 1 Mô hình FTP client-server*

Mặc dù có ưu điểm là dễ triển khai và quản lý, mô hình chia sẻ tệp tin truyền thống theo kiến trúc Client–Server bộc lộ nhiều hạn chế khi quy mô hệ thống ngày càng mở rộng:

- Vấn đề điểm nghẽn (Bottleneck):

  Toàn bộ lưu lượng truy cập đều tập trung vào máy chủ trung tâm. Khi số lượng người dùng truy cập đồng thời tăng cao, tài nguyên và băng thông của server bị phân tán, dẫn đến hiệu năng hệ thống suy giảm và tốc độ truyền tải tệp tin giảm đáng kể.
- Chi phí vận hành và hạ tầng:

  Việc duy trì một hệ thống chia sẻ tệp tin quy mô lớn đòi hỏi đầu tư đáng kể vào hạ tầng lưu trữ dung lượng cao và đường truyền mạng băng thông lớn. Bên cạnh đó, doanh nghiệp còn phải gánh thêm các chi phí liên quan đến bảo trì hệ thống, làm mát, cũng như nhân sự kỹ thuật vận hành.
- Rủi ro về bảo mật và quyền riêng tư:

  Dữ liệu người dùng được lưu trữ tập trung trên máy chủ trở thành mục tiêu hấp dẫn của các cuộc tấn công mạng. Ngoài ra, việc người quản trị hệ thống có quyền truy cập vào nội dung tệp tin tiềm ẩn nguy cơ xâm phạm quyền riêng tư, đặc biệt đối với các dữ liệu nhạy cảm.
- Độ tin cậy thấp (Single Point of Failure):

  Kiến trúc phụ thuộc vào máy chủ trung tâm khiến hệ thống dễ bị gián đoạn. Trong trường hợp server gặp sự cố phần cứng hoặc bị tấn công từ chối dịch vụ (DDoS), toàn bộ hoạt động chia sẻ tệp tin của người dùng sẽ bị ngưng trệ hoàn toàn.

Để giải quyết các hạn chế của mô hình truyền thống, mô hình Peer-to-Peer (P2P) đã ra đời và phát triển mạnh mẽ, trong đó việc chia sẻ tệp tin trở thành một trong những ứng dụng tiêu biểu và hiệu quả nhất. Mô hình này mang lại nhiều ưu điểm nổi bật, bao gồm:

- Dữ liệu được truyền trực tiếp giữa các peer, cho phép khai thác đồng thời nhiều nguồn truyền, từ đó nâng cao tốc độ chia sẻ tệp tin.

- Giảm tải cho máy chủ trung tâm, do server chỉ đóng vai trò hỗ trợ signaling hoặc điều phối kết nối.

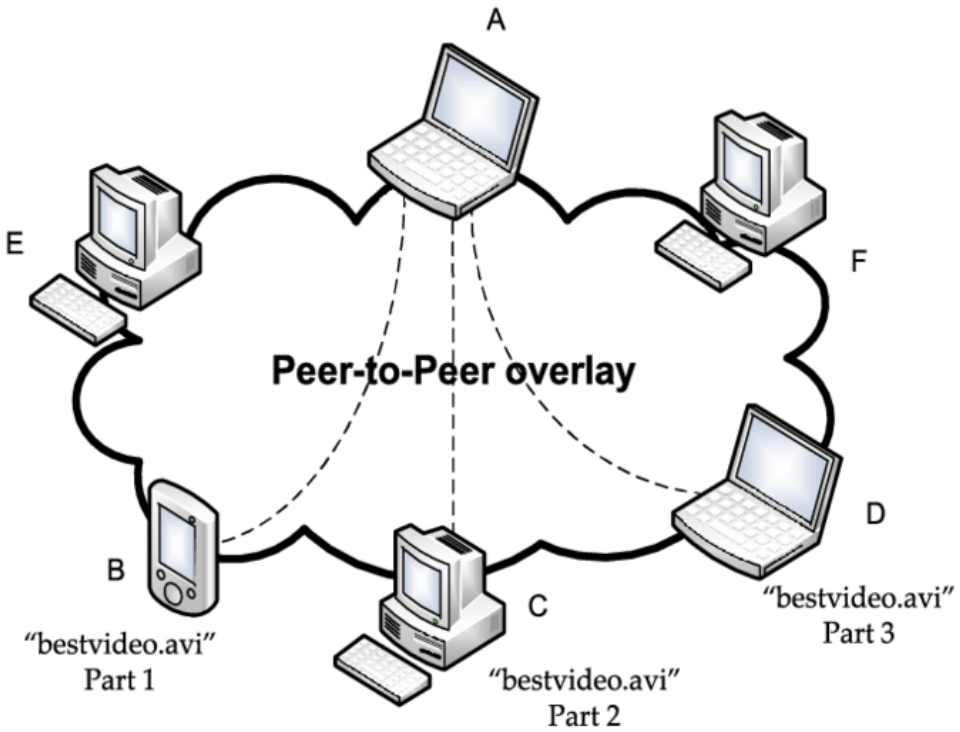
- Nâng cao mức độ bảo mật nhờ việc dữ liệu không được lưu trữ tập trung trên một máy chủ duy nhất.

- Tiết kiệm chi phí triển khai hạ tầng và có khả năng mở rộng hệ thống gần như không giới hạn.

## 2. Mô hình Peer-to-Peer (P2P)

Mô hình Peer-to-Peer (P2P) là một kiến trúc mạng phân tán, trong đó các máy tính (gọi là peer) có vai trò ngang hàng và kết nối tương tác trực tiếp với nhau để chia sẻ tài nguyên mà không phụ thuộc hoàn toàn vào một server trung tâm. Mỗi peer vừa là client yêu cầu tài nguyên vừa là server cung cấp tài nguyên cho các peer khác trong mạng.

Về mặt kiến trúc, P2P không phụ thuộc vào một máy chủ trung tâm để lưu trữ hoặc phân phối dữ liệu. Thay vào đó, dữ liệu được phân tán trên nhiều peer khác nhau và việc truy xuất tài nguyên được thực hiện thông qua các kết nối ngang hàng. Cách tiếp cận này giúp giảm sự phụ thuộc vào hạ tầng tập trung, đồng thời hạn chế các vấn đề như quá tải máy chủ và điểm nghẽn hệ thống.




*Hình 2 Sơ đồ mạng Peer-to-Peer overlay – minh họa cách các peer chia sẻ file*

Mô hình chia sẻ tệp tin P2P có một số đặc trưng kỹ thuật quan trọng như sau:

- Tính phân tán và tự tổ chức (Self-organizing):

  Các peer có thể tham gia hoặc rời khỏi hệ thống một cách linh hoạt mà không làm gián đoạn hoạt động chung của mạng. Hệ thống có khả năng tự điều chỉnh để duy trì kết nối và tiếp tục chia sẻ dữ liệu.
- Khả năng chịu lỗi cao:

  Do dữ liệu và tài nguyên không tập trung tại một điểm duy nhất, sự cố của một hoặc một số peer không làm ảnh hưởng đến toàn bộ hệ thống, từ đó nâng cao tính sẵn sàng của dịch vụ.
- Cơ chế chia nhỏ và truyền song song dữ liệu:

  Trong các hệ thống chia sẻ tệp tin P2P, dữ liệu thường được chia thành nhiều khối nhỏ và phân phối trên nhiều peer khác nhau. Khi một peer tải tệp tin, các khối dữ liệu này có thể được truyền đồng thời từ nhiều nguồn, giúp tối ưu băng thông và rút ngắn thời gian truyền tải.
- Khả năng mở rộng cao (Scalability):

  Khi số lượng peer tham gia hệ thống tăng lên, tổng tài nguyên và năng lực phân phối dữ liệu của mạng cũng tăng theo. Điều này giúp mô hình P2P đặc biệt phù hợp với các hệ thống chia sẻ tệp tin quy mô lớn trên Internet.

Nhiều hệ thống chia sẻ tệp tin phổ biến đã được xây dựng dựa trên mô hình P2P, tiêu biểu là BitTorrent, nơi người dùng có thể tải dữ liệu từ nhiều peer đồng thời thay vì phụ thuộc vào một máy chủ duy nhất.

Do đặc thù của môi trường mạng Internet hiện nay, việc triển khai kết nối P2P trực tiếp giữa các peer cần có các công nghệ hỗ trợ thiết lập kết nối trực tiếp, vượt qua các rào cản mạng như NAT và tường lửa, đảm bảo độ trễ thấp và tính bảo mật trong quá trình truyền dữ liệu. Những yêu cầu này đã thúc đẩy sự ra đời và ứng dụng của WebRTC như một giải pháp truyền thông thời gian thực phù hợp cho các hệ thống chia sẻ tệp tin P2P trên nền tảng web.

## 3. Giới thiệu về công nghệ WebRTC

WebRTC (Web Real-Time Communication) là một dự án mã nguồn mở do Google khởi xướng, cung cấp các API JavaScript được tích hợp sẵn trong các trình duyệt

hiện đại, cho phép thiết lập truyền thông thời gian thực trực tiếp giữa các peer mà không cần cài đặt plugin bổ sung (như Flash trong các hệ thống cũ).

WebRTC hỗ trợ truyền tải âm thanh, video và dữ liệu tùy ý (arbitrary data) theo mô hình ngang hàng (peer-to-peer), đồng thời tích hợp sẵn các cơ chế bảo mật nhằm đảm bảo an toàn cho dữ liệu trong quá trình truyền. Nhờ đó, WebRTC trở thành một nền tảng phù hợp để hiện thực hóa các ứng dụng chia sẻ tệp tin P2P trên môi trường web.

