ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
Khoa Khoa học và kỹ thuật máy tính

BÀI TẬP LỚN MÔN
MẠNG MÁY TÍNH
(CO3093)

Lớp: TN01 – Học kỳ: 241

NETWORK APPLICATION

Giảng viên hướng dẫn: **TS. NGUYỄN LÊ DUY LAI**

Sinh viên thực hiện: Lê Vân Anh Khoa, 2211605
Võ Thanh Tâm, 2213046

Tp. Hồ Chí Minh, tháng 11 năm 2024.

Mục lục

Danh sách hình vẽ

1 Bảng phân công nhiệm vụ

Tên thành viên	Nhiệm vụ	Mức độ đóng góp
Lê Văn Anh Khoa	Thực hiện giao tiếp giữa các peer	50%
Võ Thanh Tâm	Tracker Server, xây giao diện	50%

2 Các chức năng của ứng dụng

2.1 Chia sẻ file và folder

Một người dùng muốn chia sẻ một số file, hoặc một folder gồm nhiều file, khi nhấn nút "Share", cửa sổ File Explorer hiện lên để người dùng chọn các file và folder cần chia sẻ.
Sau khi file/folder được tải lên, phần mềm sẽ xử lý và trả một file `.torrent` chứa các thông tin cần thiết cho người khác có thể dùng và đưa vào phần mềm để tải về các file hoặc folder đã được chia sẻ.

2.2 Tải file và folder qua .torrent

Người dùng có được file `.torrent`, nếu có nhu cầu muốn tải các file hoặc folder được chia sẻ thông qua file `.torrent`, thì mở ứng dụng và nhấn nút "Download". Lúc này cửa sổ Window Explorer sẽ hiện lên để người dùng chọn file `.torrent` chứa các file cần tải, sau đó tiến trình tải sẽ bắt đầu.

2.3 Xem thông tin của mỗi torrent

Nếu người dùng muốn xem thông tin các peer đang hoạt động (tải file/seed file) trong một torrent bất kì, thì nhất vào mục "Scrape", lúc này cửa sổ Window Explorer hiện lên để người dùng chọn file `.torrent` để xem các thông tin cần thiết.

3 Các protocol của ứng dụng

3.1 Giữa các Peer và Tracker Server

Mỗi khi người dùng khởi động ứng dụng, thì phiên hoạt động này sẽ trở thành một peer và được kết nối với Tracker Server (như mô hình Client-Server).

3.1.1 Tải lên

Khi người dùng tải lên một file hoặc một folder gồm nhiều file, phần mềm sẽ xử lý các file được tải lên và cho ra một file `.torrent` gồm các thông tin cần thiết để người khác có thể dùng và tải các file đã chia sẻ. Nội dung của file `.torrent` như sau.

Một file `.torrent` chứa các thông tin được bencoded dưới dạng dictionary, với các key chính sau:

1. `info`:
   - **Mô tả**: Dictionary chứa thông tin về file hoặc các file trong torrent.
   - **Chi tiết**:
     - Với torrent chứa **một file** (single-file torrent), chỉ có thông tin về file đó.
     - Với torrent chứa **nhiều file** (multi-file torrent), thông tin bao gồm cấu trúc thư mục và danh sách các file.
2. `announce`:
   - **Mô tả**: URL của tracker (chuỗi).
   - **Vai trò**: Tracker giúp kết nối các peers trong mạng P2P.
3. `announce-list` (tuỳ chọn):
   - **Mô tả**: Danh sách mở rộng URL tracker, dạng *list of lists* (mỗi tracker có thể có nhiều endpoint).
   - **Lợi ích**: Cung cấp tính tương thích ngược với các client cũ.
4. `creation date` (tuỳ chọn):
   - **Mô tả**: Thời điểm tạo file torrent, dạng **UNIX epoch** (số giây từ 01/01/1970 00:00:00 UTC).
   - **Kiểu dữ liệu**: Integer.
5. `comment` (tuỳ chọn):
   - **Mô tả**: Nhận xét tự do từ tác giả file torrent (chuỗi).
6. `created by` (tuỳ chọn):
   - **Mô tả**: Tên và phiên bản của chương trình dùng để tạo file torrent (chuỗi).
7. `encoding` (tuỳ chọn):
   - **Mô tả**: Định dạng mã hoá chuỗi được sử dụng trong phần `pieces` của dictionary `info` (chuỗi).

- **Ví dụ**: UTF-8 hoặc các định dạng khác.

Sau khi file `.torrent` được tạo:

- Peer sẽ thông báo với Tracker Server rằng peer đang giữ file này và sẵn sàng để được leech bởi các peer khác nếu họ có nhu cầu, thông qua request `/announce`.
- Tracker Server sẽ thêm peer này vào các peer đang hoạt động trong torrent trên.
- Tracker Server gửi lại thông tin của các peer đang hoạt động trong torrent này cho peer này (có bao gồm chính nó).

3.1.2 Tải về

Người dùng tải lên một file `.torrent` để tải các file được chia sẻ từ torrent này.
Lúc này quy trình sẽ như sau:

- Peer sẽ gửi thông báo tới **Tracker Server** được liệt kê trong file torrent, với yêu cầu thông qua `/announce`.
- Tracker Server:
  - Tìm các **peer đang hoạt động** trong torrent tương ứng.
  - Thêm thông tin của peer yêu cầu vào danh sách các peer hoạt động trong torrent này.
  - Gửi lại thông tin của các peer đang hoạt động trong torrent này cho peer đang cần tải (có bao gồm chính nó).

Sau khi nhận được thông tin này, peer sẽ thực hiện kết nối trực tiếp với các peer được liệt kê để tải file.

3.1.3 Request /announce

`/announce` request gửi đến Tracker Server có định dạng như sau:

```
1 GET /announce?info_hash=b%DE%D6%C3y%8El%00%3E%07%BE%9F%9F6Z%FA%2A
    %0A%D1%FC&peer_id=-PY0001-YYbuyNCwrD1y&ip=10.130.85.145&port
    =64219&uploaded=0&downloaded=0&left=0&compact=0&event=STARTED
    HTTP/1.1
2 Host: tracker.example.com
```

Định nghĩa các tham số trong `/announce` request:

- `info_hash`:
  - Một chuỗi 20 byte SHA1 hash, URL-encoded, được lấy từ giá trị của khóa `info` trong file Metainfo (`.torrent`).
  - Giá trị của `info` là một dictionary được mã hóa bằng bencode.
- `peer_id`:
  - Một chuỗi 20 byte, URL-encoded, được sử dụng làm ID duy nhất cho peer.
  - Peer ID được tạo bởi client khi khởi động và có thể là bất kỳ giá trị nào, bao gồm dữ liệu nhị phân.

  - Một số client thường mã hóa Peer ID theo các tiêu chuẩn riêng, ví dụ: `-AZ2060-`, `-UT3530-`, ở đây dùng `-PY0001`.

