VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

**COMPUTER NETWORKS (CO3093)**

---

**Assignment**

# DEVELOP A NETWORK APPLICATION

---

Advisor(s): Nguyễn Lê Duy Lai, *CSE-HCMUT*

Students: Nguyễn Văn Đức Long - 2153535
Trần Nguyễn Gia Huy - 2153395
Lê Dương Khánh Huy - 2153380

HO CHI MINH CITY, AUGUST 2023

# Contents

## General Information

### Acknowledgement

First of all, we would like to acknowledge and give our warmest regards to our main instructor **Nguyễn Lê Duy Lai** , who has given us a lot of advice and provided us with many instructions on the subject Computer Networks.

Secondly, we would also like to thank our team members who have worked so hard on this project. Everyone has made enjoyable moment, brilliant comments and suggestions. Throughout the project, we have learned how to collaborate effectively between team members to solve many problems.

## 1 Members' Contribution

| **No.** | **Fullname** | **Student ID** | **Problems** | **Contribution** |
|---|---|---|---|---|
| 1 | Nguyễn Văn Đức Long | 2153535 | - Implement tracker-peer functionalities<br>- Sockets handling | 100% |
| 2 | Trần Nguyễn Gia Huy | 2153395 | - Implement peer functionality<br>- Establish connection of peer to peer | 100% |
| 3 | Lê Dương Khánh Huy | 2153380 | - Implement peer-peer download/upload<br>- Merge and split files into chunks | 100% |

## 2 Introduction

In the contemporary landscape of digital connectivity, where the exchange of data is ingrained in everyday practices, the importance of reliable file-sharing tools cannot be overstated. Recognizing this significance, our group embarked on a project aimed at developing a BitTorrent-like application

This report encapsulates our journey from inception to execution, offering a comprehensive narrative of our endeavors. It delineates the ideation phase, the development process, and the rigorous testing that ensued.

At the core of our application lies a centralized server, functioning as the nexus that facilitates user connectivity and orchestrates file transfers. A fundamental aspect of our approach involves the utilization of a specialized protocol, enabling clients to communicate their available files to the server without transmitting the files themselves.

When a user seeks a particular file, they interface with other peers via a given Peer Set. This streamlined process ensures efficient and secure file exchanges within our network.

Our client software is engineered to adeptly manage multiple downloads concurrently, thereby enhancing overall efficiency and user experience.

Throughout the development lifecycle, meticulous attention was devoted to optimizing user-friendliness and performance. This report delves into the intricate technical details of our BitTorrent-like application and elucidates the valuable insights garnered during its conceptualization and implementation.

## 3 Use-case flow of a new peer

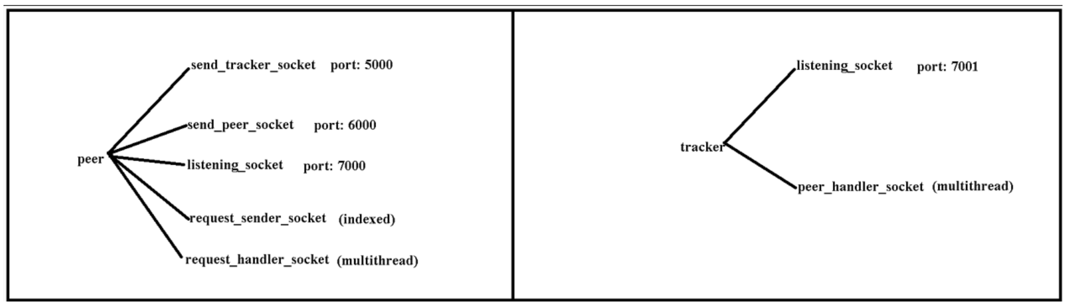
- Initially, a peer that wishes to download specific files in a torrent must connect to the tracker via the information provided in the Metainfo File (torrent file). This file contains useful information about the tracker, such as its IP address and port number. Subsequently, the peer connects to the tracker to obtain the peer set. The peer's information is then updated in the peer set on the tracker, and the tracker sends the peer set to the peer. Following this, the peer examines the peer set and decides which peers to connect to in order to retrieve the missing chunks.

- The peer then establishes connections to multiple peers in the peer set that have its missing chunks to request downloading the missing chunk. All the connected peers will check if they have the missing chunk, and then the uploading procedure will be executed. A peer can request download to multiple peers concurrently, and a peer can also upload its chunks to many other peers at the same time.

- Finally, after downloading all the chunks of a file, the peer merges those chunks together into files for personal use. The current peer can then either become a seeder or leave the torrent.

## 4 Socket Diagram

### 4.1 Overview

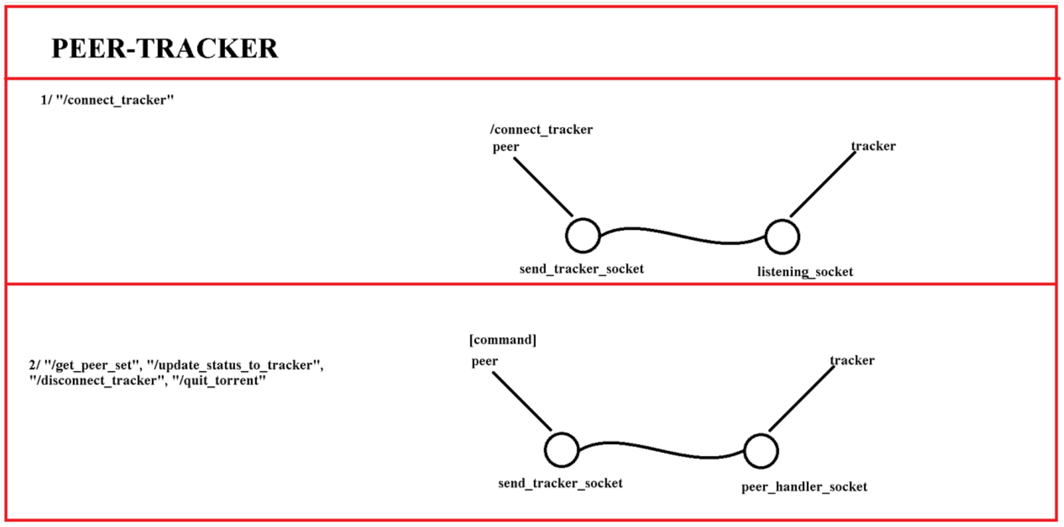

peer
send_tracker_socket port: 5000
send_peer_socket port: 6000
listening_socket port: 7000
request_sender_socket (indexed)
request_handler_socket (multithread)
tracker
listening_socket port: 7001
peer_handler_socket (multithread)

Hình 1: All sockets and their port numbers

Where:

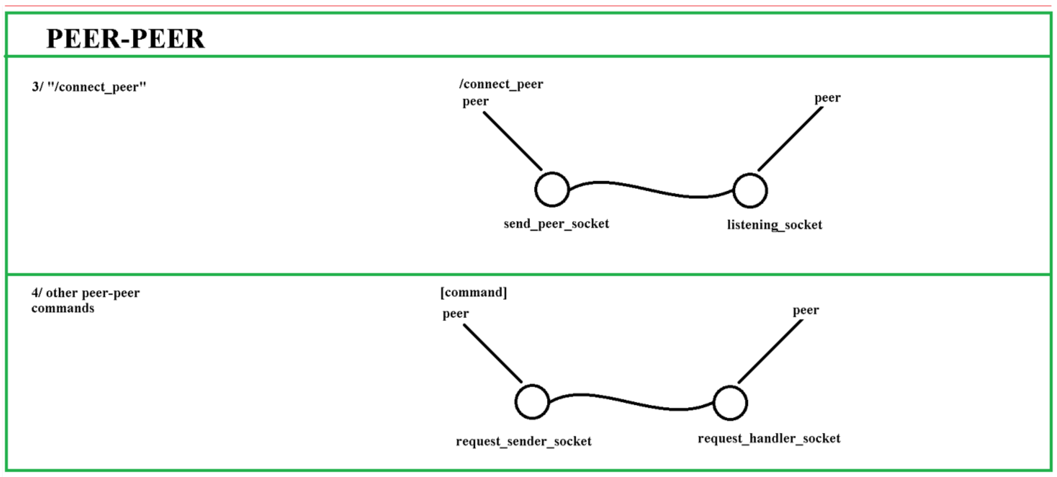
- **peer sockets:**
  - **send_tracker_socket** (5000): Peer sends commands to tracker via this socket. Peer also receive tracker's response via the same socket.
  - **send_peer_socket** (6000): this socket only handle "/connect_peer" command of current peer to other peers and receive response of the command from other peers via this socket. After that, the "request_sender_socket" is used to send other commands to a specific connected peer.
  - **listening_socket** (7000): this socket is only used to listen to any "/connect_peer" commands from other peers and respond to them. The "request_handler_socket" is used to handle specific commands of specific peers after that.
  - **request_sender_socket** (indexed): These sockets are used to send specific commands to specific peers. A peer can have as many "request_sender_socket" as how much peers it has established connection to. Each of "request_sender_socket" is indexed according to target peer's IP and listening_socket port number.
  - **request_handler_socket** (multi-threaded): These sockets are used to handle received commands from other peers's "request_sender_socket". A "request_handler_socket" is created after a thread was created to handle commands when the current peer's "listening_socket" receive a "/connect_peer" command from an other peer.
- **tracker sockets:**
  - **listening_socket** (7001): this socket is only used to listen to any "/connect_tracker" commands from other peers and respond to them. The "peer_handler_socket" is used to handle specific commands of specific peers after that.
  - **peer_handler_socker** (multi-threaded): These sockets are used to handle received commands from other peers's "send_tracker_socket". A "peer_handler_socket" is created after a thread was created to handle commands when the tracker's "listening_socket" receive a "/connect_tracker" command from a peer.

