TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

ĐỒ ÁN

**TỐT NGHIỆP ĐẠI HỌC**

NGÀNH CÔNG NGHỆ THÔNG TIN

**MÔ HÌNH CHIA SẺ BỘ NHỚ ĐỆM QUA P2P**

Sinh viên thực hiện : **Lê Hoài Nam**

Lớp AS – K55

Giáo viên hướng dẫn: PGS.TS **Hà Quốc Trung**

HÀ NỘI 6-2015

**PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

1. Thông tin về sinh viên

Họ và tên sinh viên: Lê Hoài Nam

Điện thoại liên lạc: 01669806628 Email: lhnam298@gmail.com

Lớp: AS-K55 Hệ đào tạo: Chính quy

Đồ án tốt nghiệp được thực hiện tại: Đại học bách khoa Hà Nội

Thời gian làm ĐATN: Từ ngày 1/10/2014 đến 31/5/2015

2. Mục đích nội dung của ĐATN

Mục đích nội dung của đồ án tốt nghiệp là sử dụng kết hợp giữa mô hình client – server và mô hình P2P xây dựng một hệ thống chia sẻ file nhằm tích hợp những ưu điểm của hai mô hình trên vào trong một mô hình duy nhất.

3. Các nhiệm vụ cụ thể của ĐATN

* Tìm hiểu về mô hình client – server
* Tìm hiểu về mô hình peer-to-peer
* Tìm hiểu về các phương pháp kết hợp mô hình client – server và mô hình peer-to-peer
* Xây dựng ứng dụng chia sẻ file theo phương pháp kết hợp mô hình client – server và mô hình peer-to-peer
* Thực hiện mô phỏng trên máy ảo
* Đánh giá kết quả đạt được

4. Lời cam đoan của sinh viên:

Tôi cam Lê Hoài Nam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của PGS.TS Hà Quốc Trung. Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

|  |  |
| --- | --- |
|  | *Hà Nội, ngày tháng năm*  Tác giả ĐATN  *Lê Hoài Nam* |

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ

|  |  |
| --- | --- |
|  | *Hà Nội, ngày tháng năm*  Giáo viên hướng dẫn  *PGS.TS Hà Quốc Trung* |

**LỜI CẢM ƠN**

Trước hết, em xin gửi lời cảm ơn chân thành tới PGS.TS Hà Quốc Trung, người đã tận tình dạy dỗ và hướng dẫn em trong quá trình hoàn thành đồ án cũng như trong học tập.

Đồng thời, em xin bày tỏ lòng biết ơn đến các thầy cô giáo trong Viện Công nghệ thông tin và Truyền thông – trường Đại học Bách Khoa Hà Nội, những người đã tận tâm giảng dạy, truyền đạt cho chúng em những kiến thức cơ bản làm nền tảng cho việc thực hiện đồ án cũng như trong công việc sau này.

Em cũng xin gửi lời cảm ơn tới các anh chị khóa trên, các bạn trong nhóm sinh viên nghiên cứu, những người luôn ở bên cạnh giúp đỡ em trong quá trình thực hiện đồ án.

Cuối cùng, tôi xin gửi lời cám ơn tới gia đình, bạn bè, những người luôn cổ vũ động viên tôi hoàn thiện đồ án này.

Hà Nội, ngày 20 tháng 05 năm 2015

Lê Hoài Nam

**TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP**

Hiện nay, trong khi nhu cầu sử dụng internet ngày càng tăng cao, hầu hết các ứng dụng trên mạng internet được xây dựng theo mô hình client-server đều vướng phải vấn đề là tắc nghẽn do nút cổ chai và quá tải về băng thông. Trong khuôn khổ đồ án tốt nghiệp này, em xin trình bày các giải pháp kết hợp giữa mô hình client-server và mô hình peer-to-peer, nhằm đưa ra một mô hình mà khi được ứng dụng các máy client không chỉ tương tác trực tiếp với server mà còn có thể chia sẻ dữ liệu cho nhau. Điều này có thể góp phần cải thiện đáng kể hiệu năng hệ thống.

Cấu trúc của đồ án gồm 5 phần với nội dung chính như sau:

* Phần I đưa ra các vấn đề và định hướng giải quyết trong khuôn khổ của đồ án.
* Phần II trình bày về cơ sở lý thuyết các mô hình client – server, mô hình peer-to-peer và các giải pháp kết hợp hai mô hình nêu trên.
* Phần III trình bày về các công nghệ được sử dụng trong việc cài đặt ứng dụng chia sẻ file.
* Phần IV trình bày đặc tả các chức năng của ứng dụng theo hai mô hình: mô hình client – server truyền thống và mô hình share cache.
* Phần V trình bày kết quả thực nghiệm, nhận xét và đánh giá hiệu quả của giải pháp thực hiện.

**MỤC LỤC**

[I. Đặt vấn đề 10](#_Toc419733746)

[II. Cơ sở lý thuyết 11](#_Toc419733747)

[1. Tổng quan về mô hình client – server 11](#_Toc419733748)

[1.1. Khái niệm 11](#_Toc419733749)

[1.2. Giao tiếp giữa client – server 12](#_Toc419733750)

[1.3. Kiến trúc client – server 12](#_Toc419733751)

[1.4. Ưu và nhược điểm của mô hình client - server 14](#_Toc419733752)

[2. Tổng quan về mô hình peer-to-peer 15](#_Toc419733753)

[2.1. Định nghĩa 15](#_Toc419733754)

[2.2. Kiến trúc mạng peer-to-peer 16](#_Toc419733755)

[2.3. Phân loại mô hình peer-to-peer theo cấu trúc 17](#_Toc419733756)

[2.4. Ưu và nhược điểm của mô hình peer-to-peer 19](#_Toc419733757)

[3. Giải pháp kết hợp mô hình client-server và peer-to-peer 20](#_Toc419733758)

[3.1. Nhận xét 20](#_Toc419733759)

[3.2. Caching model 21](#_Toc419733760)

[3.3. Shared caching model 23](#_Toc419733761)

[3.4. Local proxy model 24](#_Toc419733762)

[III. Công nghệ sử dụng 27](#_Toc419733763)

[1. WebRTC 27](#_Toc419733764)

[1.1. Giới thiệu 27](#_Toc419733765)

[1.2. Cấu trúc WebRTC 29](#_Toc419733766)

[1.3. Phương thức hoạt động 30](#_Toc419733767)

[2. Node.js 33](#_Toc419733768)

[3. HTML5 33](#_Toc419733769)

[IV. Đặc tả ứng dụng chia sẻ tệp 35](#_Toc419733770)

[1. Ứng dụng chia sẻ tệp theo mô hình client – server 35](#_Toc419733771)

[1.1. Chức năng đăng ký tài khoản 35](#_Toc419733772)

[1.2. Chức năng đăng nhập 36](#_Toc419733773)

[1.3. Chức năng đăng xuất 37](#_Toc419733774)

[1.4. Chức năng upload 38](#_Toc419733775)

[1.5. Chức năng xóa tệp khỏi hệ thống 39](#_Toc419733776)

[1.6. Chức năng download 40](#_Toc419733777)

[2. Ứng dụng chia sẻ tệp theo mô hình shared-cache 41](#_Toc419733778)

[2.1. Nhận xét về ứng dụng chia sẻ tệp theo mô hình client – server 41](#_Toc419733779)

[2.2. Chức năng đăng ký tài khoản 42](#_Toc419733780)

[2.3. Chức năng đăng nhập 44](#_Toc419733781)

[2.4. Chức năng đăng xuất: Như bên phần mô hình client – server 45](#_Toc419733782)

[2.5. Chức năng upload: Như bên phần mô hình client – server 45](#_Toc419733783)

[2.6. Chức năng xóa tệp khỏi hệ thống 45](#_Toc419733784)

[2.7. Chức năng download 46](#_Toc419733785)

[V. Cài đặt thử nghiệm và đánh giá 48](#_Toc419733786)

[1. Cài đặt chương trình 48](#_Toc419733787)

[2. Kịch bản thử nghiệm 49](#_Toc419733788)

[3. Đánh giá 51](#_Toc419733789)

[4. Kết luận 51](#_Toc419733790)

**DANH MỤC HÌNH VẼ**

**DANH MỤC CÁC BẢNG**

**DANH MỤC CÁC TỪ VIẾT TẮT VÀ THUẬT NGỮ**

# Đặt vấn đề

* Hiện nay, mạng internet đã trở thành một phần không thể thiếu với đời sống con người. Các ứng dụng trên mạng internet được phát triển hằng ngày để đáp ứng nhu cầu sử dụng. Hai mô hình đang được áp dụng phổ biến nhất trong các ứng dụng internet hiện nay là mô hình client-server và mô hình peer-to-peer.
* Mô hình client-server có một ưu điểm dễ nhận thấy nhất là tính đơn giản của nó. Tuy nhiên việc xử lí các nhiệm vụ trong mô hình client-server chỉ tập trung trên máy chủ, với lưu lượng sử dụng mạng lớn như hiện nay và có xu hướng ngày còn tăng cao, các máy chủ sẽ có thời điểm bị rơi vào tình trạng quá tải, khả năng mở rộng của hệ thống yếu do phụ thuộc vào khả năng xử lý của máy chủ.
* Mô hình peer-to-peer giải quyết những hạn chế của mô hình client-server bằng cách phân phối khối lượng công việc cho tất cả các nút tham gia vào hệ thống. Sử dụng mô hình này, sự tương tác giữa các thành phần trong hệ thống được thay đổi, vì vậy các giao thức và các ứng dụng thiết kế phức tạp hơn. Thay vì thiết kế một giao thức giữa máy khách và máy chủ, bây giờ phải xét đến sự tương tác phức tạp giữa các thiết bị ngang hàng.
* Xuất phát từ ưu điểm của cả hai mô hình: khả năng mở rộng của mô hình peer-to-peer và sự đơn giản của mô hình client-server, trong các phần tiếp theo của đồ án sẽ hướng đến các giải pháp kết hợp mô hình client – server và mô hình peer-to-peer, cài đặt thử nghiệm ứng dụng chia sẻ file dựa trên một trong các giải pháp kết hợp.

# Cơ sở lý thuyết

## Tổng quan về mô hình client – server

### Khái niệm

* [Mô hình client-server](http://vdo.vn/kien-thuc/client-server-la-gi.html)  là một mô hình nổi tiếng trong 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 (response) về cho máy khách.



##### Hình 1: Mô hình client – server

* Server : Là nơi cung cấp xử lí hầu hết các nghiệp vụ. Server còn được định nghĩa như là một máy tính nhiều người sử dụng (multiuser computer). Vì một server phải quản lý nhiều yêu cầu từ các client trên mạng cho nên nó hoạt động sẽ tốt hơn nếu hệ điều hành của nó là đa nhiệm với các tính năng hoạt động độc lập song song với nhau như hệ điều hành UNIX, WINDOWS... Server cung cấp và điều khiển các tiến trình truy cập vào tài nguyên của hệ thống. Các ứng dụng chạy trên server phải được tách rời nhau để một lỗi của ứng dụng này không làm hỏng ứng dụng khác.
* Client có nhiệm vụ tương tác với người dùng, nhận thông tin người dùng và hiển thị kết quả còn mọi thông tin đều được xử lí trên server. Thực tế trong các ứng dụng của mô hình client/server, các chức năng hoạt động chính là sự kết hợp giữa client và server với sự chia sẻ tài nguyên, dữ liệu trên cả 2 máy. Trong mô hình client/server, client được coi như là người sử dụng các dịch vụ trên mạng do một hoặc nhiều máy chủ cung cấp và server được coi như là người cung cấp dịch vụ để trả lời các yêu cầu của các clients. Một máy client trong mô hình này lại có thể là server trong một mô hình khác.

