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 Functions of application

1.1 BitTorrent File-Sharing Application Functions and Communication Protocols

File Discovery and Metadata Sharing *Description:* Allows users to locate and retrieve metadata about files to be shared or downloaded, typically through .torrent files or magnet links. *Communication Protocols:*

- **HTTP:** For downloading .torrent files or accessing magnet links.

Peer Discovery *Description:* Enables the identification and connection to other peers sharing the same file, ensuring distributed sharing. *Communication Protocols:*

- **BitTorrent Tracker Protocol (HTTP):** Centralized approach to finding peers.

File Piece Distribution *Description:* Divides files into smaller pieces for efficient sharing. Each piece is individually shared and verified to ensure data integrity. *Communication Protocols:*

- **BitTorrent Protocol over TCP/UDP:** Manages the exchange of file pieces.

Upload *Description:* Handles the sending of file pieces from a user's device to multiple peers in the network. Ensures balanced contribution and resource sharing. *Communication Protocols:*

- **BitTorrent Protocol over TCP/UDP:** Governs the sending of data pieces to peers.

Download *Description:* Manages receiving file pieces from other peers. Ensures optimal use of network resources by downloading from multiple peers concurrently. *Communication Protocols:*

- **BitTorrent Protocol over TCP/UDP:** Controls receiving data pieces from multiple peers simultaneously.

Piece Verification *Description:* Verifies the integrity of each downloaded piece by comparing its hash to the expected value, ensuring the reliability of the download. *Communication Protocols:*

- **Internal Hash Verification (SHA-1):** Conducted within the application to confirm the accuracy of received data.

Tit-for-Tat Function *Description:* Implements a strategy to ensure fair sharing by prioritizing peers that contribute more data. Encourages reciprocation among peers to balance the network load. *Communication Protocols:*

- **BitTorrent Protocol:** Uses a built-in mechanism to manage upload-/download ratios and prioritize reciprocating peers.

Encryption Function *Description:* Ensures secure data transfer by encrypting communications between peers. Uses key exchange mechanisms to establish a secure connection. *Communication Protocols:*

- **Diffie-Hellman Key Exchange:** Establishes a shared secret between peers to encrypt communication.
- **AES (Advanced Encryption Standard):** Encrypts the actual data transfer for security.

Error Handling and Recovery *Description:* Detects and recovers from issues like incomplete or corrupted downloads. Retries failed downloads and re-requests missing pieces from other peers. *Communication Protocols:*

- **BitTorrent Protocol:** Contains mechanisms for retrying failed piece downloads and error detection.

1.2 Tracker-Specific Functions

Tracker Registration *Description:* Registers new peers with the tracker to enable them to participate in the file-sharing network. *Communication Protocols:*

- **BitTorrent Tracker Protocol (HTTP):** Used to register peers with a centralized tracker.

Tracker Announce *Description:* Updates the tracker with a peer's status, such as upload/download progress, which files the client is seeding and connection status. *Communication Protocols:*

- **BitTorrent Tracker Protocol (HTTP):** Communicates the peer's status to the tracker.

Tracker Peer List Retrieval *Description:* Allows peers to retrieve a list of other peers sharing the same file from the tracker. *Communication Protocols:*

- **BitTorrent Tracker Protocol (HTTP):** Provides a list of active peers from the tracker.

2 Protocols

2.1 Uploader - Downloader Protocol

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 Client - Tracker Protocol

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%1
  - 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
  - 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?peerid$\bar{}$BKUTorrent836763067317"
  - 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: