

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



**BÁO CÁO BÀI TẬP LỚN MÔN HỌC**

---

**MÔN HỌC: Mạng Máy Tính - Computer Networks - CO3093**

**DEVELOP A NETWORK APPLICATION**

**HK241**

---

**Giảng viên hướng dẫn:** Nguyễn Phương Duy  
**Sinh viên:** Trần Đình Tường - 2213892  
Trần Đại Việt - 2213951  
Nguyễn Phương Duy - 2210526  
Trần Thành Long - 2153538

HO CHI MINH CITY, SEPTEMBER 2024

# Mục lục

<b>1</b>	<b>Danh Sách Thành Viên Và Phân Công Nhiệm Vụ</b>	<b>1</b>
<b>2</b>	<b>Tổng quan về ứng dụng mạng</b>	<b>1</b>
<b>3</b>	<b>Cơ sở lý thuyết hiện thực ứng dụng</b>	<b>3</b>
3.1	Mô hình mạng ngang hàng P2P . . . . .	3
3.2	Giao thức TCP/IP . . . . .	3
3.3	Mô hình BitTorrent . . . . .	4
<b>4</b>	<b>Phân tích yêu cầu chức năng</b>	<b>5</b>
4.1	Yêu cầu chức năng của Tracker . . . . .	5
4.2	Yêu cầu chức năng của Peer . . . . .	5
<b>5</b>	<b>Thiết kế kiến trúc hiện thực ứng dụng</b>	<b>6</b>
5.1	Thiết kế kiến trúc của hệ thống . . . . .	6
5.2	Package Diagram . . . . .	8
5.3	Deployment Diagram . . . . .	9
<b>6</b>	<b>Mô hình hóa các tương tác trong hệ thống</b>	<b>10</b>
6.1	Usecase Diagram . . . . .	10
6.2	Activity Diagram . . . . .	10
<b>7</b>	<b>Kết quả hiện thực:</b>	<b>12</b>
7.1	Tracker . . . . .	12
7.2	Giao tiếp giữa các Peers và Tracker thông qua HTTP . . . . .	13
7.3	Peer . . . . .	14
<b>8</b>	<b>Đường dẫn tới kết quả hiện thực và Demo ứng dụng:</b>	<b>18</b>

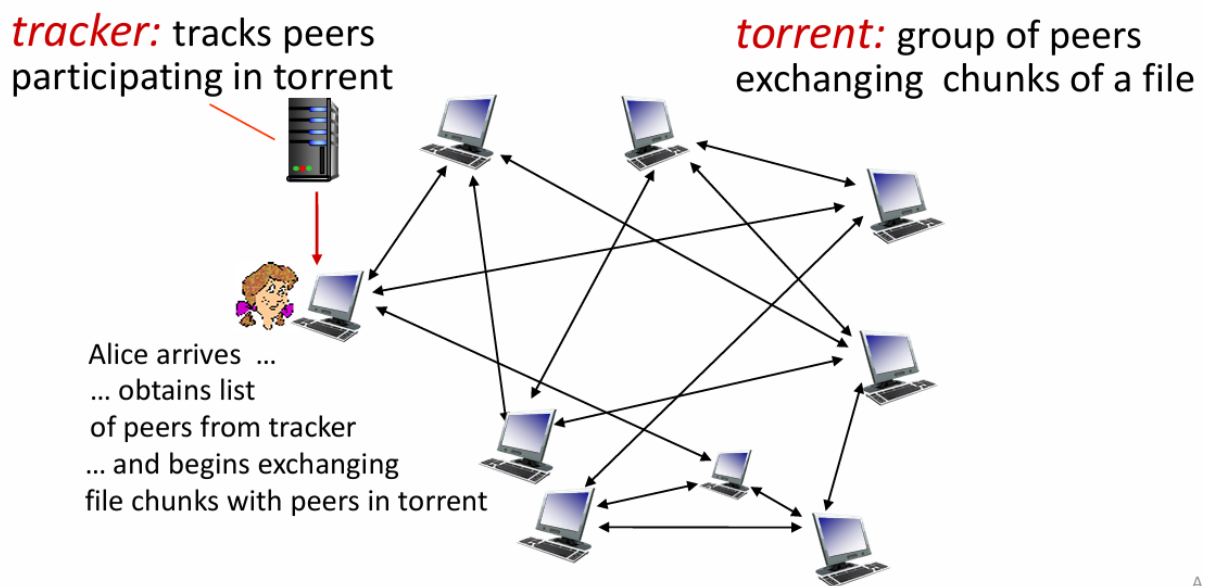
## 1 Danh Sách Thành Viên Và Phân Công Nhiệm Vụ

MSSV	Họ và tên	Nhiệm vụ	Phần trăm công việc
2213951	Trần Đại Việt	Tìm hiểu và hiện thực	100%
2213892	Trần Đình Tường	Tìm hiểu và hiện thực	100%
2210526	Nguyễn Phương Duy	Tìm hiểu và hiện thực	100%
2153538	Trần Thành Long	Tìm hiểu và hiện thực	100%

Bảng 1: Danh sách sinh viên và nhiệm vụ

## 2 Tổng quan về ứng dụng mạng

Trong bài tập lớn này nhóm chúng em sẽ xây dựng một ứng dụng Simple Torrent-like Application (STA) đây là một hệ thống chia sẻ tệp dựa trên giao thức TCP/IP, được thiết kế để hỗ trợ truyền dữ liệu đa hướng (MDDT). Ứng dụng này chủ yếu sẽ được xây dựng và hiện thực dựa trên mô hình BitTorrent của mạng ngang hàng P2P. Hệ thống bao gồm hai loại Host chính: Tracker và Peer. Tracker đóng vai trò như một Host chuyên về việc quản lý danh sách các Peer tham gia mạng và lưu trữ các thông tin cơ bản đối với các Peer đó, như Peer đó sẽ chứa Data cho những Torrent với Info\_hash nào, việc lưu trữ này sẽ hỗ trợ các thông tin cần thiết cho loại Host còn lại là Peer, những thông tin được lưu trữ ở Tracker sẽ được các Peer gửi Request thông HTTP Protocol để lấy, sau đó sẽ kết nối với các Peer cần thiết khác trong mạng để tải tệp từ các nguồn khác hay đóng vai trò làm nguồn cung cấp dữ liệu (seeder) cho các Peer khác trong mạng.

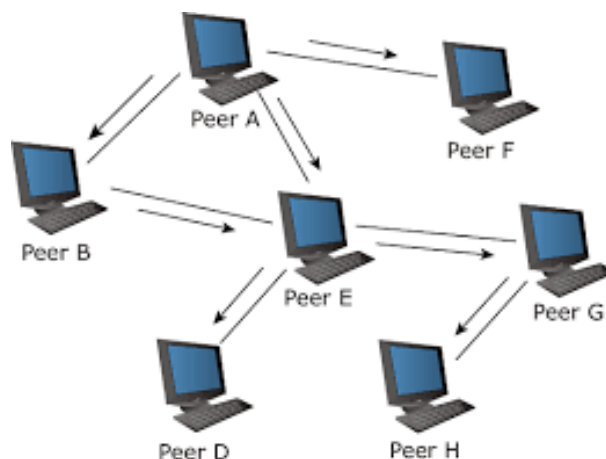


Hình 1: Simple Torrent-like Application (STA)

STA được thiết kế nhằm tối ưu hóa quá trình tải xuống nhiều tệp cùng lúc từ nhiều nguồn, yêu cầu triển khai mô hình đa luồng (multithreaded) để tối ưu hiệu suất truyền tải. Ứng dụng sử dụng các thành phần chính như Info\_hash là mã định danh cho các File Torrent (.torrent) có trong hệ thống chứa thông tin chi tiết về tệp như địa chỉ tracker, độ dài các Piece, và số lượng Piece. Các tệp lớn sẽ được chia nhỏ thành các Piece có kích thước đồng đều (thường khoảng 512KB) để tăng hiệu quả truyền tải.

Khi một Peer yêu cầu tải tệp không có sẵn trong kho lưu trữ của nó, nó sẽ gửi yêu cầu đến tracker. Tracker phản hồi bằng danh sách các Peer chứa các phần cần thiết của tệp, và node sẽ kết nối tới các peer này để tải dữ liệu. Quá trình này yêu cầu các thuật toán quản lý hàng đợi hay khả năng lưu trữ tiến trình tải giúp theo dõi trạng thái của các phần tệp đang được tải, tránh yêu cầu trùng lặp và đảm bảo khả năng vẫn tiếp tục tải tiếp tiến trình còn lại trong trường hợp mất kết nối. Ngoài chức năng tải xuống, STA còn hỗ trợ tải lên (seeding), cho phép Peer

chia sẻ tệp với các Peer khác sau khi hoàn thành tải xuống.

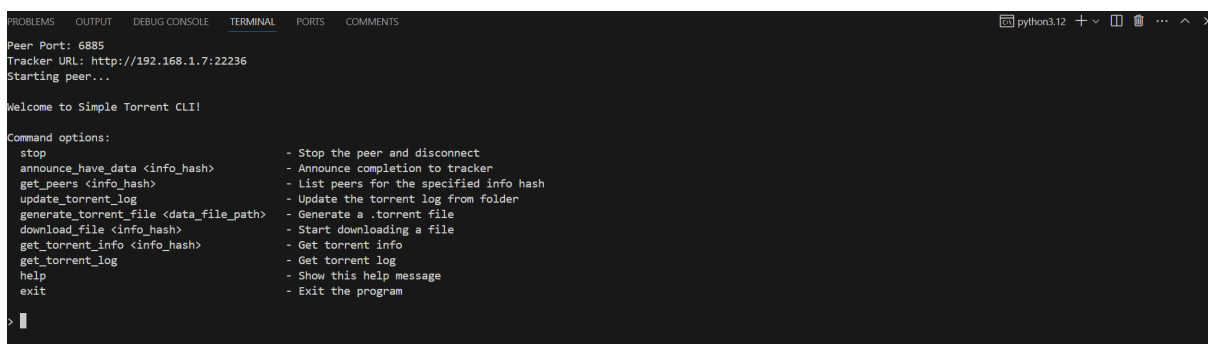


Hình 2: Chia sẻ dữ liệu giữa các Peer

Ngoài ra ứng dụng còn cung cấp một giao diện dòng lệnh (CLI) đơn giản nhưng đầy đủ, giúp người dùng dễ dàng thao tác và quản lý các tác vụ trong hệ thống chia sẻ tệp ngang hàng. Giao diện CLI được thiết kế thân thiện, với các lệnh rõ ràng, trực quan, cho phép người dùng thực hiện các thao tác như khởi động hoặc dừng peer, thông báo trạng thái tải dữ liệu, tìm kiếm danh sách các peer liên quan đến một tệp cụ thể, tạo và quản lý các tệp .torrent, cũng như tải xuống và theo dõi thông tin về các torrent đang hoạt động.

Các lệnh CLI của dự án bao gồm những chức năng cơ bản và cần thiết như:

- Dừng kết nối: Người dùng có thể dễ dàng dừng peer và ngắt kết nối khỏi tracker thông qua lệnh stop.
- Thông báo Peer đã có đủ Piece cho một Torrent: Sử dụng lệnh announce\_have\_data, người dùng có thể gửi thông báo về việc hoàn thành đến tracker.
- Lấy danh sách peer: Lệnh get\_peers cho phép người dùng liệt kê danh sách các peer đang sở hữu tệp tương ứng với mã băm (info\_hash) chỉ định.
- Quản lý log torrent: CLI hỗ trợ cập nhật nhật ký torrent từ thư mục hiện tại bằng lệnh update\_torrent\_log.
- Tạo tệp torrent: Lệnh generate\_torrent\_file giúp tạo nhanh một tệp .torrent từ đường dẫn đến tệp dữ liệu cụ thể.
- Tải xuống tệp: Người dùng có thể bắt đầu tải một tệp từ mạng ngang hàng với lệnh download\_file.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
Peer Port: 6885
Tracker URL: http://192.168.1.7:22236
Starting peer...

Welcome to Simple Torrent CLI!

Command options:
stop                                - Stop the peer and disconnect
announce_have_data <info_hash>    - Announce completion to tracker
get_peers <info_hash>              - List peers for the specified info hash
update_torrent_log                 - Update the torrent log from folder
generate_torrent_file <data_file_path> - Generate a .torrent file
download_file <info_hash>          - Start downloading a file
get_torrent_info <info_hash>       - Get torrent info
get_torrent_log                    - Get torrent log
help                               - Show this help message
exit                               - Exit the program

> |
```

Hình 3: Command Line Interface

Với khả năng cung cấp CLI đơn giản nhưng hiệu quả, dự án không chỉ đảm bảo tính tiện dụng mà còn tạo điều kiện thuận lợi cho việc tương tác với hệ thống, phù hợp cho cả người mới bắt đầu và người dùng nâng cao. CLI giúp giảm thiểu thời gian thao tác thủ công, đồng thời hỗ trợ kiểm soát chặt chẽ các tiến trình liên quan đến tải lên, tải xuống và quản lý tệp torrent,...

### 3 Cơ sở lý thuyết hiện thực ứng dụng

Ứng dụng mạng trong bài tập này sẽ được phát triển dựa trên các nguyên lý cơ bản và cơ chế của giao thức BitTorrent, sử dụng mô hình mạng ngang hàng (P2P) và giao thức TCP/IP để chia sẻ dữ liệu hiệu quả. Sau đây là cơ sở lý thuyết để làm rõ các khái niệm, cấu trúc, và giao thức mà ứng dụng này kế thừa và áp dụng.

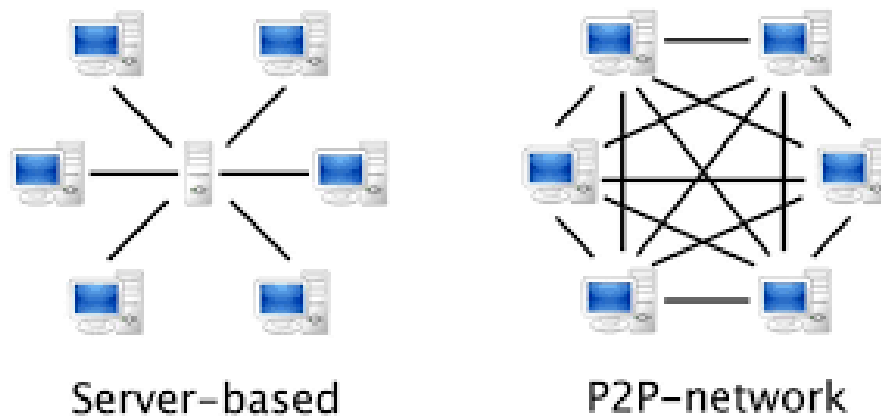
#### 3.1 Mô hình mạng ngang hàng P2P

Mạng ngang hàng Peer to Peer (P2P) là một hệ thống mạng máy tính, trong đó mỗi thiết bị có vai trò như nhau, vừa hoạt động như máy chủ (server) vừa hoạt động như máy khách (client). Điều này có nghĩa là các thiết bị trong mạng không phụ thuộc vào một Server duy nhất mà có thể trao đổi dữ liệu trực tiếp với nhau. Nói cách khác, trong mạng Peer to Peer, tất cả các máy tính đều có vai trò bình đẳng, cùng chia sẻ tài nguyên và thực hiện các nhiệm vụ.

Các mạng P2P được xây dựng dựa trên nguyên tắc tương đồng và tương tác trực tiếp giữa các thiết bị. Thay vì phải truy cập thông qua một máy chủ trung tâm, các thiết bị có thể trực tiếp kết nối với nhau qua giao thức truyền dẫn dữ liệu như BitTorrent,...

P2P phân tải dữ liệu hiệu quả bằng cách chia sẻ công việc giữa các node, giúp giảm tải cho máy chủ trung tâm và tăng khả năng mở rộng. Mô hình này cho phép tải xuống từ nhiều nguồn đồng thời, tối ưu hóa băng thông mạng, đồng thời duy trì hoạt động ngay cả khi một số node bị mất kết nối. P2P cũng hỗ trợ mở rộng mạng dễ dàng khi có thêm node tham gia, cải thiện khả năng chia sẻ dữ liệu mà không ảnh hưởng đến hiệu suất.

Tuy nhiên, P2P phức tạp trong quản lý, đòi hỏi các giao thức và thuật toán để đảm bảo phân phối dữ liệu và giám sát hiệu suất. Nó tiềm ẩn rủi ro bảo mật như phát tán phần mềm độc hại hoặc rò rỉ thông tin cá nhân do kết nối không được kiểm soát chặt chẽ. Hiệu suất mạng phụ thuộc vào số lượng và chất lượng seeders; nếu seeders ít hoặc không hoạt động, tốc độ tải xuống sẽ bị ảnh hưởng đáng kể.



Hình 4: Mô hình P2P đơn giản

#### 3.2 Giao thức TCP/IP

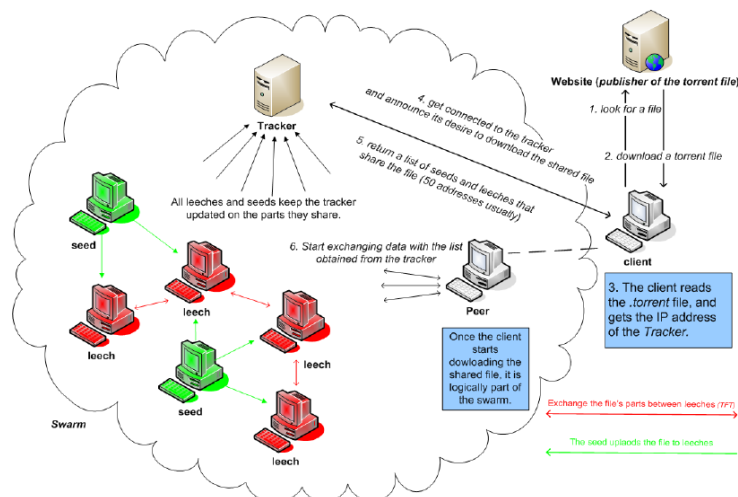
Giao thức TCP/IP là nền tảng cho việc truyền thông tin trong STA, đảm bảo tính đáng tin cậy và hiệu quả. Các tầng chính được sử dụng:

- Tầng ứng dụng: Quản lý giao tiếp giữa tracker và node thông qua giao thức HTTP.
- Tầng Transport: TCP đảm bảo dữ liệu được truyền tải toàn vẹn giữa các node và tracker.
- Tầng mạng: Sử dụng địa chỉ IP để định tuyến và xác định các node trong hệ thống.

Áp dụng trong việc hiện thực ứng dụng:

- Giao thức HTTP được sử dụng để giao tiếp giữa Peer và Tracker, hỗ trợ việc truyền tải thông tin như danh sách peers, trạng thái tệp, và yêu cầu tải.
- TCP đảm bảo các gói tin truyền đi các Peer không bị mất mát và được nhận đúng thứ tự.

### 3.3 Mô hình BitTorrent



Hình 5: Mô hình BitTorrent

Mô hình BitTorrent là một giao thức chia sẻ tệp ngang hàng (P2P) được thiết kế để tối ưu hóa việc truyền tải dữ liệu bằng cách phân chia tệp thành các phần nhỏ và cho phép nhiều peer cùng tham gia chia sẻ dữ liệu. Điều này giúp tăng hiệu quả băng thông, giảm tải cho máy chủ trung tâm và cải thiện tốc độ tải xuống.

Trong mạng BitTorrent, các thành phần chính bao gồm:

#### 1. Tracker

- Là máy chủ trung tâm quản lý thông tin về các Peers trong mạng và danh sách các Peers có dữ liệu ứng với file Torrent tương ứng.
- Tracker sử dụng giao thức HTTP/HTTPS để giao tiếp, cung cấp danh sách các peer có thể kết nối, đồng thời theo dõi trạng thái của từng peer trong mạng, bao gồm seeders (các peer có toàn bộ tệp) và leechers (các peer đang tải xuống).

#### 2. Peer

- Peer là các thiết bị tham gia vào mạng BitTorrent, đóng vai trò là khách hàng (client) vừa tải xuống dữ liệu từ các peer khác vừa chia sẻ dữ liệu của mình.
- Peer được phân thành hai loại gồm Seeder là các peer sở hữu toàn bộ tệp và chia sẻ cho các peer khác trong mạng và Leecher là các peer chưa có đầy đủ tệp và đang tải xuống từ seeders hoặc các leechers khác.
- Mỗi peer duy trì kết nối đa luồng (multithreaded), cho phép tải xuống từ nhiều peer khác đồng thời và chia sẻ dữ liệu ngay trong quá trình tải.

3. File Metainfo (.torrent): Tệp .torrent là một tệp chứa metadata được mã hóa theo định dạng bencode, cung cấp thông tin cần thiết để quản lý việc chia sẻ và tải tệp qua mạng BitTorrent. Cấu trúc của tệp bao gồm các thành phần chính:

