ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

Mạng Máy Tính

NETWORK APPLICATION
P2P FILE SHARING

GVHD: Lê Bảo Khánh
SV:

TP. HỒ CHÍ MINH, THÁNG 10/2024

Mục lục

1 Mở đầu

Trong bài tập lớn lần này, nhóm sẽ thực hiện một ứng dụng gần tương tự với BitTorrent, bao gồm các kết nối Peer-to-Peer (P2P) và tracker để hỗ trợ chia sẻ file giữa các người dùng. Ứng dụng sẽ bao gồm các thành phần chính như Tracker, đóng vai trò là máy chủ theo dõi và duy trì danh sách các peer (người dùng) đang tham gia chia sẻ file. Tracker sẽ cung cấp thông tin về các peer khác khi một peer mới tham gia để tải hoặc chia sẻ file. Mạng lưới P2P sẽ cho phép các peer kết nối trực tiếp với nhau, vừa tải xuống, vừa chia sẻ file mà không cần qua máy chủ trung gian, tối ưu hóa băng thông và hiệu suất của hệ thống. Thông tin về file chia sẻ sẽ được lưu trong một file torrent, bao gồm metadata về tên file, kích thước và thông tin về các mảnh nhỏ (piece) của file, cũng như địa chỉ của tracker. File được chia thành nhiều mảnh nhỏ để tăng tốc độ chia sẻ và đảm bảo phục hồi dữ liệu hiệu quả hơn khi xảy ra sự cố. Các mô tả của file sẽ được giải thích chi tiết ở những phần dưới đây.

2 Cơ sở lý thuyết

2.1 TCP

2.1.1 Khái niệm TCP

TCP (*Transmission Control Protocol*) là giao thức định hướng kết nối (*connection-oriented protocol*) giúp trao đổi tin nhắn giữa các thiết bị khác nhau qua mạng. Giao thức vận chuyển cơ bản này có nhiều ứng dụng, bao gồm liên kết giữa máy chủ web và trang web, email, FTP (*File Transfer Protocol*), và các giao thức có dạng peer-to-peer. TCP cùng với IP (*Internet Protocol*), giao thức giúp truyền tải các gói dữ liệu giữa các máy tính, tạo nên TCP/IP.

2.1.2 Cách TCP hoạt động

2.1.2.a Thiết lập kết nối

Quá trình này được thực hiện thông qua quá trình bắt tay 3 bước (*3-way handshake process*):

- Client gửi một gói tin SYN đến server — yêu cầu kết nối từ cổng nguồn (*source port*) của client đến cổng đích (*destination port*) của server.
- Server phản hồi bằng một gói tin SYN/ACK, xác nhận yêu cầu kết nối.
- Client nhận gói tin SYN/ACK và trả lời bằng một gói tin ACK của chính nó.

2.1.2.b Truyền tải dữ liệu

Sau khi kết nối được thiết lập thành công, trong mô hình TCP, dữ liệu được chia thành những gói nhỏ gọi là *segment*. Sau đó, các segment này được ráp lại thành dữ liệu ban đầu ở đầu bên kia, đảm bảo rằng mỗi thông điệp đến được vị trí mục tiêu một cách nguyên vẹn. Sau khi một tệp dữ liệu cụ thể được chia thành các segment, chúng có thể di chuyển qua nhiều tuyến khác nhau nếu một tuyến bị tắc nghẽn, nhưng đích đến vẫn giữ nguyên.

2.1.3 Mô hình của TCP segment

Mỗi *segment* bao gồm 2 phần: *head* (phần đầu, độ dài: 20 bytes) và *data* (dữ liệu). *Head* có 11 trường:

- **Source port (16 bit)**: Số cổng của thiết bị gửi.
- **Destination port (16 bit)**: Số cổng của thiết bị nhận.
- **Sequence number (32 bit)**: Trường này có hai chức năng: Nếu cờ SYN được đặt, trường đóng vai trò là số thứ tự ban đầu, và byte đầu tiên được gửi sẽ có số thứ tự là giá trị này cộng với 1. Nếu cờ SYN không được đặt, nó đại diện cho số thứ tự của byte đầu tiên đang được gửi.
- **Acknowledgment number (32 bit)**: Được sử dụng để xác nhận việc nhận một gói tin và chỉ ra số thứ tự của byte tiếp theo được nhận vào.
- **Data offset (DO) (4 bit)**: Xác định độ dài của toàn bộ phần đầu tính bằng từ (1 từ = 4 bytes).
- **Reserved (RSV) (4 bit)**: Luôn được đặt là 0.
- **Flags (9 bit)**: Được sử dụng để thiết lập kết nối, gửi dữ liệu và kết thúc kết nối.
  - URG: Ưu tiên dữ liệu này hơn các dữ liệu khác.
  - ACK: Dùng để xác nhận.
  - PSH: Cờ đẩy.
  - SYN: Thiết lập số thứ tự ban đầu.
  - FIN: Kết thúc kết nối TCP.
- **Windows (16 bit)**: Số byte mà thiết bị sẵn sàng nhận.
- **Checksum (16 bit)**: Kiểm tra lỗi trong toàn bộ segment TCP.
- **Urgent pointer (16 bit)**: Ưu tiên dữ liệu khẩn cấp.
- **Options (up to 32 bits)**: Cho phép thêm các tính năng bổ sung vào TCP. (Tùy chọn)

2.1.4 Ưu điểm và nhược điểm của TCP

2.1.4.a Ưu điểm

- Đây là giao thức đáng tin cậy.
- Cung cấp cơ chế kiểm tra lỗi cũng như cơ chế "recovery".
- Cung cấp công cụ kiểm soát luồng.
- Đảm bảo rằng dữ liệu đến đúng địa điểm ở đúng thứ tự mà nó được gửi.
- Là một giao thức hỗ trợ cho nhiều dạng dữ liệu, tài liệu hóa tốt và được triển khai rộng rãi, được duy trì bởi các tổ chức uy tín như IETF (*Internet Engineering Task Force*).

2.1.4.b Nhược điểm

- TCP được thiết kế cho *Mạng diện rộng (WAN)*, do đó kích thước của nó có thể trở thành vấn đề cho các mạng nhỏ với tài nguyên hạn chế.
- Phải chạy qua nhiều lớp, dẫn đến làm chậm tốc độ của mạng.
- Không mang tính tổng quát, tức là không thể đại diện cho bất kỳ ngăn xếp giao thức nào ngoài bộ TCP/IP.

2.2 HTTP

2.2.1 Khái niệm

HTTP (*HyperText Transfer Protocol*) là giao thức tiêu chuẩn cho Internet, được sử dụng để trao đổi thông tin giữa máy chủ Web (*Web server*) và máy khách (*client*), là ứng dụng của giao thức TCP/IP. Tuy đơn giản nhưng HTTP là một giao thức mạnh mẽ nhờ vào các đặc trưng cơ bản sau:

- **Sự đơn giản**: HTTP thường được thiết kế để trở nên đơn giản và thân thiện để con người có thể đọc được, cải thiện khả năng thử nghiệm (*testing*) cho các nhà phát triển, và giảm thiểu độ phức tạp cho người mới bắt đầu.
- **Có thể mở rộng**: Các header HTTP cho phép giao thức dễ dàng mở rộng và linh hoạt cho các thử nghiệm mới.
- **Stateless, nhưng không sessionless**: HTTP không liên kết các yêu cầu liên tiếp, điều này có thể gây ra vấn đề cho người dùng. Tuy nhiên, cookie của HTTP cho phép tạo các *session* có trạng thái, cho phép chia sẻ cùng một ngữ cảnh giữa nhiều yêu cầu.

2.2.2 Cách HTTP hoạt động

HTTP được thiết kế dựa trên giao thức Client/Web server. Máy tính của người dùng hoạt động như các máy khách (*client*). Sau khi người dùng thực hiện một hành động, các máy khách sẽ gửi yêu cầu tới máy chủ và chờ phản hồi từ các máy chủ này.

2.2.3 Các lớp của TCP/IP

Được thiết kế dựa trên TCP/IP, HTTP cũng được chia thành 4 tầng, mỗi tầng sử dụng các giao thức ở tầng dưới để đạt được mục đích của mình.