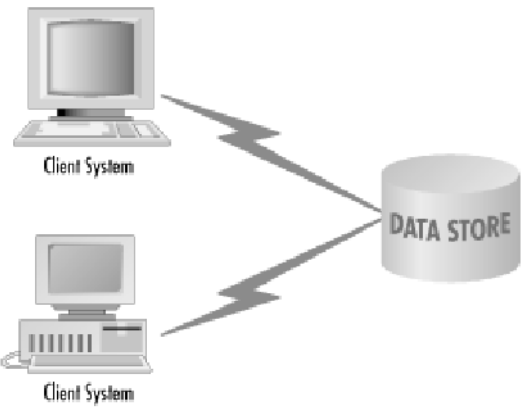
### Giao tiếp giữa client – server

* Việc giao tiếp giữa client và server được thực hiện dưới hình thức trao đổi các thông điệp (message). Để được phục vụ, client sẽ gửi một thông điệp yêu cầu (request message) mô tả về công việc muốn server thực hiện. Khi nhận được thông điệp yêu cầu, server tiến hành phân tích để xác định công việc cần phải thực thi. Nếu việc thực hiện yêu cầu này có sinh ra kết quả trả về, server sẽ gửi nó cho client trong một thông điệp trả lời (reply message). Dạng thức (format) và ý nghĩa của các thông điệp trao đổi giữa client và server được quy định rõ bởi giao thức (protocol) của ứng dụng.
* Các chế độ giao tiếp: Quá trình giao tiếp giữa client và server có thể diễn ra theo hai chế độ là nghẽn (blocked) hay không nghẽn (Non blocked).
* Chế độ nghẽn: Trong chế độ này, khi quá trình client hay server phát ra lệnh gửi dữ liệu (thông thường bằng lệnh send), sự thực thi của nó sẽ bị tạm dừng cho đến khi quá trình nhận phát ra lệnh nhận số dữ liệu đó (thường là lệnh receive). Tương tự cho trường hợp nhận dữ liệu, nếu  quá trình nào đó, client hay server, phát ra lệnh nhận dữ liệu, mà ở thời điểm đó chưa có dữ liệu gửi đến, sự thực thi của nó cũng tạm dừng cho đến khi có dữ liệu gửi đến.
* Chế độ không nghẽn: Trong chế độ này, khi quá trình client hay server phát ra lệnh gửi dữ liệu, sự thực thi của nó vẫn được tiếp tục mà không quan tâm đến việc có quá trình nào phát ra lệnh nhận số dữ liệu đó hay không. Tương tự cho trường hợp nhận dữ liệu, khi quá trình phát ra lệnh nhận dữ liệu, nó sẽ nhận được số lượng dữ liệu hiện có (bằng 0 nếu chưa có quá trình nào gửi dữ liệu đến). Sự thực thi của quá trình vẫn được tiếp tục.

### Kiến trúc client – server

1. Client/ Server hai tầng

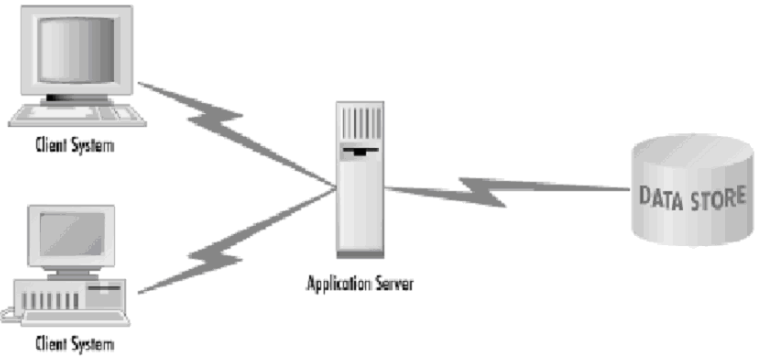
* Là kiến trúc client/server đơn giản nhất. Trong thực tế hầu hết các kiến trúc client/server là kiến trúc hai tầng. Một ứng dụng hai tầng cung cấp nhiều trạm làm   
  việc với một tầng trình diễn thống nhất, tầng này truyền tin với tầng lưu trữ dữ liệu tập trung. Tầng trình diễn thông thường là client và tầng lưu trữ dữ liệu là server.
* Trong ứng dụng hai tầng truyền thống, khối lượng công việc xử lý được dành cho   
  phía client trong khi server chỉ đơn giản đóng vai trò như là chương trình kiểm soát luồng vào ra giữa ứng dụng và dữ liệu. Kết quả là không chỉ hiệu năng của ứng dụng bị giảm đi do tài nguyên hạn chế của PC, mà khối lượng dữ liệu truyền đi trên mạng cũng tăng theo. Khi toàn bộ ứng dụng được xử lý trên một PC, ứng dụng bắt buộc phải yêu cầu nhiều dữ liệu trước khi đưa ra bất kỳ kết quả xử lý nào cho người dùng. Nhiều yêu cầu dữ liệu cũng làm giảm hiệu năng của mạng. Một vấn đề thường gặp khác đối với ứng dụng hai tầng là vấn đề bảo trì. Chỉ cần một thay đổi nhỏ đối với ứng dụng cũng cần phải thay đổi lại toàn bộ ứng dụng client và server.



##### Hình 2: Kiến trúc client/ server hai tầng

1. Client/ Server ba tầng

* Ta có thể tránh được các vấn đề của kiến trúc client/server hai tầng bằng cách mở rộng kiến trúc thành ba tầng. Một kiến trúc ba tầng có thêm một tầng mới tác biệt việc xử lý dữ liệu ở vị trí trung tâm.
* Theo kiến trúc ba tầng, một ứng dụng được chia thành ba tầng tách biệt nhau về mặt logic. Tầng đầu tiên là tầng trình diễn thường bao gồm các giao diện đồ họa. Tầng thứ hai, còn được gọi là tầng trung gian hay tầng tác nghiệp. Tầng thứ ba chứa dữ liệu cần cho ứng dụng. Tầng thứ ba về cơ bản là chương trình thực hiện các lời gọi hàm để tìm kiếm dữ liệu cần thiết. Tầng trình diễn nhận dữ liệu và định dạng nó để hiển thị. Sự tách biệt giữa chức năng xử lý với giao diện đã tạo nên sự linh hoạt cho việc thiết kế ứng dụng. Nhiều giao diện người dùng được xây dựng và triển khai mà không làm thay đổi logic ứng dụng. Tầng thứ ba chứa dữ liệu cần thiết cho ứng dụng. Dữ liệu này có thể bao gồm bất kỳ nguồn thông tin nào, bao gồm cơ sở dữ liệu như Oracale, SQL Server hoặc tài liệu XML.



##### Hình 3: Kiến trúc client/ server hai tầng

1. Kiến trúc n-tầng

Kiến trúc n-tầng được chia thành các tầng như sau:

* Tầng giao diện người dùng: quản lý tương tác của người dùng với ứng dụng
* Tầng logic trình diễn: Xác định cách thức hiển thị giao diện người dùng và các yêu cầu của người dùng được quản lý như thế nào.
* Tầng logic tác nghiệp: Mô hình hóa các quy tắc tác nghiệp
* Tầng các dịch vụ hạ tầng: Cung cấp một chức năng bổ trợ cần thiết cho ứng dụng như các thành phần (truyền thông điệp, hỗ trợ giao tác).

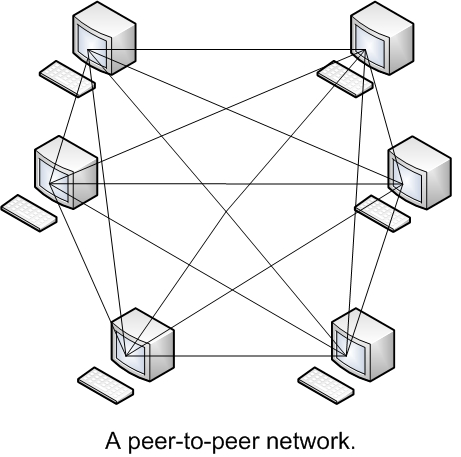
### Ưu và nhược điểm của mô hình client - server

* Ưu điểm:
* Quản lí tài nguyên một cách tập trung vì mọi dữ liệu đều được lưu trữ ở một nơi.
* Có thể back-up và phục hồi: Tất cả các dữ liệu được lưu trữ trên máy chủ dễ dàng để thực hiện back-up. Ngoài ra, trong trường hợp của một số hỏng hóc nếu dữ liệu bị mất, nó có thể được phục hồi dễ dàng và hiệu quả.
* Có khả năng nâng cấp và mở rộng: Thay đổi có thể được thực hiện dễ dàng bằng cách chỉ cần nâng cấp máy chủ. Ngoài ra nguồn lực mới và hệ thống có thể được thêm bằng cách thêm những thay đổi cần thiết trong máy chủ.
* Khả năng tiếp cận dễ dàng từ các nền tảng trong hệ thống.
* Những thông tin mới được đăng tải trên cơ sở dữ liệu, mỗi máy trạm không cần phải có khả năng lưu trữ của nó tăng lên. Tất cả các thay đổi được thực hiện chỉ trong máy tính trung tâm mà cơ sở dữ liệu máy chủ này tồn tại.
* Quyền bảo mật và truy cập có thể được xác định tại thời điểm thiết lập của máy chủ.
* Nhược điểm:
* Tắc nghẽn trong mạng: Quá nhiều yêu cầu từ các khách hàng có thể dẫn đến tắc nghẽn. Tình trạng quá tải có thể dẫn đến làm sập các máy chủ.
* Một khi máy chủ gặp sự số, toàn bộ hệ thống sẽ bị ảnh hưởng theo.
* Chi phí rất tốn kém: Các máy chủ trung tâm phải đủ mạnh để duy trì và chia sẻ tài nguyên với các máy tính khác trên mạng.
* Cần có đội ngũ để quản lý và bảo trì hệ thống.

## Tổng quan về mô hình peer-to-peer

### Định nghĩa

Hệ thống peer-to-peer là hệ thống bao gồm các nút mạng kết nối với nhau, có khả năng tự tổ chức hình thái mạng nhằm mục đích chia sẻ tài nguyên phân tán trong môi trường mạng mà không cần thông qua một dịch vụ trung tâm

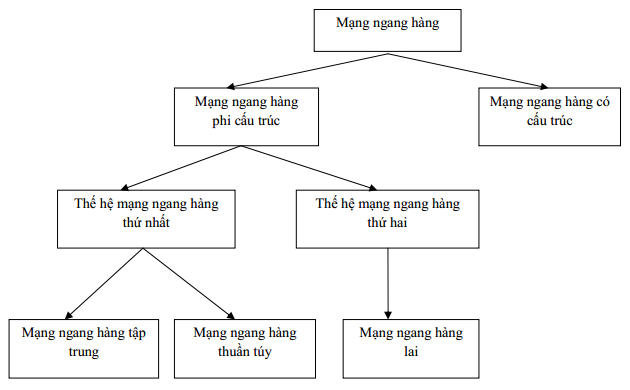


##### Hình 4: Mô hình peer-to-peer

### Kiến trúc mạng peer-to-peer

* Mạng đồng đẳng không cấu trúc (Unstructured):
* Một mạng đồng đẳng không cấu trúc khi các liên kết giữa các nút mạng trong mạng phủ được thiết lập ngẫu nhiên (tức là không theo quy luật nào).
* Những mạng như thế này dễ dàng được xây dựng vì một máy mới khi muốn tham gia mạng có thể lấy các liên kết có sẵn của một máy khác đang ở trong mạng và sau đó dần dần tự bản thân nó sẽ thêm vào các liên kết mới của riêng mình.
* Khi một máy muốn tìm một dữ liệu trong mạng đồng đẳng không cấu trúc, yêu cầu tìm kiếm sẽ được truyền trên cả mạng để tìm ra càng nhiều máy chia sẻ càng tốt.
* Hệ thống này thể hiện rõ nhược điểm: không có gì đảm bảo tìm kiếm sẽ thành công. Đối với tìm kiếm các dữ liệu phổ biến được chia sẻ trên nhiều máy, tỉ lệ thành công là khá cao, ngược lại, nếu dữ liệu chỉ được chia sẻ trên một vài máy thì xác suất tìm thấy là khá nhỏ. Tính chất này là hiển nhiên vì trong mạng đồng đẳng không cấu trúc, không có bất kì mối tương quan nào giữa một máy và dữ liệu nó quản lý trong mạng, do đó yêu cầu tìm kiếm được chuyển một cách ngẫu nhiên đến một số máy trong mạng. Số lượng máy trong mạng càng lớn thì khả năng tìm thấy thông tin càng nhỏ. Một nhược điểm khác của hệ thống này là do không có định hướng, một yêu cầu tìm kiếm thường được chuyển cho một số lượng lớn máy trong mạng làm tiêu tốn một lượng lớn băng thông của mạng, dẫn đến hiệu quả tìm kiếm chung của mạng thấp.
* Mạng ngang hàng phi cấu trúc được chia thành ba mô hình mạng ngang hàng khác: mạng ngang hàng tập trung, mạng ngang hàng thuần túy, mạng ngang hàng lai
* Mạng đồng đẳng có cấu trúc (Structured):
* Mạng đồng đẳng có cấu trúc khắc phục nhược điểm của mạng không cấu trúc bằng cách sử dụng hệ thống DHT (Bảng băm phân tán - Distributed Hash Table).
* Hệ thống này định nghĩa liên kết giữa các nút mạng trong mạng phủ theo một thuật toán cụ thể, đồng thời xác định chặt chẽ mỗi nút mạng sẽ chịu trách nhiệm đối với một phần dữ liệu chia sẻ trong mạng.
* Với cấu trúc này, khi một máy cần tìm một dữ liệu, nó chỉ cần áp dụng một giao thức chung để xác định nút mạng nào chịu trách nhiệm cho dữ liệu đó và sau đó liên lạc trực tiếp đến nút mạng đó để lấy kết quả.

### Phân loại mô hình peer-to-peer theo cấu trúc



##### Hình 5: Phân loại mạng ngang hàng

* + 1. Mạng ngang hàng tập trung
* Mạng ngang hàng tập trung là một trong những thế hệ mạng ngang hàng đầu tiên, đặc trưng của mạng này vẫn dựa vào một máy chủ tìm kiếm trung tâm. Topo xếp chồng của một mạng ngang hàng tập trung do đó có thể được miêu tả như là một mạng hình sao.
* Trong mô hình mạng này, mỗi điểm nút kết nối tới máy chủ tìm kiếm trung tâm để có thể gửi truy vấn tìm kiếm tài nguyên, sau khi gửi yêu cầu tới máy chủ tìm kiếm trung tâm, máy chủ tìm kiếm trung tâm trả về thông tin phản hồi tương ứng với từ khóa được quy định trong truy vấn. Tức là tại máy chủ tìm kiếm trung tâm, từ khóa trong thông báo truy vấn sẽ được ánh xạ với bảng danh sách tài nguyên mà máy chủ có. Nếu máy chủ tìm kiếm trung tâm có thông tin mà điểm nút đó yêu cầu thì nó sẽ trả về thông tin vị trí truy cập tới các điểm nút chia sẻ (đa phần là trả về các địa chỉ IP và các cổng). Sau khi điểm nút đã nhận được thông tin từ máy chủ tìm kiếm trung tâm thì lúc này quá trình trao đổi thông tin cần tìm được thực hiện theo đúng cơ chế của mạng ngang hàng, tức là trao đổi trực tiếp giữa các nút mạng với nhau mà không cần qua máy chủ tìm kiếm trung tâm.
* Như vậy, trong mô hình này bao gồm :
* Một máy chủ tìm kiếm trung tâm, máy chủ này chứa danh sách thông tin về các điểm nút trong mạng do nó quản lý (bao gồm: địa chỉ IP, cổng, băng thông kết nối, …) và danh sách thông tin về các tài nguyên (tên tài nguyên, dung lượng tài nguyên, kiểu tài nguyên, …) mà mỗi điểm nút trong mạng chia sẻ.
* Các điểm nút: các điểm nút này lưu trữ các tài nguyên cần chia sẻ với các điểm nút khác trong mạng.
* Cơ chế hoạt động của mô hình này bao gồm 2 hoạt động: hoạt động giữa các điểm nút với máy chủ tìm kiếm trung tâm, hoạt động giữa các điểm nút với nhau.
* Ưu điểm :
* Tìm kiếm nhanh và hiệu quả.
* Quản lý tập trung, quản trị tin cậy.
* Dễ xây dựng.
* Nhược điểm:
* Dễ dàng bị tấn công.
* Khả năng tắc nghẽn.
* Khó mở rộng.
* Cần quản trị.
  + 1. Mạng ngang hàng thuần túy
* Mạng ngang hàng thuần túy là một kiểu mạng của thế hệ mạng ngang hàng thứ nhất, đặc trưng nổi bật của mô hình này là không có máy chủ tìm kiếm tập trung như trong mô hình mạng ngang hàng tập trung, do đó nó không gặp phải vấn đề nút cổ chai. Các điểm nút giao tiếp trực tiếp với điểm nút khác trong mạng mà không cần các máy chủ trung tâm riêng biệt nào, các điểm nút thiết lập kết nối với nhau ngẫu nhiên.
* Trong mô hình mạng ngang hàng này, việc tìm kiếm file sử dụng phương pháp phát tràn (phương pháp này có sử dụng giá trị giới hạn phạm vi tìm kiếm là TTL và sử dụng GUID để trao đổi). Khi muốn tìm kiếm một file nào đó thì yêu cầu tìm kiếm được gửi từ điểm nút nguồn tới tất cả các điểm nút mạng là hàng xóm của nó. Đây vừa là đặc trưng của mạng ngang hàng thuần túy và cũng là một điểm yếu của các mạng ngang hàng này bởi vì phương pháp tìm kiếm làm tăng đáng kể lưu lượng trong mạng và gây ra dư thừa hay trùng lặp thông báo truy vấn. Nếu tài nguyên được tìm thấy là tồn tại thì khi đó điểm nút có tài nguyên chia sẻ sẽ trao đổi với điểm nút yêu cầu dựa vào GUID của điểm nút yêu cầu.
* Ưu điểm:
* Không có điểm duy nhất chịu lỗi nên khó bị tấn công.
* Có thể thích nghi với mạng vật lý.
* Cho phép nặc danh.
* Dễ xây dựng.
* Các điểm nút tham gia và rời khỏi mạng một cách tùy mà không ảnh hưởng đến cấu trúc của toàn mạng.
* Nhược điểm:
* Tốn tài nguyên băng thông.
* Tìm kiếm phức tạp
* Các điểm nút có khả năng khác nhau (sức mạnh xử lý của CPU, băng thông, không gian lưu trữ, …) đều có thể phải chịu tải như nhau.
  + 1. Mạng ngang hàng lai
* Mạng ngang hàng lai là mạng ngang hàng thuộc thế hệ thứ hai. Chúng được phát triển để khắc phục nhược điểm của các mô hình mạng ngang hàng trước đó. Mô hình mạng ngang hàng lai bao gồm các: các siêu điểm nút, các điểm nút thông thường (hay còn gọi là điểm nút client). Trong đó các siêu điểm nút tạo thành một mạng không có cấu trúc, và mỗi siêu điểm nút kết nối đến nhiều điểm nút thông thường, mỗi siêu điểm nút quản lý vùng của nó, vai trò của siêu điểm nút giống như một máy chủ trong mô hình mạng ngang hàng tập trung.
* Ưu điểm:
* Không có điểm duy nhất chịu lỗi vì có nhiều siêu điểm nút.
* Cho phép nặc danh.
* Phù hợp với các nhóm lợi ích đặc biệt.
* Khả năng mở rộng quy mô tốt.
* Hiệu quả thỏa mãn các truy vấn.
* Hạn chế phát tràn các truy vấn và tránh được hiện tượng nút cổ chai.
* Nhược điểm: Phân chịu tải không cân bằng - các siêu điểm nút chịu tải cao hơn.

### Ưu và nhược điểm của mô hình peer-to-peer

* Ưu điểm:
* Một mục đích quan trọng của mạng ngang hàng là trong mạng tất cả các máy tham gia đều đóng góp tài nguyên, bao gồm băng thông, lưu trữ, và khả năng tính toán. Do đó khi càng có nhiều máy tham gia mạng thì khả năng tổng thể của hệ thống mạng càng lớn. Ngược lại, trong cấu trúc máy chủ-máy khách, nếu số lượng máy chủ là cố định, thì khi số lượng máy khách tăng lên khả năng chuyển dữ liệu cho mỗi máy khách sẽ giảm xuống.
* Tính chất phân tán của mạng ngang hàng cũng giúp cho mạng hoạt động tốt khi một số máy gặp sự cố. Đối với cấu trúc tập trung, chỉ cần máy chủ gặp sự cố thì cả hệ thống sẽ ngưng trệ. Còn đối với mạng ngang hàng các máy tính có thể tham gia và rời khỏi mạng bất kì lúc nào mà mạng vẫn hoạt động bình thường, các máy tính còn lại vẫn có thể trao đổi thông tin và chia sẻ tài nguyên với nhau.
* Trong mạng ngang hàng dữ liệu trên các máy tính được đem ra chia sẽ nên một máy tính có thể thực hiện vai trò giống server để chia sẽ cho các máy tính khác. Các máy tính sau khi được chia sẻ dữ liệu cũng có thể tham gia chia sẻ cho các máy tính khác. Như vậy sẽ tăng số bản sao dữ liệu và giúp cho việc chia sẻ dữ liệu nhanh chóng.
* Nhược điểm:
* Nhược điểm của P2P là vì tài nguyên của mạng nằm trên các máy tính cá nhân và không phải lúc nào các máy này cũng liên kết với mạng nên có thể dẫn tới sư biến mất của một dịch vụ trong khoảng thời gian nhất định. Tuy nhiên nhược điểm này có thể khắc phục khi số lượng thành viên trên mạng P2P đủ lớn, lúc đó sẽ có nhiều điểm cung cấp dịch vụ cho toàn mạng hơn.

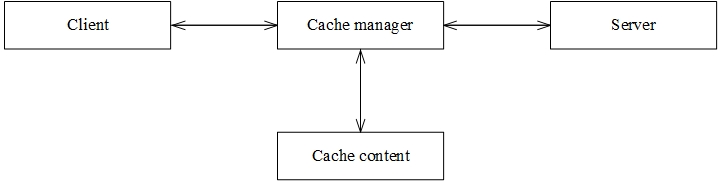
## Giải pháp kết hợp mô hình client-server và peer-to-peer

### Nhận xét

* Mô hình client – server truyền thống gặp phải 2 khó khăn cơ bản là:
* Nút thắt cổ chai do tất cả mọi yêu cầu từ client đều thông qua một server chung và do chính server đó quản lý
* Khả năng mở rộng hệ thống với cộng đồng do yêu cầu về chi phí cho tài nguyên máy chủ, chi phí quản trị, …
* Mô hình P2P giải quyết được vấn đề trên do khả năng phân tán các xử lý ra nhiều node thay vì tập trung tại server. Tuy nhiên mô hình P2P có nhược điểm:
* Khó triển khai hơn client – server.
* Chuyển đổi từ mô hình client – server sang mô hình P2P có thể ảnh hưởng tới dữ liệu sẵn có của hệ thống
* Trong một số trường hợp, cần quản lý dữ liệu tập trung nên ứng dụng cần phải triển khai theo mô hình client – server. Vì vậy mô hình client – server kết hợp P2P là một mô hình phù hợp có thể áp dụng nhằm cải thiện hiệu năng của hệ thống.

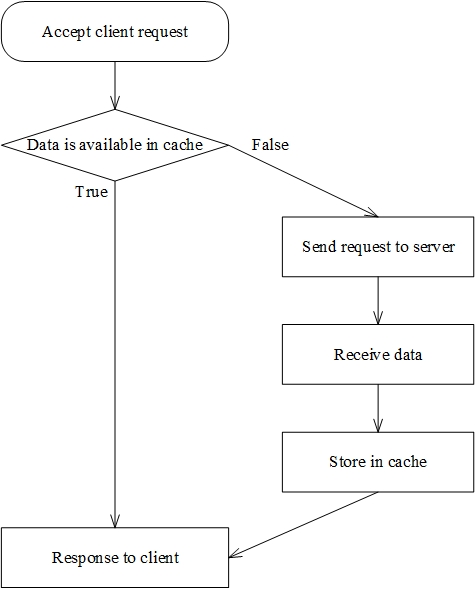
### Caching model

* Trong ứng dụng client – server không sử dụng cache, client gửi yêu cầu dữ liệu tới server và server sẽ xử lý trả dữ liệu về theo hướng ngược lại. Mô hình này có nhược điểm là trong trường hợp có nhiều dữ liệu được sử dụng lại sẽ làm tốn nhiều chi phí để lấy lại cùng một dữ liệu.
* Caching trong client – server được sử dụng để cải thiện hiệu năng hệ thống. Việc truy cập dữ liệu trong cache content có thể giúp làm giảm thời gian truy cập và giảm nhu cầu băng thông kết nối tới server.



##### Hình 6: Caching model

* Giải pháp caching model sử dụng bộ nhớ máy client để lưu trữ dữ liệu đã lấy về. Khi client gửi một truy vấn, cache manager sẽ tìm kiếm dữ liệu đã lưu trong cache content, nếu tìm thấy dữ liệu cache manager sẽ trả về cho client, trong trường hợp không tìm thấy dữ liệu trong cache content, cache manager sẽ chuyển tiếp truy vấn tới server. Khi nhận được dữ liệu trả về từ server, cache manager sẽ lưu lại và trong cache content trước khi trả về cho client.

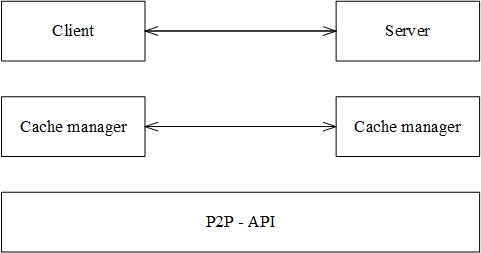


##### Hình 7: Sơ đồ hoạt động của caching model

* Nhược điểm của caching model chỉ phù hợp với các loại dữ liệu tĩnh, ít thay đổi do không cần cơ chế cập nhật các thay đổi dữ liệu từ server. Với những dữ liệu có tần suất thay đổi lớn, việc cập nhật lại dữ liệu thường xuyên từ server có thể khiến việc sử dụng cache không mang lại hiệu quả cao. Trong nhiều trường hợp chi phí cập nhật thay đổi dữ liệu có thể lớn hơn chi phí lấy toàn bộ file từ server.
* Ngoài ra, khi sử dụng caching model dữ liệu được lưu trữ trong cache không thể chia sẻ với các client khác trong mạng dẫn tới lãng phí tài nguyên hệ thống như: tài nguyên ổ cứng lưu trữ cache, tài nguyên mạng trong truyền dữ liệu giữa client – server, …

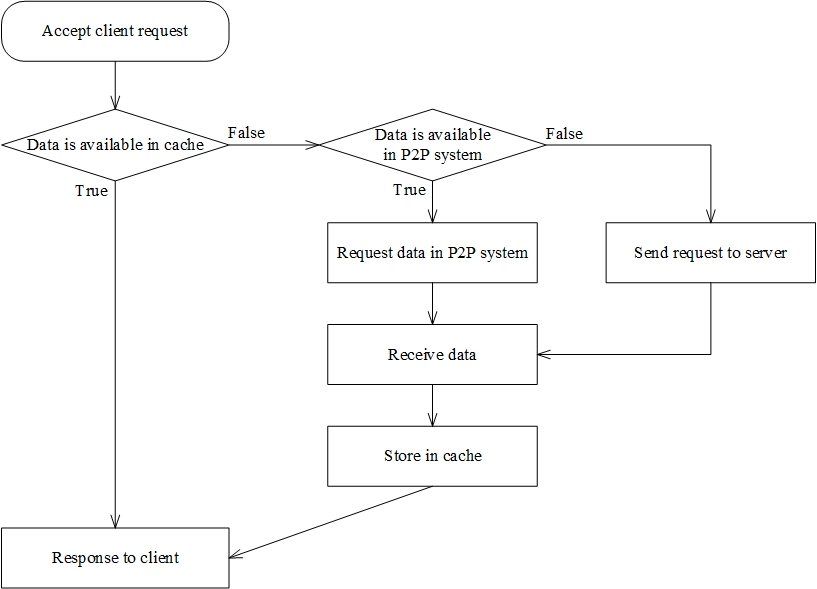
### Shared caching model

* Như đã phân tích trong phần caching model, dữ liệu lưu trong cache chỉ được sử dụng cục bộ trong máy client sẽ làm lãng phí tài nguyên ở máy client. Nếu có thể chia sẻ dữ liệu trong cache tới các client khác sẽ góp phần cải thiện đáng kể hiệu năng hệ thống do:
* Các client ngoài lấy dữ liệu từ server còn chia sẻ dữ liệu với nhau, giúp làm giảm băng thông server
* Tận dụng được khả năng xử lý và lưu trữ của client để tăng hiệu năng hệ thống
* Để có thể chia sẻ dữ liệu cache giữa các client, giải pháp được đưa ra là:
* Sử dụng P2P để kết nối và chia sẻ cache folder
* Thay đổi cache manager để truy cập dữ liệu từ máy xa thông qua P2P
* Tổ chức dữ liệu trong bộ nhớ cache theo cách mà chúng có thể được sử dụng trong nhiều trường hợp phân phối dữ liệu của ứng dụng.



##### Hình 8: Shared – caching model

* Trong mô hình shared – caching, cache manager sử dụng hệ thống P2P thông qua các P2P-API gồm các thao tác: đưa dữ liệu lên hệ thống P2P, lấy dữ liệu từ hệ thống P2P, tìm kiếm dữ liệu trong hệ thống P2P, xóa dữ liệu trong hệ thống P2P.

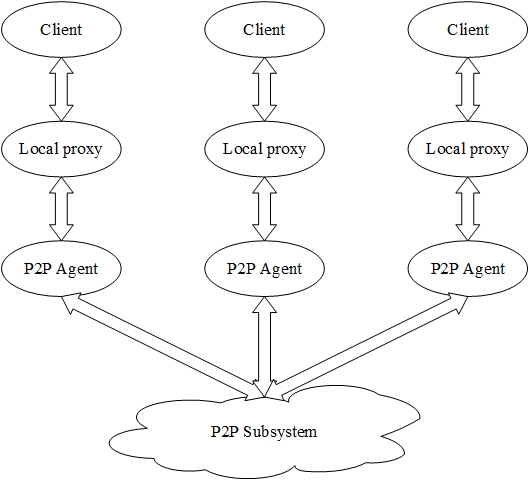


##### Hình 9: Sơ đồ hoạt động của mô hình shared-caching

* Khi nhận được yêu cầu download từ client, hệ thống thực hiện kiểm tra dữ liệu trong cache, nếu dữ liệu đã tồn tại trong cache lập tức trả về client xử lý. Ngược lại, gửi yêu cầu download dữ liệu tới các máy trong mạng P2P, nếu tìm thấy dữ liệu trong mạng P2P thì client lấy dữ liệu từ các máy trong mạng đồng thời lưu vào cache. Nếu không tìm thấy dữ liệu trong mạng P2P thì hệ thống gửi yêu cầu dữ liệu của client tới server, client nhận dữ liệu từ server và cũng lưu vào cache.

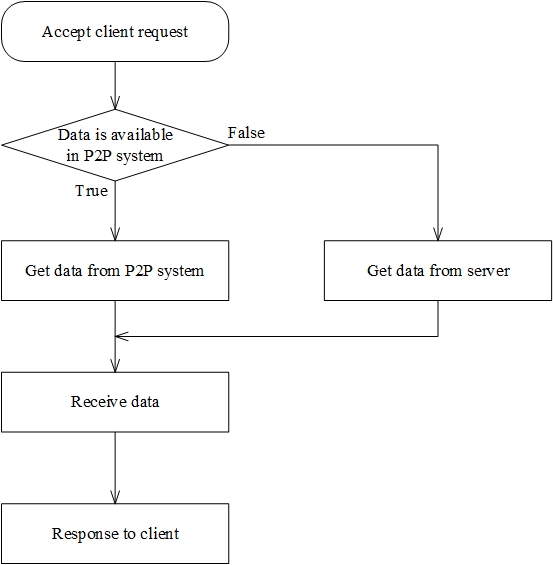
### Local proxy model

* Tương tự như mô hình shared cache, trong mô hình local proxy, chúng ta cũng chia sẻ dữ liệu lưu trữ cục bộ trên máy client để cải thiện hiệu năng của hệ thống.



##### Hình 10: Mô hình local proxy

* Trong mô hình local proxy bao gồm những thành phần:
* Module client: giống như module máy khách trong kiến trúc client-server truyền thống.
* Module local proxy: mỗi máy client được cài đặt một module local proxy. Module này tương tác với module client và cập nhật dữ liệu cục bộ.
* P2P agent: tương tác với các P2P agent khác thông qua mạng P2P.
* Mô hình này không chỉ truyền dữ liệu đơn thuần giữa các client mà mỗi client được xây dựng để có thể hoạt động như một client host. Module local proxy hoạt động như một server, tương tác với module client. Do vậy, có thể sử dụng lại được module client trong mô hình client - server mà không cần phải thiết kế lại. Việc đồng bộ dữ liệu giữa các client host sẽ được thực hiện thông qua hệ thống P2P.
* Để áp dụng mô hình này, chúng ta cần tổ chức dữ liệu server một cách phù hợp để có thể áp dụng các thao tác: đưa dữ liệu lên hệ thống P2P, lấy dữ liệu từ hệ thống P2P, tìm kiếm dữ liệu trong hệ thống P2P, xóa dữ liệu trong hệ thống P2P. Đồng thời cũng phải phù hợp với giao thức client – server mà ta muốn triển khai.
* Ưu điểm của mô hình này là phân phối vai trò của server trên các client host.
* Nhược điểm của mô hình này là độ trễ trong việc lấy dữ liệu từ hệ thống P2P.



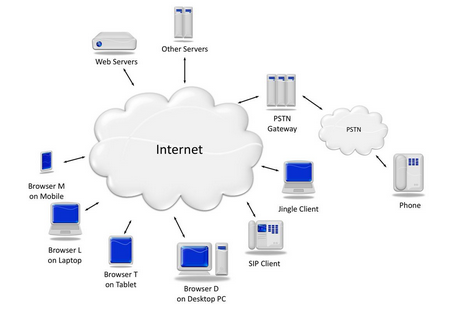
##### Hình 11: Nhận dữ liệu trong mô hình local proxy

# Công nghệ sử dụng

## WebRTC

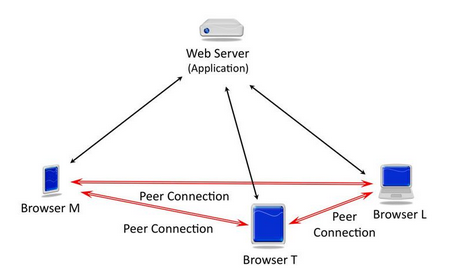
### Giới thiệu

* WebRTC (Web Real Time Communication) là tập hợp các tiêu chuẩn và giao thức cho phép các trình duyệt Web thực hiện trực tiếp các tính năng truyền thông đa phương tiện thời gian thực như gọi điện, tin nhắn hình, truyền dữ liệu bằng các API Javascript.
* Các phần tử của một hệ thống WebRTC: Một hệ thống WebRTC gồm web server, trình duyệt chạy trên các hệ điều hành khác nhau trên các thiết bị khác nhau gồm: desktop PCs, tablets, mobile phones. Ngoài ra hệ thống WebRTC còn có sự tham gia của các server khác (signaling server, media server, …), PSTN gateway, Jingle client, SIP client.



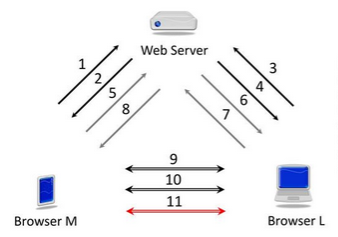
##### Hình 12: Các phần tử trong môi trường WebRTC

* Đa kết nối ngang hàng giữa các trình duyệt: WebRTC hỗ trợ việc thiết lập đa kết nối giữa các trình duyệt với nhau, tạo thành một mạng kết nối.



