VIETNAM NATIONAL UNIVERSITY
BACH KHOA UNIVERSITY
COMPUTER SCIENCE AND ENGINEERING FACULTY

COMPUTER NETWORK

Project 1 REPORT

Develop a Network Application

Lecturer:	Diep Dang Thanh
Student:	Tran Quoc Trung - 2252859
	Tran Quoc Hieu - 2252217
	Nguyen Anh Khoa -

Ho Chi Minh city, November 2024

Contents

1 Overview about BitTorrent

1.1 BitTorrent File-sharing Application

In the traditional file-sharing application, there is only one host that contain all of the files and take responsibility for sending file across multiple end user. The drawback of this system is that the congestion in the sender side, cause the downloading rate decrease as the number of client increase.

BitTorrent is an architecture that allow the files source distribute and be transferred across multiple clients. When a client, or peer request for a file, it can download it from other peers, instead of directly from the owner, which avoid the congestion in sender side. Here is some features in BitTorrent application:

- **Tracker:** a server at the center of the BitTorrent Network, which responsible for connect multiple clients.
- **Swarm:** a group of clients who are interested in downloading/uploading a specific file.
- **Torrent file:** Each swarm has a ".torrent" file, which save the metadata info for the file, include list of **Pieces**, owner, size of files,... and the IP address of the tracker. A client must download and execute the Torrent file to join the swarm and start downloading the file.
- **Magnet link:** ...
- **Seeder/ Leacher:** ...
-

1.2 How BitTorrent work ?

Step 1: Client download the Torrent file or using magnet link to request for the download file to **tracker**.

Step 2: **Tracker** valid the request and send the list of peers in the corresponding swarm back to the client.

Step 3: Client contact to other peers in the swarm and exchange the list of available pieces.

Step 4: Client request pieces from other peers. In this step, we can design algorithms to setup strategy for client to connect with the other peers.

2 Overview about Torrent-like Application in this Project

2.1 Project architecture

```
|  client_1234:          # seeder client
|   |__textbook.txt
|   |__file_1.txt
|   |__...
|   |__torrents
|       |__torrent_textbook.json
|       |__torrent_file_1.json
|       |__torrent_file_2.json    # file belong to client_2217
|__client_1235
|__client_2217:
|   |__file_2.txt
|   |__torrents
|       |__torrent_file_2.json
|__data structure:          # Classes declaration directory
|   |__Connection.py
|   |__DownloadTask.py
|__server.py
|__client.py
```

2.1.1 Files and directories

- **server.py:** Save the source code to execute the tracker side. The PORT server is .... To run the server, we execute: *python server.py*
- **client.py:** Save the source code for the client side. To active the client, we execute: *python client.py <port_number>*, e.g: *python client.py* 2441.
- **client_<port_number>:** We regard each port as a single client and also has particular data directory to save the data files.
- **torrents:** Directory that store .torrent files.

2.1.2 Torrent file

The structure of .torrent file that we implementation:

```
{
    file_name: str,
    file_size: int,
    owner: Tuple(str, int),             # Ip address and port of owner.
    tracker: Tuple(str, int),           # Ip address and port of tracker server.
    number_of_pieces: int
}
```

Listing 1: .torrent file structure

2.2 File-sharing implementation

2.2.1 Swarm implementation

In this project, we save swarm metadata as dict object in tracker server, where the structure of swarm is:

```
{
    key: hash(torrent_data),                    # Hash value of torrent file
    value:
    {
        file_name: str,
        file_size: int,
        owner: Tuple(str, int),                 # Ip address and port of owner
        tracker: Tuple(str, int),               # Ip address and port of tracker server
        number_of_pieces: int,
        seeders: [
            Tuple(str, int),
        ]
    }
}
```

Listing 2: Swarm data in tracker

2.2.2 File-sharing process

Step 1: Client first create the .torrent file in their directory. They then send it to the server to create swarm.

Step 2: Server create swarm with the owner as the first seeder.

Step 3: The other clients get the .torrent file.

- The .torrent are sent between clients *(outside scope of the program, maybe through Messenger, Zalo or copy between directories)*.
- The client get list of available swarm directly from the tracker.

Step 4: The other client use the .torrent file to retrieve tracker IP address and send request for downloading file.

Step 5: The server valid the the request and send back the list of other peers in the swarm.

Step 6: The client contact with other peers in the swarm, ask for their pieces and their congestion status, i.e, number of seeding tasks that they are performing. Then the client pick one for sending data file.

Step 7: After complete downloading the file, the client trigger the server to be added into file seeders list.

2.3 Server Task

The server side, or tracker server, take responsibility for connecting the clients. Their main tasks that we implement in this project including:

- **Creating swarm:** After receiving *uploading request* with torrent data, the task of server is creating the swarm, as format in Section 2.2.1.
- **Retrieving and add new peer to swarm:** After receiving *downloading request*, server has to give back the list of clients in the swarm and add the requested client into the swarm.
- **Deleting offline peer from the swarm:** After a peer disconnecting, the server must remove it from all of the swarms.
- **Server command line interface**

3 Command Line Interface program

In this project, we build the command line interface where the main command interaction is carried in the Client side (See Section 3.2). Also, for managing

3.1 Server CLI

For managing reason, we implement server command line interface:

3.1.1 Show all swarms

- **Description:** Show all current swarms.
- **Example:**

```
server> show swarms
---------------------------------
Key: asdasdas55fas4f1a4s1f4as
File name: textbook.txt
Length: 513 B
Owner: Client ("localhost", 1234)
Number of pieces: 9
Seeders: [
    ("localhost", 1234)
    ("localhost", 1235)
]
---------------------------------
Key: bdabsyyxxx33525524sadsdt
File name: file_1.txt
Length: 25 B
Owner: Client ("localhost", 1234)
Number of pieces: 1
Seeders: [
    ("localhost", 1234)
    ("localhost", 2217)
]
---------------------------------
```