1. **Lớp Network Access**: Xác định chi tiết về cách dữ liệu được truyền qua mạng bởi các thiết bị phần cứng giao tiếp trực tiếp với môi trường mạng, chẳng hạn như cáp đồng trục, cáp quang hoặc dây đồng xoắn đôi.
2. **Lớp Internet**: Đóng gói dữ liệu thành các gói IP (*Internet Protocol*), chứa địa chỉ nguồn và địa chỉ đích (địa chỉ logic hoặc địa chỉ IP) được sử dụng để định tuyến các gói tin giữa các máy chủ và qua các mạng.
3. **Lớp Transport**: Cho phép các thiết bị trên các máy chủ nguồn và đích trao đổi dữ liệu, đồng thời xác định mức độ dịch vụ và trạng thái kết nối được sử dụng cho việc truyền dữ liệu. Giao thức chính ở đây là TCP.
4. **Application Layer**: Cung cấp các chức năng cho phép người dùng trao đổi dữ liệu ứng dụng qua mạng.

2.2.4 HTTP Request và Response

Việc giao tiếp giữa máy khách và máy chủ được thực hiện dựa trên cặp yêu cầu-phản hồi (*request-response*). Client gửi các yêu cầu, và Server phản hồi lại các yêu cầu này.

2.2.4.a HTTP Request

Một *Request* bao gồm 2 phần chính: *Request line* và *Header*. *Request line* bao gồm 3 phần: *Method*, *URI*, và phiên bản HTTP. *Method* chỉ ra phương thức được sử dụng trong yêu cầu HTTP, thường là **GET** hoặc **POST** (hoặc *HEAD*, *PUT*, *DELETE*, *OPTIONS*, *CONNECT*). *URI* là địa chỉ định danh của tài nguyên, và phiên bản HTTP chỉ định phiên bản HTTP đang được sử dụng. *Header* cho phép máy khách gửi thêm thông tin về thông điệp yêu cầu HTTP và về chính máy khách. Một số trường *Header* phổ biến bao gồm:

- **Accept**: Các loại nội dung có thể chấp nhận từ response (ví dụ: *text/plain*, *text/html*,...).
- **Accept-Encoding**: Các kiểu nén được chấp nhận (ví dụ: *gzip*, *deflate*,...).
- **Connection**: Các tùy chọn kiểm soát cho kết nối hiện tại (*keep-alive*, *Upgrade*,...).
- **Cookie**: Thông tin HTTP cookie từ Server.
- **User-Agent**: Thông tin về tác nhân người dùng (*user agent*) của người dùng.

HTTP request có một số phương thức như *HEAD*, *PUT*, *DELETE*, *OPTIONS*, *CONNECT*, nhưng phổ biến nhất vẫn là **GET** và **POST**.

- Với **GET**, câu truy vấn sẽ được đính kèm vào đường dẫn của HTTP request. Một số đặc điểm của phương thức GET bao gồm:
  - GET request có thể được cached, bookmark và lưu trong lịch sử của trình duyệt.
  - GET request bị giới hạn về chiều dài, do chiều dài của URL là có hạn.
  - GET request không nên dùng với dữ liệu quan trọng, chỉ dùng để nhận dữ liệu.
- Với **POST**, câu truy vấn sẽ được gửi trong phần message body của HTTP request. Một số đặc điểm của phương thức POST bao gồm:
  - POST không thể cached, bookmark hay lưu trong lịch sử trình duyệt.
  - POST không bị giới hạn về độ dài.

2.2.4.b HTTP Response

Cấu trúc HTTP response gần giống với HTTP request, chỉ khác nhau là thay vì *Request line* thì là *Status line*. Và giống như *Request line*, *Status line* cũng có ba phần:

- **HTTP-version**: Phiên bản HTTP cao nhất mà server hỗ trợ.
- **Status-Code**: Mã kết quả trả về.
- **Reason-Phrase**: Mô tả về Status-Code.

Một số loại Status-Code thông dụng mà server trả về cho client:

- **1xx Information Message**: Các status code này chỉ có tính chất tạm thời, client có thể không quan tâm.
- **2xx Successful**: Khi đã xử lý thành công request của client, server trả về status dạng này (ví dụ: 200 OK — request thành công; 202 Accepted — request đã được nhận, nhưng không có kết quả nào trả về, thông báo cho client tiếp tục chờ đợi,...).

- **3xx Redirection**: Server thông báo cho client phải thực hiện thêm thao tác để hoàn tất request (ví dụ: 301 Moved Permanently — tài nguyên đã được chuyển hoàn toàn tới địa chỉ *Location* trong HTTP response,...).
- **4xx Client Error**: Lỗi của client (ví dụ: 404 Not Found — không tìm thấy tài nguyên,...).
- **5xx Server Error**: Lỗi của server (ví dụ: 500 Internal Server Error — có lỗi trong quá trình xử lý của server,...).