##### Hình 13: Đa kết nối ngang hàng

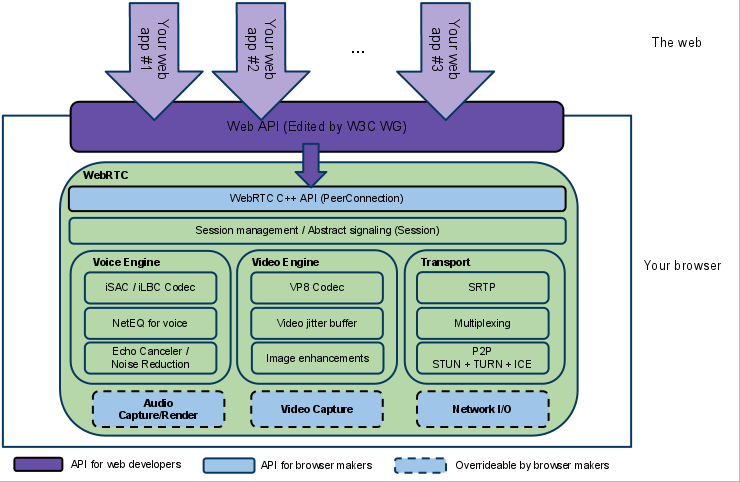
* Quy trình thiết lập peer connection



##### Hình 14: Quá trình thiết lập phiên làm việc của WebRTC

* Browser M yêu cầu trang web từ web server
* Web server cung cấp trang web cho M với WebRTC Javascript
* Browser L yêu cầu trang web từ web server
* Web server cung cấp trang web cho L với WebRTC Javascript
* Browser M xác định kết nối với L, Javascript tại browser M sẽ tạo đối tượng session description của M (offer) và gửi tới web server
* Web server sẽ gửi đối tượng session description của M tới Javascript tại browser L
* Javascript tại browser L sẽ tạo đối tượng session description của L (answer) và gửi tới web server
* Web server sẽ gửi đối tượng session description của L tới Javascript tại browser M
* Browser M và browser L bắt đầu sử dụng phương pháp “hole punching” để xác định đường đi tốt nhất có thể đi đến phía đối phương
* Sau khi “hole punching” hoàn thành, browser M và browser L tạo khóa để bảo mật cho media
* Browser M và browser L bắt đầu trao đổi voice, media và data

### Cấu trúc WebRTC

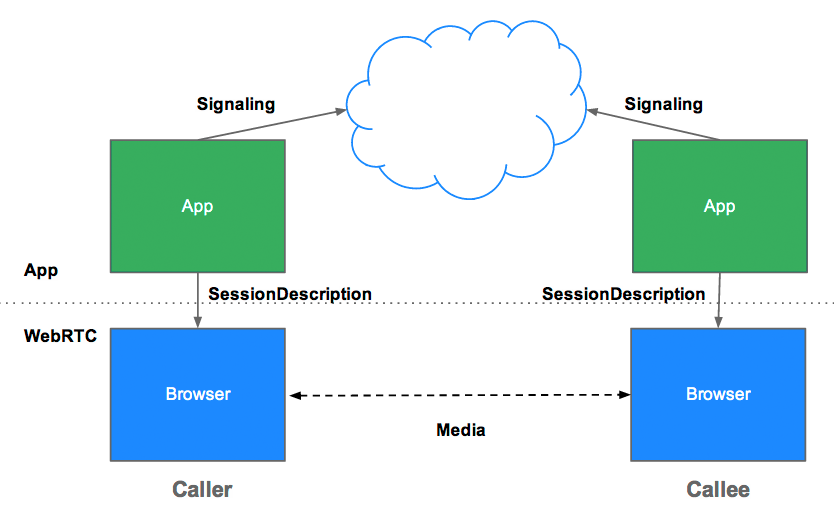


##### Hình 15:

* Để có thể tích hợp các ứng dụng thời gian thực vào web, các browser phải thêm vào các khối chức năng hỗ trợ dành cho WebRTC:
* Voice/video engine: Thu nhận âm thanh và hình ảnh từ thiết bị ngoại vi (microphone, camera), điều chế mã hoá âm thanh và hình ảnh dựa trên các chuẩn mã hoá cơ bản trước khi truyền. Các chuẩn mã hoá cơ bản dành cho voice và video bao gồm: Opus, iSac, iLBC <voice>; VP8, H263, H264 (video).
* Transport: cũng cấp chức năng kết nối với các thành phần khác cùng tham gia trong WebRTC (STUN, TURN, ICE ...)
* Session management: đóng vai trò điều khiển hoạt động của ứng dụng
* Application Programmable Interface - API: cung cấp các hàm cơ bản để các nhà phát triển có thể sử dụng. API ở đây có thể xét nằm trên 2 mức cơ bản: API dành cho Browser developer và API dành cho Front end developer.

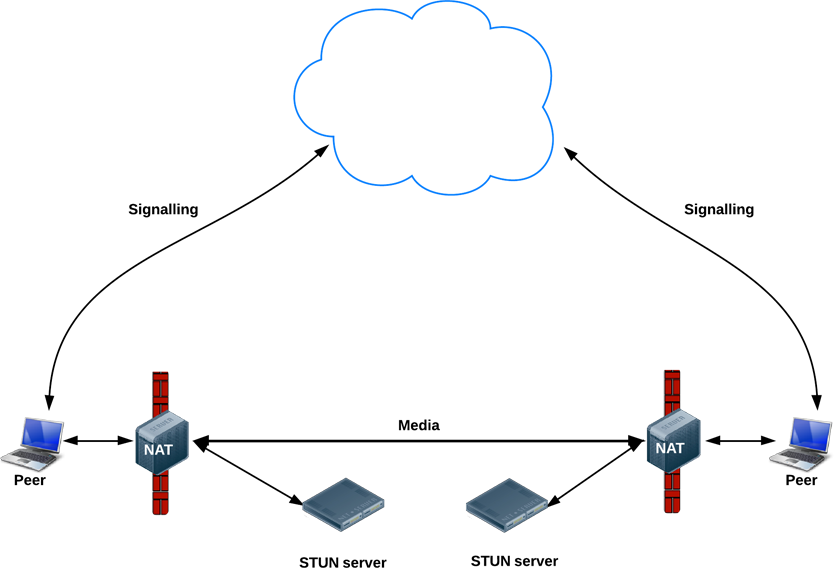
### Phương thức hoạt động

* Trong mô hình hoạt động truyền thống của Web, mô hình client - server được sử dụng khá là phổ biến khi mà server sẽ đảm nhiệm tất cả bao gồm: điều khiển, truyền dữ liệu, security, ... Tuy nhiên, nếu mang mô hình này vào trong các ứng dụng thời gian thực sẽ ảnh hướng tới hiệu năng hoạt động của ứng dụng vì các lý do sau:
* Khối lượng dữ liệu trao đổi giữa các client là cực kỳ lớn. Chuẩn mã hoá voice OPUS có bitrate dao động từ 6 Kbps cho tới 510 Kbps. Nếu dữ liệu này truyền thằng trực tiếp tới server để server truyền tải tới client thì sẽ xảy ra việc quá tải không chỉ dành cho server mà cho cả hệ thống mạng.
* Chất lượng dịch vụ (Quality of service - QoS). Trong QoS dành cho các ứng dụng thời gian thực, có hai tham số cực kỳ quan trọng cần được giảm thiểu tối đa là deplay (trễ client ) và jitter (trễ giữa các gói tin). Việc phải truyền tải toàn bộ dữ liệu tới server sẽ gia tăng rủi do trong delay và jitter.
* Do đó, các ứng dụng thời gian thực đòi hỏi một mô hình khác có thể đáp ứng được hai yêu cầu trên, đó là mô hình P2P. Trong mô hình này, dữ liệu truyền tải được chia ra làm hai loại: dữ liệu điều khiển, báo hiệu (Signalling) và dữ liệu nội dung (Data stream). Các máy client thường sẽ giao tiếp với server để gửi/nhận các tín hiệu điều khiển và sau đó sẽ truyền nhận trực tiếp dữ liệu với các client khác.



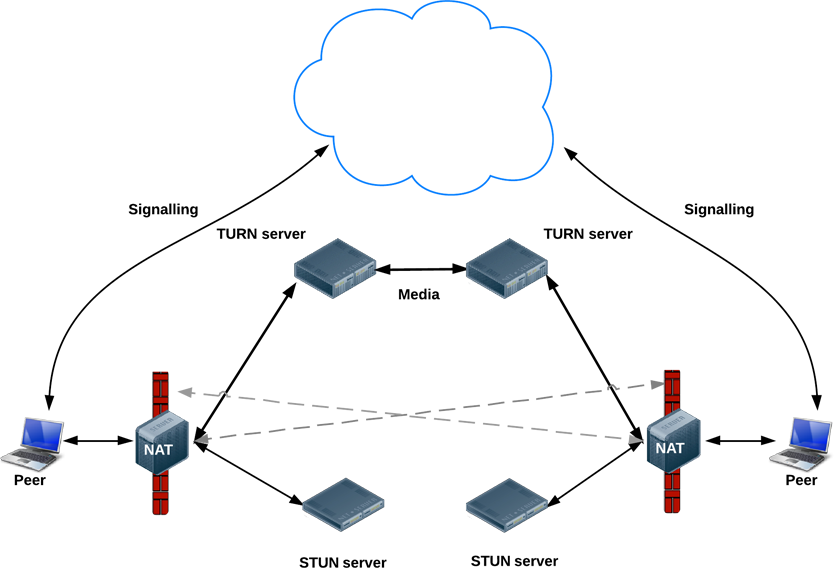
##### Hình 16:

* WebRTC tạo ra kết nối peer-to-peer giữa các trình duyệt mà không có một cơ chế báo hiệu (signaling) nào được cài đặt sẵn. Người lập trình sẽ quyết định sử dụng cơ chế báo hiệu cho phù hợp với nhu cầu và với hệ thống sẵn có. Do đó việc khởi tạo, quản lý, định danh, xác thực sẽ được thực hiện theo các phương thức bất kỳ. Đây là thiết kế hợp lý của WebRTC, nó cho phép kết nối WebRTC với các hệ thống sẵn có như SIP. Ngoài ra, tất cả các luồng tín hiệu và dữ liệu truyền qua WebRTC đều được mã hóa sử dụng DTLS và SRTP.
* WebRTC bị cản trở khi tạo kết nối peer-to-peer bởi tường lửa và NAT. WebRTC sử dụng kỹ thuật ICE để tạo kênh giao tiếp giữa các peer nhờ STUN và TURN server.
* Đầu tiên, ICE sẽ ưu tiên thử kết nối trực tiếp giữa 2 peer, STUN server được sử dụng để xác định địa chỉ public của peer nếu có mặt của NAT.



##### Hình 17:

* Nếu STUN không hoạt động được do peer nằm trong NAT đối xứng, dữ liệu sẽ được truyền qua server trung gian vận chuyển gói tin là TURN server. Điều này nâng cao không những chất lượng dịch vụ của ứng dụng mà còn đảm bảo an toàn thông tin khi truyền dẫn.



##### Hình 18:

## Node.js

* Node.js là một nền tảng (platform) chạy trên môi trường V8 Javascript runtime. Node.js cho phép lập trình viên xây dụng các ứng dụng có tính mở rộng cao sử dụng Javascript trên server. Vì đươc porting từ C nên tốc độ xử lý khá nhanh.
* Với cách lập trình truyền thống, mỗi kết nối máy chủ sẽ tạo ra một luồng (thread). Khi thực hiện một thao tác I/O, tiến trình sẽ không thể tiếp tục cho đến khi thao tác I/O hoàn thành. Cơ chế này được gọi là blocking. Với Node.js, cơ chế này không được sử dụng vì khi có một tiến trình thao tác I/O hay truy xuất dữ liệu, các tiến trình sẽ bị ngưng hoàn toàn, điều này làm phí thời gian chờ. Thay vào đó, Node.js sử dụng cơ chế non-blocking, nghĩa là trong khi thao tác I/O, tiến trình vẫn được tiếp tục. Để quản lý các tiến trình, Node.js sử dụng mô hình event-driven, nghĩa là các luồng xử lý sẽ được xác định bởi các sự kiện. Khi có một sự kiến đến, ví dụ thao tác truy xuất dữ liệu, sự kiện này sẽ được ghi nhớ (register), một event-callback sẽ được đánh dấu. Một event-callback là một hàm được định nghĩa cho phép tiến trình gọi thực thi khi một thao tác hoàn thành. Khi event-callback được thực thi, sự kiện sẽ được loại bỏ, tiến trình vẫn được tiếp tục cho đến khi kết thúc.
* Lợi ích mà Node.js có thể mang lại:
* Node.js kế thừa công nghệ V8 của google, khả năng tự build server bằng ngôn ngữ client giúp cho việc trung chuyển dữ liệu diễn ra nhanh hơn. Đồng thời giảm khả năng rùi ro bị ngắt (interupt). Ngoài ra nhờ cơ chế Non I/O Blocking, tận dùng tối đa tài nguyên của server, không tạo ra độ trễ như các ngôn ngữ phía server khác.
* Node.js có thể điều khiển hàng nghìn kết nối đồng thời với chi phí tối thiểu trên một quá trình duy nhất.
* Javascript là một ngôn ngữ phổ biến, và chính điều này đã giúp node.js trở nên quen thuộc và dễ dàng được áp dụng rộng rãi hơn.
* Phù hợp để xây dựng những ứng dụng thời gian thực (real-time applications) như ứng dụng chat, các dịch vụ mạng xã hội như Facebook, Twitter, …

## HTML5

* HTML5 là một ngôn ngữ được thiết kế để thiết lập nội dung web. Nó nhằm làm cho việc thiết kế và phát triển web dễ dàng hơn bằng cách tạo một giao diện ngôn ngữ đánh dấu chuẩn hóa và trực quan. HTML5 cung cấp các phương tiện phân tích và phân định các trang của bạn, và nó cho phép bạn tạo các thành phần rời rạc không chỉ được thiết kế để cấu tạo trang web một cách hợp lý mà còn được tạo ra để cung cấp cho trang web các khả năng thông tin. HTML5 có thể được gọi là “cách tiếp cận thông tin thông qua thiết kế” do nó kết hợp yếu tố cơ bản về lập bản đồ thông tin, phân chia và ghi nhãn thông tin giúp dễ dàng sử dụng và hiểu thông tin. Đây là nền tảng của tiện ích ngữ nghĩa và thẩm mỹ gây ấn tượng sâu sắc của HTML5. HTML5 cung cấp khả năng xuất bản tất cả mọi thứ trên thế giới từ nội dung văn bản đơn giản đến đa phương tiện phong phú, tương tác cho các nhà thiết kế và các nhà phát triển ở mọi trình độ.
* HTML5 cung cấp các công cụ quản lý dữ liệu, đồ họa, video, và âm thanh có hiệu quả. Nó tạo điều kiện cho sự phát triển của các ứng dụng giữa các trình duyệt với nhau cho trang web cũng như cho các thiết bị di động. HTML5 là một trong những công nghệ thúc đẩy những cải tiến trong các dịch vụ điện toán đám mây di động, vì nó tính đến tính linh hoạt rộng hơn, cho phép phát triển các trang web thú vị và có khả năng tương tác. Nó cũng đưa vào thẻ và các cải tiến mới, bao gồm cấu trúc thu nhỏ, các nút điều khiển của biểu mẫu, các API, đa phương tiện, hỗ trợ cơ sở dữ liệu, và tốc độ xử lý nhanh hơn đáng kể.
* HTML5 Filesystem API cung cấp một phương tiện cho phép ứng dụng web tạo mới, đọc, ghi, điều hướng tới hệ thống tệp tin cục bộ của người dùng. Các Filesystem API có thể được chia thành các kỹ thuật khác nhau:
* Đọc và thao tác với tệp tin: File/Blob, FileList, FileReader
* Tạo và ghi tệp tin: BlobBuilder, FileWriter
* Truy cập thư mục và hệ thống tệp tin: DirectoryReader, FileEntry/DirectoryEntry, LocalFileSystem

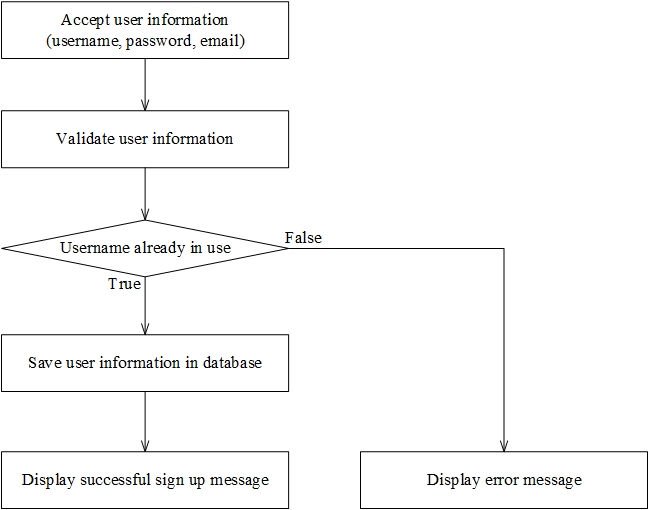
# Đặc tả ứng dụng chia sẻ tệp

* Các chức năng của ứng dụng:
* Đăng ký tài khoản
* Đăng nhập
* Đăng xuất
* Tải tệp lên server
* Xóa tệp khỏi hệ thống
* Tải tệp về máy
* Ứng dụng xây dựng trên nền tảng ứng dụng web, người dùng tương tác với hệ thống thông qua trình duyệt cài đặt trên máy người dùng
* Sử dụng HTTP request và HTTP response message để giao tiếp và truyền tải dữ liệu giữa client và server

## Ứng dụng chia sẻ tệp theo mô hình client – server

### Chức năng đăng ký tài khoản

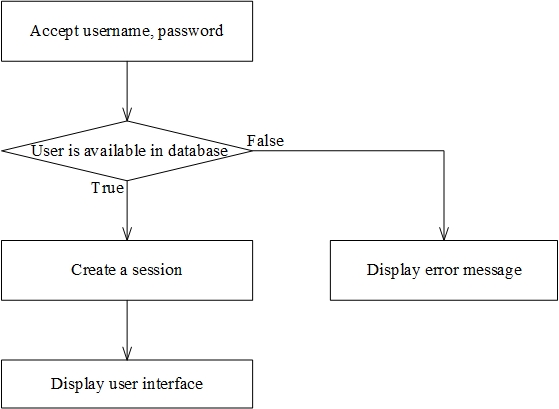
* Về phía người dùng:
* Người dùng bấm vào nút “Sign up” để thấy form đăng ký.
* Người dùng điền thông tin vào form đăng ký.
* Người dùng nhấn vào nút “Sign up” trong form đăng ký để hoàn tất.
* Về phía hệ thống
* Hệ thống hiển thị form đăng ký.
* Hệ thống ghi nhận thông thông tin người dùng tin từ form đăng ký và kiểm tra tính hợp lệ của thông tin người dùng.
* Hệ thống kiểm tra tên đăng nhập đã được sử dụng chưa, nếu tên đăng nhập đã được sử dụng thì hiển thị thông báo lỗi.
* Nếu tên đăng nhập chưa được sử dụng, hệ thống lưu thông tin người dùng vào CSDL, hiển thị thông báo đăng ký thành công.



##### Hình 19: Chức năng đăng ký tài khoản theo mô hình client – server

### Chức năng đăng nhập

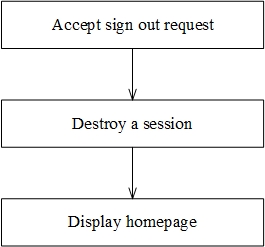
* Về phía người dùng:
* Người dùng bấm vào nút “Sign in” để thấy form đăng nhập.
* Người dùng điền thông tin vào form đăng nhập.
* Người dùng nhấn vào nút “Sign in” trong form đăng nhập để hoàn tất.
* Về phía hệ thống:
* Hệ thống hiển thị form đăng nhập.
* Hệ thống ghi nhận thông tin từ form đăng nhập.
* Hệ thống tìm trong cơ sở dữ liệu có người dùng tương ứng với tên đăng nhập và mật khẩu vừa ghi nhận được không. Nếu không tìm thấy thì hiển thị thông báo đăng nhập thất bại, ngược lại thì hệ thống chuyển qua giao diện người dùng.



##### Hình 20: Chức năng đăng nhập theo mô hình client – server

### Chức năng đăng xuất

* Về phía người dùng: Người dùng phải đang trong phiên làm việc
* Người dùng bấm vào nút “Sign out” để đăng xuất khỏi hệ thống
* Về phía hệ thống:
* Hệ thống nhận yêu cầu đăng xuất
* Hệ thống xóa phiên làm việc của người dùng
* Hệ thống hiển thị giao diện trang chủ



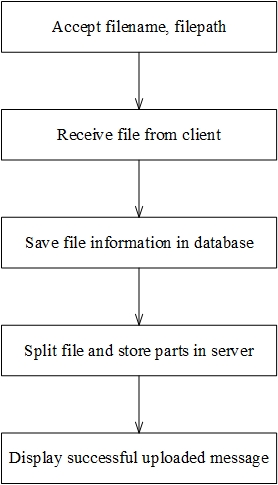
##### Hình 21: Chức năng đăng xuất theo mô hình client – server

### Chức năng upload

* Về phía người dùng: Người dùng phải thực hiện đăng nhập trước khi upload tệp lên server
* Người dùng bấm nút “Browser” để chọn tệp.
* Người dùng bấm nút “Upload” để tải tệp lên server.
* Về phía hệ thống:
* Hệ thống nhận yêu cầu upload từ người dùng.
* Hệ thống nhận dữ liệu từ người dùng.
* Hệ thống lưu thông tin file người dùng vừa tải lên vào cơ sở dữ liệu gồm các trường:

|  |  |
| --- | --- |
| Id | Chỉ số file |
| Filename | Tên file |
| Size | Kích thước file |
| Piece | Số mảnh của file sau khi chia nhỏ |
| Path | Đường dẫn file trên server |
| Uploader | Tài khoản upload file |
| Type | Kiểu file |
| Uploaded\_at | Thời gian upload |

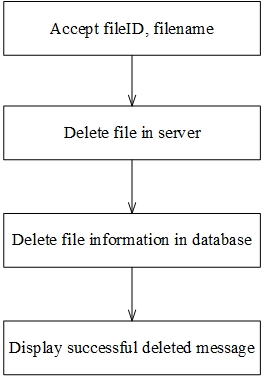
* Hệ thống tạo một thư mục với tên: filename-id (ví dụ: abc-1)
* Hệ thống lưu dữ liệu nhận từ client vào các mảnh với kích thước bằng nhau (4096 bytes, mảnh cuối cùng có thể nhỏ hơn), và lưu vào folder vừa tạo.
* Hệ thống thông báo upload thành công.



##### Hình 22: Chức năng upload theo mô hình client – server

### Chức năng xóa tệp khỏi hệ thống

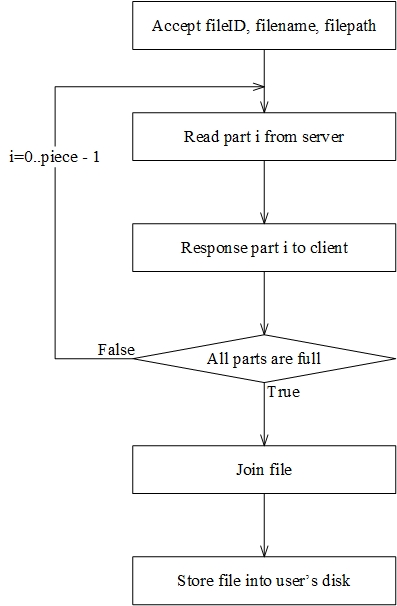
* Về phía người dùng: Người dùng phải thực hiện đăng nhập trước khi xóa tệp của mình khỏi hệ thống.
* Người dùng bấm vào nút “My files” để xem danh sách các file đã tải lên.
* Người dùng lựa chọn file muốn xóa và nhấn vào nút “Delete” để xóa.
* Về phía hệ thống:
* Hệ thống hiển thị danh sách các tệp do người dùng đã tải lên
* Hệ thống nhận yêu cầu xóa tệp từ người dùng, ghi nhận các thông tin của tệp cần xóa
* Hệ thống xóa thông tin tệp trong database và xóa nội dung tệp của người dùng khỏi server.
* Hệ thống thông báo việc xóa tệp hoàn tất



##### Hình 23: Chức năng xóa tệp theo mô hình client – server

### Chức năng download

* Về phía người dùng:
* Người dùng chọn file cần download trong danh sách
* Người dùng ấn vào nút “Download” để tải file về máy
* Về phía hệ thống:
* Hệ thống nhận yêu cầu download file từ người dùng.
* Hệ thống dựa theo đường dẫn lưu trong cơ sở dữ liệu tìm đến thư mục chứa tệp cần download.
* Server đọc dữ liệu trong thư mục theo từng mảnh sau sau đó gửi trả về cho client
* Khi client nhận đủ số mảnh của tệp, sẽ thực hiện nối thành tệp hoàn chỉnh, trả về cho người dùng.



