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

HỌC KỲ

Bài tập lớn

Computer Network

GVHD: Hoàng Lê Hải Thanh
SV:

Huỳnh Minh Khoa	- 2252346
Thân Nguyễn Minh Khoa	- 2252361
Trần Lương Yến Nhi	- 2252586
Nguyễn Lê Vân Tú	- 2252881

TP. HỒ CHÍ MINH, THÁNG 10/2024

Mục lục

1 Chức năng của ứng dụng

1.1 Chức năng ứng dụng chia sẻ tệp BitTorrent và giao thức giao tiếp

File Discovery và chia sẻ Metadata *Mô tả:* Cho phép người dùng định vị và truy xuất siêu dữ liệu về các tệp được chia sẻ hoặc tải xuống, thường thông qua các tệp .torrent hoặc magnet link.

Peer Discovery *Mô tả:* Cho phép nhận dạng và kết nối với các peer khác chia sẻ cùng một tệp, đảm bảo chia sẻ phân tán.

Phân phối tập tin *Mô tả:* Chia tệp thành các mảnh nhỏ hơn để chia sẻ hiệu quả. Mỗi mảnh được chia sẻ và xác minh riêng lẻ để đảm bảo tính toàn vẹn của dữ liệu.

Tải lên *Mô tả:* Xử lý việc gửi các phần tệp từ thiết bị của người dùng đến nhiều peer trong mạng. Đảm bảo cân bằng sự đóng góp và chia sẻ tài nguyên.

Tải xuống *Mô tả:* Quản lý việc nhận các mảnh tệp từ các peer khác. Đảm bảo sử dụng tối ưu tài nguyên mạng bằng cách tải xuống từ nhiều peer đồng thời.

Xác minh mảnh *Mô tả:* Xác minh tính toàn vẹn của từng mảnh được tải xuống bằng cách so sánh giá trị băm của nó với giá trị mong đợi, đảm bảo độ tin cậy của quá trình tải xuống.

Chức năng Tit-for-Tat *Mô tả:* Triển khai chiến lược để đảm bảo chia sẻ công bằng bằng cách ưu tiên các peer đóng góp nhiều dữ liệu hơn. Khuyến khích sự tương hỗ giữa các peer để cân bằng tải mạng.

Chức năng mã hóa *Mô tả:* Đảm bảo truyền dữ liệu an toàn bằng cách mã hóa thông tin liên lạc giữa các peer. Sử dụng cơ chế trao đổi khóa để thiết lập kết nối an toàn.

Xử lý lỗi và phục hồi *Mô tả:* Phát hiện và phục hồi các sự cố như tải xuống không đầy đủ hoặc bị hỏng. Thử lại các lần tải xuống không thành công và yêu cầu lại các phần bị thiếu từ các peer khác.

1.2 Chức năng dành riêng cho Tracker

Đăng kí Tracker *Mô tả:* Đăng ký những peer mới với trình theo dõi để cho phép tham gia vào mạng chia sẻ tệp.

Thông báo Tracker *Mô tả:* Cập nhật trình theo dõi với trạng thái của peer, chẳng hạn như tiến trình tải lên/tải xuống, các tệp mà máy khách

đang seeding và trạng thái kết nối.

Truy xuất danh sách Tracker Peer *Mô tả:* Cho phép các peer lấy danh sách các peer khác chia sẻ cùng một tệp từ trình theo dõi.

2 Giao thức

2.1 Giao thức Uploader - Downloader

2.1.1 Downloader và Uploader

Lần lượt chờ đợi tin nhắn của nhau ứng theo các trường cần lưu

B1: Downloader kết nối với Uploader:

- Downloader: Client gửi kết nối
- Uploader: Gửi lại message: ““Got connection from” + client_addr” để xác nhận.

B2: Downloader sau khi nhận tin xác nhận kết nối sẽ bắt đầu tải file từ Uploader:

- Downloader: Gửi message: reponame (với reponame là torrent hash)
- Uploader: Từ reponame truy suất ra một mảng mà các phần tử là giá trị nhị phân tượng trưng cho các Piece mà Uploader đã tải xuống

B3: Downloader chọn chunk muốn được tải từ Uploader: Uploader sẽ nằm trong vòng lặp chờ nhận các integer cho đến khi ngắt kết nối:

- B3.1: Downloader: Sẽ gửi 1 interger ứng với vị trí Piece cần tải
- B3.2: Uploader: Sau khi nhận sẽ bắt đầu gửi dữ liệu từ Piece với độ dài Piece theo Piece_length được lưu trong file Torrent cho Downloader
- B3.3: Downloader: Sẽ nhận và xử lý dữ liệu và lập lại bước 3.1 cho đến khi hết Piece để download từ Uploader này hoặc đã tải hết tất cả các Piece

2.1.2 Lý do sử dụng socket

- Socket cho phép kết nối trực tiếp giữa các thiết bị, giúp truyền dữ liệu nhanh chóng và hiệu quả.
- Socket cung cấp tốc độ và hiệu quả cao nhờ giao thức TCP đảm bảo độ tin cậy và đúng thứ tự, cũng như có thể kiểm soát chặt chẽ việc truyền tải và linh hoạt chọn giữa các giao thức như TCP hoặc UDP tùy theo nhu cầu.
- Socket cho phép truyền tải dữ liệu hai chiều, tức là cả hai bên (client và server) đều có thể gửi và nhận dữ liệu qua đó thuận tiện cho việc trao đổi liên tục giữa Uploader và Downloader.
- Sockets cho phép các ứng dụng có thể thực hiện nhiều kết nối đồng thời (thông qua các thư viện hỗ trợ như threading, asyncio trong Python), giúp tăng cường khả năng xử lý của ứng dụng. Ví dụ: một máy chủ có thể sử dụng socket để xử lý nhiều yêu cầu từ nhiều client cùng lúc mà không bị gián đoạn.
- Socket có thể được sử dụng trên nhiều hệ điều hành và nền tảng khác nhau như Windows, macOS, Linux, và các thiết bị di động. Điều này giúp ứng dụng mạng sử dụng socket có thể dễ dàng tương thích và hoạt động trên nhiều thiết bị khác nhau.

2.2 Giao thức Client - Tracker

Được xây dựng dựa trên giao thức HTTP, các client gửi các thông báo đến tracker thông qua giao thức HTTP GET “announce”+ /<command> ”+ “? ”+ tham số

2.2.1 Client và Tracker

- Client tham gia vào Tracker:
  - “/announce/join?peerid=BKU−Torrent−836763067317&port=8333”
  - Tracker sẽ dựa vào peerid để đưa Client vào swarm và lưu lại port mà client đang mở để chấp nhận kết nối từ các client khác.
- Client thông báo có đủ file cho Tracker:

  - "/announce/have?torrent_hash=%C1%27%12%94%D7%F6%CB5%94%FB%09%10%7Bs%C1%BF%85%24%16B&peerid=BKU−Torrent−836763067317"
  - Tracker sẽ lưu thông tin của client có peerid đang chia sẻ file theo tham số torrent_hash là mã hash đại diện cho file torrent đó.

- Client thông báo muốn tải file cho Tracker:
  - "/announce/down?torrent_hash%C1%27%12%94%D7%F6%CB5%94%FB%09%10%7Bs%C1%BF%85%24%16B&peerid=BKU−Torrent−836763067317"
  - Tracker sẽ dựa vào torrent_hash để gửi cho peerid một danh sách các IP và port của các client đang có file và đồng thời thêm peerid vào danh sách đó.

- Client thông báo thoát khỏi Tracker:
  - "/announce/exit?peeridBKUTorrent836763067317"
  - Tracker sẽ xóa peerid ra khỏi swarm và danh sách các file mà peerid có.

- Client kiểm tra Tracker có hoạt động:
  - "/announce/ping"
  - Tracker sẽ trả về "OK"nếu còn hoạt động.

2.2.2 Một số lý do chính khi dùng HTTP để giao tiếp giữa client và tracker trong BitTorrent:

Phổ biến và dễ triển khai

- HTTP là giao thức phổ biến và đã được sử dụng rộng rãi, hỗ trợ trên hầu hết các mạng và thiết bị. Điều này giúp việc triển khai tracker dễ dàng hơn vì có thể sử dụng cơ sở hạ tầng mạng sẵn có, như các máy chủ web và cổng HTTP.

Tương thích tốt với tường lửa và NAT

- HTTP hoạt động trên cổng 80 hoặc 443, các cổng này thường được mở trên hầu hết các tường lửa và thiết bị NAT. Điều này giúp client dễ dàng kết nối với tracker mà ít gặp trở ngại, tránh được các hạn chế mạng mà các giao thức không phổ biến hơn có thể gặp phải.