- announce: URL của tracker, đóng vai trò kết nối các peer.
- info: Một dictionary chứa thông tin chi tiết về tệp hoặc thư mục, bao gồm:
  - files: Danh sách các tệp
  - name: Tên của thư mục lưu trữ các tệp hoặc tên tệp trong trường hợp chia sẻ một tệp.
  - piece length: Số byte trong mỗi phần tệp (piece).
  - pieces: Danh sách các giá trị hash SHA1 của từng phần tệp, dùng để kiểm tra tính toàn vẹn của dữ liệu.
- Tệp .torrent là cầu nối để thiết lập quá trình tải xuống từ nhiều nguồn một cách hiệu quả trong mạng BitTorrent.

## 4 Phân tích yêu cầu chức năng

### 4.1 Yêu cầu chức năng của Tracker

- **Quản lý trạng thái Tracker**

Tracker cần theo dõi trạng thái của từng torrent và các Peer liên quan, bao gồm:

- **info\_hash**: Đại diện cho tệp hoặc bộ tệp đang được chia sẻ.
- **complete**: Danh sách các Peer (*seeders*) đã tải xong toàn bộ tệp và đang chia sẻ.
- **incomplete**: Danh sách các Peer (*leechers*) đang tải xuống và chưa hoàn thành.
- **peers**: Danh sách tất cả các Peer tham gia vào mạng chia sẻ.

- **Xử lý các sự kiện từ Peer**

Tracker cần hỗ trợ các sự kiện chính của Peer:

- **started**: Khi một Peer bắt đầu tham gia mạng, Tracker thêm Peer này vào danh sách.
- **completed**: Khi một Peer hoàn tất tải xuống, trạng thái của nó chuyển từ "incomplete" sang "complete".
- **stopped**: Khi một Peer rời khỏi mạng, Tracker xóa Peer này khỏi tất cả các danh sách.

- **Giao tiếp với Peer thông qua HTTP**

Tracker sử dụng giao thức HTTP để xử lý các yêu cầu từ Peer:

- **/announce**: Nhận thông tin từ Peer (*info\_hash*, *peer\_id*, *port*, *event*) và trả về danh sách các Peer khác để kết nối.
- Phản hồi gồm:
  - \* Số lượng *seeders* (**complete**) và *leechers* (**incomplete**).
  - \* Danh sách các Peer khác mà Peer yêu cầu có thể kết nối.
  - \* Thông tin cảnh báo hoặc lỗi nếu có.

- **Xử lý thông tin Peer**

Tracker cần:

- Thêm Peer mới vào danh sách phù hợp (**complete** hoặc **incomplete**).
- Cập nhật trạng thái của Peer khi nhận được sự kiện mới.
- Loại bỏ Peer khi nhận sự kiện "*stopped*" hoặc khi mất kết nối.

- **Cơ chế mã hóa và truyền dữ liệu**

Tracker sử dụng **bencode** để mã hóa phản hồi dưới dạng nhị phân, phù hợp với giao thức BitTorrent.

- Dữ liệu trả về bao gồm địa chỉ IP, cổng và trạng thái của từng Peer.

### 4.2 Yêu cầu chức năng của Peer

Peer trong mô hình hoạt động BitTorrent cần thực hiện các chức năng chính sau:

- **Kết nối với Tracker**

- Gửi các thông báo (*announce*) đến Tracker để báo cáo trạng thái hoặc nhận danh sách các Peer khác.
- Hỗ trợ sự kiện **announce\_have\_data <info\_hash>**: Thông báo rằng dữ liệu đã được tải hoàn tất.

- **Quản lý danh sách Peer**

- Yêu cầu danh sách Peer từ Tracker (**get\_peers <info\_hash>**).
- Lưu trữ thông tin của các Peer khác để thiết lập kết nối và quản lý trạng thái.

- **Quản lý dữ liệu Torrent**

- **update\_torrent\_log**: Cập nhật trạng thái tệp torrent từ thư mục nội bộ.
- **generate\_torrent\_file <data\_file\_path>**: Tạo tệp .torrent chứa thông tin của dữ liệu được chia sẻ.
- **get\_torrent\_info <info\_hash>**: Lấy thông tin chi tiết về torrent.
- **get\_torrent\_log**: Hiển thị nhật ký hoạt động của torrent.

- Tải xuống và chia sẻ dữ liệu

- **download\_file <info\_hash>**: Bắt đầu tải tệp từ các Peer khác dựa trên *info\_hash*.
- Quản lý kết nối song song (đa luồng) với các Peer để tải nhiều phần dữ liệu (pieces) từ nhiều nguồn cùng lúc. Đảm bảo đáp ứng yêu cầu không gửi yêu cầu tải xuống một mảnh dữ liệu mà Peer không có và không gửi yêu cầu tải xuống một mảnh mà đã được tải về
- 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.
- Sau khi tải xuống dữ liệu, nó sẽ có khả năng chia sẻ dữ liệu đó (seeding) cho các peers khác, để các peers mới có thể tải xuống từ nó.

- Hỗ trợ giao diện người dùng CLI

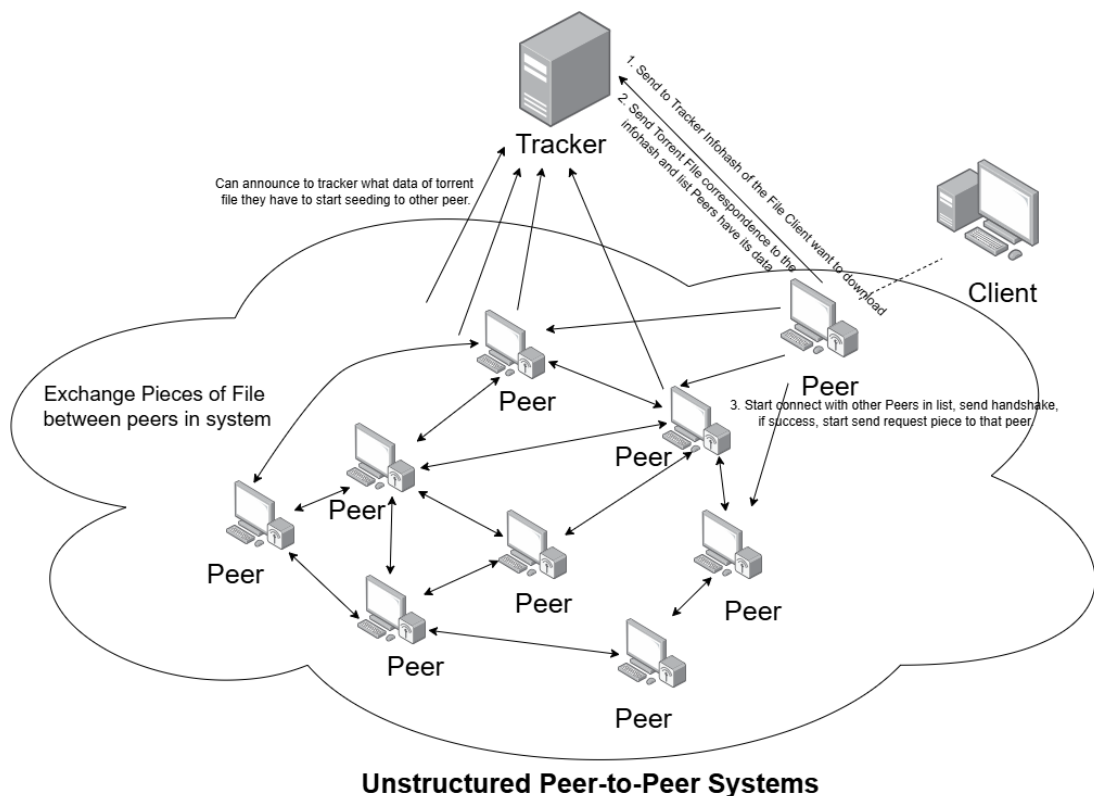
- Hiển thị các lệnh trợ giúp (**help**): Cung cấp thông tin chi tiết về các lệnh CLI.
- Kết thúc hoạt động (**exit**): Ngắt kết nối và dừng chương trình.

## 5 Thiết kế kiến trúc hiện thực ứng dụng

Sau đây là phần nội dung chi tiết về việc thiết kế kiến trúc để hiện thực ứng dụng, từ Tracker đến các Peer, hay cấu trúc Request, Response dùng để trao đổi dữ liệu qua giao thức HTTP của Tracker và Peer.

Đường dẫn tới phần hiện thực các class diagram sẽ nằm ở [đây](#).

### 5.1 Thiết kế kiến trúc của hệ thống



Hình 6: System Structure Design