##### Hình 24: Chức năng tải tệp theo mô hình client – server

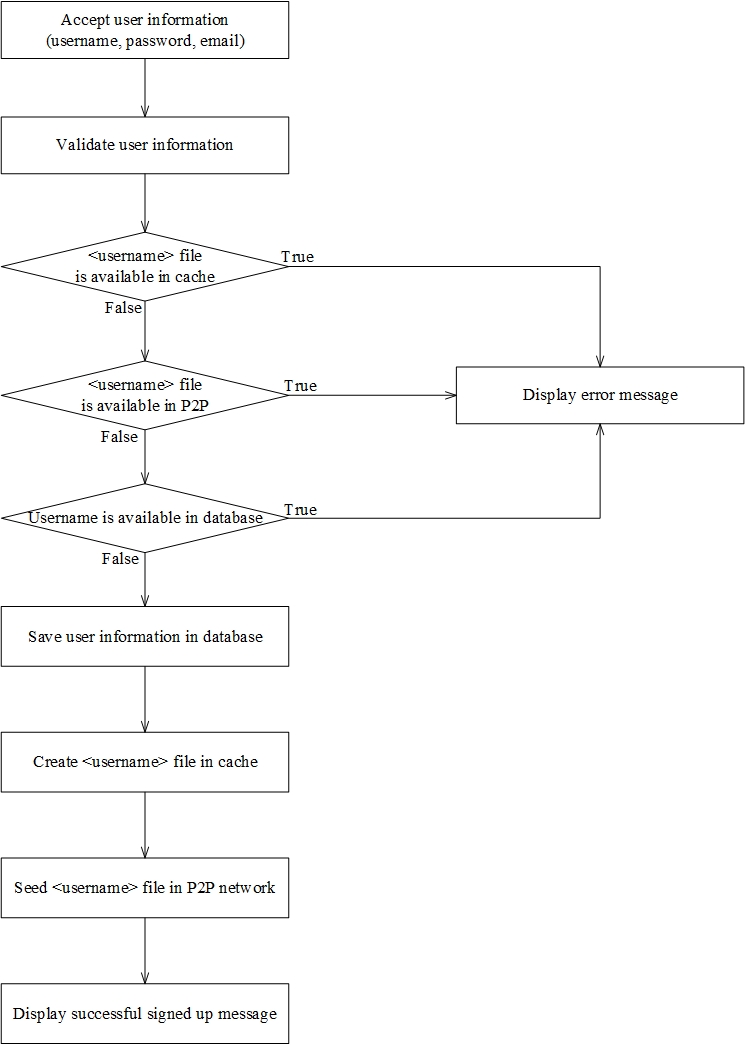
## Ứng dụng chia sẻ tệp theo mô hình shared-cache

### Nhận xét về ứng dụng chia sẻ tệp theo mô hình client – server

* Ứng dụng xây dựng theo mô hình client – server nên kế thừa các nhược điểm của mô hình này như đã phân tích ở trên
* Cụ thể trong ứng dụng này, các tác vụ kiểm tra tên đăng nhập khi đăng ký tài khoản, xác thực tài khoản khi đăng nhập và tải các mảnh dữ liệu về máy, những tác vụ này đều phải thực hiện tại server, điều này có thể gây ảnh hưởng tới hiệu năng của server
* Dữ liệu được chia sẻ trong ứng dụng là các dữ liệu tĩnh, ứng dụng được xây dựng trên nền web, HTML5 cho phép cache dữ liệu trên trình duyệt, phù hợp để áp dụng caching model. Tuy nhiên để tối ưu hóa khả năng của hệ thống, có thể áp dụng mô hình shared cache tận dụng khả năng xử lý của máy client, giảm băng thông server
* Mô hình shared-cache được áp dụng vào ứng dụng chia sẻ tệp này bằng cách thiết lập một mạng P2P giữa các client ở tầng trung gian giữa client và server, mạng này sẽ cho phép client chia sẻ dữ liệu trong cache của mình.

### Chức năng đăng ký tài khoản

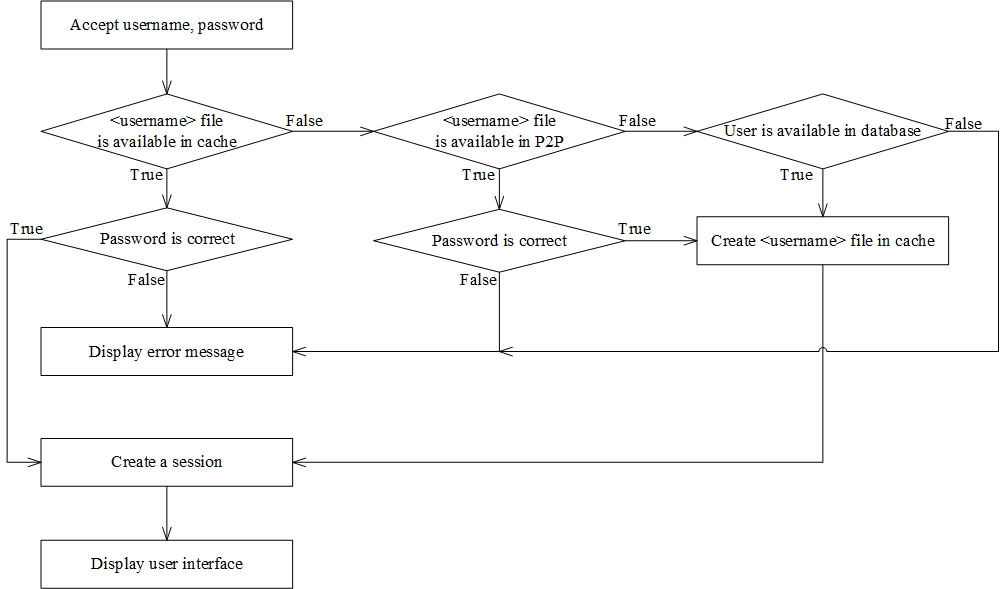
* Về phía người dùng: như bên phần mô hình client – server
* Về phía hệ thống
* Hệ thống hiển thị form đăng ký.
* Hệ thống ghi nhận thông tin từ form đăng ký và kiểm tra tính hợp lệ của thông tin người dùng nhập vào, nếu có lỗi hệ thống hiển thị thông báo lỗi.
* Hệ thống chuyển tên đăng nhập sang dạng mã sha-256, tìm kiếm trong cache file trùng tên với tên đăng nhập đã mã hóa, nếu tìm thấy thì hiển thị thông báo tên đăng nhập đã được sử dụng.
* Nếu không tìm thấy file trong cache, hệ thống gửi yêu cầu tìm kiếm lên mạng P2P, sau 500ms, nếu nhận được phản hồi từ bất kì peer nào thì thông báo tên đăng nhập đã được sử dụng.
* Nếu sau 500ms từ khi gửi yêu cầu tìm kiếm, không nhận được bất kì phản hồi từ peer nào, hệ thống sẽ gửi yêu cầu đăng ký tài khoản tới server
* Hệ thống kiểm tra tên đăng nhập có tồn tại trong CSDL, nếu tồn tại thì hiển thị thông báo tên đăng nhập đã được sử dụng.
* Nếu tất cả các trường thông tin là hợp lệ, hệ thống lưu thông tin người dùng vào CSDL, hiển thị thông báo đăng ký thành công.
* Sau khi đăng ký thành công, hệ thống tạo một file với tên là tên đăng nhập mã hóa, nội dung file là mật khẩu mã hóa, lưu vào cache của trình duyệt, đồng thời, gửi file này đến các máy đang trong mạng P2P, các máy trong mạng nhận được file sẽ lưu vào cache của mình.



##### Hình 25: Chức năng đăng ký tài khoản theo mô hình shared cache

### Chức năng đăng nhập

* Về phía người dùng: như bên phần mô hình client – server
* Về phía hệ thống:
* Hệ thống hiển thị form đăng nhập.
* Hệ thống ghi nhận thông tin từ form đăng nhập.
* Hệ thống chuyển tên đăng nhập và mật khẩu sang dạng mã sha-256, tìm kiếm trong cache file trùng tên với tên đăng nhập đã mã hóa, nếu tìm thấy file thì đọc nội dung file sau đó so sánh nội dung file với mật khẩu đã mã hóa. Nếu mật khẩu trùng với nội dung file thì chuyển trạng thái đăng nhập thành công, nếu mật khẩu không khớp với nội dung file thì hiển thị thông báo đăng nhập thất bại.
* Nếu không tìm thấy file trong cache, hệ thống gửi yêu cầu đăng nhập lên mạng P2P, các máy trong mạng P2P nhận được yêu cầu đăng nhập sẽ thực hiện tìm kiếm trong cache của mình. Nếu tìm thấy file thì thực hiện so khớp tên đăng nhập, mật khẩu và trả về trạng thái đăng nhập thành công hoặc thất bại cho máy gửi yêu cầu. Nếu không tìm thấy file thì không gửi phản hồi.
* Sau 500ms, máy gửi yêu nhận được phản hồi thì chuyển qua trạng thái đăng nhập thành công hoặc thất bại tùy theo trạng thái nhận về. Nếu không nhận được phản hồi từ bất kì peer nào thì gửi yêu cầu đăng nhập lên server
* Hệ thống tìm trong cơ sở dữ liệu server có người dùng tương ứng với tên đăng nhập và mật khẩu vừa ghi nhận được không. Nếu không tìm thấy thì hiển thị thông báo đăng nhập thất bại, ngược lại thì hệ thống chuyển qua giao diện người dùng.
* Khi người dùng đăng nhập thành công dưới sự xác nhận từ mạng P2P hoặc server thì hệ thống sẽ tạo file tương ứng với tên đăng nhập và mật khẩu của người dùng, cache lại trên trình duyệt.



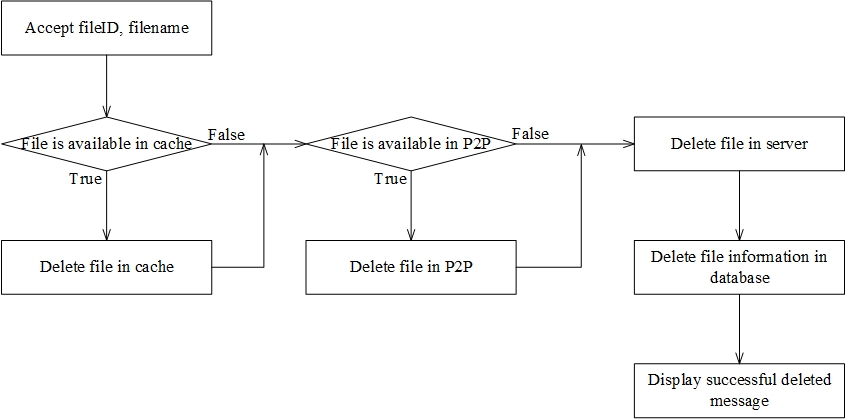
##### Hình 26: Chức năng đăng nhập theo mô hình shared cache

### Chức năng đăng xuất: Như bên phần mô hình client – server

### Chức năng upload: Như bên phần mô hình client – server

### Chức năng xóa tệp khỏi hệ thống

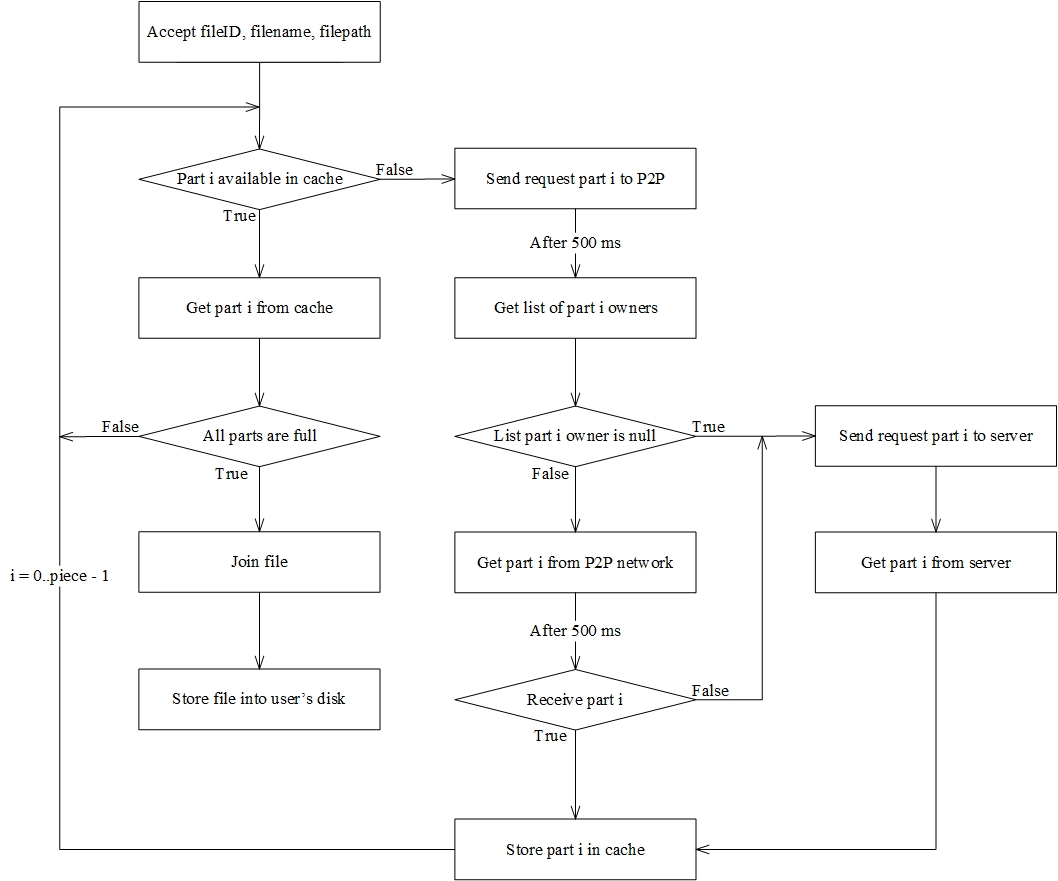
* Về phía người dùng: như bên phần mô hình client – server
* Về phía hệ thống:
* Hệ thống hiển thị danh sách các tệp do người dùng đã tải lên.
* Hệ thống nhận yêu cầu xóa tệp từ người dùng, ghi nhận các thông tin của tệp cần xóa.
* Hệ thống tìm trong cache của người dùng, nếu tồn tại file người dùng chọn xóa thì xóa file khỏi cache.
* Hệ thống tìm kiếm trong cache của các máy đang tham gia kết nối P2P, nếu tìm thấy file thì xóa file khỏi cache.
* Hệ thống xóa tệp của người dùng khỏi server.
* Hệ thống thông báo việc xóa tệp hoàn tất



##### Hình 27: Chức năng xóa tệp theo mô hình shared cache

### Chức năng download

* Về phía người dùng: như bên phần mô hình client – server
* Về phía hệ thống:
* Hệ thống nhận yêu cầu download file từ người dùng.
* Hệ thống nhận tên file và id của file sau đó tìm trong cache thư mục có tên theo định dạng filename-id. Nếu thư mục tồn tại, hệ thống sẽ đọc các mảnh có trong thư mục, nếu số mảnh là đủ thì hệ thống nối thành file hoàn chỉnh sau đó trả về cho người dùng, nếu thiếu mảnh nào, hệ thống sẽ tìm trong mạng P2P hoặc server để tải về.
* Nếu không tồn tại thư mục trong cache, hệ thống sẽ tìm kiếm trong mạng P2P bằng cách gửi một tin nhắn yêu cầu file đến tất cả các máy trong mạng. Các máy trong mạng P2P nhận được tin nhắn yêu cầu file từ máy khác sẽ tìm trong cache của mình, nếu tìm thấy file thì gửi tin nhắn phản hồi. Sau 500ms hệ thống sẽ lấy ra danh sách những máy phản hồi (tức có chứa file cần download), hệ thống gửi yêu cầu download các mảnh tới những máy trong danh sách này và nhận mảnh từ máy trả về nhanh nhất, lưu vào cache. Sau 1s từ khi gửi yêu cầu download một mảnh trong mạng P2P mà không nhận về được dữ liệu, hệ thống sẽ gửi yêu cầu mảnh đó tới server, nhận mảnh từ server và lưu vào cache. Khi nhận được đầy đủ các mảnh, hệ thống sẽ nối thành file hoàn chỉnh sau đó trả về cho người dùng.
* Nếu danh sách các máy chứa file cần download là rỗng, hệ thống sẽ gửi yêu cầu file tới server. Server trả về các mảnh, hệ thống nhận mảnh, lưu các mảnh vào cache, nối thành file hoàn chỉnh, trả về cho người dùng.
* Quá trình tìm kiếm và yêu cầu một mảnh sẽ được thực hiện theo thứ tự từ cache, mạng P2P rồi đến server.



##### Hình 28: Chức năng tải tệp theo mô hình shared cache

* Khi nhận về một file mới (từ mạng P2P hay server), hệ thống lưu trữ trong cache theo cấu trúc thư mục tương tự trên server. Tuy nhiên do tài nguyên của client có giới hạn hơn nên nếu dung lượng lưu trữ trong cache đến ngưỡng giới hạn, hệ thống sẽ xóa bớt các file trong cache để thu hồi bộ nhớ theo nguyên tắc các file được cache trước sẽ bị xóa trước.

# Cài đặt thử nghiệm và đánh giá

## Cài đặt chương trình

* 1. Server
* Xây dựng server bằng ngôn ngữ javascript với nền tảng node.js
* Trong ứng dụng này, server có hai nhiệm vụ chính: vừa làm kênh tín hiệu điều khiển giúp các trình duyệt thiết lập kết nối peer-to-peer với nhau, vừa làm nhiệm vụ của một server nhưng trong mô hình client-server truyền thống, nhận và xử lý các yêu cầu từ phía client.
* Để thực hiện hai nhiệm vụ trên, xây dựng hai module phía server gồm:
* handlePeerConnection module với các phương thức xử lý kết nối P2P

|  |  |
| --- | --- |
| connect() | Xử lý yêu cầu tham gia vào hệ thống P2P, cấp phát id và hộp tin nhắn cho peer. |
| sendMessage() | Nhận mô tả các thông số khởi tạo phiên của một peer, sau đó gửi đến hộp tin nhắn của các peer khác trong mạng P2P. |
| getMessages() | Trả về cho peer các tin nhắn mới trong hộp tin nhắn của peer đó. |
| sendCheckOutMessage() | Gửi tin kiểm tra tới hộp tin nhắn của tất cả các peer theo chu kì để kiểm tra peer còn trong mạng P2P không. |
| getResponseCheckOutMessage() | Nhận các phản hổi từ các peer vẫn còn trong mạng P2P sau khi gửi tin nhắn kiểm tra. |
| clearMemory() | Giải phóng bộ nhớ đã cấp phát cho các peer đã rời hệ thống. |

* handleUserAction module với các phương thức xử lý yêu cầu từ người dùng

|  |  |
| --- | --- |
| download() | Xử lý yêu cầu download file từ server của người dùng. |
| upload() | Xử lý yêu cầu upload file lên server của người dùng. |
| signIn() | Xử lý yêu cầu đăng nhập của người dùng. |
| signUp() | Xử lý yêu cầu đăng ký tài khoản của . |
| deleteFile() | Xóa file trên server do người dùng đã tải lên. |
| getMyfiles() | Trả về cho client danh sách các file do người dùng đã tải lên. |
| getAllFiles() | Trả về cho client danh sách tất cả các file trong hệ thống. |

* 1. Client
* Xây dựng bằng ngôn ngữ HTML5 kết hợp javascript
* Giao diện web sử dụng HTML5, cho phép người dùng tương tác một cách dễ dàng.
* Sử dụng API của WebRTC để thiết lập kết nối trực tiếp giữa các client.
* Các module xây dựng bên phía client:
* GUI: quản lý các thành phần giao diện của ứng dụng
* User: quản lý các tác vụ liên quan đến người dùng
* CacheManager: quản lý bộ nhớ cache
* File: quản lý tiến trình khi người dùng download một file.

## Kịch bản thử nghiệm

* ***Kịch bản 1***: Sử dụng ba bộ dữ liệu sau

Account 1: { username: user1, password: 123456, email: [user1@gmail.com](mailto:user1@gmail.com)}

Account 2: { username: user1, password: 123456, email: [user2@yahoo.com](mailto:user2@yahoo.com)}

Account 3: { username: user2, password: 123456, email: [user2@yahoo.com](mailto:user2@yahoo.com)}

|  |  |  |
| --- | --- | --- |
| STT | Thao tác | Kết quả |
| 1 | Máy A kết nối đến hệ thống |  |
| 2 | Đăng ký tài khoản Account 1 trên máy A | Đăng ký thành công |
| 3 | Đăng ký tài khoản Account 2 trên máy A | Tên người dùng đã được sử dụng – xác nhận từ cache |
| 4 | Máy A thoát khỏi hệ thống, máy B kết nối đến hệ thống |  |
| 5 | Đăng ký tài khoản Account 2 trên máy B | Tên người dùng đã được sử dụng – xác nhận từ server |
| 6 | Máy A kết nối đến hệ thống |  |
| 7 | Đăng ký tài khoản Account 2 trên máy B | Tên người dùng đã được sử dụng – xác nhận từ mạng p2p |
| 8 | Đăng ký tài khoản Account 3 trên máy A | Đăng ký thành công |
| 9 | Đăng ký tài khoản Account 3 trên máy B | Tên người dùng đã được sử dụng – xác nhận từ cache |

* ***Kịch bản 2***: Sử dụng bộ dữ liệu sau, trong đó Account 1 là tài khoản đã được đăng ký thành công.

Account 1: { username: user1, password: 123456}

|  |  |  |
| --- | --- | --- |
| STT | Thao tác | Kết quả |
| 1 | Máy A kết nối đến hệ thống |  |
| 2 | Đăng nhập tài khoản Account 1 trên máy A | Đăng nhập thành công – xác nhận từ server |
| 3 | Đăng xuất khỏi máy A |  |
| 4 | Đăng nhập tài khoản Account 1 trên máy A | Đăng nhập thành công – xác nhận từ cache |
| 5 | Đăng xuất khỏi máy A |  |
| 6 | Máy B kết nối đến hệ thống |  |
| 7 | Đăng nhập tài khoản Account 1 trên máy B | Đăng nhập thành công – xác nhận từ mạng p2p |

* ***Kịch bản 3***:
* Chuẩn bị 10 file với kích thước: 284Kb, 891Kb, 1764Kb, 3267Kb, 6528Kb, 13864Kb, 18837Kb, 24668Kb, 40639Kb, 59040Kb
* Máy client kết nối với hệ thống, cache của client không chứa bất cứ file nào, hệ thống lúc thực hiện kịch bản chỉ có duy nhất máy client kết nối đến (không tồn tại mạng p2p)
* Thực hiện download với từng file, sau khi download hoàn tất, tiến hành download lần hai với tất cả các file, thời gian download của mỗi lần sẽ được ghi lại trong bảng bên dưới.

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Dung lượng tệp | Tải từ server | Tải từ cache |
| 1 | 284Kb |  |  |
| 2 | 891Kb |  |  |
| 3 | 1764Kb |  |  |
| 4 | 3267Kb |  |  |
| 5 | 6528Kb |  |  |
| 6 | 13864Kb |  |  |
| 7 | 18837Kb |  |  |
| 8 | 24668Kb |  |  |
| 9 | 40639Kb |  |  |
| 10 | 59040Kb |  |  |

Kết quả: ở lần download thứ nhất, dữ liệu được lấy ra từ server – tương ứng với kết quả khi triển khi theo mô hình client-server. Sau đó dữ liệu được cache lại trong cache của máy client, ở lần download thứ hai dữ liệu được lấy từ trong cache của client – tương ứng với kết quả khi triển khai theo caching model.

* ***Kịch bản 4***: dữ liệu cần download không có trong cache của client nhưng có trong cache của máy khác trong mạng P2P

## Đánh giá

## Kết luận

**TÀI LIỆU THAM KHẢO**