- `ip`:
  - Địa chỉ IP của peer, thường được sử dụng để xác định peer trên mạng.
  - Đây là một tham số tùy chọn. Nếu không được cung cấp, Tracker sẽ sử dụng địa chỉ IP từ kết nối hiện tại.
- `port`:
  - Số cổng mà client đang lắng nghe.
  - Các cổng phổ biến cho BitTorrent nằm trong dải `6881-6889`.
  - Nếu không thiết lập được cổng trong dải này, client có thể chuyển sang cổng ngẫu nhiên.
- `uploaded`:
  - Tổng số byte đã tải lên kể từ khi sự kiện `started` được gửi tới Tracker, biểu diễn dưới dạng ASCII.
- `downloaded`:
  - Tổng số byte đã tải xuống kể từ khi sự kiện `started` được gửi tới Tracker, biểu diễn dưới dạng ASCII.
- `left`:
  - Số byte còn lại mà peer cần tải xuống để hoàn thành torrent, biểu diễn dưới dạng ASCII.
  - Giá trị này giúp Tracker biết peer đã hoàn thành việc tải file hay chưa.
- `compact`:
  - Nếu được đặt thành `1`, client sẽ nhận phản hồi dạng compact (danh sách peers được thay thế bằng một chuỗi byte gồm 6 byte mỗi peer: 4 byte địa chỉ IP và 2 byte cổng).
  - Nếu được đặt thành `0`, client yêu cầu phản hồi không compact (thường sẽ bị từ chối bởi một số Tracker để tiết kiệm băng thông).
- `event`:
  - Loại sự kiện mà peer gửi tới Tracker. Các giá trị có thể bao gồm:
    * `STARTED`: Peer bắt đầu tải torrent.
    * `STOPPED`: Peer dừng tải torrent.
    * `COMPLETED`: Peer đã hoàn thành việc tải xuống toàn bộ torrent.

Tracker Server nhận yêu cầu `/announce`, phản hồi sẽ có cấu trúc:

```
{
  "tracker_id": "TRACKER123",
  "info_hash": "abc123",
  "peers": [
    {
```

```
      "peer_id": "-AZ2060-6wfG2wk6wF00",
      "ip": "192.168.1.5",
      "port": 6881
    },
    {
      "peer_id": "-UT3530-9dbXwJgQ8Kmn",
      "ip": "203.0.113.25",
      "port": 51413
    }
  ]
}
```

- `tracker_id`: ID của Tracker Server.
- `info_hash`: Mã băm của torrent đang được yêu cầu.
- `peers`: Danh sách các peer đang hoạt động, bao gồm:
  - `peer_id`: Chuỗi định danh của peer.
  - `ip`: Địa chỉ IP của peer.
  - `port`: Cổng kết nối của peer.

3.1.4 Lấy thông tin scrape của file torrent

Nếu người dùng không có nhu cầu tải hoặc chia sẻ file, mà chỉ muốn xem thông tin các peer đang hoạt động với một torrent bất kì, thì peer tương ứng với phiên đăng nhập của người dùng sẽ gửi scrape request của file `.torrent` đến Tracker Server.

Một yêu cầu `/scrape` cơ bản có thể có định dạng như sau:

```
GET /scrape?info_hash=%D1%FC%3E%07%BE%9F%9F6Z%FA%2A%0A HTTP/1.1
Host: tracker.example.com
Connection: close
```

Định nghĩa các tham số trong `/scrape` Request

- `info_hash`:
  - Một chuỗi 20 byte SHA1 hash, URL-encoded, đại diện cho torrent mà client muốn lấy thông tin.
  - Nhiều `info_hash` có thể được gửi trong một yêu cầu `/scrape` bằng cách phân tách các giá trị bằng ký tự `&`.

Tracker Server sẽ tìm kiếm thông tin các peer đang hoạt động với file torrent này và trả về kết quả như sau:

```
{
    "tracker_id": "tracker-12345",
    "info_hash": "b2d6c3798e6c003e07be9f9f365afa2a0ad1fc",
    "total_peers": 42
}
```

Với:

- `tracker_id`:
  - Chuỗi định danh duy nhất của Tracker Server đang phản hồi.
- `info_hash`:
  - Chuỗi 20 byte (SHA1 hash) xác định torrent mà yêu cầu liên quan đến.
- `total_peers`:
  - Số lượng peer đang hoạt động liên quan đến `info_hash`.

Peer nhận được thông tin từ scrape request, peer sẽ không làm gì khác, trong khi với announce request, peer sẽ kết nối với các peer được liệt kê trong thông tin trả về nếu là để tải file.

3.2 Giữa các Peer với nhau

Khi một peer đã nhận được danh sách các peer đang seed hoặc leech torrent từ tracker, bước tiếp theo là thiết lập kết nối trực tiếp với các peer này để trao đổi dữ liệu.

3.2.1 2-way Handshake

Handshake là thông điệp bắt buộc và phải là thông điệp đầu tiên được gửi giữa hai peer muốn kết nối. Thông điệp này xác minh rằng cả hai peer đang sử dụng cùng giao thức và cùng tham gia vào một torrent cụ thể.

Cấu trúc của thông điệp handshake:

$$\text{handshake} : \langle pstrlen \rangle \langle pstr \rangle \langle reserved \rangle \langle info\_hash \rangle \langle peer\_id \rangle$$

Với:

- **pstrlen**: Độ dài của chuỗi giao thức (`<pstr>`), được biểu diễn bằng một byte.
- **pstr**: Chuỗi định danh giao thức, thường là `"BitTorrent protocol"`.
- **reserved**: 8 byte dự phòng, tất cả thường được đặt là 0.
- **info_hash**: Băm SHA-1 dài 20 byte của khóa `info` trong tệp `.torrent`.
- **peer_id**: Chuỗi dài 20 byte, là định danh duy nhất cho mỗi peer.

Sau khi nhận được thông điệp handshake từ một peer khác, peer cần kiểm tra `info_hash` để xác nhận rằng torrent được tham gia là cùng một torrent, trước khi tiến hành giao tiếp thêm.

Quy trình:

1. Peer A gửi thông điệp handshake cho Peer B.
2. Peer B nhận, kiểm tra `info_hash` và phản hồi bằng thông điệp handshake tương ứng.

3.2.2 Choke & Unchoke

Các thông điệp `Choke` và `Unchoke` điều khiển quyền truy cập của một peer để tải xuống dữ liệu:

- **Choke** (`<id=0>`): Thông điệp chặn, thông báo rằng peer sẽ không gửi dữ liệu cho peer khác. Thông điệp này không có payload.

$$\text{choke} : \langle \text{len}=0001 \rangle \langle \text{id}=0 \rangle$$

- **Unchoke** (`<id=1>`): Thông điệp bỏ chặn, cho phép peer yêu cầu và nhận dữ liệu.

$$\text{unchoke} : \langle \text{len}=0001 \rangle \langle \text{id}=1 \rangle$$

Việc sử dụng `choke` và `unchoke` giúp các peer ưu tiên tài nguyên mạng cho những kết nối mang lại hiệu quả cao hơn.

Quy trình:

1. Ban đầu, Peer A gửi `Choke` để từ chối yêu cầu từ Peer B.
2. Khi sẵn sàng chia sẻ, Peer A gửi `Unchoke`.

3.2.3 Gửi Interest & Uninterest

Các thông điệp `interested` và `not interested` được sử dụng để thể hiện trạng thái quan tâm hoặc không quan tâm đến việc nhận dữ liệu từ một peer.

- `interested`: Peer thông báo quan tâm đến dữ liệu của peer khác.

$$\text{interested} : \langle \text{len}=0001 \rangle \langle \text{id}=2 \rangle$$

- `not interested`: Peer không còn quan tâm đến dữ liệu.

$$\text{not interested} : \langle \text{len}=0001 \rangle \langle \text{id}=3 \rangle$$

Quy trình:

1. Peer A gửi `Interested` nếu bitfield của Peer B chứa phần dữ liệu mà Peer A cần.
2. Khi đã hoàn tất hoặc không cần thêm dữ liệu, Peer A gửi `Not Interested`.

3.2.4 Gửi Bitfield

Thông điệp `bitfield` cung cấp thông tin về trạng thái sở hữu các mảnh dữ liệu của một peer. `Bitfield` thường được gửi ngay sau khi hoàn tất handshake, với dạng là một chuỗi bit.

$$\text{bitfield} : \langle \text{len}=0001+\text{X} \rangle \langle \text{id}=5 \rangle \langle \text{bitfield} \rangle$$

- `bitfield`: biểu diễn trạng thái các mảnh:
  - Bit cao nhất trong byte đầu tiên tương ứng với `index=0`.
  - Bit được đặt giá trị 1 nếu mảnh tương ứng đã được tải xong, giá trị 0 nếu chưa tải.
  - Các bit dư thừa được đặt giá trị 0.

Có thể hiểu:

- Bit `1`: Peer sở hữu phần dữ liệu tương ứng.
- Bit `0`: Peer không sở hữu phần dữ liệu.

Nếu peer nhận được thông điệp `bitfield` không hợp lệ (sai độ dài hoặc chứa bit dư thừa không bằng 0), kết nối sẽ bị ngắt.

Quy trình:

1. Sau khi handshake, Peer A gửi `Bitfield` để thông báo trạng thái dữ liệu.
2. Peer B phân tích bitfield để xác định các phần dữ liệu cần tải xuống.

3.2.5 Gửi Pieces

Sau khi nhận được yêu cầu, peer gửi lại các khối dữ liệu được yêu cầu dưới dạng các thông điệp `piece`. Mỗi khối dữ liệu được truyền là một phần của mảnh, giúp đảm bảo tính linh hoạt và hiệu quả trong việc quản lý truyền tải.
Các thông điệp chính liên quan đến quá trình gửi một file bao gồm:

- **Request (`<id=6>`)**:

  $$\text{request} : \langle \text{len=0013} \rangle \langle \text{id=6} \rangle \langle \text{index} \rangle \langle \text{begin} \rangle \langle \text{length} \rangle$$

  Peer gửi yêu cầu tải một phần dữ liệu, bao gồm:

  - `index`: Vị trí của phần dữ liệu.
  - `begin`: Offset trong phần dữ liệu.
  - `length`: Kích thước khối dữ liệu cần tải.

- **Piece (`<id=7>`)**: Peer phản hồi với khối dữ liệu được yêu cầu, gồm:

  - `index, begin`: Thông tin về vị trí và offset.
  - `block`: Dữ liệu thực tế.

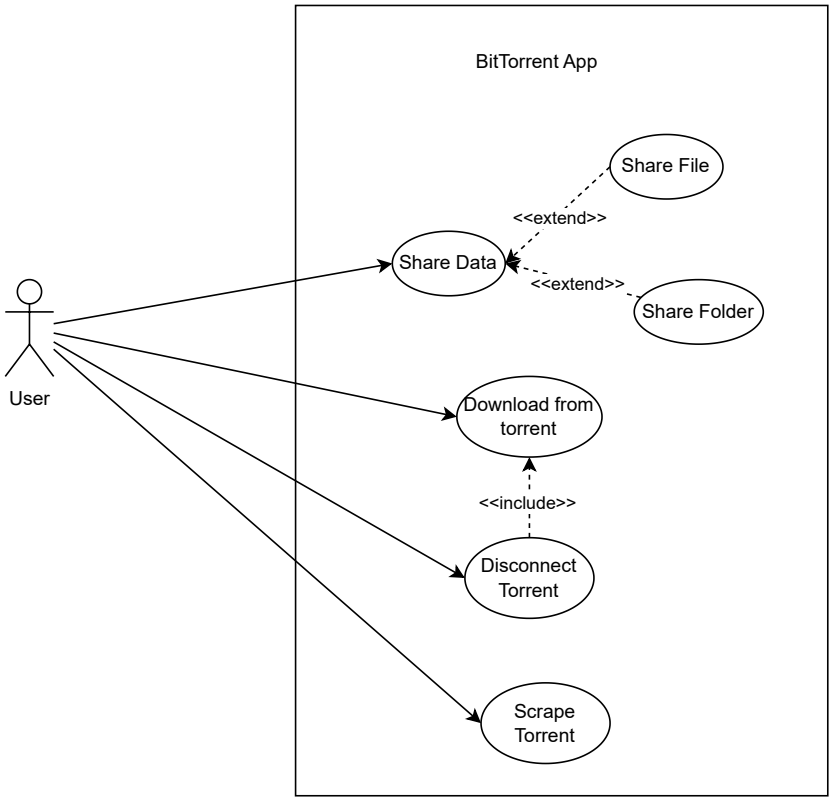
- **Cancel (`<id=8>`)**: Peer hủy yêu cầu tải xuống.

  Quy trình:

1. Peer A gửi `Request` đến Peer B.
2. Peer B kiểm tra trạng thái `Choke`, nếu được phép, gửi `Piece` với khối dữ liệu tương ứng.
3. Sau khi nhận đủ pieces, Peer A lắp ráp chúng thành file hoàn chỉnh.

4 Lược đồ Use case và mô tả

4.1 Lược đồ Use case



Hình 1: Use case Diagram

Use Case ID	Tên Use Case	Mô tả
UC-1	Share data	Người dùng chia sẻ dữ liệu
UC-1.1	Share file	Người dùng chia sẻ file
UC-1.2	Share folder	Người dùng chia sẻ folder
UC-2	Download from torrent	Người dùng tải xuống từ một torrent file
UC-2.1	Disconnect Torrent	Người dùng ngừng tham gia vào một mạng torrent
UC-3	Scrape Torrent	Người dùng gửi yêu cầu lấy thông tin của torrent từ Tracker Server

Bảng 1: Danh sách Use Case cho BitTorrent App

4.2 Mô tả chi tiết các use case

Use case ID	UC-1
Use case name	Share data
Created by	Lê Văn Anh Khoa
Last updated by	None
Date Created	19/11/2024
Date Last Updated	None
Actors	Actor-1: Người dùng
Description	Người dùng chia sẻ dữ liệu
Trigger	Người dùng chọn chia sẻ dữ liệu
Preconditions	Người dùng đã đăng nhập vào hệ thống
Postconditions	Người dùng tham gia vào mạng torrent mà mình đã chia sẻ
Normal Flow	1. Người dùng chọn chức năng chia sẻ ở transfer page. 2. Hệ thống xử lý yêu cầu chia sẻ. 3. Người dùng tham gia vào mạng torrent mà mình đã chia sẻ.
Alternative Flow	1a. Người dùng chia sẻ file 1b. Người dùng chia sẻ folder *Use case kết thúc*
Notes and issues	None

Bảng 2: Chi tiết use case UC-1: Share data

Use case ID	UC-2
Use case name	Download from torrent
Created by	Lê Văn Anh Khoa
Last updated by	None
Date Created	19/11/2024
Date Last Updated	None
Actors	Actor-1: Người dùng
Description	Người dùng tải xuống dữ liệu
Trigger	Người dùng chọn tham gia vào mạng torrent
Preconditions	Người dùng đã đăng nhập vào hệ thống
Postconditions	Người dùng tham gia vào mạng torrent
Normal Flow	1. Người dùng chọn chức năng thêm torrent ở transfer page. 2. Hệ thống xử lý yêu cầu tham gia vào mạng torrent. 3. Người dùng tham gia vào mạng torrent, bắt đầu tải xuống và tải lên các piece.
Exception Flow	2a. Hệ thống không tìm thấy torrent người dùng yêu cầu
Notes and issues	None

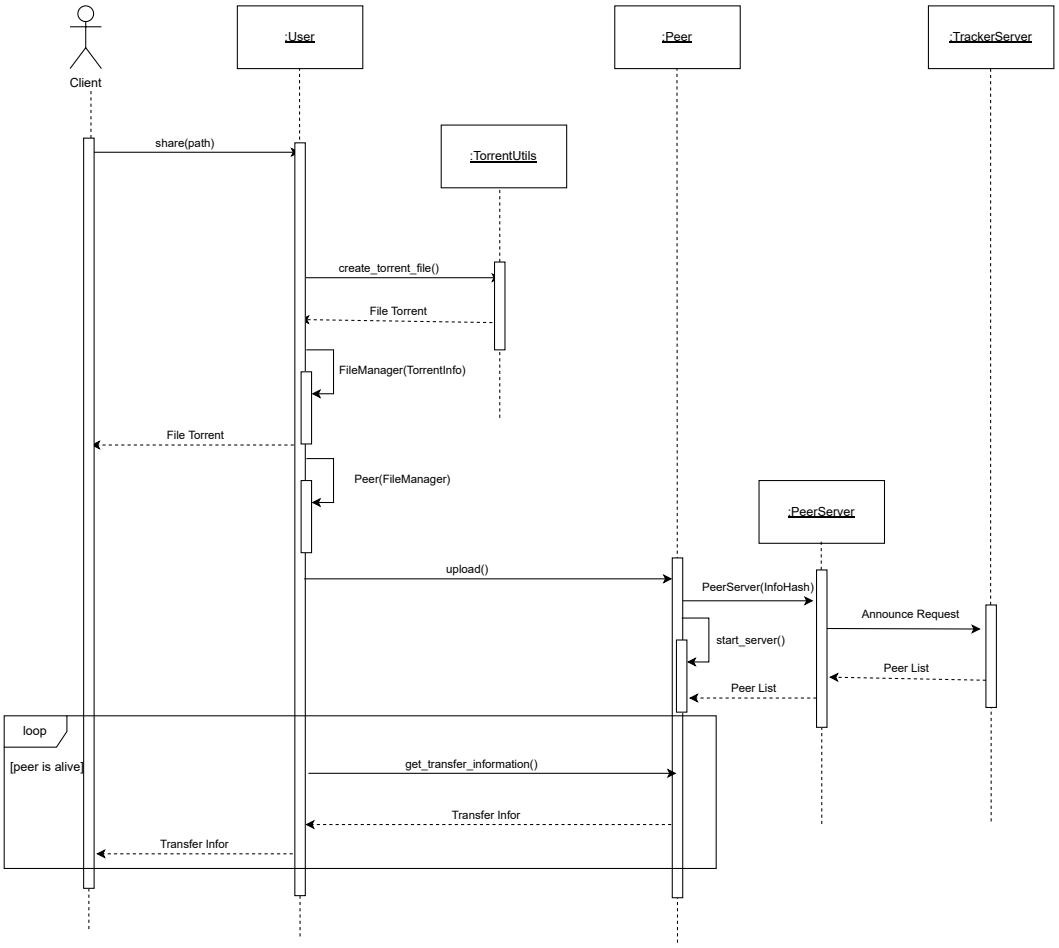
Bảng 3: Chi tiết use case UC-2: Download from torrent

Use case ID	UC-3
Use case name	Scrape data
Created by	Lê Văn Anh Khoa
Last updated by	None
Date Created	19/11/2024
Date Last Updated	None
Actors	Actor-1: Người dùng
Description	Người dùng yêu cầu xem thông tin của mạng torrent
Trigger	Người dùng chọn chức năng scrape torrent
Preconditions	Người dùng đã đăng nhập vào hệ thống
Postconditions	Hệ thống hiển thị thông tin của torrent
Normal Flow	1. Người dùng chọn chức năng scrape torrent. 2. Hệ thống xử lý yêu cầu. 3. Hệ thống hiển thị thông tin của torrent .
Exception Flow	2a. Hệ thống không tìm thấy torrent người dùng yêu cầu
Notes and issues	None