WebRTC bao gồm ba thành phần cốt lõi chính:

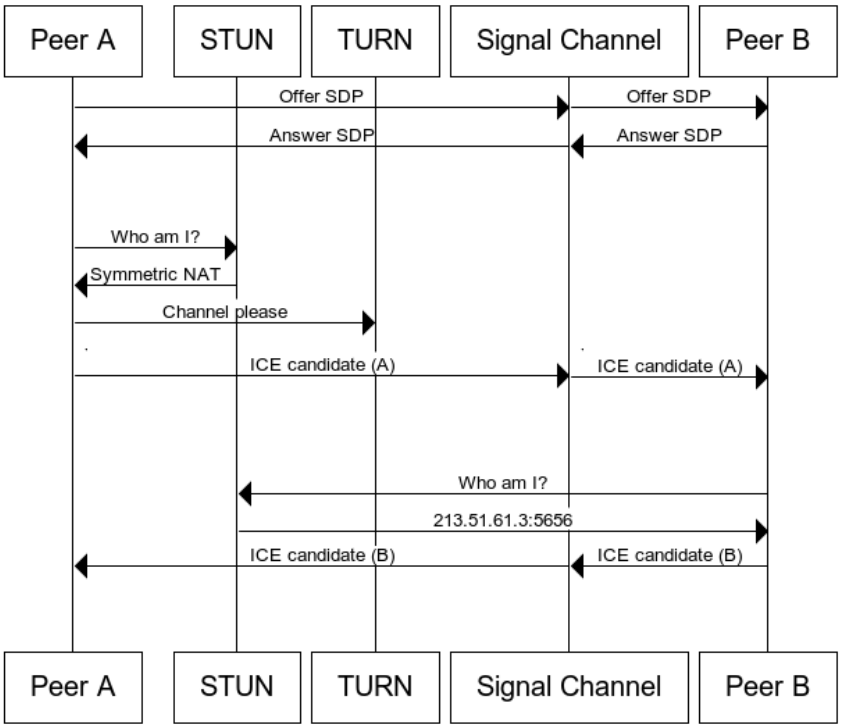
- **RTCPeerConnection**: Đây là thành phần phức tạp nhất, chịu trách nhiệm tối ưu hóa băng thông, xử lý các vấn đề mất gói tin, và quan trọng nhất là thực hiện quá trình "bắt tay" (handshake) để thiết lập kết nối. Nó tự động tính toán các thông số mạng nhằm thiết lập đường truyền dữ liệu ngắn nhất và hiệu quả nhất.
- **RTCDataChannel**: Khác với các luồng âm thanh/video, RTCDataChannel được thiết kế dành riêng cho việc truyền dữ liệu thô (binary data) như văn bản, JSON hoặc file. DataChannel cho phép cấu hình chế độ truyền:
    - Reliable – đảm bảo dữ liệu đến đích đầy đủ và chính xác, phù hợp cho chia sẻ tệp tin.
    - Unreliable – ưu tiên độ trễ thấp, phù hợp cho các ứng dụng thời gian thực như trò chơi.
- **Cơ chế NAT Traversal**: WebRTC sử dụng giao thức ICE (Interactive Connectivity Establishment) kết hợp với STUN và TURN server để vượt qua các rào cản mạng như NAT và tường lửa, cho phép các peer có thể thiết lập kết nối trực tiếp trong môi trường Internet công cộng.

Ưu điểm của WebRTC trong ứng dụng P2P file sharing

- Truyền trực tiếp P2P: Dữ liệu được truyền trực tiếp giữa các peer mà không đi qua server trung tâm, giúp giảm độ trễ và tăng tốc độ truyền tải.

- Bảo mật cao: WebRTC tự động mã hóa dữ liệu bằng DTLS đối với DataChannel và SRTP đối với luồng media, đảm bảo tính bảo mật end-to-end.
- Hỗ trợ NAT traversal tự động: Cơ chế ICE cùng STUN/TURN giúp tăng tỷ lệ thiết lập kết nối thành công ngay cả khi các peer nằm sau router hoặc tường lửa.
- Không cần plugin: WebRTC hoạt động trực tiếp trên trình duyệt, giúp dễ dàng triển khai và tương thích đa nền tảng.
- Hỗ trợ truyền dữ liệu thời gian thực và khả năng mở rộng: Dữ liệu lớn có thể được chia nhỏ và truyền qua DataChannel, phù hợp cho các hệ thống chia sẻ tệp tin P2P quy mô nhỏ đến trung bình.

## 4. Quy trình thiết lập kết nối trong WebRTC




*Hình 3 Quá trình thiết lập kết nối trong WebRTC*

Trong quá trình thiết lập kết nối, WebRTC sử dụng cơ chế ICE để tự động tìm ra con đường truyền dữ liệu khả thi và hiệu quả nhất giữa các peer. Quá trình này diễn ra theo thứ tự ưu tiên, từ kết nối trực tiếp nhất đến phương án dự phòng cuối cùng.

### Giai đoạn 1: Trao đổi thông tin kết nối (SDP Signaling)

Ban đầu, Peer A và Peer B chưa thể kết nối trực tiếp do bị ngăn cách bởi NAT hoặc firewall. Vì vậy, cả hai sử dụng **Signal Channel** (thông qua Signaling Server) để trao đổi thông tin kết nối ban đầu:

- Peer A tạo một **SDP Offer**, mô tả các khả năng kết nối, giao thức truyền dữ liệu và cơ chế mã hóa mà trình duyệt hỗ trợ.
- SDP Offer được gửi tới Signaling Server và chuyển tiếp đến Peer B.
- Peer B phân tích Offer, thiết lập cấu hình tương ứng, sau đó tạo **SDP Answer** và gửi ngược lại cho Peer A thông qua Signal Channel.

Sau giai đoạn này, hai peer đã thống nhất cấu hình truyền thông, nhưng chưa biết địa chỉ mạng thực tế để có thể kết nối trực tiếp.

**Giai đoạn 2: Thu thập và trao đổi địa chỉ mạng (ICE Candidate Gathering)**

Để vượt qua NAT và xác định các địa chỉ có thể kết nối, mỗi peer tiến hành thu thập ICE candidates:

- Trước hết, peer thu thập host candidate (IP nội bộ trong LAN). Trường hợp hai peer cùng mạng, kết nối có thể được thiết lập ngay tại bước này.
- Nếu không cùng mạng, peer gửi yêu cầu “Who am I?” đến STUN Server để xác định địa chỉ IP công khai và cổng mà NAT ánh xạ ra Internet.
- Thông tin này được đóng gói thành server reflexive candidate (srflx) và gửi tới peer còn lại thông qua Signal Channel.

Quy trình trên diễn ra song song ở cả hai phía:

- Peer A gửi ICE candidate (A) cho Peer B.
- Peer B gửi ICE candidate (B) cho Peer A.

**Giai đoạn 3: Kiểm tra kết nối và fallback qua TURN (ICE Connectivity Check)**

Sau khi trao đổi danh sách ICE candidates, WebRTC tự động thực hiện kiểm tra kết nối, tìm kiếm con đường truyền ngắn nhất, trực tiếp nhất giữa Peer A và Peer .

- Các cặp candidate được thử theo thứ tự ưu tiên:
  **host → srflx (STUN) → relay (TURN).**
- Nếu hai peer nằm cùng mạng nội bộ, WebRTC sẽ ưu tiên sử dụng host candidate để thiết lập kết nối trực tiếp với độ trễ thấp nhất và không cần đi qua Internet.
- Khi các peer nằm sau NAT, WebRTC gửi yêu cầu đến **STUN server** để xác định địa chỉ IP và cổng công khai mà NAT ánh xạ ra Internet.\
- Trong trường hợp cả host và srflx candidate đều thất bại, thường xảy ra khi peer nằm sau **Symmetric NAT** hoặc firewall nghiêm ngặt, WebRTC sẽ sử dụng **TURN server** để cấp một địa chỉ relay trung gian. Khi đó, TURN server đóng vai trò trung gian chuyển tiếp dữ liệu giữa hai peer, đảm bảo kết nối vẫn được thiết lập dù không thể kết nối P2P trực tiếp.

**Giai đoạn 4: Thiết lập kết nối P2P và truyền dữ liệu**

Khi một cặp ICE candidate hợp lệ được xác nhận:

- Kết nối P2P giữa Peer A và Peer B được thiết lập thành công.
- **RTCDataChannel** được mở trên nền **SCTP/UDP**, đảm bảo truyền dữ liệu tin cậy, đúng thứ tự và kiểm soát luồng.
- Dữ liệu file được truyền trực tiếp giữa hai peer mà không đi qua Signaling Server, STUN hay TURN server, trừ trường hợp relay bắt buộc qua TURN.

Nhờ cơ chế này, hệ thống đạt được hiệu năng cao, giảm tải cho server trung tâm và đảm bảo tính phân tán thực sự trong quá trình truyền dữ liệu.

## II. PHÂN TÍCH YÊU CẦU

### 1. Mô tả bài toán

Bài toán đặt ra là xây dựng một hệ thống chia sẻ file ngang hàng (Peer-to-Peer) hoạt động trên nền tảng web, cho phép người dùng truyền tệp tin trực tiếp giữa các trình duyệt mà không cần thông qua server trung tâm để lưu trữ hay trung chuyển dữ liệu.

Hệ thống được triển khai dựa trên công nghệ WebRTC, tận dụng khả năng thiết lập kết nối P2P của trình duyệt thông qua cơ chế Signaling, ICE/STUN và RTCDataChannel để truyền dữ liệu nhị phân. Một server signaling nhẹ được sử dụng với vai trò trung gian ban đầu nhằm trao đổi thông tin kết nối giữa các peer, nhưng không tham gia vào quá trình truyền file.

Ứng dụng cho phép người dùng:

- Kết nối trực tiếp với nhau thông qua Peer ID.
- Thiết lập kết nối P2P ngay cả khi các peer nằm sau NAT hoặc firewall.
- Chia sẻ file dung lượng bất kỳ bằng cách chia nhỏ dữ liệu và truyền qua Data Channel.
- Theo dõi tiến trình gửi và nhận file theo thời gian thực.

## 2. Yêu cầu chức năng

### - Thiết lập kết nối Peer-to-Peer

- Cho phép hai người dùng kết nối trực tiếp với nhau thông qua trình duyệt web.
- Mỗi người dùng được gán một **Peer ID** duy nhất để phục vụ quá trình kết nối.
- Hỗ trợ tự động lựa chọn đường truyền tối ưu thông qua cơ chế ICE (LAN → STUN → TURN nếu cần).
- Hiển thị trạng thái kết nối theo thời gian thực (đang kết nối, đã kết nối, mất kết nối).

### - Chia sẻ và truyền file

- Cho phép người dùng chọn hoặc kéo-thả (drag & drop) file từ thiết bị.
- Chia file thành các khối dữ liệu nhỏ (chunk) có kích thước 16KB để truyền qua RTCDataChannel.
- Truyền dữ liệu trực tiếp giữa các peer, không lưu file trên server.
- Hiển thị tiến trình gửi và nhận file theo thời gian thực (progress bar).
- Tự động ghép (reassemble) các chunk thành file hoàn chỉnh ở phía nhận.
- Cho phép tải xuống file sau khi quá trình truyền hoàn tất.

- **Quản lý peer và phiên kết nối**

- Hiển thị danh sách các peer đang online trong hệ thống.
- Thông báo khi peer tham gia hoặc rời khỏi hệ thống (join/leave).
- Quản lý vòng đời kết nối WebRTC, đảm bảo giải phóng tài nguyên khi ngắt kết nối.

### 3. Yêu cầu phi chức năng

- **Tính toàn vẹn của dữ liệu:** File nhận được phải trùng khớp hoàn toàn với file gửi đi (kiểm soát qua cơ chế truyền tin tin cậy của SCTP).
- **Hiệu năng:** Tốc độ truyền tải phải đạt mức tối đa của băng thông mạng giữa hai Peer, độ trễ thấp.
- **Bảo mật:** Toàn bộ dữ liệu truyền đi phải được mã hóa đầu cuối (End-to-End Encryption) bằng giao thức DTLS.

## III. THIẾT KẾ PHẦN MỀM

### 1. Công nghệ và thư viện sử dụng

Đề đảm bảo tối ưu hóa tốc độ thời gian thực, đảm bảo tính bảo mật và khả năng tương thích cao cho hệ thống truyền file P2P, hệ thống được xây dựng dựa trên các thư viện và công nghệ sau:

**- Frontend:**

- HTML, CSS, JavaScript thuần
- Giao diện đơn giản, tập trung vào chức năng kết nối và chia sẻ file

**- Backend (Signaling Server):**

- Node.js, Express.js
- Socket.io (phục vụ trao đổi thông tin signaling): đây là thư viện quan trọng nhất ở tầng Server. Trong WebRTC, Signaling không có giao thức chuẩn hóa, do đó Socket.io được sử dụng để thiết lập một kênh giao tiếp hai chiều (Bi-directional)

thông qua WebSocket. Nó chịu trách nhiệm trung chuyển các gói tin cấu hình mạng (SDP và ICE Candidates) giữa các Peer.

**- WebRTC:**

- RTCPeerConnection: thiết lập và quản lý kết nối P2P
- RTCDataChannel: truyền dữ liệu file dạng nhị phân

**- STUN Server:**

- **STUN (Session Traversal Utilities for NAT)** hỗ trợ phát hiện địa chỉ public và vượt NAT do hầu hết người dùng nằm sau các bộ định tuyến NAT, IP nội bộ của họ không thể truy cập từ bên ngoài Internet.
- Hệ thống sử dụng các máy chủ STUN miễn phí của Google (ví dụ: stun:stun.l.google.com:19302). Khi Client gửi một truy vấn đến STUN Server, Server này sẽ phản hồi lại địa chỉ IP công cộng (Public IP) và Port mà Client đó đang sử dụng. Thông tin này được đóng gói thành "ICE Candidate" và trao đổi qua Signaling Server để hai Peer có thể "nhìn thấy" nhau và thiết lập đường truyền trực tiếp xuyên qua tường lửa của router.