Để hiện thực được ứng dụng mạng này, trước hết chúng ta sẽ sử dụng thiết kế Unstructured Peer-to-Peer Systems, đây là hệ thống mạng ngang hàng không cấu trúc, đây là một kiến trúc cho phép các Peer trong hệ thống có thể chủ động trong việc kết nối với các Peer khác để tải xuống hay tải lên các Piece dữ liệu của các File Torrent, ngoài ra kiến trúc này còn cho phép các Peer trong mạng tham gia và rời đi một cách tự do.

Nhóm chúng em chọn hiện thực kiến trúc này bởi vì hệ thống P2P không có cấu trúc để triển khai hơn so với hệ thống có cấu trúc. Chúng không yêu cầu các thuật toán phức tạp để sắp xếp và định tuyến dữ liệu.

Trong hệ thống bao gồm ba thành phần chính:

### 1. Client

- Đây là đối tượng khởi tạo yêu cầu tải tập tin từ mạng ngang hàng.
- Client gửi thông tin định danh về tập tin cần tải (infohash) đến Tracker để nhận về file Torrent ứng với dữ liệu đó nếu cần và danh sách các Peers đang lưu trữ dữ liệu liên quan.
- Sau khi nhận danh sách các Peer, Client bắt đầu kết nối và yêu cầu dữ liệu từ các Peer này.

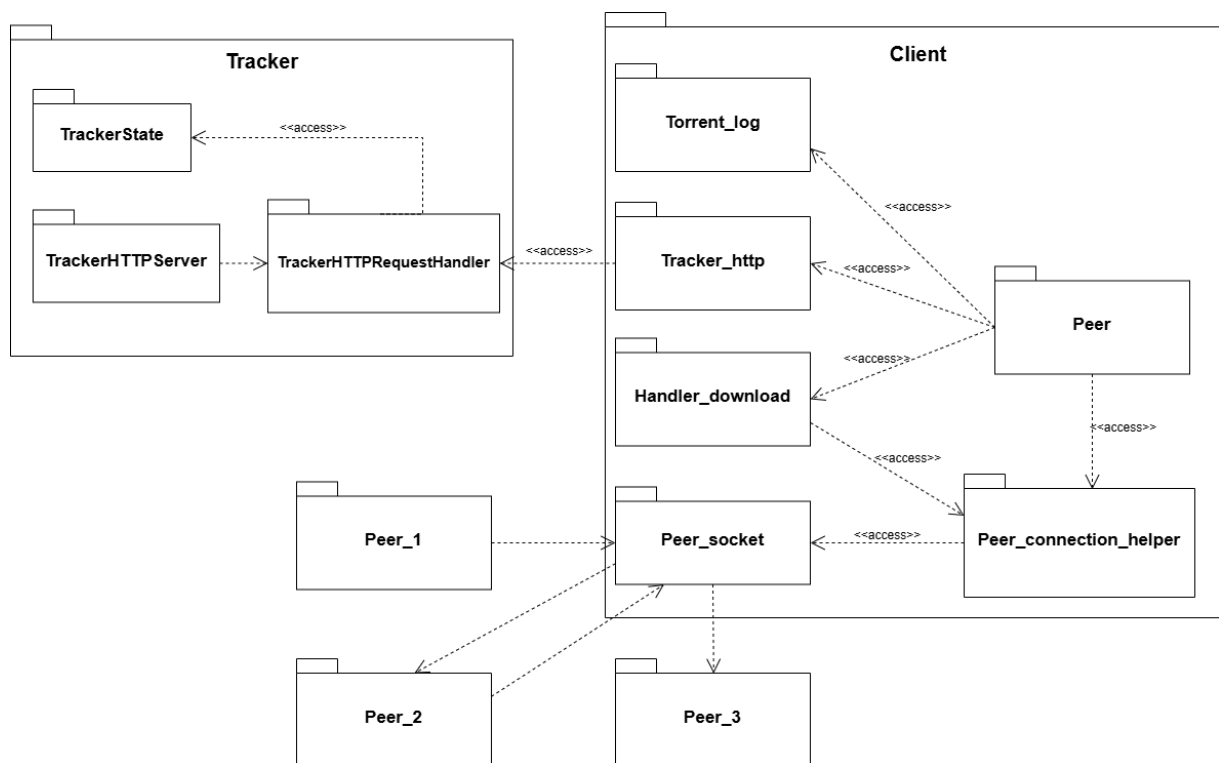
### 2. Tracker

- Đây là một máy chủ quản lý thông tin về các Peer trong mạng và danh sách các Peers sở hữu dữ liệu ứng với file Torrent nào, từ đó nếu nhận yêu cầu từ Client (thông tin về tập tin cần tải), Tracker trả về danh sách các Peer có chứa tập tin hoặc các phần của tập tin đó.
- Bình thường Tracker không lưu trữ dữ liệu của tập tin mà chỉ lưu thông tin về các Peer đang tham gia vào mạng. Tuy nhiên để giảm bớt các thành phần cần phải hiện thực, Tracker lúc này còn đóng vai trò là một Server lưu trữ các Torrent File đại diện cho các tập tin tương ứng, việc lưu trữ này sẽ giúp các Peer có được các Torrent File đại diện cho dữ liệu cần tải về bằng cách gửi Request tới Tracker.

### 3. Peer

- Là các thiết bị trong mạng P2P, đóng vai trò vừa là người tải (download) vừa là người chia sẻ (upload) dữ liệu.
- Peer lưu trữ các mảnh dữ liệu (pieces) của tập tin. Một Peer có thể không cần lưu toàn bộ tập tin, chỉ cần lưu các mảnh nhỏ.
- Các Peer trao đổi dữ liệu với nhau thông qua giao thức mạng ngang hàng mà không cần qua Tracker.

## 5.2 Package Diagram



Hình 7: Package Diagram

Package Diagram trong hình cung cấp một cái nhìn tổng quan hơn về từng thành phần cần hiện thực của hệ thống, bằng cách tổ chức các thành phần thành những package riêng biệt dựa trên chức năng của chúng.

### 1. Package Tracker:

Package này đại diện cho server Tracker, chịu trách nhiệm quản lý các Peer có trong mạng. Nó bao gồm:

- TrackerState: Quản lý trạng thái nội bộ của tracker, lưu trữ thông tin về các peer, bao gồm trạng thái (hoàn thành/chưa hoàn thành) và chúng đã có dữ liệu ứng với file Torrent nào.
- TrackerHTTPServer: Khởi tạo HTTP server và xử lý các yêu cầu đến từ client.
- TrackerHTTPRequestHandler: Xử lý các yêu cầu HTTP cụ thể, như đăng ký thông tin peer hoặc phản hồi danh sách các peer liên quan đến một file Torrent.

### 2. Package Client:

Package này bao gồm logic xử lý phía client trong mạng ngang hàng (peer-to-peer). Nó bao gồm:

- Torrent\_log: Ghi lại trạng thái và tiến trình tải xuống, bao gồm thông tin về các mảnh dữ liệu đã tải.
- Tracker\_http: Quản lý giao tiếp với tracker, gửi các yêu cầu để đăng ký peer hoặc lấy thông tin peer.
- andler\_download: Đảm nhiệm việc tải dữ liệu từ các peer khác.
- Peer: Đại diện cho một peer trong mạng và xử lý việc trao đổi dữ liệu với các peer khác.
- Peer\_connection\_helper: Cung cấp hỗ trợ cho việc mã hóa/giải mã thông điệp của peer và duy trì kết nối ổn định.
- Peer\_socket: Quản lý giao tiếp socket giữa các peer.

### 3. Tương tác giữa các Peer:

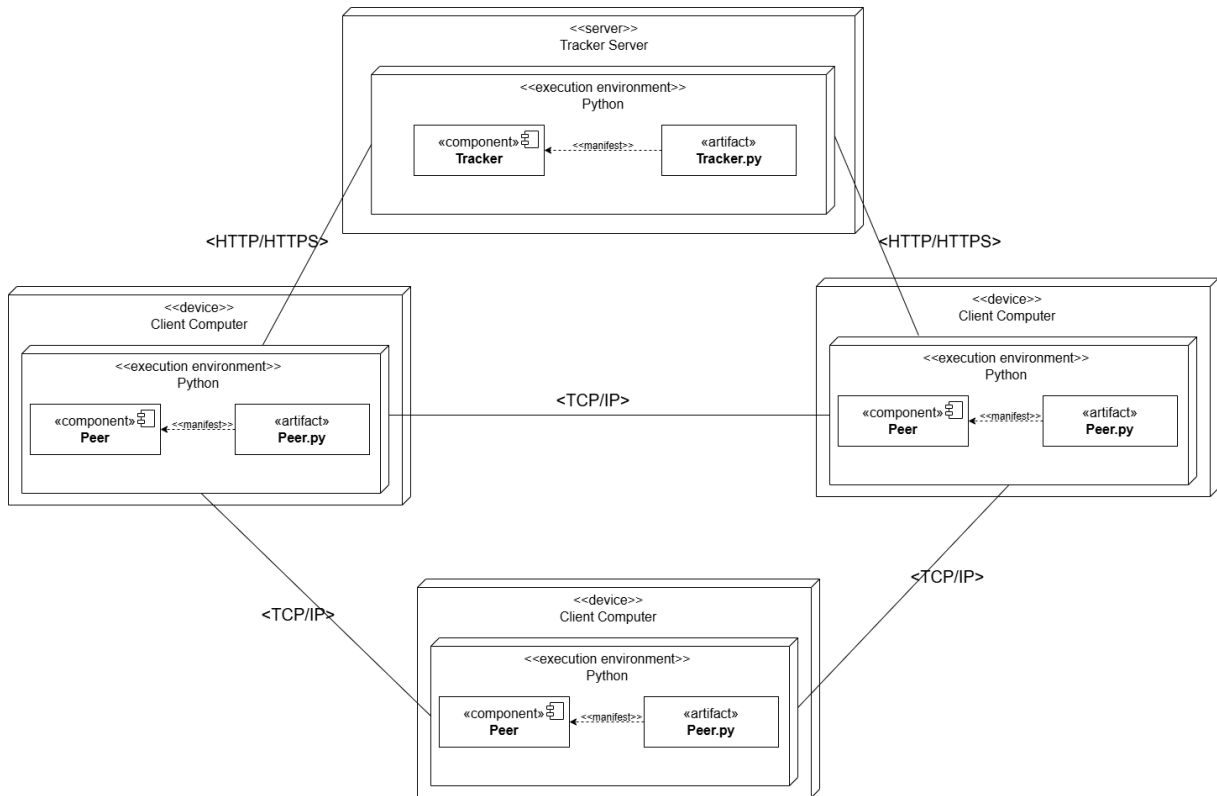
Các peer (Peer\_1, Peer\_2, Peer\_3) được hiển thị trong sơ đồ để biểu diễn các Peer có trong mạng, các Peer này sẽ trao đổi dữ liệu thông qua Peer\_socket, từ đó tạo khả năng chia sẻ dữ liệu giữa các Peer trong mạng.

### 4. Môi quan hệ và phụ thuộc:

Sơ đồ làm nổi bật rõ ràng các mối quan hệ phụ thuộc và truy cập giữa các package, ví dụ như việc:

- Package Client giao tiếp với Tracker thông qua module Tracker\_http, cho phép thông báo với Tracker sự tham gia của Peer tương ứng, cung cấp thông tin về việc nắm giữ dữ liệu các Torrent tương ứng hay lấy danh sách các Peers cần thiết.
- Trong Package Client, các thành phần liên kết chặt chẽ với nhau để cung cấp các chức năng cần thiết cho ứng dụng.

### 5.3 Deployment Diagram



Hình 8: Deployment Diagram

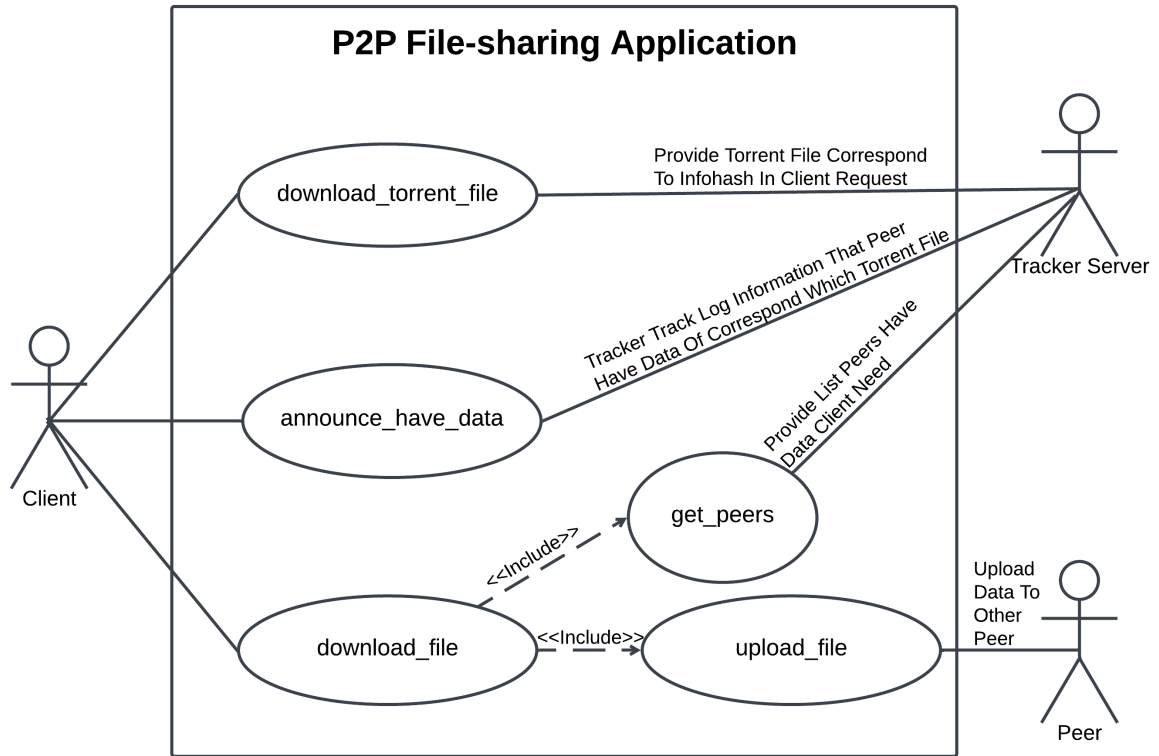
Sơ đồ Deployment Diagram trên biểu diễn cách triển khai hệ thống mạng ngang hàng (P2P) với hai thành phần chính là Tracker Server và Peer. Tracker Server đóng vai trò trung tâm để quản lý thông tin và hỗ trợ giao tiếp giữa các Peer trong mạng.

Giao tiếp giữa Tracker Server và các Peer được thực hiện qua giao thức HTTP/HTTPS. Các Peer gửi yêu cầu đến Tracker để nhận danh sách các Peer khác hoặc thông báo trạng thái của mình. Sau đó, các Peer giao tiếp trực tiếp với nhau qua giao thức TCP/IP để trao đổi dữ liệu trong mạng.