2.3 Lý thuyết về Bittorent

2.3.1 Lý thuyết về mã hóa bencode

Bencode là một dạng mã hóa được dùng cho việc truyền file peer-to-peer trong hệ thống **Bittorent**.
Hỗ trợ bốn kiểu dữ liệu là:

- byte strings
- integers
- lists
- dictionaries

Cách hoạt động của Bencode:

- Đầu tiên là các byte strings sẽ được biểu diễn dưới dạng hệ cơ số 10 theo sau là dấu hai chấm và chuỗi kí tự. Ví dụ: "**4:spam**" tương ứng với chuỗi "spam".
- Tiếp theo là các integers sẽ được biểu diễn bằng chữ "**i**" theo sau là số ở hệ cơ số 10 và kết thúc bằng chữ "**e**". Ví dụ: "**i3e**" tương ứng với số "3" và "**i-3e**" tương ứng với số "-3". Integers không có giới hạn và tất cả các mã hóa với số 0 đầu đều không hợp lệ như "**i-0e**" hoặc "**i03e**" thế nhưng "**i0e**" tương ứng với số "0" là hợp lệ.
- Kế tiếp là đối với lists sẽ được ký hiệu là "**l**" theo sau là các phần tử của nó (cũng được mã hóa theo bencode) và kết thức bằng chữ "**e**". Ví dụ như: "**l4:spam4:eggse**" tương ứng với ['spam', 'eggs'].
- Cuối cùng là đối với dictionaries thì sẽ được encode bằng cách ký hiệu "**d**" theo sau là cặp khóa và giá trị tương ứng (cũng được bencode), và kết thúc bằng chữ "**e**" tương tự như lists. Ví dụ: "**d3:cow3:moo4:spam4:eggse**" tương ứng với dictionaries 'cow': 'moo', 'spam': 'eggs'. Các khóa phải là chuỗi và được sắp xếp theo thứ tự (sắp xếp theo rawstrings).

Lợi ích của mã hóa theo kiểu bencode là nó đơn giản (vì các số được mã hóa dưới dạng văn bản theo ký hiệu thập phân) và không bị tình trạng Endianness. Bencode hoạt động với mục đích giống như JSON và YAML, cho phép lưu trữ dữ liệu phức tạp nhưng có cấu trúc lỏng lẻo theo cách độc lập với nền tảng. Điều này cho phép lưu trữ bộ nhớ tuyến tính cho dữ liệu phức tạp.

2.3.2 Lý thuyết về file torrent

File torrent là một loại tệp nhỏ chứa thông tin cần thiết để tải và chia sẻ dữ liệu qua mạng BitTorrent, một giao thức chia sẻ tệp tin P2P (peer-to-peer). File torrent không chứa dữ liệu thực tế của tệp tin cần tải mà chỉ chứa siêu dữ liệu giúp quá trình tải tệp diễn ra.

Cấu trúc của một file torrent:

- **Announce URL**: URL của tracker – một máy chủ theo dõi các địa chỉ IP của những người dùng đang chia sẻ tệp tin này.
- **Info**: Đây là phần chứa thông tin chi tiết về tệp tin hoặc nhóm tệp tin. Nó gồm có:
  - **Name**: Tên của tệp tin hoặc thư mục được chia sẻ.
  - **Piece length**: Kích thước của mỗi mảnh (piece) của tệp tin.
  - **Pieces**: Mã băm của mỗi mảnh (để kiểm tra tính toàn vẹn của dữ liệu).
  - **File length**: Độ dài của tệp tin hoặc các tệp tin.
  - **Path** (nếu có nhiều tệp tin): Đường dẫn đến các tệp tin bên trong.

Ưu điểm của file torrent là:

- **Tối ưu băng thông**: Vì người dùng tải về từ nhiều nguồn khác nhau, tổng băng thông được phân phối.
- **Tải nhanh hơn với nhiều peer**: Khi có nhiều người tham gia vào mạng lưới chia sẻ, tốc độ tải có thể tăng lên.

Nhược điểm của file torrent là:

- **Tính pháp lý**: Torrent thường bị lạm dụng để chia sẻ các tệp tin vi phạm bản quyền, dẫn đến các vấn đề pháp lý.
- **Tính bảo mật**: Có thể trong file torrent có chứa một mã độc mà seeder cố tình đưa vào.
- **Yêu cầu seeder**: Nếu không có ai làm seeder (chia sẻ tệp tin hoàn chỉnh), việc tải tệp có thể bị dừng lại.

2.4 Lý thuyết về Tracker

Tracker trong mạng torrent có trách nhiệm theo dõi các người dùng và quản lý thông tin về tiến trình tải dữ liệu. Các yêu cầu GET từ client đến tracker bao gồm các key sau:

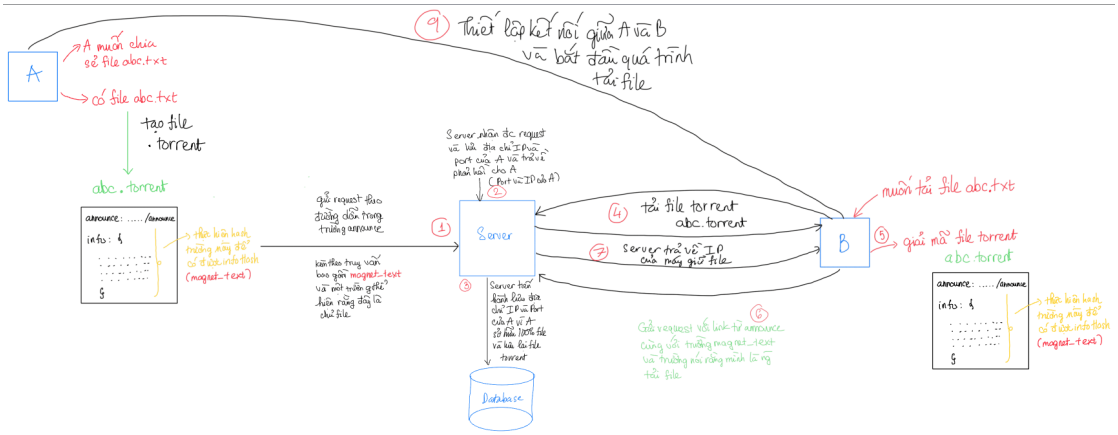
- **info_hash:** Mã băm SHA-1 dài 20 byte ở dạng được mã hóa của giá trị info trong tệp metainfo. Giá trị này được mã hóa và từng torrent có một giá trị duy nhất.
- **peer_id:** Chuỗi dài 20 ký tự, mỗi trình tải xuống tự tạo một id ngẫu nhiên khi bắt đầu một lượt tải mới (giá trị này phải được mã hóa).
- **ip:** Địa chỉ IP hoặc tên miền của peer thường là địa chỉ của chính người dùng nếu cùng máy với tracker.
- **port:** số cổng mà peer này đang nghe. Trình tải xuống sẽ nghe lần lượt từ cổng 6881 đến cổng 6889 nếu cổng nào đã được sử dụng thì cứ lần lượt đi qua các cổng sau.
- **uploaded:** Tổng số byte đã tải lên đến thời điểm hiện tại, mã hóa ở dạng ASCII cơ số 10.
- **downloaded:** Tổng số byte đã tải xuống đến thời điểm hiện tại, mã hóa ở dạng ASCII cơ số 10.

- **left:** Số byte mà peer này còn phải tải, mã hóa ở dạng ASCII cơ số 10. Lưu ý rằng giá trị này không thể tính toán từ downloaded và tổng chiều dài tệp vì quá trình tải có thể bị gián đoạn và một số dữ liệu tải xuống không hợp lệ phải tải lại.
- **event:** Là key tùy chọn, có thể là "started", "completed", "stopped" hoặc để trống. Nếu để trống, đây là một thông báo được thực hiện đều đặn. Thông báo"started" được gửi khi quá trình tải xuống bắt đầu lần đầu và thông báo "completed" được gửi khi quá trình tải xuống hoàn tất. Không có "completed" nào được gửi nếu tệp đã hoàn tất khi bắt đầu. Người tải xuống sẽ gửi thông báo "stopped" khi ngừng tải xuống.

Phản hồi từ tracker là các từ điển được mã hóa. Nếu phản hồi có chứa lý do fail, tức là yêu cầu thất bại và không cần thêm thông tin nào khác. Nếu thành công, phản hồi sẽ có hai key chính: interval (thời gian chờ giữa các lần yêu cầu) và peers (danh sách các peer hiện tại). Một số tracker còn sử dụng giao thức UDP để thông báo.