## 2. Kiến trúc tổng thể

Hệ thống được thiết kế theo mô hình Hybrid P2P, kết hợp giữa signaling tập trung và truyền dữ liệu phân tán hoàn toàn. Kiến trúc gồm hai thành phần chính: **Signaling Server** và **Client (Browser)**.

### 2.1. Signaling Server

Signaling Server được xây dựng bằng Node.js, Express và Socket.io, đóng vai trò trung gian trong giai đoạn thiết lập kết nối ban đầu giữa các peer. Server không lưu trữ và không truyền dữ liệu file, do đó không trở thành điểm nghẽn về băng thông.

Các chức năng chính của Signaling Server bao gồm:

- **Quản lý peer online:**
  Lưu trữ thông tin Peer ID tương ứng với socket ID, cho phép theo dõi trạng thái kết nối của các peer trong room và cập nhật realtime.
- **Chuyển tiếp thông điệp signaling:**
  Trung chuyển các gói tin SDP Offer/Answer và ICE Candidates giữa các peer, hỗ trợ quá trình thiết lập kết nối P2P và vượt NAT/firewall.
- **Thông báo join/leave:**
  Khi một peer tham gia hoặc rời khỏi hệ thống, server broadcast sự kiện đến các peer còn lại để cập nhật danh sách online và trạng thái kết nối.

Nhờ sử dụng Socket.io, quá trình signaling được đảm bảo độ tin cậy cao thông qua WebSocket và cơ chế fallback (long-polling) trong các môi trường mạng khác nhau.

### 2.2. Client (Browser)

Client là nơi thực hiện toàn bộ logic kết nối P2P và truyền file, được triển khai bằng JavaScript thuần sử dụng API WebRTC có sẵn trong trình duyệt.

Các chức năng chính phía client bao gồm:

- **Khởi tạo Peer ID:**
  Mỗi client tự sinh một Peer ID ngẫu nhiên khi tải trang, dùng để chia sẻ và thiết lập kết nối với peer khác.
- **Thiết lập kết nối WebRTC:**
  Tạo đối tượng **RTCPeerConnection** với cấu hình **Google STUN servers**, cho phép thu thập ICE candidates và hỗ trợ NAT traversal tự động.
- **Quản lý RTCDataChannel:**
  Peer gửi tạo DataChannel khi khởi tạo kết nối; peer nhận xử lý sự kiện **ondatachannel** để sử dụng kênh truyền dữ liệu tin cậy.
- **Xử lý truyền file:**
  File được chia thành các khối nhỏ (16KB), gửi qua DataChannel, theo dõi luồng truyền thông qua **bufferedAmount**, hiển thị tiến trình realtime và ghép lại thành file hoàn chỉnh ở phía nhận.

Với kiến trúc này, server chỉ xử lý signaling nhẹ (dữ liệu vài KB), trong khi toàn bộ băng thông truyền file được thực hiện trực tiếp giữa các peer, giúp hệ thống đạt hiệu năng cao và tính phân tán thực sự.