### 4.2 Peer-Tracker socket diagram



Hình 2: Peer-Tracker socket diagram

### 4.3 Peer-Peer socket diagram



Hình 3: Peer-Peer socket diagram

# 5 Function description

## 5.1 Peer-Tracker communication commands

### 5.1.1 /connect_tracker

#### 5.1.1.a Command syntax

**/connect_traker**

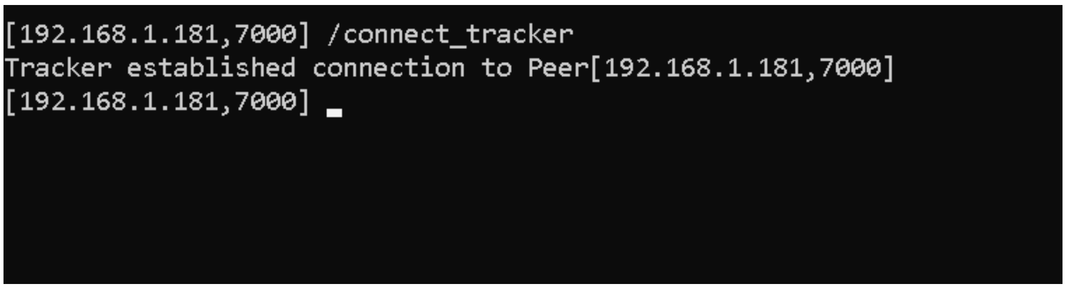
#### 5.1.1.b Explanation

1. This command helps the peer establishs connection (using TCP protocol) to tracker, then send the tracker its information including IP address, port number, current chunks, missing chunks.

2. If connecting successfully, the tracker response: "Tracker established connection to Peer[peer_ip]" (tracker creates a request_handler_socket in a thread to handle the upcoming commands from this peer)

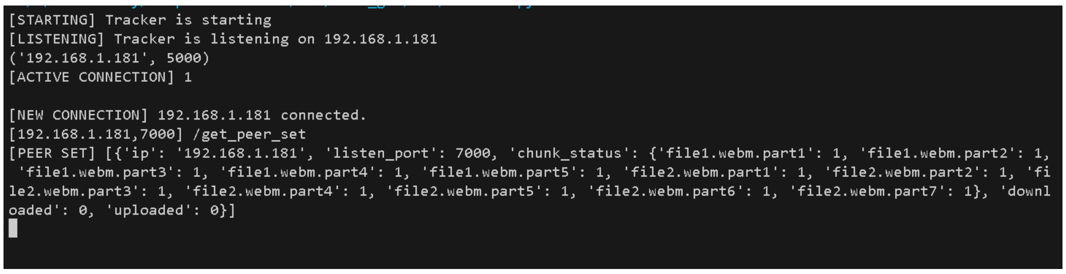
3. The tracker then sends the peer_set to the peer.

#### 5.1.1.c Demonstration



```
[192.168.1.181,7000] /connect_tracker
Tracker established connection to Peer[192.168.1.181,7000]
[192.168.1.181,7000] _
```

Hình 4: /connect_tracker Demo



```
[STARTING] Tracker is starting
[LISTENING] Tracker is listening on 192.168.1.181
('192.168.1.181', 5000)
[ACTIVE CONNECTION] 1

[NEW CONNECTION] 192.168.1.181 connected.
[192.168.1.181,7000] /get_peer_set
[PEER SET] [{'ip': '192.168.1.181', 'listen_port': 7000, 'chunk_status': {'file1.webm.part1': 1, 'file1.webm.part2': 1,
 'file1.webm.part3': 1, 'file1.webm.part4': 1, 'file1.webm.part5': 1, 'file2.webm.part1': 1, 'file2.webm.part2': 1, 'fi
le2.webm.part3': 1, 'file2.webm.part4': 1, 'file2.webm.part5': 1, 'file2.webm.part6': 1, 'file2.webm.part7': 1}, 'downl
oaded': 0, 'uploaded': 0}]
```