3 Phần mềm

3.1 Biểu đồ sơ lược về đường đi (flow) của phần mềm



Hình 1: Flow hoạt động

3.2 Đối tượng sử dụng

- Những người dùng cá nhân có nhu cầu chia sẻ tệp tin nhanh chóng.
- Các nhóm làm việc, dự án có nhu cầu chia sẻ tệp tin cùng nhau.
- Các cộng đồng có nhu cầu chia sẻ tệp tin với nhau.
- Các tổ chức nghiên cứu học thuật có nhu cầu chia sẻ tệp tin.

3.3 Yêu cầu về phần mềm

3.3.1 Yêu cầu về chức năng

Yêu cầu chức năng của Người dùng (Node):

- Đăng nhập/Đăng ký với hệ thống: Người dùng có thể đăng nhập hoặc đăng ký tài khoản để kết nối với máy chủ tracker.
- Cập nhật thông tin tệp tin: Sau khi kết nối người dùng thông báo với máy chủ tracker về các tệp tin sẵn sàng chia sẻ.
- Tải tệp từ các peer khác: Khi người dùng cần tải xuống một tệp không có sẵn trong kho lưu trữ, người dùng sẽ yêu cầu máy chủ tracker và nhận danh sách các peer khác để tải về.
- Kết nối và tải từ nhiều nguồn cùng lúc: Người dùng có thể kết nối với nhiều peer để tải các phần của tệp từ nhiều nguồn đồng thời.
- Chia sẻ tệp tin: Sau khi tải xuống người dùng có thể chia sẻ các tệp với các peer khác trong hệ thống.
- Xem tiến độ và thống kê: Người dùng có thể xem tiến độ tải lên/tải xuống cũng như các thông tin về tốc độ và danh sách các peer đang kết nối.

Yêu cầu chức năng của máy chủ theo dõi (Tracker):

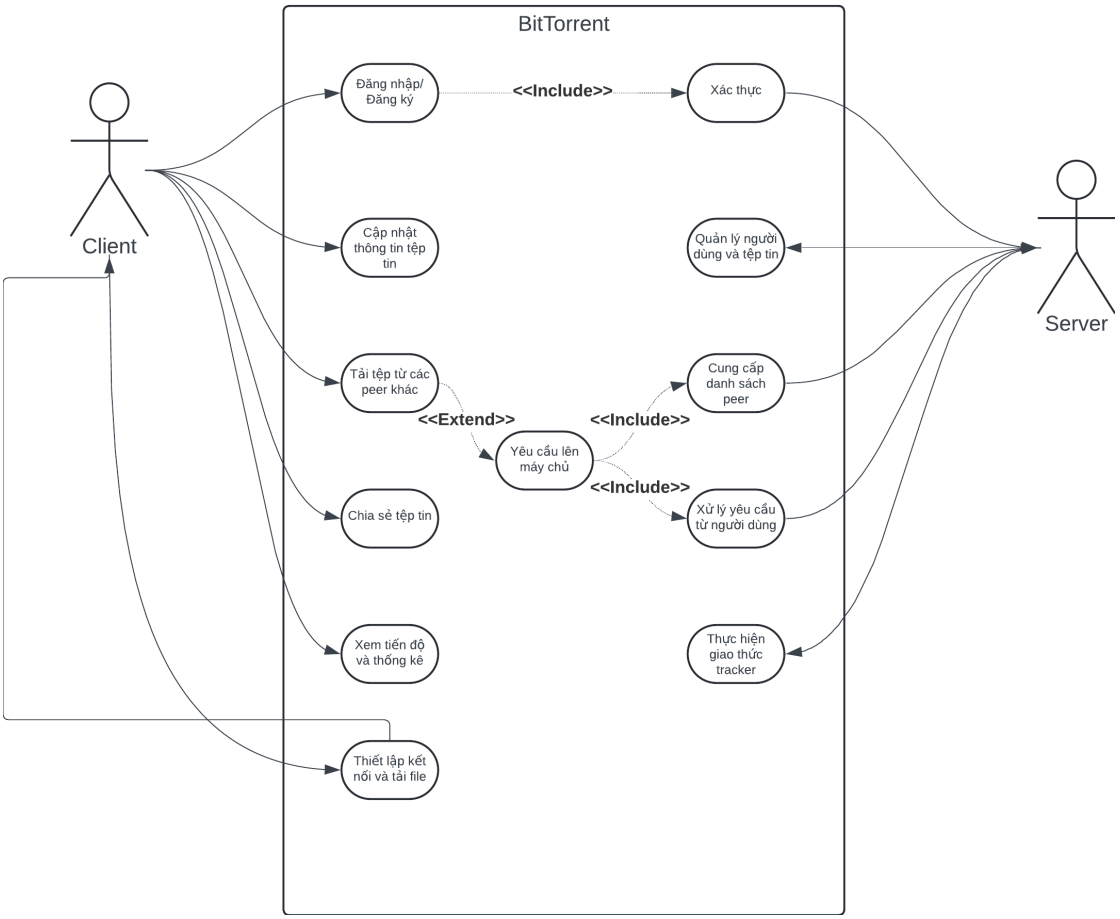
- Quản lý danh sách người dùng và tệp tin: Máy chủ lưu trữ thông tin về những người dùng đang kết nối và những tệp tin mà họ đang chia sẻ.
- Cung cấp danh sách peer: Khi một người dùng yêu cầu tải tệp, máy chủ cung cấp danh sách các peer đang có tệp đó để giúp người dùng tải về.
- Thực hiện giao thức tracker: Máy chủ theo dõi các peer, tệp tin và quản lý các kết nối giữa các peer với nhau thông qua giao thức tracker.
- Xử lý yêu cầu từ người dùng: Máy chủ nhận và phản hồi lại các yêu cầu từ người dùng về các thông tin tệp và danh sách các peer thông qua giao thức HTTP.

3.3.2 Yêu cầu phi chức năng

- Hệ thống phảo đảm bảo đạt hiệu suất khi upload và download cao, giảm thiểu độ trễ và quản lý các băng thông hiệu quả.
- Hệ thống phải đảm bảo bảo mật thông tin của từng Node.
- Hệ thống phải đảm bảo tính nhất quán của dữ liệu, tính toàn vẹn khi tải tệp tin về máy.
- Hệ thống phải đảm bảo việc tiếp tục tải tệp ngay cả khi một số peers bị ngắt kết nối hoặc không còn hoạt động.
- Hệ thống phải cung cấp giao diện thân thiện, dễ sử dụng cho người dùng, cung cấp tiến trình tải tệp tin cụ thể.

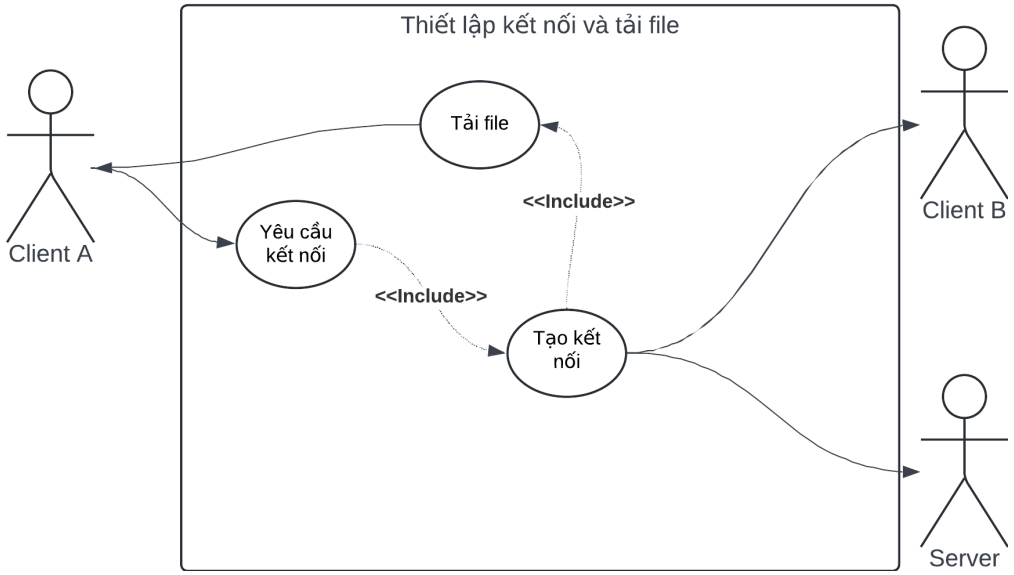
3.4 Sơ đồ Usecase (Usecase Diagram)