Listing 3: Show all swarms

3.1.2 Show all peers

- **Description:** Show all current conencted peers.
- **Example:**

```
server> show peers
---------------------------------
("localhost", 1234)
("localhost", 2217)
("localhost", 1235)
---------------------------------
```

Listing 4: Show all peers

3.2 Client CLI

This is where our main program happen. We implement the command line interface for the client. The list of command line including:

- create torrent <torrent_file_path> <server_ip> <server_port>
- upload torrent <torrent_file_path>
- download <torrent_file_path>
- skip <file_id>
- show progress
- show directory
- show swarms

3.2.1 Create torrent file

- **Description:** Create the torrent file and save to the directory torrents in data directory.
- **Example:**

```
client_1234> create torrent file_1.txt localhost 1232
---------------------------------
[SUCCESSFUL]
New torrent file "torrent_file_1.json" has been created to torrents directory !
---------------------------------
```

Listing 5: Upload swarm into the server

3.2.2 Upload torrent file

- **Description:** Upload the torrent file to the server and create the swarm.
- **Example:**

```
client_1234> upload torrent_file_1.json
--------------------------------
Upload "file_1.txt" to tracker server in ("localhost", 1230)...
Server: new swarm has been created !
Hash key: bdabsyyxxx33525524sadsdt
--------------------------------
```

Listing 6: Upload swarm into the server

3.2.3 Download file from server

- **Description:** Download the data file verified in the the torrent file
- **Example:**

```
client_1234> download torrent_file_2.json
--------------------------------
Server: found swarm bdabsyyxxx33525524sadsdt match.
List of seeders: [
    client_2217,
    client_1235
]
--------------------------------
Connect to seeder client_2217 !
Starting download from client_2217...
--------------------------------
```

Listing 7: Downloading task in the server

3.2.4 Skip downloading file

- **Description:** Skip the downloading process of a file.
- **Example:**

```
client_1234> skip 1
--------------------------------
Skip downloading file text_1.txt !
--------------------------------
```

Listing 8: Downloading task in the server

3.2.5 Show download progress

- **Description:** show all download progress.
- **Example:**

```
client_1234> show progress
--------------------------------
[1] text_2.txt - 0.04% - Skipping
[2] text_1.txt - 0.01% - Downloading
[3] textbook.txt - 100% - Commplete
--------------------------------
```

Listing 9: Showing all downloading progress

3.2.6 Show directory

- **Description:** show all file in the directory.
- **Example:**

```
client_1234> show directory
---------------------------------
-> file_1.txt
-> textbook.txt
-> torrents
---> torrent_file_1.json
---> torrent_file_2.json
---> torrent_textbook.json
---------------------------------
```

Listing 10: Showing all files in directory

3.2.7 Show all swarms

- **Description:** show all swarms that the client is joining.
- **Example:**

```
client_1234> show swarms
---------------------------------
Server: ("localhost", 1232)
Swarms:
[1] Key: bdabsyyxxx33525524sadsdt
    Value: [
        File name: text_1.txt
        Pieces: 1
    ]
---------------
[2] Key: asdasdas55fas4f1a4s1f4as
    Value: [
        File name: textbook.txt
        Pieces: 111111000
    ]
---------------------------------
```

Listing 11: Showing all swarms that the client has been joined

3.2.8 Quit

- **Description:** disconnecting all connections to the servers and terminate the process.
- **Example:**

```
client_1234> quit
---------------------------------
Disconnect to server ("localhost", 1232)...
The program terminate !
---------------------------------
```

Listing 12: Quit the process

4 Seeding algorithms

—— TODO HERE ——

5 Sample program

In this section, we run the program follow the steps in section 2.2.2. Assume that server side is running in localhost:1232.

5.1 File uploading process

```
client_1234> show directory
---------------------------------
-> file_1.txt
-> textbook.txt
-> torrents
---------------------------------

client_1234> create torrent file_2.txt localhost 1232
---------------------------------
[ERROR]
Not found file !
---------------------------------

client_1234> create torrent file_1.txt localhost 1232
---------------------------------
[SUCCESSFUL]
New torrent file "torrent_file_1.json" has been created to torrents directory !
---------------------------------

client_1234> upload torrent_file_1.json
---------------------------------
Upload "file_1.txt" to tracker server in ("localhost", 1232)...
Server: new swarm has been created !
Hash key: bdabsyyxxx33525524sadsdt
---------------------------------

client_1234> show swarms
---------------------------------
Server: ("localhost", 1232)
Swarms:
[1] Key: bdabsyyxxx33525524sadsdt
    Value: [
        File name: text_1.txt
        Pieces: 1
    ]
---------------------------------
```

Listing 13: Client 1234 terminal

5.2 File downloading process

Assume that the client 1235 has the torrent file in it's directory.

```
client_1235> show directory
---------------------------------
-> torrents
---> torrent_file_1.json
---------------------------------

client_1235> download torrent_file_1.json
---------------------------------
Server: found swarm bdabsyyxxx33525524sadsdt match.
List of seeders: [
    client_1234
]
---------------------------------
Connect to seeder client_1234 !
Starting download from client_1234...
---------------------------------

client_1235> show progress
---------------------------------
[1] text_1.txt - 22.2% - Downloading
---------------------------------

client_1235> show swarms
---------------------------------
Server: ("localhost", 1232)
Swarms:
[1] Key: bdabsyyxxx33525524sadsdt
    Value: [
        File name: text_1.txt
        Pieces: 110000000
    ]
---------------------------------
```
Listing 14: Client 1234 terminal