**ASSIGNMENT 1: PHÁT TRIỂN ỨNG DỤNG MẠNG**

**I. Mô tả đề tài**

**1. Mục tiêu**

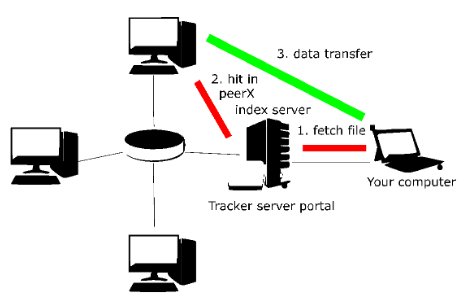
Xây dựng một Ứng Dụng Giống Torrent Đơn Giản (Simple Torrent-like Application - STA) với các giao thức do mỗi nhóm tự định nghĩa, sử dụng ngăn xếp giao thức TCP/IP và phải hỗ trợ việc truyền dữ liệu đa hướng (Multi-direction Data Transferring - MDDT).

**2. Mô tả ứng dụng**

**a. Tổng quan ứng dụng**

* Ứng dụng bao gồm hai loại máy chủ: tracker và node.
* Một tracker tập trung theo dõi nhiều node và lưu trữ thông tin về phần của các tệp.
* Thông qua giao thức tracker, một node thông báo với server về các tệp có trong kho lưu trữ cục bộ của nó, nhưng không thực sự truyền tệp lên server.
* Khi một node yêu cầu tệp mà không có trong kho của nó, một yêu cầu sẽ được gửi đến tracker.
* **MDDT**: Client có thể tải xuống nhiều tệp từ nhiều node nguồn cùng một lúc.

Yêu cầu rằng mã của node phải được triển khai đa luồng (multithreaded).



Hình 1. Illustration of file-sharing system

**b. Các thành phần cần thiết của ứng dụng giống Torrent**

Trước khi bắt đầu với việc truyền tải mạng, bạn cần biết về tất cả các thành phần nhỏ nhưng rất quan trọng sau đây:

* **Magnet text:** là văn bản đơn giản chứa tất cả thông tin cần thiết để dẫn tới tệp Metainfo trên tracker. Yêu cầu tối thiểu là một mã băm (hash code) trỏ đến thông tin tệp/metainfo được lưu trữ trên portal tracker tập trung của bạn (còn gọi là tracker-portal địa phương của bạn).
* **Tệp Metainfo:** (còn gọi là tệp .torrent) chứa tất cả thông tin về torrent của bạn, bao gồm địa chỉ của tracker (địa chỉ IP), tệp là gì, chiều dài của từng phần, và số lượng phần. Điều này có thể coi là thông tin nơi mà magnet text trỏ đến và nó kết hợp với danh sách các tracker và các node.
* **Các phần (Pieces):** là đơn vị của tệp. Các phần là các phần có kích thước bằng nhau của tệp được tải về. Kích thước thông thường của mỗi phần khoảng 512KB. Phần này được liệt kê trong tệp Metainfo.
* **Files:** cũng được chỉ định trong tệp Metainfo. Tệp có thể bao gồm nhiều phần. Có thể có nhiều tệp trong một torrent. Điều này có nghĩa là bạn có thể cần ánh xạ không gian địa chỉ của tệp với không gian địa chỉ tệp nếu bạn có N phần và M tệp.
* **Lỗi nhỏ:** Các lỗi nhỏ trong toán học có khả năng xảy ra rất cao tại đây, nhưng bạn PHẢI thực hiện toàn bộ ánh xạ giữa tệp và các phần được liên kết thủ công và định nghĩa nó cho các mục tham chiếu sau này.
* **Điểm khởi đầu:** Bạn có thể triển khai một phần tệp trên mỗi torrent như là một điểm tối thiểu, nhưng dừng lại tại giai đoạn này có thể hạn chế đánh giá điểm của bạn.

**c. Giao thức HTTP của Tracker**

**Các tham số yêu cầu của Tracker:** Khi khởi động, bạn được yêu cầu liên hệ với máy chủ tracker (portal tracker tập trung toàn cầu) và gửi tất cả các trường thông tin hỗ trợ phù hợp.

* Bạn nên đưa vào cờ nén (compact flag) hoặc metainfo trong yêu cầu của bạn, yêu cầu tối thiểu là một tệp magnet của torrent của bạn có thể phân tích được (để dẫn tới một tệp Metainfo được đặt trên server).
* Chúng ta có thể tham khảo ví dụ mẫu với hai giai đoạn: mô hình client-server và mô hình peer-to-peer.
* Cuối cùng, bạn cần phải gửi các yêu cầu bắt đầu, dừng và hoàn thành tới server tracker khi cần thiết. Khi bạn gửi yêu cầu bắt đầu, bạn cần đưa vào số byte bạn đã tải xuống cho đến lúc đó (theo chi tiết cấp độ phần).

**Phản hồi của Tracker:** Chúng tôi xác định một tập hợp con giới hạn của đặc tả cho BitTorrent(<https://wiki.theory.org/BitTorrentSpecification>)

* Tracker ban đầu được định nghĩa như một tracker tập trung và duy nhất, nhưng không có giới hạn trong việc nâng cấp nó thành tracker nhiều mục tiêu với lý thuyết DNS sử dụng phương pháp lặp hoặc đệ quy.

Tracker phản hồi với tài liệu "text/plain" chứa từ điển (dictionary) với các khóa sau:

* **Lý do lỗi (Failure reason):** Nếu có, các khóa khác không cần có mặt. Giá trị là thông báo lỗi dạng văn bản (string) cho biết lý do yêu cầu bị từ chối.
* **Thông báo cảnh báo (Warning message):** (mới, tùy chọn) Tương tự như lý do lỗi, nhưng phản hồi vẫn được xử lý như bình thường. Thông báo cảnh báo giống như một lỗi nhẹ.
* **Tracker ID:** Một chuỗi mà client nên gửi lại trong các thông báo tiếp theo. Nếu vắng mặt và client chưa gửi ID tracker trước đó, không nên bỏ qua giá trị cũ; tiếp tục sử dụng nó.
* **Peers:** (giá trị từ điển) Giá trị này là một danh sách các từ điển, mỗi từ điển có các khóa sau:
* **peer id:** ID tự chọn của peer, như đã được mô tả trong yêu cầu của tracker (chuỗi ký tự).
* **ip:** Địa chỉ IP của peer, có thể là IPv6 (dạng hex) hoặc IPv4 (dạng số) hoặc tên DNS (dạng chuỗi).
* **port:** Số cổng của peer (dạng số nguyên).

**d. PEER - Tải xuống (Downloading)**

Sau khi bạn nhận được danh sách các peer từ Tracker, bạn cần kết nối với càng nhiều peer càng tốt và bắt đầu tải xuống từ họ.

* Bắt buộc phải hỗ trợ MDDT.
* **Thuật toán:** Thuật toán để chọn các block để yêu cầu từ peer hoàn toàn phụ thuộc vào bạn, với yêu cầu tối thiểu là bạn không gửi yêu cầu cho một phần mà peer không có, và không gửi yêu cầu cho một phần mà bạn đã có.

Phát triển nâng cao:

Để tối ưu hóa băng thông mạng, bạn nên có một hàng đợi yêu cầu (request queue), trong đó lưu giữ danh sách tất cả các block bạn đã gửi yêu cầu tới các peer.

Vì bạn sẽ có nhiều peer cho cùng một torrent, bạn cần giữ trạng thái hiện tại về tiến trình của các block đã được yêu cầu từ tất cả các peer, để tránh gửi yêu cầu cho cùng một block tới nhiều peer, và hỗ trợ phục hồi từ việc ngắt kết nối trong quá trình truyền tệp đang diễn ra.

**e. PEER - Tải lên (Uploading)**

Không thể tải xuống các tệp từ client của bạn nếu không có người seeder. Do đó, ứng dụng của bạn phải bắt đầu **chia sẻ tệp** cho các peer khác khi họ cũng có nhu cầu tải xuống tệp sau khi bạn đã tải xong nó.

Phát triển nâng cao:

* Lý thuyết **"ăn miếng trả miếng" (tit-for-tat)** là nền tảng cho việc chia sẻ tệp kiểu peer-to-peer, ngụ ý rằng càng chia sẻ nhiều, càng có nhiều tệp khác sẽ được chia sẻ với bạn. Dự án này là cơ hội để áp dụng lý thuyết đã học của bạn.
* Thuật toán hiện tại để chọn peer nhằm ngăn chặn hành vi lợi dụng (free-riders) chỉ thay đổi quyết định lựa chọn mỗi mười giây.

**f. Giao diện người dùng (User Interface)**

Phần này của bài tập cho phép bạn sáng tạo rất nhiều: bạn có thể sử dụng và triển khai bất kỳ kiểu client nào bạn muốn, miễn là nó đáp ứng các yêu cầu tối thiểu về việc tải xuống và chia sẻ một torrent với nhiều peer, và cung cấp khả năng xem tất cả các thống kê tải lên/tải xuống.

Nói chung, <http://www.transmissionbt.com/> có mô tả đầy đủ. Bạn có thể hỗ trợ một số tham số và tùy chọn cơ bản của các dòng lệnh của transmission-client. Chúng tôi giữ lại mô tả chi tiết cho bạn tham khảo trong tương lai nếu cần.

* **transmission-cli:** một client torrent <https://linux.die.net/man/1/transmission-cli>. Ví dụ: transmission-cli <tệp torrent>
* **transmission-create:** <https://linux.die.net/man/1/transmission-create>
* **transmission-daemon:** <https://linux.die.net/man/1/transmission-daemon>
* **transmission-edit:** https://linux.die.net/man/1/transmission-edit
* **transmission-remote:** https://linux.die.net/man/1/transmission-remote
* **transmission-show:** <https://linux.die.net/man/1/transmission-show>

Bạn cũng có thể phát triển một giao diện đồ họa (GUI) nếu bạn có nhiều thời gian rảnh. Dù bạn có thể sẽ không được cộng điểm cho việc này, nhưng bạn có thể kiếm được một vài "ngôi sao" trên Github.

**g. Điểm thưởng (Extra credit)**

Bạn có thể chọn triển khai bất kỳ hoặc tất cả các mục sau (hoặc điều gì đó không có trong danh sách này nhưng có liên quan đến bittorrent và mạng):

* **Bảng băm phân tán (Distributed hash table - DHT):** truyền torrent của bạn mà không cần tracker tập trung. Điều này là tốt hơn để tìm thêm nhiều peer bằng cách sử dụng DHT thay vì tracker tập trung.
* **Tải xuống đồng thời (Simultaneous torrents):** yêu cầu client của bạn hiển thị cách tải xuống và tải lên nhiều torrent đồng thời (cùng lúc), cũng như truy cập các thống kê qua giao diện người dùng của bạn.
* **Tracker scrape:** <https://en.wikipedia.org/wiki/Tracker_scrape> là việc trao đổi để lấy thông tin metainfo thay mặt cho client. Nó nên được nhắc đến rằng sự tham gia của peer vào việc truyền tải dữ liệu không bị ảnh hưởng bởi các trao đổi scrape.
* **Chiến lược tải xuống/tải lên:** một chiến lược tải xuống đơn giản là yêu cầu đối với client của bạn, nhưng nếu bạn chọn áp dụng các chiến lược phức tạp hơn (như Super Seeding, Rarest-First, End Game), bạn cần phải ghi lại nó và cung cấp tài liệu bổ sung.
* **Tối ưu hóa việc chọn peer:** một chiến lược chọn peer đơn giản là bắt buộc đối với client của bạn, có nghĩa là bạn chỉ có thể phản hồi tới tất cả các peer của mình. Nếu bạn muốn sử dụng tiêu chuẩn thực tế của wiki (chỉ 4+1 peer tại một thời điểm), vui lòng đảm bảo rằng bạn ghi lại nó.