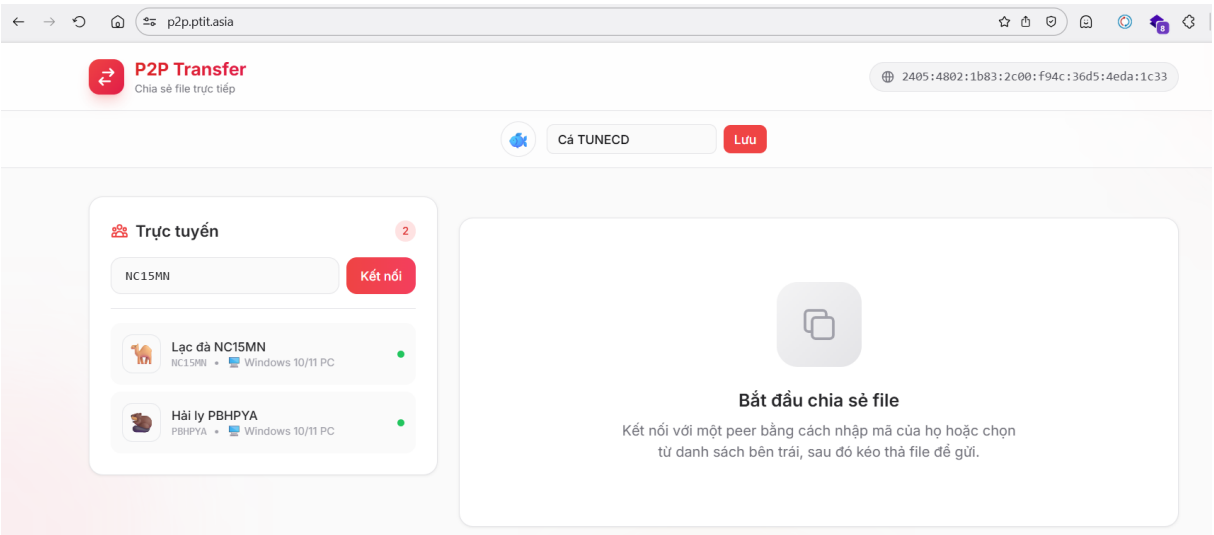
## 3. Demo

Đường dẫn truy cập ứng dụng: https://p2p.ptit.asia/

### Bước 1: Kết nối peers

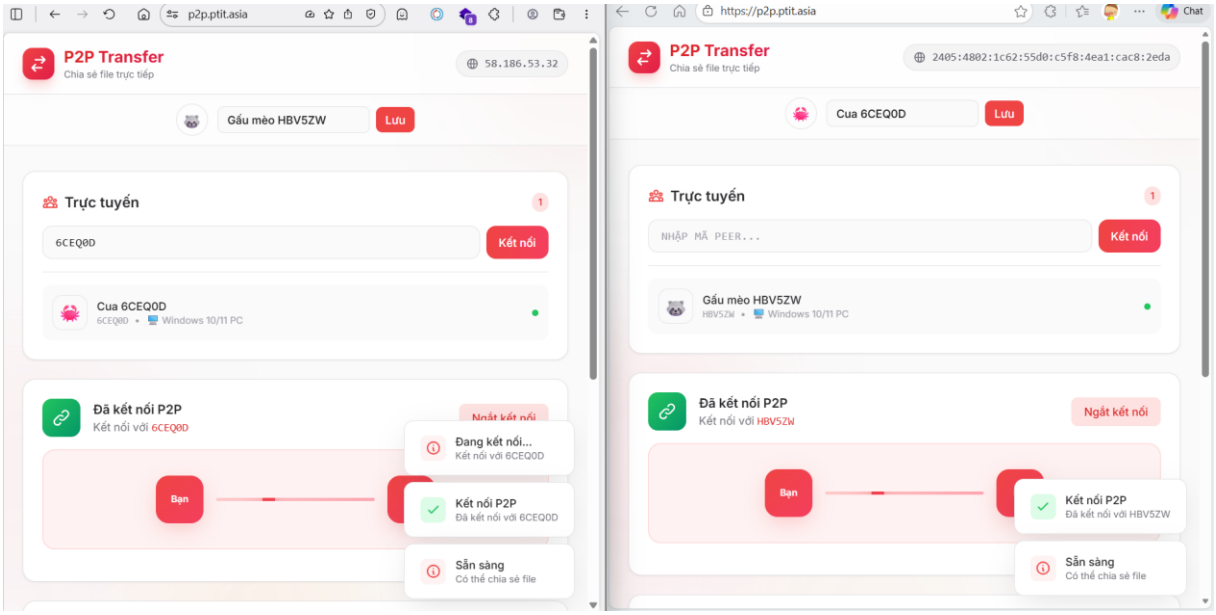
1. User A mở app → nhận được Peer ID (ví dụ: ABC123)
2. User A share Peer ID cho User B
3. User B nhập Peer ID — click "Connect". Ngoài ra ứng dụng còn hiển thị danh sách người dùng đang Trực tuyến, có thể click vào tên người dùng để kết nối.
4. WebRTC handshake tự động diễn ra
5. Hiển thị thông báo "Đã kết nối" khi thành công và trạng thái Sẵn sàng chia sẻ file

Khi truy cập trình duyệt Web, user A được cấp Peer ID HBV5ZW, user B được cấp Peer ID 6CEQ0D. Trên giao diện hiển thị danh sách user đang Trực tuyến, sẵn sàng kết nối.