Dễ mở rộng và hỗ trợ

- HTTP là giao thức linh hoạt, hỗ trợ nhiều tính năng nâng cao như truyền tải qua HTTPS, giúp bảo mật dữ liệu. Sử dụng HTTP cũng giúp tracker dễ dàng mở rộng hoặc thay đổi để hỗ trợ nhiều loại client khác nhau, kể cả khi các client sử dụng phần mềm và phiên bản khác nhau.

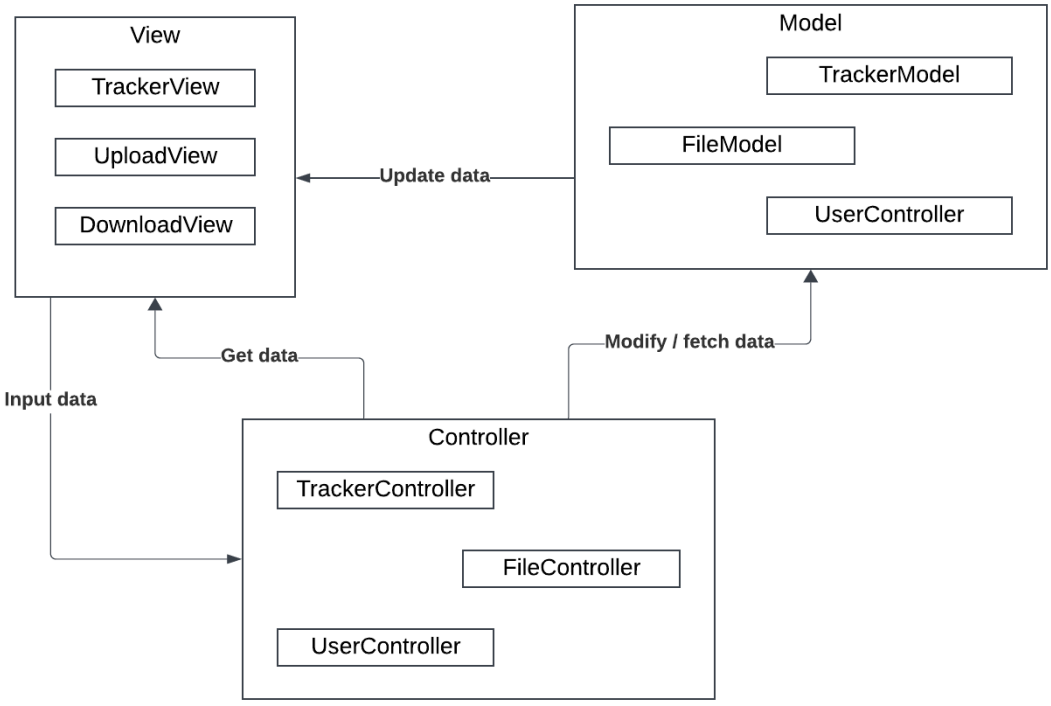
Tận dụng các công nghệ sẵn có

- Các công nghệ web khác như caching (bộ nhớ đệm) và load balancing (cân bằng tải) có thể dễ dàng áp dụng cho tracker khi dùng HTTP. Ví dụ, nếu nhiều client cùng yêu cầu thông tin về một torrent, tracker có thể tận dụng cache để trả về thông tin này nhanh chóng mà không cần xử lý lại từ đầu.

Hiệu quả và tiết kiệm tài nguyên

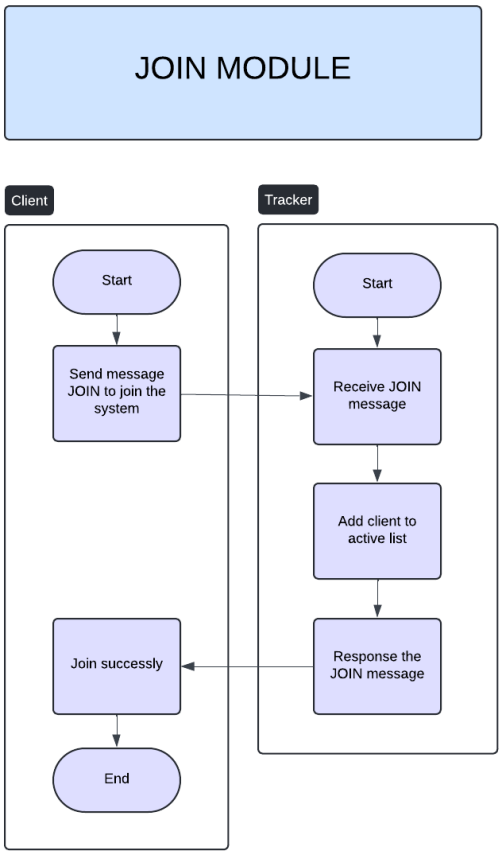
- Các yêu cầu HTTP thường là đơn giản và nhẹ, chứa các tham số cần thiết trong URL (như /announce?torrent_hash= ...&peer_id=...). Điều này giúp tracker xử lý nhiều yêu cầu cùng lúc mà không tốn quá nhiều tài nguyên, thích hợp cho hệ thống theo dõi mạng ngang hàng (P2P) có lượng yêu cầu rất lớn.

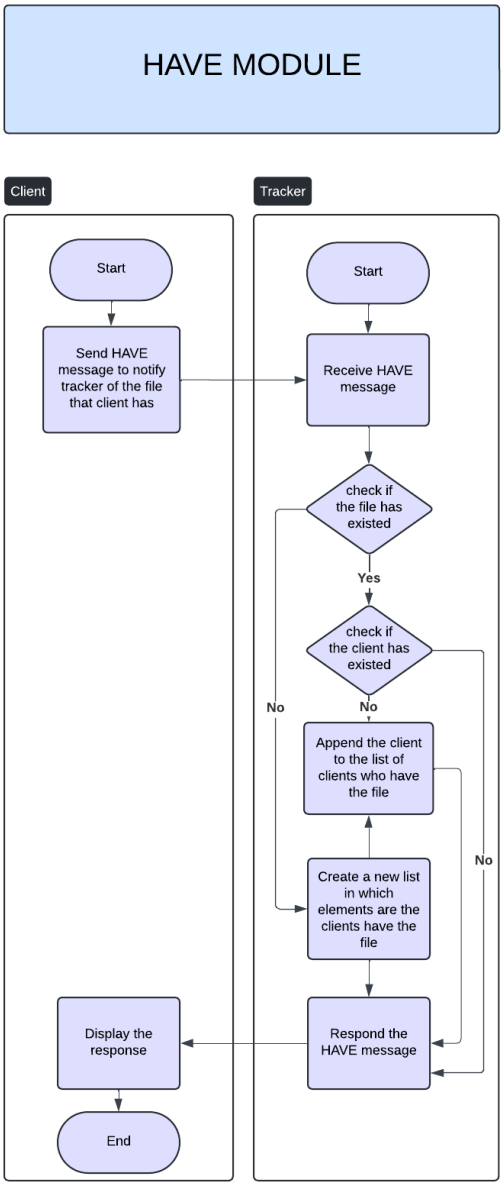
2.3 Biểu đồ:

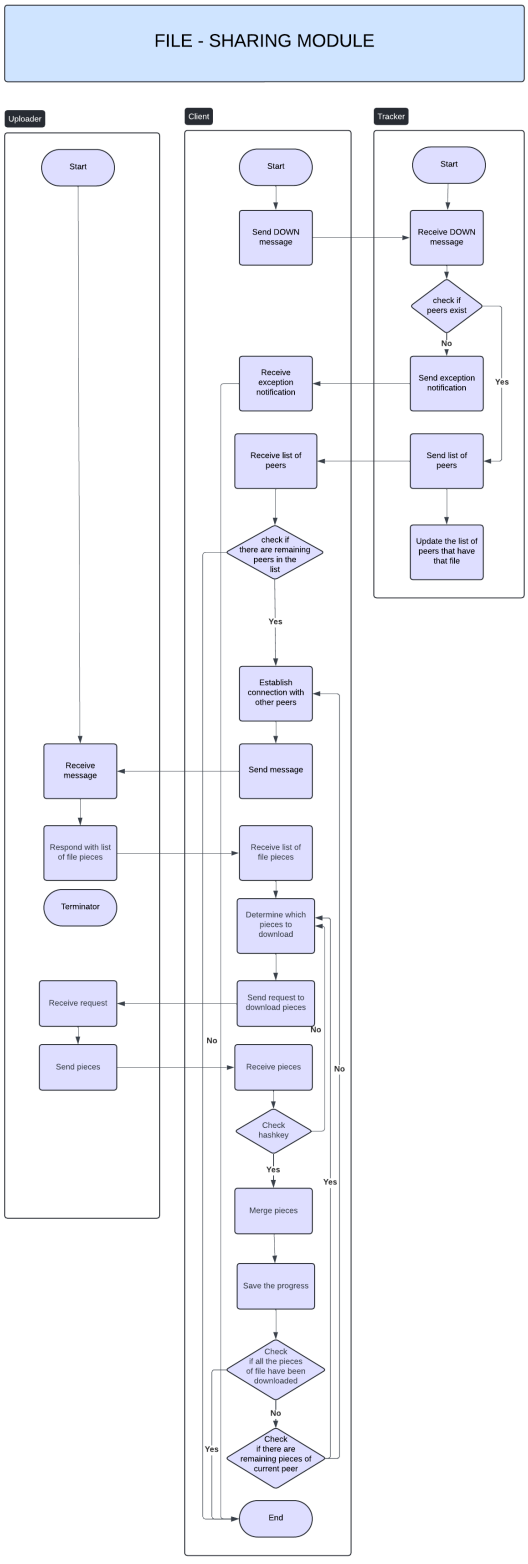


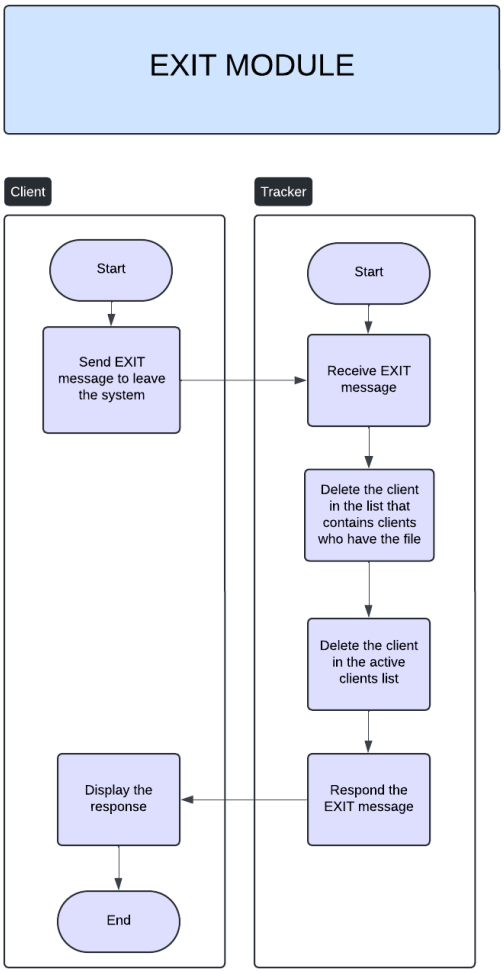
MVC

Flow chart của các module:


JOIN MODULE
Client
Tracker
Start
Send message JOIN to join the system
Join successly
End
Start
Receive JOIN message
Add client to active list
Response the JOIN message


HAVE MODULE
Client
Tracker
Start
Send HAVE message to notify tracker of the file that client has
Display the response
End
Start
Receive HAVE message
check if the file has existed
Yes
No
check if the client has existed
No
No
Append the client to the list of clients who have the file
Create a new list in which elements are the clients have the file
Respond the HAVE message


FILE - SHARING MODULE
Uploader
Client
Tracker
Start
Send DOWN message
Receive DOWN message
check if peers exist
No
Yes
Send exception notification
Receive exception notification
Send list of peers
Receive list of peers
Update the list of peers that have that file
check if there are remaining peers in the list
Yes
Establish connection with other peers
Receive message
Send message
Respond with list of file pieces
Receive list of file pieces
Terminator
Determine which pieces to download
Receive request
Send request to download pieces
No
Send pieces
Receive pieces
No
Check hashkey
Yes
Yes
Merge pieces
Save the progress
Check if all the pieces of file have been downloaded
No
Yes
Check if there are remaining pieces of current peer
End


EXIT MODULE
Client
Tracker
Start
Send EXIT message to leave the system
Display the response
End
Start
Receive EXIT message
Delete the client in the list that contains clients who have the file
Delete the client in the active clients list
Respond the EXIT message