3.4.1 Sơ đồ Usecase cho toàn bộ hệ thống



Hình 2: Sơ đồ Usecase cho toàn bộ hệ thống

3.4.2 Sơ đồ Usecase cho module *Thiết lập kết nối và tải file*

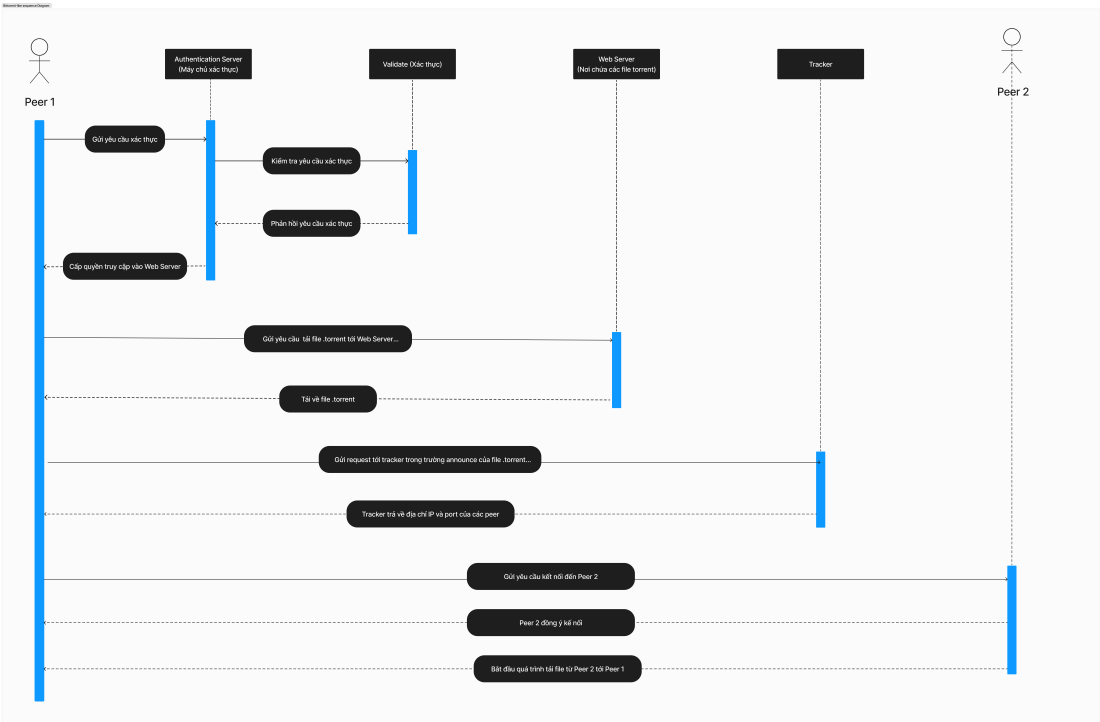


Hình 3: Sơ đồ Usecase cho module thiết lập kết nối và tải file

Use-case Name	Thiết lập kết nối và tải file
Usecase ID	TLKN&TF
Description	Tạo một cổng kết nối giữa 2 Client để cho việc tải file
Trigger	Khi client A yêu cầu tải file hoặc ngược lại
Pre-conditions	Cả hai client đều phải đang kết nối với Internet và file mà client A hoặc B yêu cầu phải tồn tại trên server
Post-conditions	Client A hoặc B tải được đúng file đã yêu cầu trước đó
Normal Flow	1. Client A (hoặc B) gửi yêu cầu tải file lên server. 2. Server tìm được file torrent dựa trên magnet text mà client A(hoặc B) gửi lên server. 3. Server tìm được magnet-text được lưu trong database và trả về địa chỉ IP của seeder (Client A hoặc B) cho Client nào muốn tải file. 4. Bên mong muốn tải file (A hoặc B) gửi yêu cầu thiết lập kết nối TCP cho bên còn lại. 5. Bên seeder phản hồi lại cho bên muốn tải file là việc tải file đã sẵn sàng và tải file theo yêu cầu của bên mong muốn tải file.
Exception Flow	1. Sẽ hiển thị ra lỗi cho client A. 2. Quay lại màn hình yêu cầu tải file cho client A.
Alternative Flow	1. Ở bước 5, nếu một bên không chấp nhận tải file thì sẽ huỷ toàn bộ quá trình và quay lại màn hình yêu cầu tải file.

Bảng 1: Usecase Table cho module Thiết lập kết nối và tải file

3.5 Sơ đồ Sequence (Sequence Diagram)



Hình 4: Sequence Diagram

Tài liệu