*Hình 4 Danh sách user trực tuyến sẵn sàng kết nối để chia sẻ file*

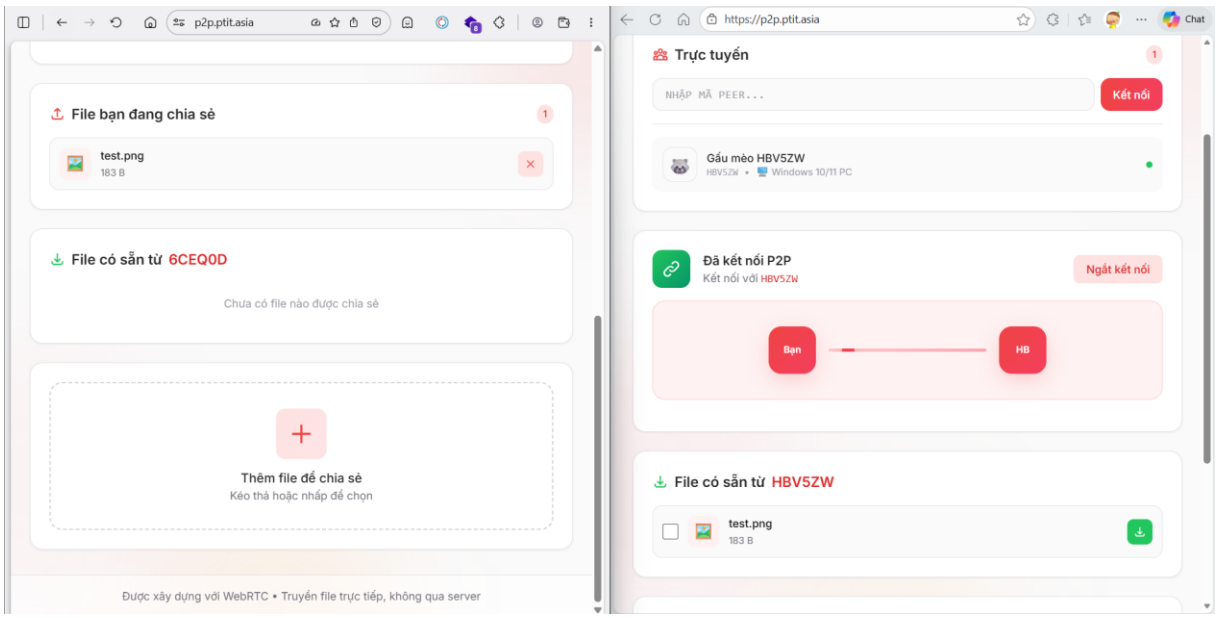
User A nhập Peer ID 6CEQ0D của user B và ấn Kết nối, WebRTC sẽ tự động thiết lập kết nối và hiển thị thông báo thành công, sẵn sàng chia sẻ file.




*Hình 5 Quá trình kết nối Peer giữa 2 User*

**Bước 2: Gửi file**

1. User A drag & drop file vào vùng upload
2. File được chunk thành 16KB và gửi qua Data Channel
3. User B thấy progress bar nhận file, hiển thị file có sẵn từ User A
4. User B click download và file sẽ được tải xuống khi hoàn tất




*Hình 6 Quá trình chia sẻ file giữa 2 user*

## IV. KẾT LUẬN

Sau quá trình nghiên cứu và thực hiện đề tài **"Hệ thống chia sẻ file ngang hàng "**, hệ thống đã đạt được các mục tiêu đề ra về mặt kỹ thuật và tính ứng dụng.

### 1. Kết quả đạt được

- **Về kỹ thuật:**
    - Triển khai thành công ứng dụng chia sẻ file P2P hoàn chỉnh dựa trên công nghệ WebRTC.
    - Xây dựng hệ thống Signaling Server hoạt động ổn định trên nền tảng Node.js và Socket.io.
    - Tối ưu hóa quy trình truyền dữ liệu qua Data Channel với cơ chế Reliable (SCTP).
- **Về tính năng:**
    - Kết nối ngang hàng thành công giữa các trình duyệt khác nhau thông qua cơ chế Peer ID.

- Hỗ trợ truyền tải tệp tin đa định dạng với tốc độ cao (đặc biệt nhanh trong mạng LAN).
- Giao diện người dùng trực quan, hỗ trợ kéo thả (Drag & Drop) và hiển thị tiến trình (Progress bar) thời gian thực.

**2. Hạn chế tồn tại**

- Signaling Server vẫn là điểm trung tâm, nếu lỗi sẽ không thiết lập kết nối mới.
- Kết nối giảm khi gặp NAT nghiêm ngặt (Symmetric NAT) do chưa triển khai TURN server riêng.
- Chưa hỗ trợ Resume khi kết nối bị gián đoạn.

**3. Phương hướng phát triển đề tài**

- Triển khai TURN server để cải thiện tỷ lệ kết nối trong môi trường NAT phức tạp.
- Cải thiện trải nghiệm người dùng: giao diện trực quan, QR Code trao đổi Peer ID, phát triển ứng dụng di động và desktop.

# TÀI LIỆU THAM KHẢO


[1] "WebRTC connectivity," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Connectivity. [Accessed 28 12 2025].

[2] A. M. G, "What is WebRTC? A Beginner's Guide to Real-Time Communication," [Online]. Available: https://www.fastpix.io/blog/what-is-webrtc-a-beginners-guide-to-real-time-communication. [Accessed 28 12 2025].

[3] a. dhamija, "P2P (Peer To Peer) File Sharing," [Online]. Available: https://www.geeksforgeeks.org/computer-networks/p2p-peer-to-peer-file-sharing/. [Accessed 28 12 2025].

[4] P. T. P. V. Cường and T. N. X. Anh, Giáo trình Các hệ thống phân tán, Hà Nội, 2024.

[5] S. Dutton, "Get started with WebRTC," [Online]. Available: https://web.dev/articles/webrtc-basics. [Accessed 28 12 2025].