## 6 Mô hình hóa các tương tác trong hệ thống

Đường dẫn tới phần hiện thực các Diagram sẽ nằm ở [đây](#).

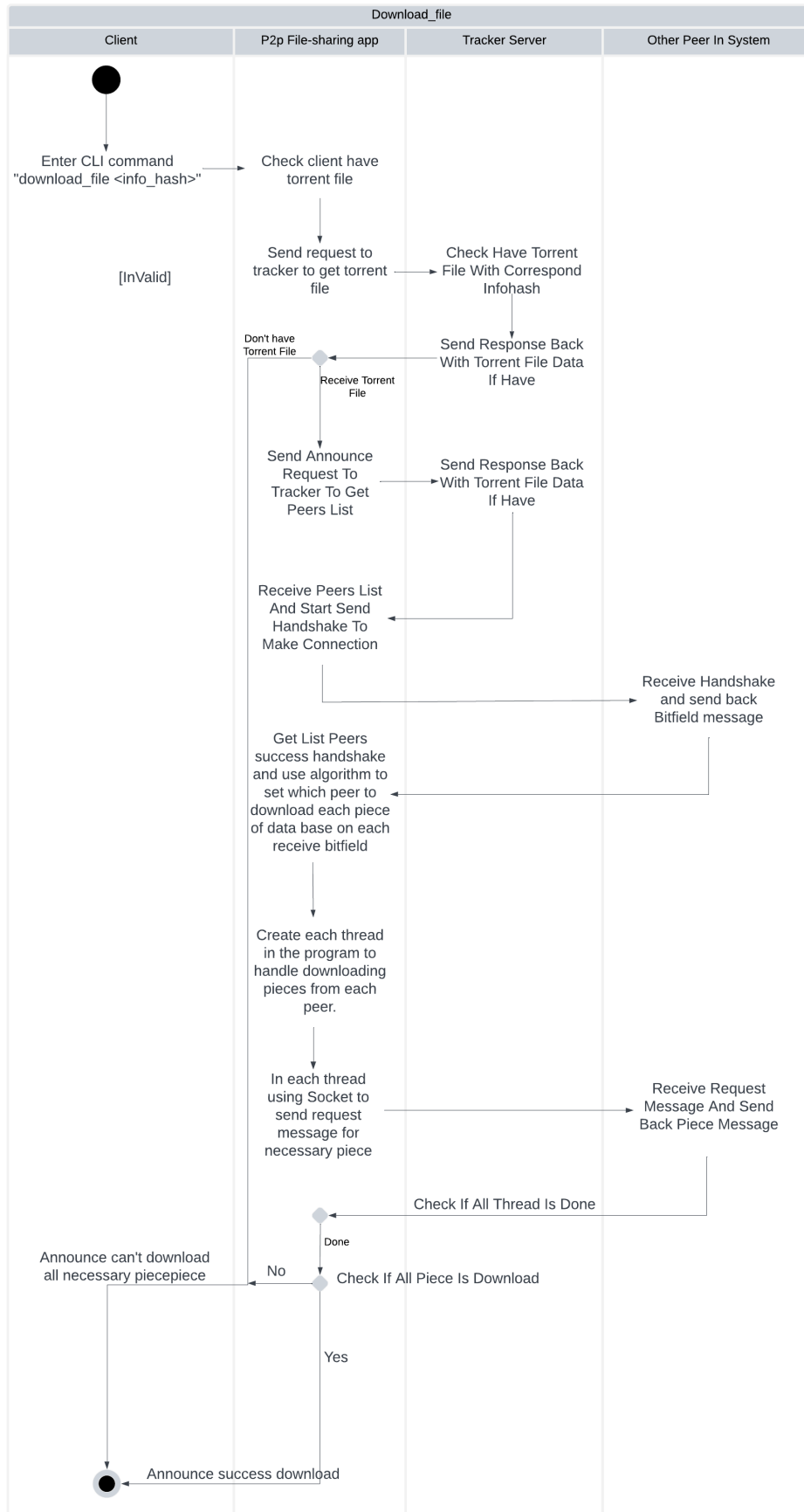
### 6.1 Usecase Diagram



Hình 9: Usecase Diagram

Ứng bao gồm các Use Case chính: `download_torrent_file` cho phép Client tải tệp torrent từ Tracker Server, `announce_have_data` để thông báo cho Tracker Server về dữ liệu mà Peer sở hữu, và `get_peers` giúp Tracker Server cung cấp danh sách các Peer có dữ liệu. Use Case `download_file` cho phép Client tải dữ liệu từ các Peer, trong khi `upload_file` cho phép Peer chia sẻ dữ liệu với các Peer khác.

### 6.2 Activity Diagram



**Hình 10:** Activity Diagram

Hình trên là Activity Diagram của chức năng **download\_file** trong ứng dụng của nhóm. Các Actor chính tham gia bao gồm **Client**, **P2P File-sharing App**, **Tracker Server**, và **Other Peer in System**. Dưới đây là mô tả rõ hơn

về luồng hoạt động của chức năng trên:

1. **Sử dụng lệnh CLI để bắt đầu tải dữ liệu xuống:** Người dùng nhập lệnh từ giao diện dòng lệnh (CLI) với cú pháp `download_file <info_hash>`. Ứng dụng kiểm tra xem **Client** đã có tệp torrent tương ứng với `info_hash` hay chưa. Nếu chưa có, Client gửi yêu cầu tới **Tracker Server** để tải tệp torrent.
2. **Nhận tệp torrent:** Tracker Server kiểm tra và gửi tệp torrent tương ứng với `info_hash` nếu tồn tại. Sau khi nhận được tệp torrent, Client tiếp tục gửi yêu cầu "announce" đến Tracker Server để nhận danh sách các Peer có dữ liệu cần thiết.
3. **Nhận danh sách Peer:** Tracker Server phản hồi danh sách các Peer có dữ liệu liên quan. Client nhận danh sách này và tiến hành thực hiện kết nối "handshake" để thiết lập liên lạc với các Peer.
4. **Tải dữ liệu từ Peer:** Sau khi thiết lập kết nối thành công, Client:
  - Sử dụng thuật toán để phân công nhiệm vụ tải các mảnh dữ liệu (pieces) từ các Peer khác nhau.
  - Tạo các luồng xử lý song song (threads) để tải dữ liệu từ nhiều Peer đồng thời.

Trong mỗi luồng, Client gửi yêu cầu thông qua giao thức Socket đến các Peer để nhận các mảnh dữ liệu cần thiết. Peer phản hồi và gửi dữ liệu về Client.

5. **Kiểm tra và hoàn tất:** Sau khi tất cả các luồng tải dữ liệu hoàn tất, Client kiểm tra xem toàn bộ các mảnh dữ liệu đã được tải đầy đủ chưa:
  - Nếu tất cả các mảnh dữ liệu đã được tải, Client thông báo quá trình tải hoàn tất thành công.
  - Nếu còn thiếu dữ liệu, Client thông báo lỗi về việc không thể tải đầy đủ.

## 7 Kết quả hiện thực:

Trong phần này, các Class Diagram sẽ được cung cấp để cho ra một cái nhìn chi tiết hơn về các thành phần được hiện thực trong hệ thống như từng thuộc tính, hay phương thức của các đối tượng được hiện thực,... Đường dẫn tới phần hiện thực các Diagram sẽ nằm ở [đây](#).

### 7.1 Tracker

Tracker được thiết kế theo mô hình hướng đối tượng với mục tiêu quản lý trạng thái của các torrent và hỗ trợ giao tiếp giữa các peer. Khi khởi chạy, TrackerHTTPServer đảm nhận việc khởi tạo server, sử dụng địa chỉ IP cục bộ, cổng giao tiếp và tracker\_id. Đồng thời, một instance của Class Tracker được tạo ra để quản lý trạng thái của các torrent thông qua biến tracker\_state. Server bắt đầu lắng nghe các yêu cầu HTTP từ peer thông qua Class xử lý TrackerHTTPRequestHandler.

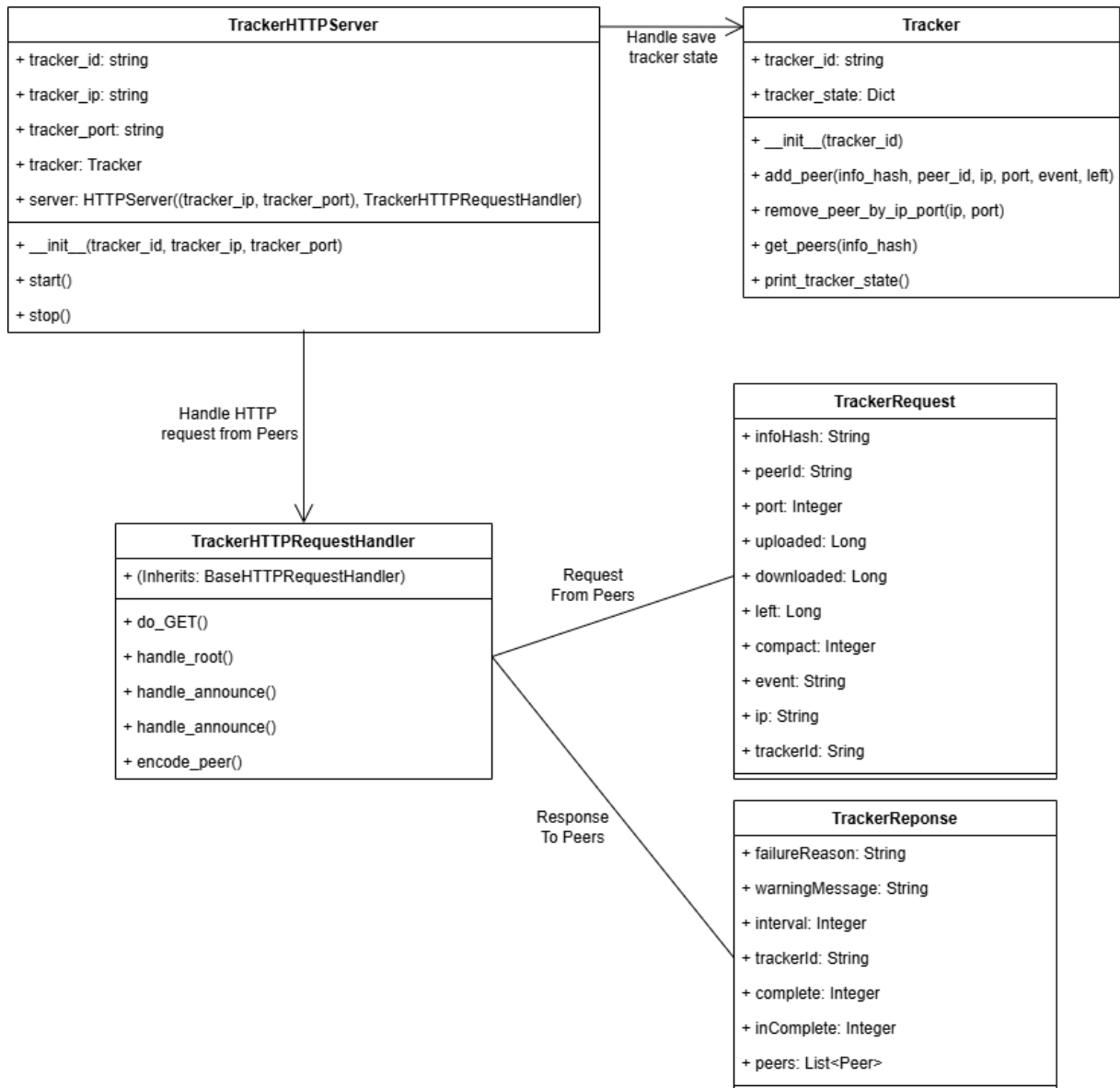
Khi nhận được yêu cầu từ peer qua endpoint /announce, Class TrackerHTTPRequestHandler trích xuất các tham số từ URL như `info_hash` (đại diện cho tệp torrent), `peer_id`, `port`, và event (loại sự kiện: started, completed, stopped). Tùy thuộc vào sự kiện, Class Tracker sẽ thêm peer mới, cập nhật trạng thái (chuyển từ incomplete sang complete nếu đã tải xong), hoặc xóa peer khỏi danh sách khi sự kiện stopped xảy ra.

Sau khi xử lý yêu cầu, Tracker trả về danh sách các peer đang hoạt động cùng thông tin về số lượng seeders (trong complete) và leechers (trong incomplete). Dữ liệu phản hồi được mã hóa bằng bencode để tuân thủ giao thức BitTorrent, đảm bảo tính tương thích và hiệu quả trong giao tiếp.

Ngoài ra dữ liệu liên quan tới danh sách các Peers ứng với từng Info\_hash sẽ được lưu trong tracker\_state và được tổ chức dưới dạng dictionary, trong đó mỗi info\_hash đại diện cho một torrent. Dữ liệu bao gồm ba danh sách: complete (các seeders đã tải xong), incomplete (các leechers đang tải), và peers (tổng hợp các peer tham gia). Mỗi peer được lưu với thông tin peer\_id, địa chỉ IP, cổng kết nối, và trạng thái tải (left). Cách tổ chức này đảm bảo việc theo dõi và cập nhật trạng thái của các peer được thực hiện hiệu quả và chính xác.