Bảng 4: Chi tiết use case UC-3: Scrape torrent

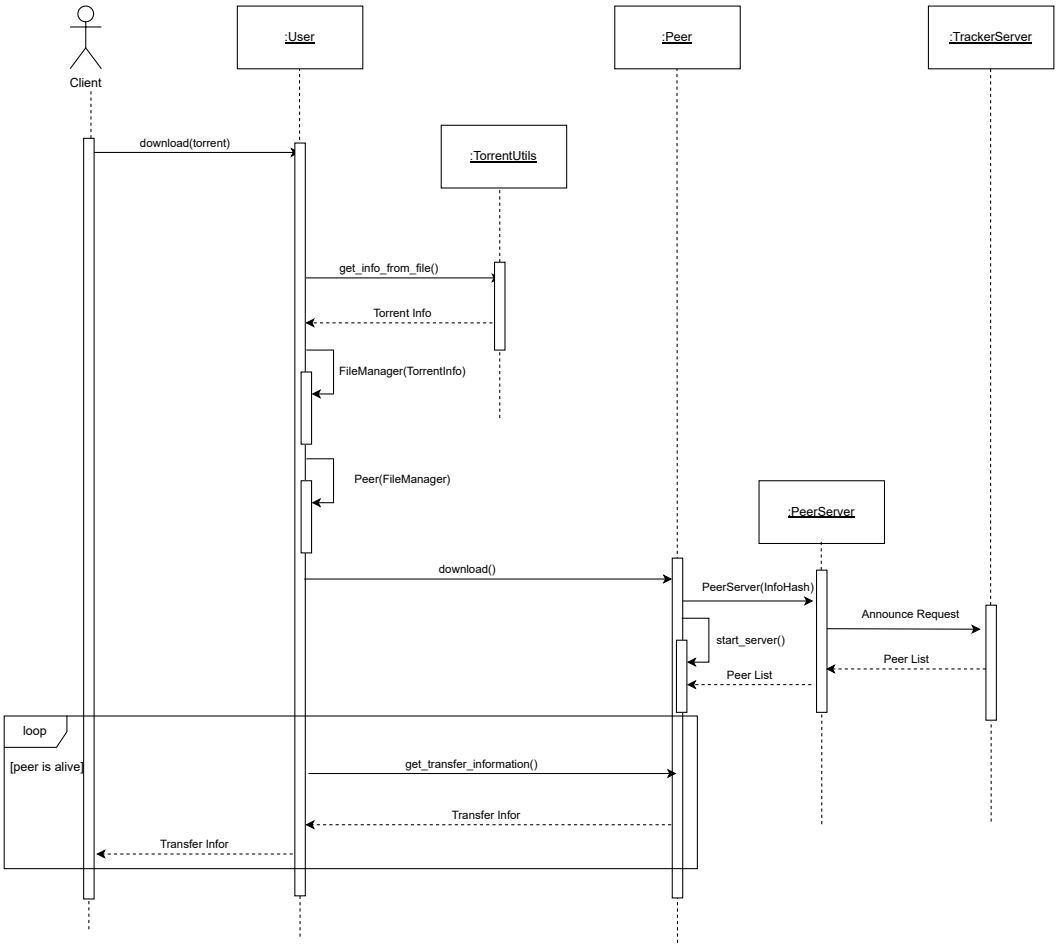
5 Lược đồ tuần tự

5.1 Share data



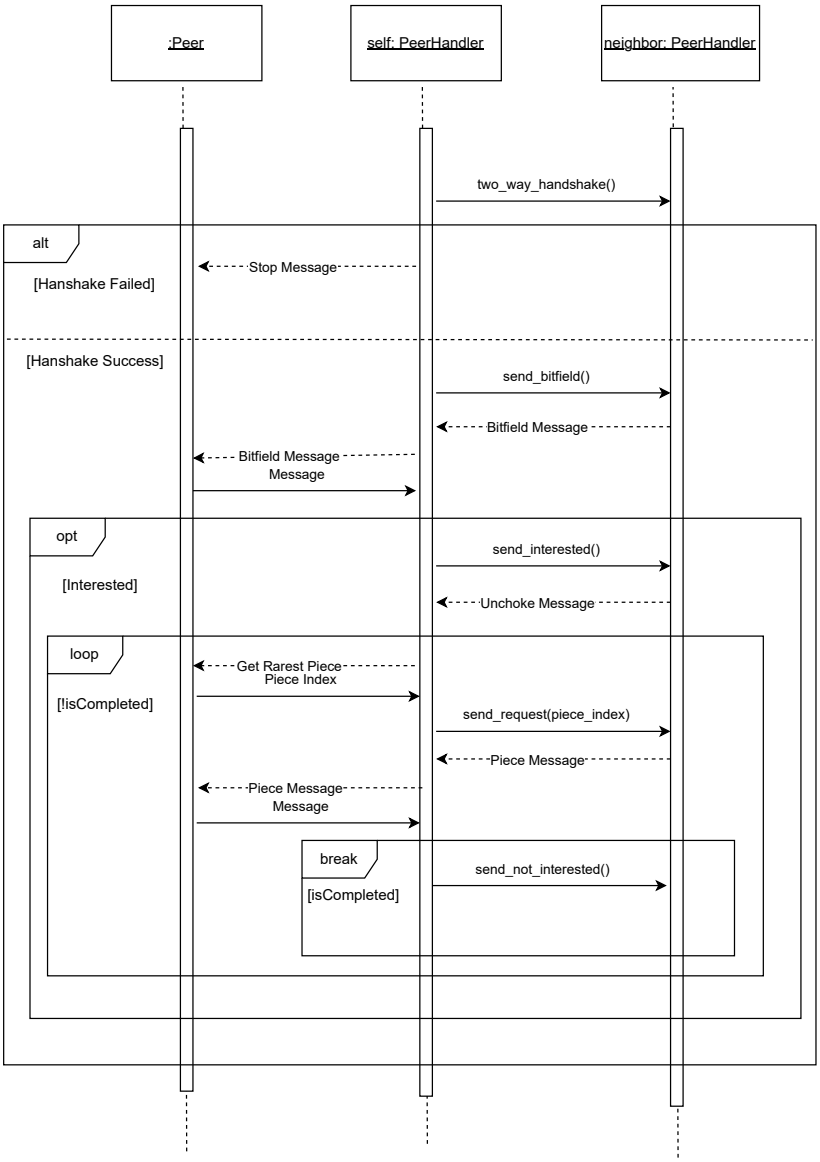
Hình 2: Sequence Diagram Share Data

5.2 Download from torrent



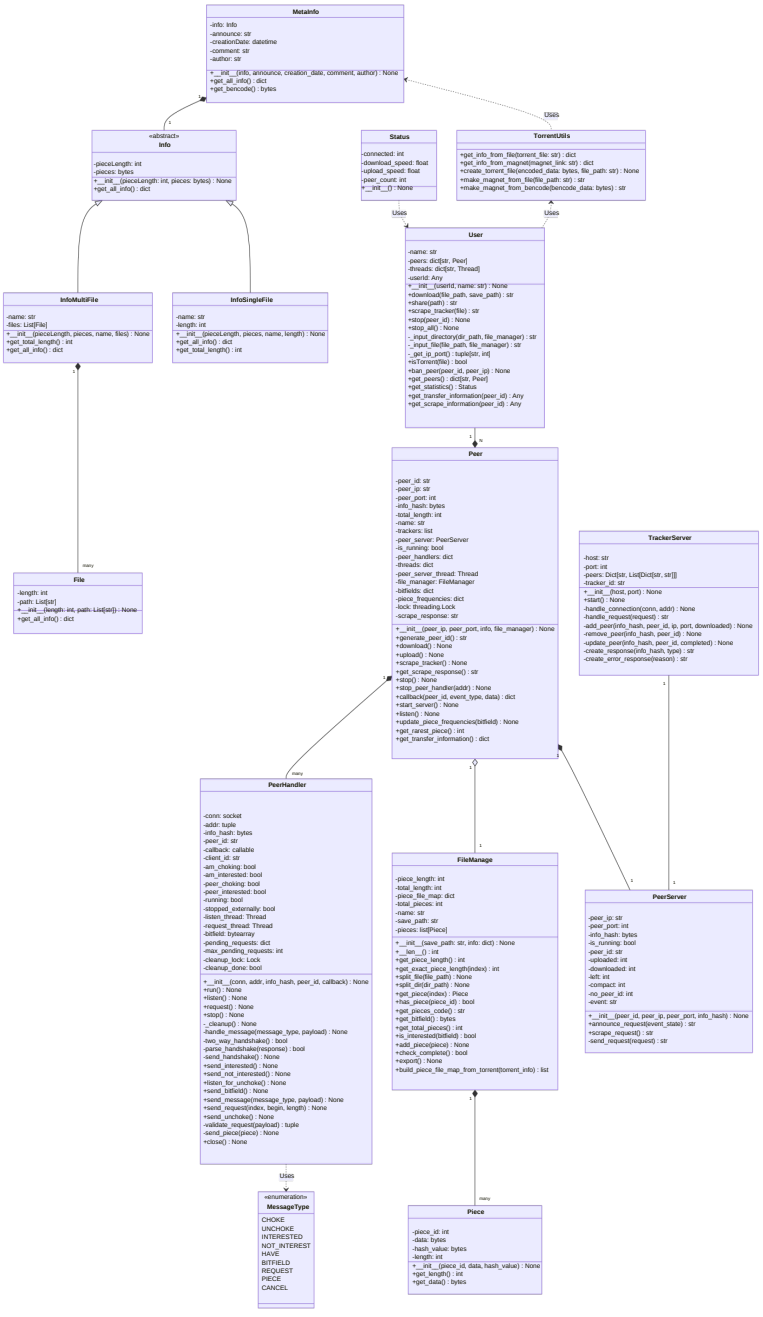
Hình 3: Sequence Diagram Download

5.3 Peer to Peer Protocol



Hình 4: Sequence Diagram Peer to Peer Protocol

6 Class Diagram



Hình 5: Class Diagram

7 Hướng dẫn sử dụng

7.1 Khởi động ứng dụng

Đầu tiên hãy clone app theo đường link Github được cung cấp bên dưới. Sau đó chạy lệnh

```
pip install -r requirements.txt
```

để tải các thư viện cần thiết.

Để mở Tracker Server, chạy lệnh

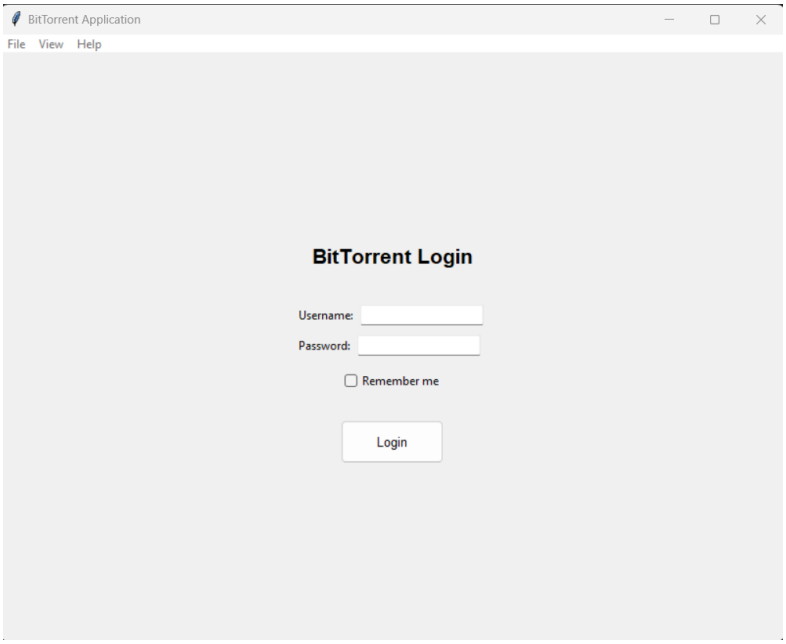
```
python TrackerServer.py
```

Để chạy application, chạy lệnh

```
python app.py
```

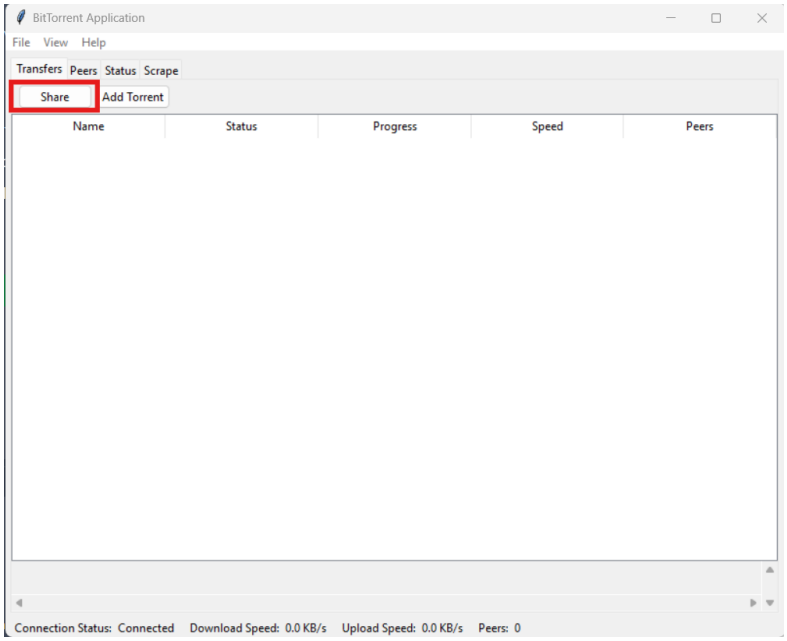
7.2 Chia sẻ

Khi vừa khởi động app bạn sẽ thấy giao diện đăng nhập như sau, cứ hãy nhập username và password bất kỳ để vào trong giao diện chính, vì nhóm vẫn chưa hiện thực bảo mật.



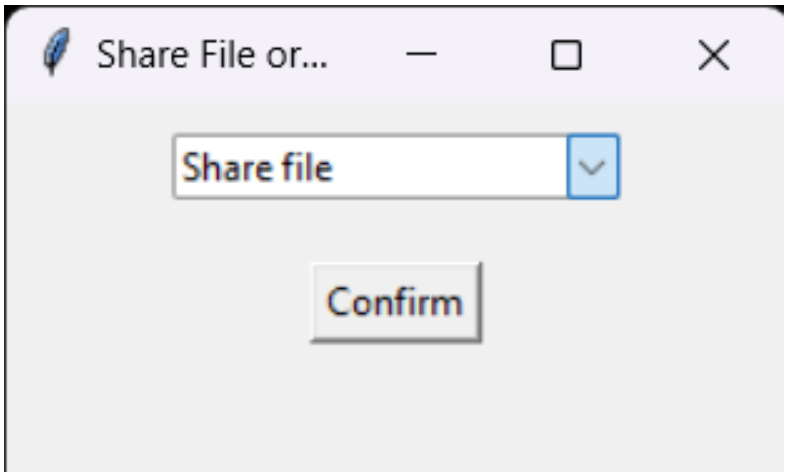
Hình 6: Giao diện đăng nhập

Nhấn vào share ở góc trái trên cùng để thực hiện chia sẻ.

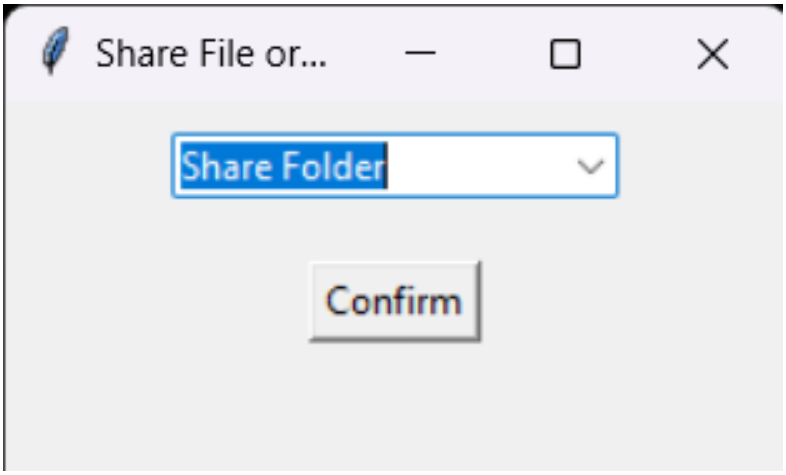


Hình 7: Giao diện chính

Các bạn có thể chọn chia sẻ 1 file hoặc 1 folder.

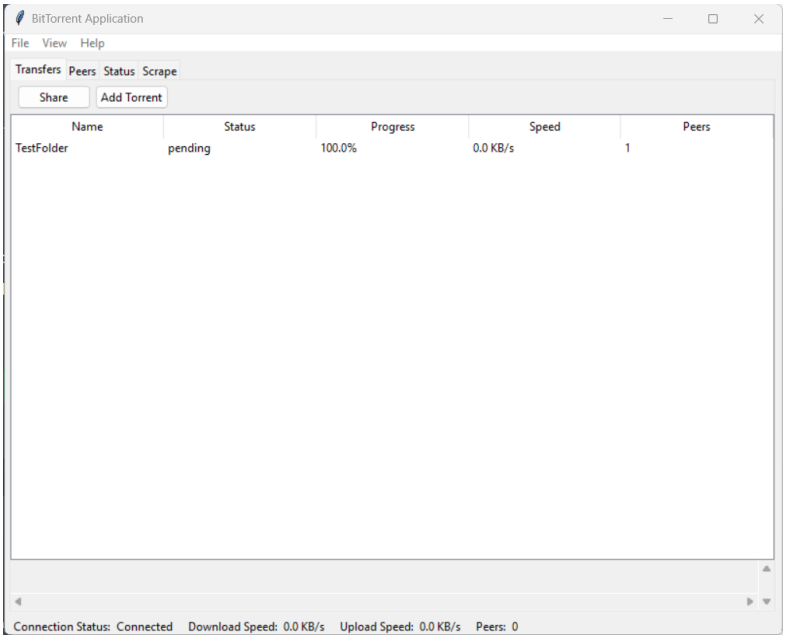


Hình 8: Chia sẻ file



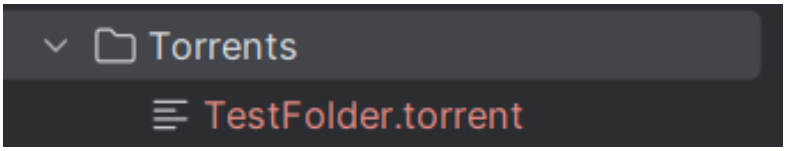
Hình 9: Chia sẻ folder

Sau khi chọn file để chia sẻ giao diện transaction sẽ hiện như sau.



Hình 10: Sau khi chia sẻ

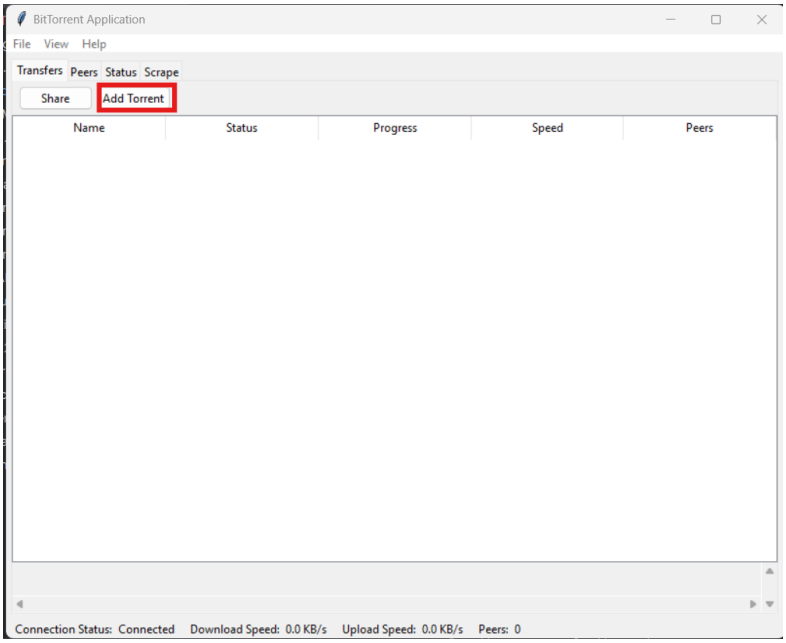
Bạn có thể thấy trong thư mục Torrent có một file torrent mới vừa được tạo ra như sau, chúng ta sẽ dùng file torrent này để tải xuống. Trên thực tế file torrent này sẽ phải được chia sẻ rộng rãi trên mạng Internet thông qua website như Internet Archive, Linux Tracker,...