**h. Chấm điểm (Grading)**

Yêu cầu chính cho bài tập này bao gồm:

* Tải xuống một torrent nhiều tệp từ nhiều peer đồng thời.
* Tải lên một torrent nhiều tệp cho nhiều peer đồng thời.
* Hiển thị các thống kê hữu ích về việc tải xuống/tải lên, giúp chúng tôi kiểm tra thông tin của peer, các torrent đang được sử dụng.

**k. Cơ cấu chấm điểm của bạn sẽ như sau**

* **Giao thức Tracker (15%):** Phân tích đúng tệp metainfo, gửi yêu cầu đến tracker, nhận phản hồi với danh sách các peer tương ứng và phân tích danh sách peer thành các thông tin hữu ích như 'ip' và 'port'.
* **Tải xuống Torrent (30%):** Thiết lập kết nối tới nhiều peer, tải torrent từ tất cả chúng đồng thời, và sử dụng đúng ánh xạ để xác định các block cần yêu cầu. Bạn không được gửi yêu cầu cho các phần bạn đã lưu trên đĩa và phải xử lý việc tải xuống nhiều tệp đúng cách.
* **Tải lên Torrent (15%):** Cho phép nhiều peer kết nối với bạn để bạn có thể phục vụ nhiều luồng yêu cầu đồng thời.
* **Cài đặt nhiều máy chủ (Multiple host deployment) (20%):** Bài tập này liên quan đến kết nối mạng giữa nhiều host máy tính, vì vậy triển khai bản demo với nhiều hơn 3-4 host là bắt buộc (kỹ thuật máy ảo sẵn sàng để thực hiện yêu cầu này).
* **Tính năng MDDT (10%).**
* **Tính năng nâng cao (10%).**

**THỰC HÀNH**

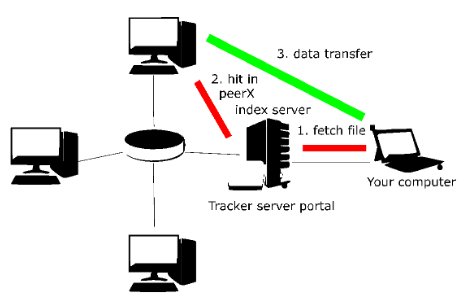
**I. Chuẩn bị Lý thuyết**

**1. Nghiên cứu về Torrent và hệ thống P2P (Peer-to-Peer)**

**Kiến trúc Torrent**

Hệ thống chia sẻ file Torrent: Đây là hệ thống phân tán cho phép các máy tính trong mạng P2P (peer-to-peer) chia sẻ các tệp với nhau mà không cần một server trung tâm quản lý tệp. Điều này làm tăng tốc độ tải xuống do người dùng có thể tải từ nhiều nguồn (peers) đồng thời.

* Cơ chế chia sẻ file: Trong hệ thống này, một tệp được chia thành nhiều phần nhỏ (pieces). Mỗi peer có thể tải xuống một hoặc nhiều phần từ các peer khác. Khi một peer đã tải xong một phần, nó có thể trở thành một nguồn (seeder) để chia sẻ phần đó với các peer khác.
* Vai trò của tracker: Tracker là một server đóng vai trò quản lý các peer trong mạng. Nó không lưu trữ file mà chỉ quản lý thông tin về các peer (địa chỉ IP, số cổng, các phần file mà peer có). Tracker giúp các peer tìm thấy nhau và trao đổi dữ liệu.
* Seeder và Leecher: Seeder: Peer có toàn bộ file và sẵn sàng chia sẻ; Leecher: Peer đang tải xuống file nhưng chưa có toàn bộ file, vừa tải vừa chia sẻ những phần đã tải được.
* Magnet Link: Magnet link là một URL chứa thông tin về một tệp torrent mà không cần tải xuống tệp .torrent. Nó chỉ chứa thông tin hash của tệp Metainfo để dẫn đến tracker, từ đó tải tệp torrent hoặc bắt đầu tải dữ liệu từ các peer.



**Tệp Metainfo**

Tệp Metainfo (.torrent): Đây là tệp chứa tất cả thông tin về torrent, bao gồm:

* Tên tệp và các phần nhỏ của nó (pieces).
* Địa chỉ IP của tracker để kết nối với mạng peer-to-peer.
* Hash của các phần nhỏ để đảm bảo tính toàn vẹn của dữ liệu.

Tệp .torrent không chứa dữ liệu file thực tế mà chỉ chứa thông tin về cách tải và xác thực file từ nhiều nguồn.

**Pieces (Các phần của tệp)**

* Pieces: Tệp torrent được chia thành nhiều phần có kích thước cố định (thường là 512KB), gọi là pieces. Mỗi piece được đánh số và hash riêng biệt để đảm bảo rằng file tải về không bị hỏng. Các peer tải xuống từng piece và kiểm tra tính toàn vẹn của piece đó dựa trên giá trị hash đã có trong Metainfo file.
* Pieces được tải ngẫu nhiên: Các phần của tệp không được tải xuống theo thứ tự từ đầu đến cuối. Thay vào đó, các phần sẽ được tải từ nhiều peer khác nhau, ưu tiên những phần mà ít peer đang có, điều này giúp tối ưu hóa quá trình chia sẻ file.

**Seeder và Leecher**

* Seeder: Khi một peer đã tải xong toàn bộ tệp và tiếp tục chia sẻ nó với các peer khác, nó trở thành seeder. Một torrent với nhiều seeder sẽ nhanh hơn vì nhiều peer có thể cung cấp file.
* Leecher: Một leecher là peer đang tải tệp về nhưng chưa có toàn bộ tệp. Các leecher vừa tải tệp từ seeder hoặc từ các leecher khác, vừa chia sẻ các phần mà họ đã tải được.

**2. Nghiên cứu về Giao thức TCP/IP và MDDT (Multi-direction Data Transferring)**

**Mô hình TCP/IP**

Giao thức TCP/IP là một bộ giao thức kết nối mạng phổ biến nhất được sử dụng trong internet. Nó quản lý cách dữ liệu được chia thành các gói nhỏ, gửi qua mạng, và được lắp ráp lại tại đích đến.

Tầng TCP (Transmission Control Protocol): Tầng này quản lý việc truyền dữ liệu đáng tin cậy giữa hai máy tính. Dữ liệu được phân mảnh thành các gói và gửi đi. Mỗi gói chứa một số thứ tự để đảm bảo rằng dữ liệu được lắp ráp lại theo đúng thứ tự.

Tầng IP (Internet Protocol): Đây là tầng cung cấp địa chỉ và quản lý định tuyến để các gói dữ liệu đi từ nguồn đến đích.

**MDDT (Multi-direction Data Transferring)**

MDDT là một khái niệm quan trọng trong ứng dụng mạng giống torrent. Nó cho phép một peer tải xuống các phần của một file từ nhiều nguồn khác nhau đồng thời (multi-source downloading). Điều này giúp tăng tốc độ tải xuống bằng cách sử dụng băng thông của nhiều peer.

Ví dụ về MDDT: Một peer tải một phần của file từ peer A, một phần khác từ peer B, và một phần khác từ peer C. Kết quả là quá trình tải xuống được phân tán, nhanh chóng và tối ưu hóa băng thông mạng.

Lợi ích của MDDT:

* Tối ưu băng thông: Bằng cách tải xuống từ nhiều nguồn đồng thời, băng thông tổng thể được sử dụng hiệu quả hơn, giúp tăng tốc độ tải xuống.
* Độ tin cậy cao hơn: Nếu một peer bị ngắt kết nối, bạn vẫn có thể tiếp tục tải xuống từ các peer khác mà không cần bắt đầu lại từ đầu.

**3. Nghiên cứu về Thuật toán Chia sẻ File P2P**

**Thuật toán Tit-for-tat (Ăn miếng trả miếng)**

Tit-for-tat là một thuật toán cơ bản trong việc chia sẻ file P2P để đảm bảo tính công bằng giữa các peer. Nguyên lý của nó là: "Nếu bạn chia sẻ với tôi, tôi sẽ chia sẻ lại với bạn". Peer chỉ tải tệp từ những peer cũng chia sẻ tệp ngược lại.

Ví dụ về Tit-for-tat: Peer A sẽ chỉ gửi dữ liệu cho Peer B nếu Peer B cũng gửi dữ liệu cho Peer A. Điều này ngăn chặn việc một peer chỉ nhận mà không chia sẻ, gọi là "free-rider".

**Chiến lược tải xuống (Download Strategies)**

Rarest-First (Chiến lược hiếm nhất trước): Peer sẽ tải xuống những mảnh tệp hiếm nhất trước. Điều này giúp đảm bảo rằng mọi phần của tệp luôn được chia sẻ trong mạng và không có phần nào bị thiếu hụt. Ví dụ: Nếu phần 5 của một tệp chỉ có ở một peer, trong khi các phần khác có ở nhiều peer, hệ thống sẽ ưu tiên tải phần 5 trước.

End-Game (Trò chơi cuối): Khi một peer gần tải xong tệp, nó sẽ gửi yêu cầu cho tất cả các peer để đảm bảo rằng tất cả các phần còn lại của tệp được tải về nhanh nhất có thể.

Super Seeding (Siêu gieo hạt): Đây là chiến lược mà seeder đầu tiên chỉ chia sẻ một vài mảnh tệp với các peer khác để các peer đó có thể chia sẻ lẫn nhau, thay vì chỉ tải từ seeder.

**4. Nghiên cứu về Giao thức HTTP của Tracker**

**Giao thức Tracker HTTP**

Tracker là một máy chủ trung gian giúp các peer trong hệ thống P2P tìm thấy nhau. Nó không lưu trữ file mà chỉ giữ thông tin về các peer nào đang có phần nào của tệp.

* Yêu cầu GET/POST: Các peer sử dụng HTTP để gửi yêu cầu tới tracker. Yêu cầu có thể bao gồm các thông tin như:
* **peer id:** ID của peer.
* **IP và port:** Địa chỉ IP và số cổng của peer.
* **hash của tệp Metainfo** mà peer đang tải xuống.
* Tracker scrape: Tracker trả về danh sách các peer có sẵn để tải dữ liệu, bao gồm thông tin về địa chỉ IP, port, và phần tệp mà mỗi peer có.

**Kết nối giữa các peer**

Sau khi nhận được danh sách peer từ tracker, mỗi peer sẽ cố gắng kết nối trực tiếp với các peer khác để bắt đầu tải xuống các phần tệp.

**II. Hiện thực ứng dụng Torrent-like**

**1. Phân tích và Thiết kế Kiến trúc Hệ thống**

**1.1. Xác định yêu cầu**

Trước khi bắt đầu viết mã, cần hiểu rõ các yêu cầu của ứng dụng:

* Hỗ trợ tải và chia sẻ file từ nhiều nguồn khác nhau (Multi-direction Data Transferring - MDDT): Máy khách (peer) phải có khả năng kết nối với nhiều peer cùng lúc để tải xuống các phần khác nhau của file.
* Giao thức trao đổi thông tin giữa tracker và peer: Máy chủ tracker quản lý thông tin về các peer, cung cấp danh sách peer cho mỗi yêu cầu từ peer.
* Tối ưu hóa băng thông mạng: Khi tải file từ nhiều nguồn, hệ thống cần đảm bảo sử dụng băng thông hiệu quả nhất, tránh lãng phí.
* **Khả năng xử lý đa luồng (multithreaded)**: Peer cần có khả năng xử lý nhiều kết nối đồng thời để có thể tải xuống từ nhiều peer một cách hiệu quả.

**1.2. Thiết kế kiến trúc ứng dụng**

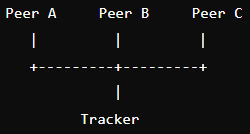
Kiến trúc hệ thống sẽ được chia làm hai phần chính: **Tracker** và **Peer**.

**Tracker**: Là thành phần quản lý thông tin về các peer và các file mà chúng đang chia sẻ. Tracker không chứa file, mà chỉ giữ danh sách các peer có file và cung cấp danh sách này cho các peer khác khi cần.

**Peer**: Mỗi peer đóng vai trò vừa là máy khách (client) tải file, vừa là máy chủ (server) chia sẻ file với các peer khác. Peer sẽ:

* Nhận danh sách peer từ tracker.
* Kết nối với các peer khác để tải xuống file.
* Khi đã tải xong một phần của file, peer sẽ bắt đầu chia sẻ phần đó với các peer khác.

**Sơ đồ kiến trúc**

****

Các thành phần chính trong Peer:

* **Downloading Module**: Quản lý quá trình tải xuống file từ các peer khác.
* **Uploading Module**: Quản lý quá trình chia sẻ file cho các peer khác.
* **Communication Module**: Giao tiếp với Tracker để nhận danh sách các peer có file.

**1.3. Sơ đồ dữ liệu và luồng thông tin**

**Sơ đồ dữ liệu**

* Tracker sẽ chứa thông tin về các file và danh sách các peer tương ứng.
* Peer sẽ chứa trạng thái của file mà nó đang tải, bao gồm các phần đã tải xong, phần đang tải, và danh sách các peer mà nó đang kết nối.

Luồng thông tin:

* **Bước 1:** Peer gửi yêu cầu tới Tracker để nhận danh sách các peer có file.
* **Bước 2:** Tracker trả về danh sách peer và các phần file mà mỗi peer có.
* **Bước 3:** Peer kết nối với nhiều peer khác nhau và bắt đầu tải xuống các phần của file.
* **Bước 4:** Sau khi tải xong một phần, Peer sẽ bắt đầu chia sẻ phần đó với các peer khác.
* **Bước 5:** Khi tải xong toàn bộ file, Peer trở thành một Seeder, giúp các peer khác hoàn thành việc tải file.

**2. Hiện thực mã cho Tracker**

**2.1. Giao thức HTTP cho Tracker**

Tracker HTTP Protocol: Tracker sử dụng giao thức HTTP để giao tiếp với các peer. Khi một peer muốn tải file, nó sẽ gửi yêu cầu HTTP đến Tracker, yêu cầu danh sách các peer có sẵn để tải file.

Các API cơ bản của Tracker:

* **GET /announce**: Peer sẽ gửi yêu cầu đến URL này để nhận danh sách các peer có file mà nó đang tìm.
* **POST /completed**: Khi một peer đã tải xong file, nó sẽ gửi thông tin đến tracker để báo rằng nó đã trở thành Seeder.

Ví dụ về luồng yêu cầu giữa Peer và Tracker: Peer A muốn tải một file, nó sẽ gửi yêu cầu HTTP đến Tracker thông qua API GET /announce. Tracker sẽ trả về danh sách các peer hiện có sẵn file đó (Peer B, Peer C). Sau đó, Peer A sẽ kết nối trực tiếp với Peer B và C để bắt đầu tải file.

**2.2. Quản lý danh sách các peer**

Danh sách peer: Tracker sẽ quản lý một cơ sở dữ liệu (hoặc danh sách trong bộ nhớ) chứa thông tin về tất cả các peer đang tham gia chia sẻ file. Mỗi peer sẽ được gán một ID, địa chỉ IP và số cổng mà nó sử dụng để kết nối.

Cập nhật thông tin peer: Khi một peer kết thúc việc tải file và trở thành Seeder, nó sẽ gửi yêu cầu đến Tracker để cập nhật trạng thái của nó. Tracker sẽ lưu trữ thông tin này để chia sẻ với các peer khác.