```
...
tracker_state = {
    "info_hash": {
        "complete": [list of complete peers],
        "incomplete": [list of incomplete peers],
        "peers": [list of all peers]
    }
}
...
```

Hình 11: Cấu trúc Tracker\_state đơn giản



Hình 12: Class Diagram của Tracker

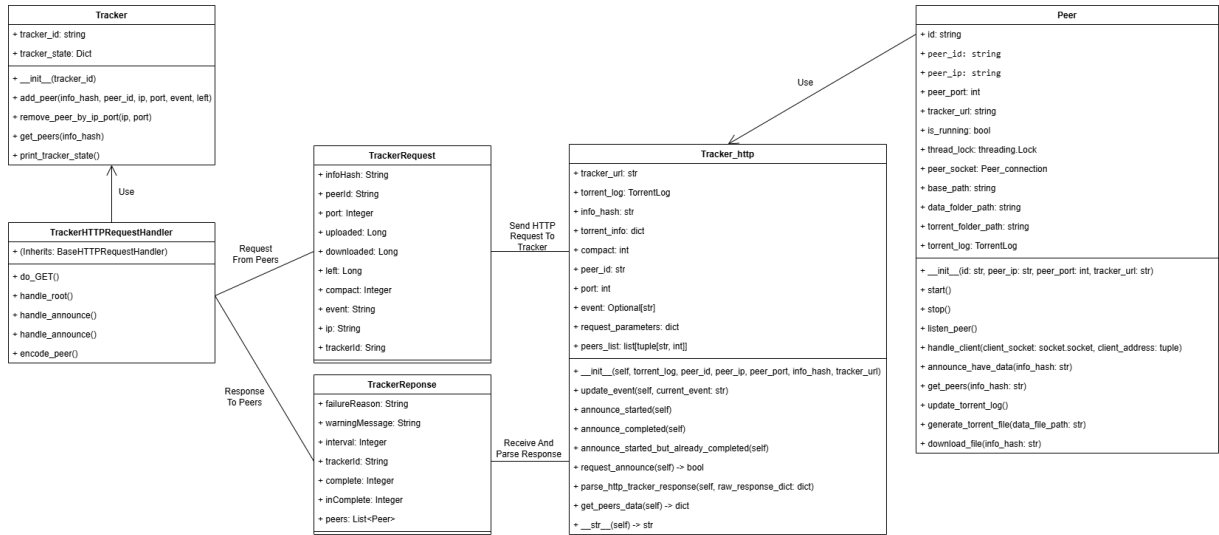
Luồng hoạt động của Tracker bắt đầu từ việc khởi chạy server, lắng nghe yêu cầu từ peer, cập nhật trạng thái trong tracker\_state, và kết thúc bằng việc trả phản hồi. Kiến trúc này đảm bảo Tracker hoạt động ổn định, đáp ứng nhanh chóng các yêu cầu, và duy trì sự đồng bộ trong mạng chia sẻ ngang hàng (P2P).

## 7.2 Giao tiếp giữa các Peers và Tracker thông qua HTTP

Kết nối giữa Peer và Tracker được thực hiện thông qua Tracker\_http, đây là một Class cung cấp các hàm hỗ trợ việc sử dụng giao thức HTTP để gửi các Request và xử lý các Response nhận được ở mỗi Peer. Peer gửi yêu cầu HTTP GET đến Tracker thông qua endpoint /announce, bao gồm các thông tin như info\_hash, peer\_id, port, tiến trình tải tệp (uploaded, downloaded, left), và sự kiện (started, completed, stopped). Class Tracker\_http đảm bảo việc quản lý việc tạo và gửi yêu cầu này, đảm bảo trạng thái của Peer được cập nhật trên ở tracker\_state trong Tracker.

Tracker phản hồi yêu cầu bằng dữ liệu chứa các thông tin quan trọng như interval (thời gian tối thiểu giữa các yêu cầu), số lượng complete (seeders), incomplete (leechers), và danh sách các Peer khác (peers). Class Tracker\_http phân tích phản hồi này, cập nhật danh sách các Peer và chia sẻ thông tin với các thành phần khác của Peer để thiết lập kết nối P2P.

Class Tracker\_http đóng vai trò quan trọng trong việc duy trì sự đồng bộ giữa Peer và Tracker. Nó quản lý việc gửi thông báo trạng thái của Peer, như bắt đầu tải (started), hoàn tất tải xuống (completed), hoặc dừng hoạt động (stopped). Đồng thời, Class này lưu trữ danh sách Peer trả về từ Tracker, hỗ trợ Peer thiết lập kết nối trực tiếp với các thành viên khác trong mạng lưới P2P.

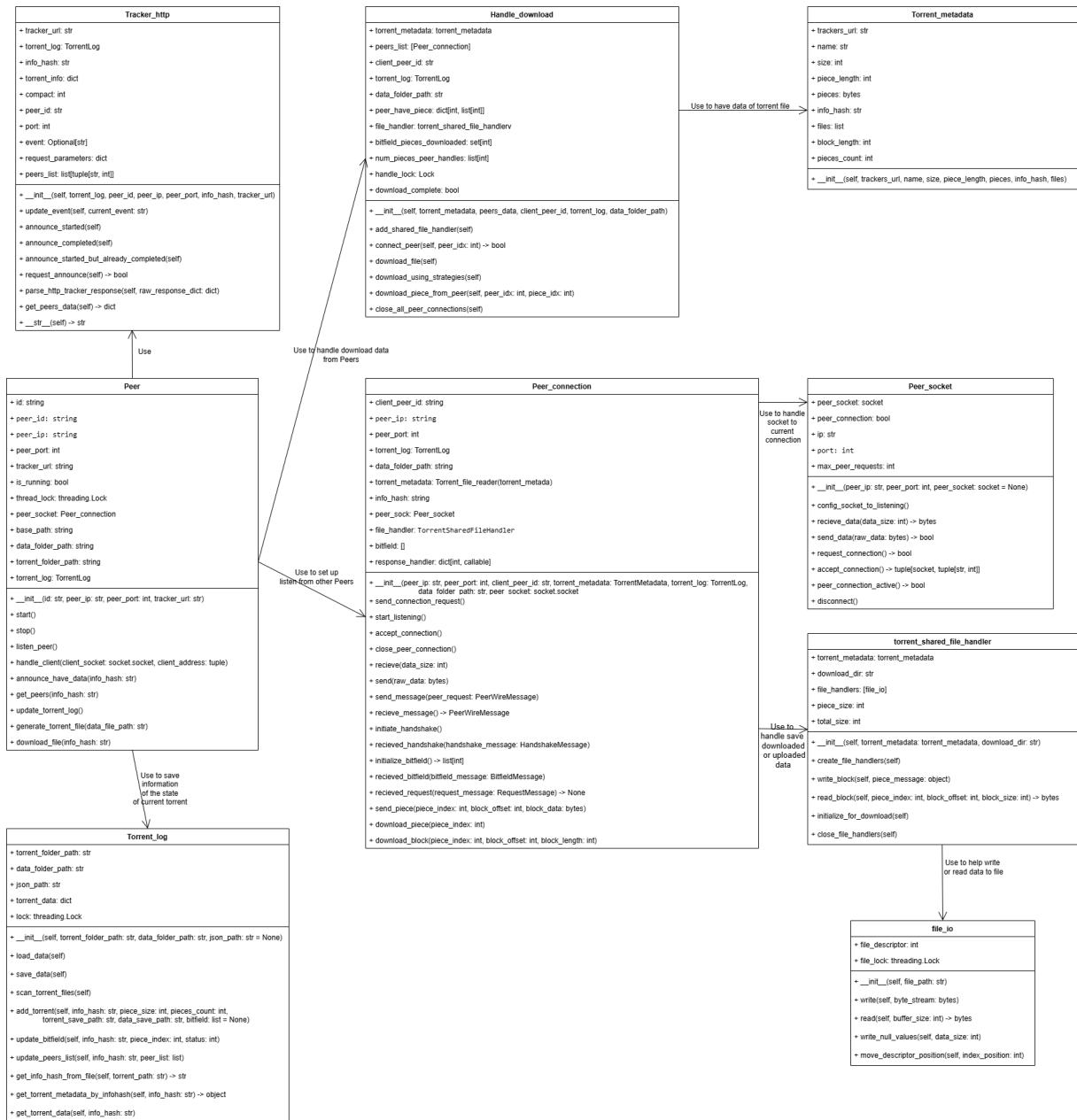


Hình 13: Giao tiếp giữa Tracker và Peers thông qua HTTP

Giao thức HTTP được sử dụng trong mô hình này đảm bảo sự đơn giản hóa việc trao đổi dữ liệu giữa Peer và Tracker. Tracker chỉ đóng vai trò điều phối và quản lý danh sách kết nối, không trực tiếp tham gia vào quá trình truyền dữ liệu, giúp tối ưu hóa hoạt động mạng lưới phi tập trung.

### 7.3 Peer





Hình 14: Kiến trúc Peer hiện thực

Peer là một thành phần chính trong hệ thống chia sẻ tệp ngang hàng (Peer-to-Peer - P2P), đóng vai trò như một thực thể độc lập, có khả năng giao tiếp với các Peers khác và Tracker để tham gia vào quá trình tải xuống hoặc chia sẻ dữ liệu. Thành phần này được thiết kế để hoạt động hiệu quả trong hệ thống chia sẻ tệp phân tán như giao thức BitTorrent, cho phép tối ưu hóa tài nguyên mạng và cải thiện tốc độ tải dữ liệu.

Trong thiết kế Class Diagram dưới đây, Peer sẽ được chạy và sử dụng những đối tượng Class liên quan để đáp ứng được các mục tiêu cơ bản của mình. Các đối tượng Class trên được xây dựng và thiết kế với mục tiêu quan trọng nhất là đảm bảo sự phân luồng rõ ràng, các class nhất định sẽ có những nhiệm vụ nhất định, để từ đó việc hiện thực và đưa vào sử dụng sẽ được dễ dàng và trơn tru hơn.

Trước hết như đã nói, các Class trên đều được hiện thực đảm bảo các mục tiêu cơ bản của một Peer nên chúng em sẽ đi qua từng mục tiêu từ đó giới thiệu về những Class và ứng dụng của nó mà có liên quan tới mục tiêu đó.

### 1. Kết nối với Tracker

Để một Peer có thể tham gia vào mạng thì điều tối thiểu là nó phải kết nối được tới Tracker để thông báo về tình trạng cũng như có khả năng thông báo rằng nó đang có dữ liệu cho những Torrent nào. Thì để đáp ứng được điều này, trong kiến trúc Peer của chúng em, đối tượng **Tracker\_http** sẽ được thiết kế để xử lý những điều này. Nó sẽ chịu trách nhiệm cho việc nhận vào các thông tin đầu vào cần thiết như Peer\_id, port, info\_hash, tracker\_url để tự động tạo các Request theo cấu trúc quy định và gửi tới Tracker, ngoài ra

đôi tượng này còn chịu trách nhiệm cho việc phân tích Response nhận lại thành các dữ liệu cần thiết tương ứng.

Vì thiết lập kết nối với một DB sẽ tạo nên một khối lượng công việc lớn, nên để đơn giản, nhóm chúng em sẽ lưu các thông tin cần thiết liên quan tới trạng thái dữ liệu hiện tại của các file Torrent tại một file Json, trong đó sẽ chứa các thông tin cơ bản theo định dạng:

- **Key:** `info_hash` – Đây là một chuỗi hash duy nhất (SHA-1) đại diện cho tệp torrent. Nó đảm bảo mỗi tệp torrent được nhận diện duy nhất trong hệ thống.
- **Value:** Một đối tượng chứa thông tin chi tiết về tệp torrent và trạng thái liên quan như:
  - `piece_size` (int) - kích thước của mỗi Piece dữ liệu (piece) trong tệp torrent, tính bằng byte. Đây là đơn vị cơ bản của quá trình tải xuống.
  - `piece_count` (int) - số lượng mảnh (pieces) mà tệp torrent được chia nhỏ. Giá trị này được tính dựa trên tổng kích thước tệp và kích thước mỗi mảnh.
  - `torrent_save_path` (string) - đường dẫn lưu trữ tệp .torrent trên hệ thống.
  - `data_save_path` (string) - đường dẫn nơi lưu trữ dữ liệu đã tải xuống.
  - `bitfield` (list[int]) - danh sách trạng thái các mảnh dữ liệu, phần tử có giá trị 0 (chưa tải) hoặc 1 (đã tải).
  - `list_peers` (list) - danh sách các peer liên kết với tệp torrent này, chứa thông tin về địa chỉ IP và cổng của các peer,...

[illegible]

**Hình 15:** File Json mẫu để lưu thông tin của Peer

Và để tạo sự tiện lợi trong việc ghi đọc dữ liệu vào file Json này thì đối tượng **TorrentLog** duy nhất được khởi tạo khi Peer chạy sẽ lo xử lý các vấn đề liên quan tới việc này.

## 2. Kết nối với các Peers khác

Việc tạo kết nối với các Peers khác là một yêu cầu rất quan trọng, vì vậy để việc xử lý kết nối với từng Peer khác nhau một cách gọn gàng nhất, nhóm sẽ hiện thực đối tượng **Peer\_connection** đây sẽ là đối tượng cung cấp các hàm chức năng như tạo kết nối, gửi message và xử lý yêu cầu qua lại của các Peer. Thì sau đây là một số chức năng nổi bật của Peer\_connection trong việc xử lý kết nối giữa các Peers trong mạng với Peer đang chạy.

- Thiết lập và quản lý kết nối TCP với Peer khác  
Đối tượng `Peer_connection` đảm nhiệm việc thiết lập kết nối TCP với Peer ứng với thông tin lúc nó được khởi tạo. Thông qua phương thức `send_connection_request`, nó gửi yêu cầu kết nối tới một Peer cụ thể, đồng thời đảm bảo kiểm tra trạng thái kết nối để xử lý các trường hợp kết nối thất bại. Ngoài ra, với chức năng `start_listening` và `accept_connection`, đối tượng này còn có khả năng chờ và chấp nhận các kết nối từ các Peer khác trong mạng, đóng vai trò như một nút giao tiếp hai chiều.
- Thực hiện quy trình bắt tay (handshake) với các Peer  
Quy trình bắt tay là bước đầu tiên và quan trọng nhất trong giao thức BitTorrent để xác thực kết nối giữa các Peer. Đối tượng `Peer_connection` thực hiện việc gửi thông điệp bắt tay thông qua phương thức `initiate_handshake`. Sau đó, nó kiểm tra và xác thực thông điệp phản hồi từ Peer nhận được. Nếu

quá trình bắt tay thành công, cờ handshake\_flag được bật để đánh dấu rằng kết nối đã sẵn sàng cho các hoạt động trao đổi dữ liệu.

- Trao đổi thông điệp giao thức Peer Wire  
Đối tượng Peer\_connection chịu trách nhiệm gửi và nhận thông điệp giao thức Peer Wire – giao thức cốt lõi trong mạng BitTorrent. Thông qua các phương thức như send\_message và receive\_message, nó gửi các thông điệp (như request, bitfield, piece) tới các Peer khác và xử lý thông điệp phản hồi. Tính năng này đảm bảo việc đồng bộ hóa trạng thái giữa các Peer trong mạng, từ đó cải thiện hiệu suất tải và chia sẻ dữ liệu.
- Gửi thông tin Bitfield  
Một phần quan trọng trong kết nối với các Peer là trao đổi và cập nhật trạng thái của các mảnh dữ liệu (pieces). Peer\_connection đảm bảo rằng Bitfield – biểu thị trạng thái các Piece dữ liệu mà Peer đang có – được nhận diện chính xác thông qua phương thức initialize\_bitfield. Sau khi nhận thông tin, đối tượng này sẽ được tự động kiểm tra và cập nhật để sử dụng sau này.
- Tải xuống dữ liệu từ Peer khác  
Peer\_connection thực hiện việc tải xuống dữ liệu từ Peer khác thông qua các phương thức như download\_piece và download\_block. Trong quá trình này, đối tượng đảm bảo rằng dữ liệu nhận được đúng về kích thước, vị trí, và giá trị băm (hash) trước khi ghi vào tệp cục bộ. Điều này giúp đảm bảo tính toàn vẹn của dữ liệu tải xuống, giảm thiểu lỗi xảy ra trong quá trình chia sẻ tệp.
- Chia sẻ dữ liệu với các Peer khác  
Không chỉ tải xuống, Peer\_connection còn hỗ trợ chia sẻ dữ liệu mà nó có với các Peer khác. Các phương thức như send\_piece hoặc send\_have được sử dụng để gửi thông tin hoặc dữ liệu Piece mà Peer này sở hữu tới Peer khác khi được yêu cầu, góp phần vào việc duy trì chức năng của mạng P2P.

Nói chung đối tượng Peer\_connection đóng vai trò như một cầu nối quan trọng giữa các Peer trong mạng BitTorrent. Nó không chỉ đảm nhiệm việc thiết lập kết nối, mà còn quản lý các hoạt động trao đổi dữ liệu và đảm bảo tính toàn vẹn, đồng bộ trong quá trình chia sẻ tệp.

### 3. Tải xuống dữ liệu

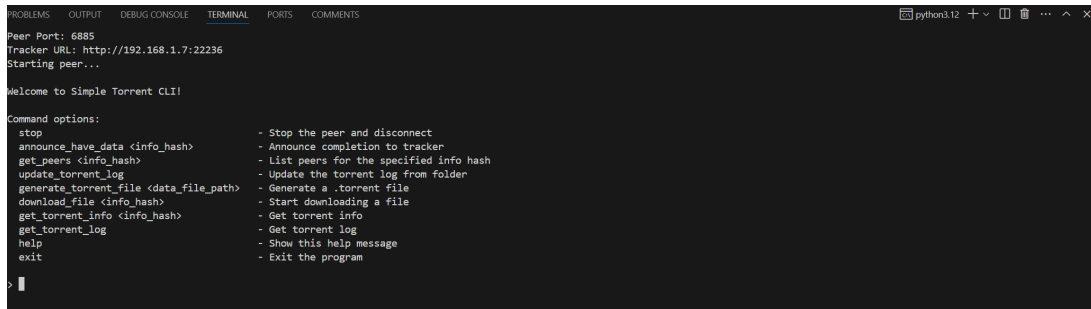
Đây là một chức năng quan trọng, đòi hỏi phải có một chiến lược và quản lý rõ ràng để tải xuống các phần dữ liệu từ nhiều Peer khác nhau. Vì vậy, đối tượng **Handle\_download** sẽ được thiết kế và hiện thực để đảm bảo tối thiểu những yêu cầu của ứng dụng.

- Khởi tạo và quản lý thông tin tải xuống  
Handle\_download được khởi tạo với thông tin từ tệp torrent (torrent\_metadata), danh sách các Peer (peers\_data), và các tài nguyên hỗ trợ khác như log torrent (torrent\_log) và đường dẫn lưu trữ dữ liệu (data\_folder\_path). Đối tượng này chịu trách nhiệm tạo danh sách các kết nối Peer (peers\_list) và cấu trúc dữ liệu để theo dõi trạng thái tải xuống, bao gồm danh sách các Peer có các Piece dữ liệu cần thiết (peer\_have\_piece) và trạng thái các Piece đã tải xuống (bitfield\_pieces\_downloaded).
- Quản lý kết nối tới các Peer  
Handle\_download sử dụng phương thức connect\_peer để thiết lập kết nối với từng Peer. Phương thức này thực hiện quy trình bắt tay (handshake) và nhận bitfield từ các Peer để xác định Piece dữ liệu mà Peer đó có. Thông qua việc cập nhật danh sách peer\_have\_piece, đối tượng đảm bảo rằng trạng thái các Piece dữ liệu sẵn có trong mạng được duy trì chính xác.
- Áp dụng chiến lược tải xuống đúng đắn  
Handle\_download sử dụng phương thức download\_using\_strategies để tối ưu hóa quá trình tải xuống. Thay vì tải dữ liệu một cách ngẫu nhiên, đối tượng này ưu tiên chọn Peer đang phải upload ít Piece dữ liệu hơn, nhằm đảm bảo rằng các Peer trong mạng được phân phối tải một cách đồng đều, từ đó tăng hiệu quả tải xuống.  
Ngoài ra, chiến lược tải xuống phải đảm bảo được các yêu cầu của đề là không gửi yêu cầu tải xuống một Piece mà đã được tải về và không gửi yêu cầu tải xuống một Piece dữ liệu mà Peer tương ứng không có.
- Quản lý đa luồng cho việc tải dữ liệu  
Với mỗi Peer được kết nối để tải dữ liệu, Handle\_download tạo một luồng riêng biệt để thực hiện việc tải các Piece cần thiết từ Peer đó. Điều này cho phép nhiều Piece dữ liệu được tải xuống đồng thời, tăng tốc độ tải và tận dụng tối đa tài nguyên mạng.

Nói chung, Handle\_download đóng vai trò trung tâm trong việc quản lý và thực thi quá trình tải xuống dữ liệu từ mạng BitTorrent. Nó không chỉ chịu trách nhiệm kết nối và giao tiếp với các Peer khác, mà còn tối ưu hóa và điều phối việc tải dữ liệu để đảm bảo tốc độ, độ tin cậy, và hiệu quả cao nhất. Nhờ có đối tượng này, toàn bộ quá trình tải xuống trong mạng P2P diễn ra một cách trơn tru, đáp ứng các yêu cầu phức tạp của giao thức BitTorrent.

#### 4. Cung cấp CLI đơn giản, hiểu quả cho người dùng

Cung cấp các lệnh rõ ràng, cụ thể, giúp người dùng thực hiện các chức năng quan trọng như bắt đầu và dừng Peer, quản lý file torrent, tải file từ mạng, và thông báo trạng thái với Tracker. Các lệnh được tổ chức gọn gàng với cú pháp dễ hiểu, ví dụ: stop để dừng Peer, download\_file <info\_hash> để tải file dựa trên mã hash, hoặc get\_peers <info\_hash> để liệt kê danh sách Peer có sẵn.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
Peer Port: 6885
Tracker URL: http://192.168.1.7:22236
Starting peer...

Welcome to Simple Torrent CLI!

Command options:
stop                                - Stop the peer and disconnect
announce_have_data <info_hash>    - Announce completion to tracker
get_peers <info_hash>              - List peers for the specified info hash
update_torrent_log                 - Update the torrent log from folder
generate_torrent_file <data_file_path> - Generate a .torrent file
download_file <info_hash>         - Start downloading a file
get_torrent_info <info_hash>      - Get torrent info
get_torrent_log                    - Get torrent log
help                               - Show this help message
exit                               - Exit the program

> |
```

Hình 16: CLI đơn giản của ứng dụng

## 8 Đường dẫn tới kết quả hiện thực và Demo ứng dụng:

Sau đây là các đường dẫn tới từng kết quả hiện thực ứng dụng:

- **Các Diagram mô tả kiến trúc hệ thống hay các Class Diagram**  
<https://drive.google.com/file/d/1eCU3qB3yYAlcBpjPPsRoevOMNw6BzA1l/view?usp=sharing>
- **Usecase Diagram hay Activity Diagram:**  
[https://lucid.app/lucidchart/96a93da6-c995-4f48-a6e4-827dfd6b920a/edit?viewport\\_loc=-1898%2C-1364%2C6381%2C3401%2C0\\_0&invitationId=inv\\_7a6a9ebf-271d-4099-b03c-be08d298b08d](https://lucid.app/lucidchart/96a93da6-c995-4f48-a6e4-827dfd6b920a/edit?viewport_loc=-1898%2C-1364%2C6381%2C3401%2C0_0&invitationId=inv_7a6a9ebf-271d-4099-b03c-be08d298b08d)
- **Repository Github chứa mã nguồn hệ thống:**  
[https://github.com/VietTranDai/241\\_CN\\_Assignment\\_1.git](https://github.com/VietTranDai/241_CN_Assignment_1.git)
- **Video Demo hoạt động của ứng dụng:**  
[https://youtu.be/GrsQM\\_Sw0YQ](https://youtu.be/GrsQM_Sw0YQ)