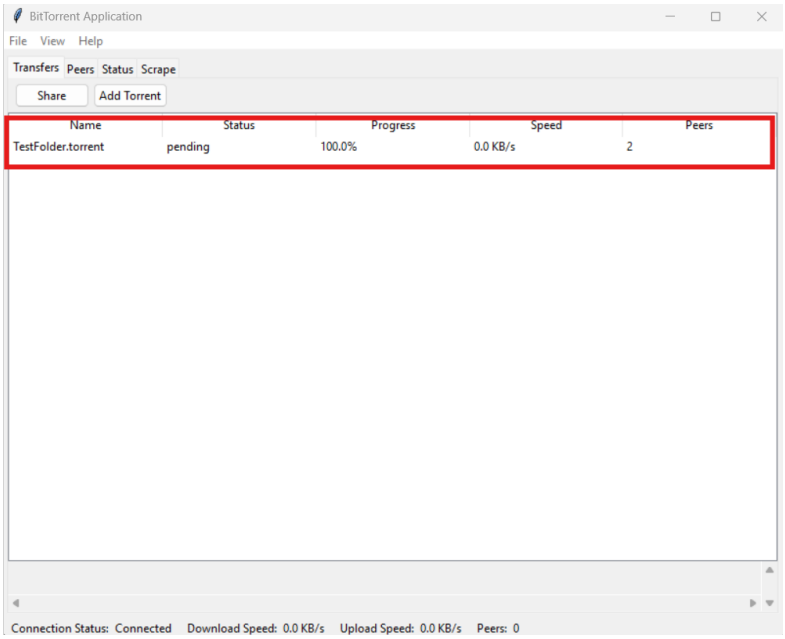
Hình 11: Torrent file

7.3 Tải xuống

Chúng ta sẽ chạy một process khác và dùng file torrent trên để tải TestFolder xuống.



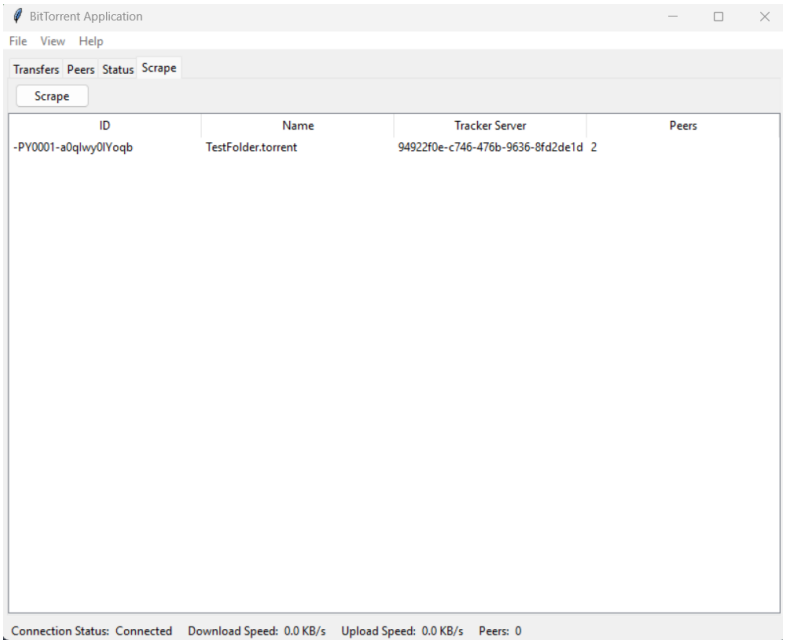
Hình 12: Thêm torrent



Hình 13: Tải thành công

7.4 Xem thông tin

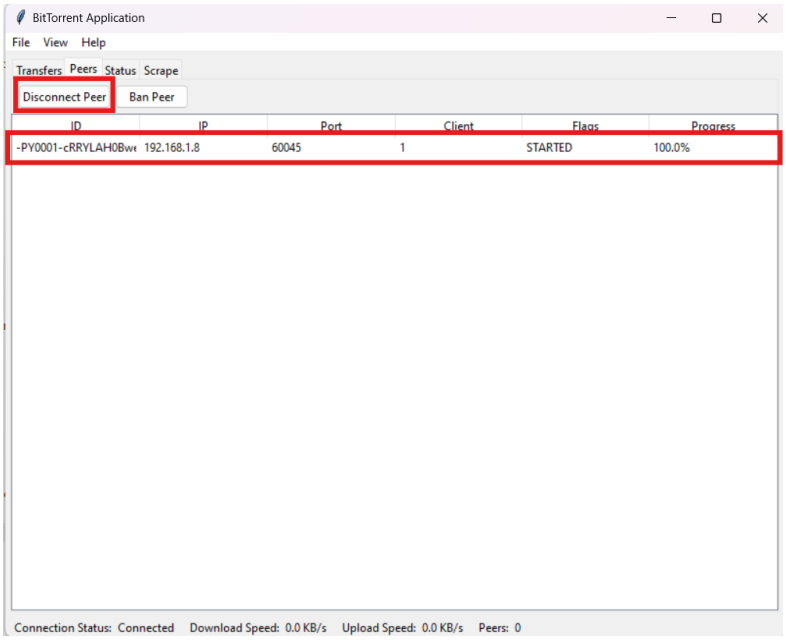
Bây giờ chúng ta sẽ thực hiện xem thông tin của file torrent thông qua chức năng scrape. Hãy tạo một process khác và thêm file torrent.



Hình 14: Xem thông tin của torrent

7.5 Ngắt kết nối

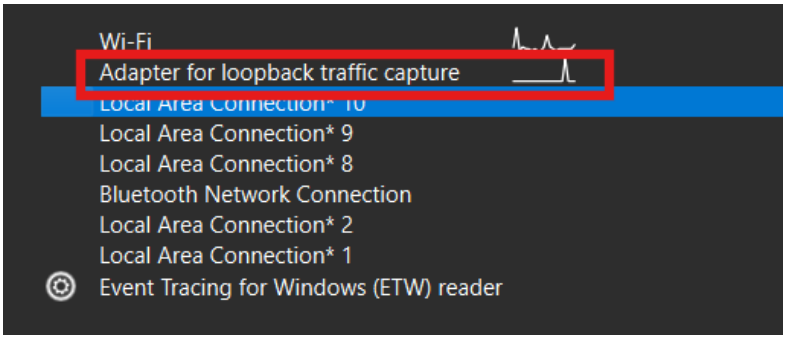
Sau khi đã tải xong bạn có thể ngừng tham gia vào mạng torrent này ở thu mục Peers chọn peer cần ngắt kết nối và chọn disconnect.