**3. Hiện thực mã cho Peer**

**3.1 Tải xuống tệp (Downloading Module)**

Kết nối tới nhiều peer: Peer sẽ phải mở nhiều kết nối TCP/IP đồng thời đến các peer khác để tải xuống các phần của file. Mỗi kết nối sẽ sử dụng một luồng (thread) để xử lý tải xuống, do đó việc quản lý đa luồng là rất quan trọng.

Thuật toán chọn block:

* Peer cần chọn phần (block) nào của file để tải trước. Điều này có thể dựa trên chiến lược "Rarest-First", tức là tải các phần hiếm nhất (được ít peer chia sẻ) trước.
* Đảm bảo rằng peer không tải lại các block mà nó đã có.

Quản lý hàng đợi yêu cầu (Request Queue):

* Mỗi block được yêu cầu từ một peer sẽ được đặt vào một hàng đợi (queue). Hàng đợi này giúp theo dõi các block nào đã được yêu cầu nhưng chưa tải xong.

**3.2 Tải lên tệp (Uploading Module)**

* Chia sẻ các block đã tải xuống: Khi peer đã tải xong một block, nó sẽ bắt đầu chia sẻ block này với các peer khác.
* Quản lý quá trình chia sẻ theo nguyên tắc "Tit-for-Tat": Peer chỉ chia sẻ nhiều dữ liệu với các peer cũng chia sẻ lại với mình. Điều này giúp tránh tình trạng "free-riders" (những peer chỉ tải mà không chia sẻ).

**4. Hỗ trợ Truyền Dữ liệu Đa Hướng (MDDT)**

**4.1 Tải từ nhiều nguồn cùng lúc**

MDDT yêu cầu Peer có khả năng kết nối với nhiều peer cùng lúc để tải xuống các phần khác nhau của file. Điều này giúp tăng tốc độ tải và tối ưu hóa băng thông mạng.

Quản lý tiến trình tải xuống:

* Peer cần giữ trạng thái của mỗi phần file đang tải từ mỗi peer. Ví dụ, Peer A có thể tải phần 1 từ Peer B, phần 2 từ Peer C, và phần 3 từ Peer D.
* Đảm bảo rằng các phần file đã tải xong sẽ không bị yêu cầu tải lại từ các peer khác.

**5. Xây dựng Giao diện Người Dùng (UI)**

**5.1 Thiết kế giao diện người dùng đơn giản**

Giao diện người dùng có thể là dòng lệnh (CLI) hoặc đồ họa (GUI), tùy thuộc vào yêu cầu của bài tập. Trong CLI, giao diện sẽ hiển thị:

* Danh sách các tệp đang tải.
* Tiến trình tải: Hiển thị các phần của file đã tải được và phần còn lại.
* Tốc độ tải xuống/tải lên: Hiển thị băng thông đang sử dụng.

**5.2 Hiển thị các kết nối Peer-to-Peer**

Danh sách các peer kết nối: Hiển thị danh sách các peer mà máy của bạn đang kết nối.

Thống kê tải xuống/tải lên: Thống kê số lượng block đã tải xuống từ mỗi peer và số lượng block bạn đã chia sẻ với các peer khác.

**6. Kiểm thử và Tối ưu hóa**

**6.1 Kiểm tra hệ thống**

Mạng thử nghiệm: Tạo một mạng lưới giả lập với nhiều peer để kiểm tra xem hệ thống có thể quản lý tốt các kết nối đồng thời hay không.

Kiểm tra tính ổn định: Đảm bảo rằng hệ thống không bị lỗi khi có nhiều peer cùng kết nối và tải file.

**6.2 Tối ưu hóa mã nguồn**

Tối ưu hóa thuật toán chọn peer: Đảm bảo rằng hệ thống không bị quá tải khi có quá nhiều kết nối. Tối ưu hóa việc chọn peer để cân bằng tải giữa các peer.