Hình 15: Ngừng tải

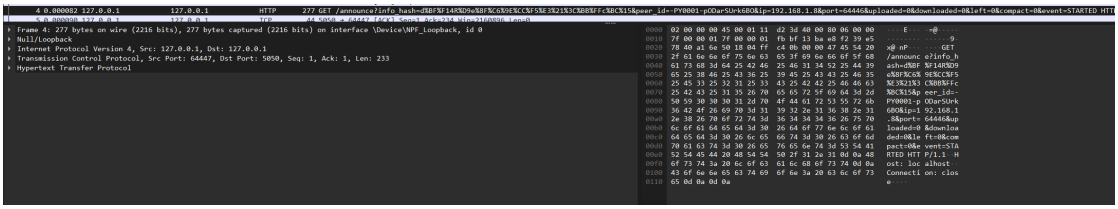
8 Bắt gói tin

Do ứng dụng chưa hỗ trợ chạy trên hai máy khác nhau (Vì chưa cấu hình được tường lửa) nên chúng ta sẽ thực hiện bắt gói bằng chức năng loopback trong wireshake để bắt các gói tin localhost.



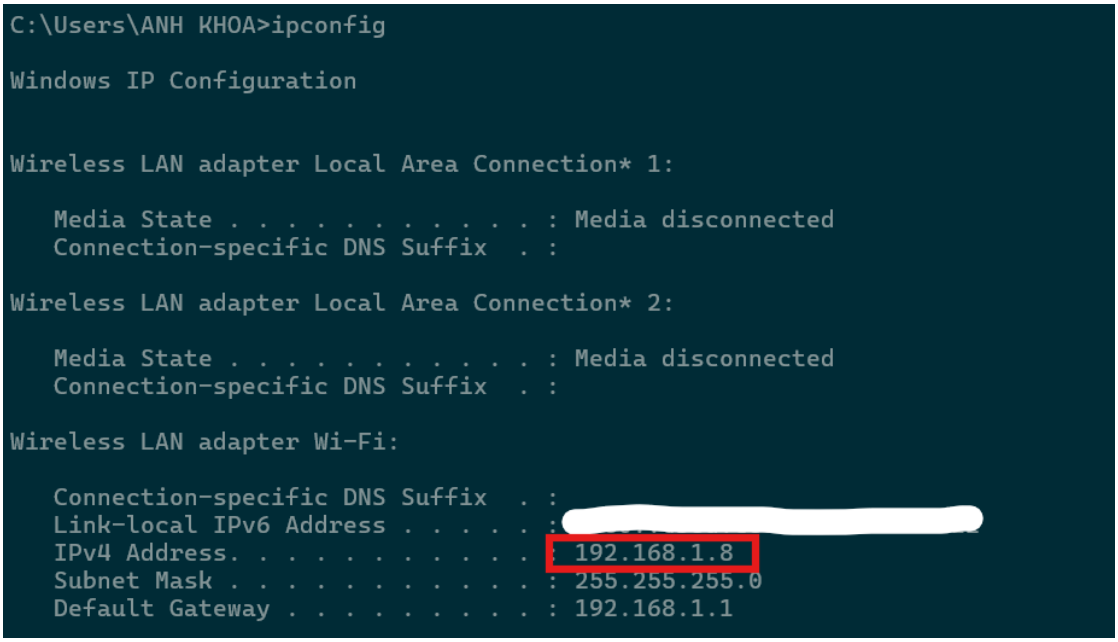
Hình 16: Chọn loopback traffic capture

Gói tin HTTP đầu tiên các bạn có thể thấy là gói thông báo bắt đầu tham gia vào mạng của người seeding.



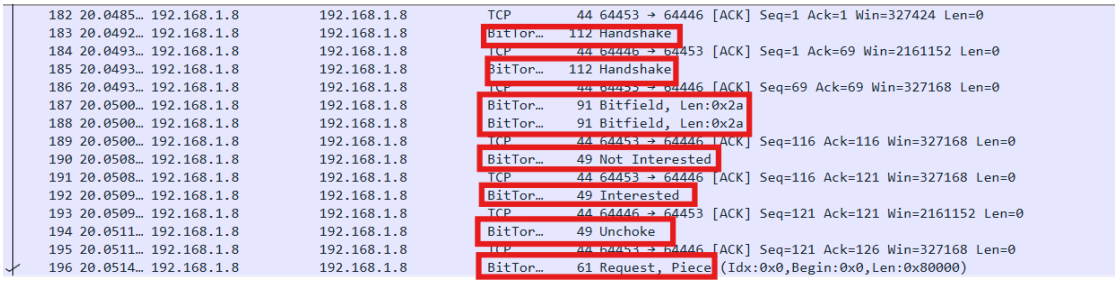
Hình 17: Thông báo bắt đầu cho server

Bạn hãy dùng lệnh ipconfig trên terminal để tìm địa chỉ IPv4 hiện tại



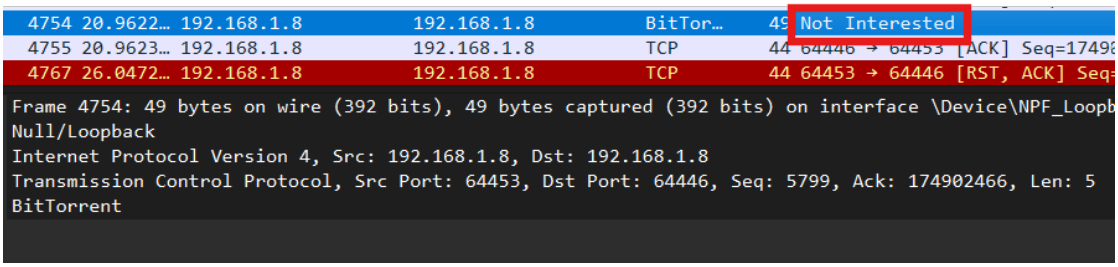
Hình 18: Ipconfig

Sau đó bạn hãy dùng filter ip.dst = 192.168.1.8 (địa chỉ IPv4 của bạn). Dưới đây là các protocol bittorent đã được giới thiệu ở phần trên. Các peer đồng loạt gửi Handshake cho nhau nếu thành công sẽ tiếp tục, thất bại sẽ hủy kết nối. Sau đó gửi Message BITFIELD để xác nhận các pieces. Sau khi xác nhận sẽ gửi thông báo Interested hoặc Not Interested. Nếu có peer Interested thì phía peer kia sẽ gửi thông báo Unchoke để bắt đầu gửi nhận Piece.



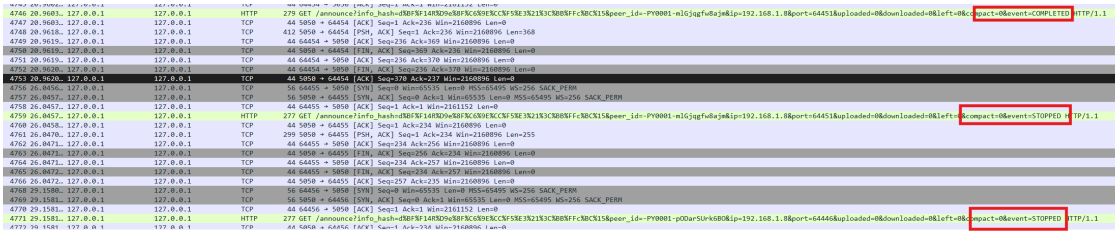
Hình 19: Peer Protocol

Sau khi kết thúc sẽ thông báo, cho Not Interested cho peer còn lại và gửi Complete cho Tracker server.



Hình 20: Not Intersested Message

Ba Message thông báo cho phía Tracker Server sau khi tải xong và ngắt kết nối từ 2 peer.



Hình 21: Thông báo tải xong và ngắt kết nối

9 Nguồn code

Mã nguồn code của nhóm tác giả được thực hiện bằng ngôn ngữ Python và được đăng tải lên Github với đường link: https://github.com/AK087204/CO3093-P2P-File-Transfer.