Hình 5: /connect_tracker Demo, tracker's POV

### 5.1.2 /get_peer_set

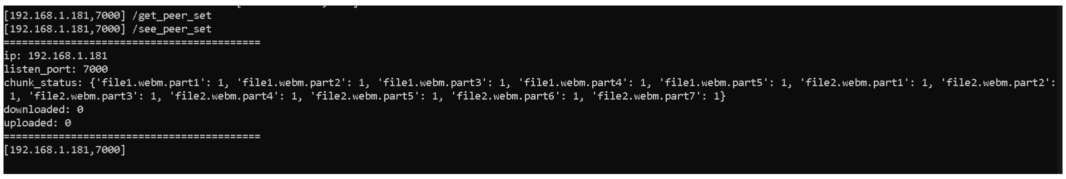
#### 5.1.2.a Command syntax

**/get_peer_set**

#### 5.1.2.b Explanation

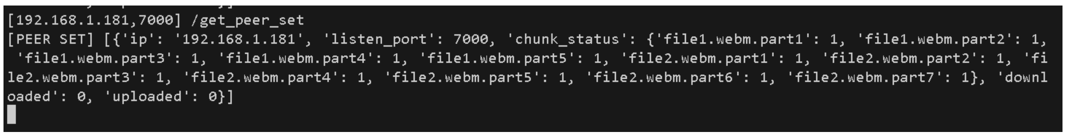
1. This command will request tracker to get the list of information of peers in the torrent. It can help us track the coming and leaving peer in our torrent.
2. This command always run whenever a new peer connects to the tracker.

#### 5.1.2.c Demonstration



```
[192.168.1.181,7000] /get_peer_set
[192.168.1.181,7000] /see_peer_set
========================================
ip: 192.168.1.181
listen_port: 7000
chunk_status: {'file1.webm.part1': 1, 'file1.webm.part2': 1, 'file1.webm.part3': 1, 'file1.webm.part4': 1, 'file1.webm.part5': 1, 'file2.webm.part1': 1, 'file2.webm.part2':
 1, 'file2.webm.part3': 1, 'file2.webm.part4': 1, 'file2.webm.part5': 1, 'file2.webm.part6': 1, 'file2.webm.part7': 1}
downloaded: 0
uploaded: 0
========================================
[192.168.1.181,7000]
```

Hình 6: /get_peer_set Demo



```
[192.168.1.181,7000] /get_peer_set
[PEER SET] [{'ip': '192.168.1.181', 'listen_port': 7000, 'chunk_status': {'file1.webm.part1': 1, 'file1.webm.part2': 1,
 'file1.webm.part3': 1, 'file1.webm.part4': 1, 'file1.webm.part5': 1, 'file2.webm.part1': 1, 'file2.webm.part2': 1, 'fi
le2.webm.part3': 1, 'file2.webm.part4': 1, 'file2.webm.part5': 1, 'file2.webm.part6': 1, 'file2.webm.part7': 1}, 'downl
oaded': 0, 'uploaded': 0}]
```

Hình 7: /get_peer_set Demo, tracker's POV

#### 5.1.3 /update_status_to_tracker

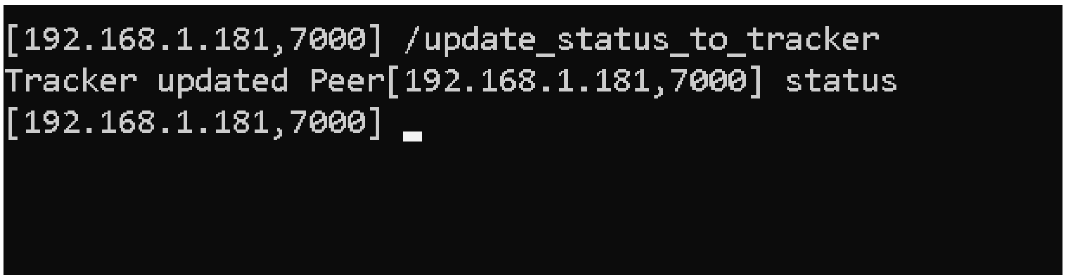
##### 5.1.3.a Command syntax

**/update_status_to_tracker**

##### 5.1.3.b Explanation

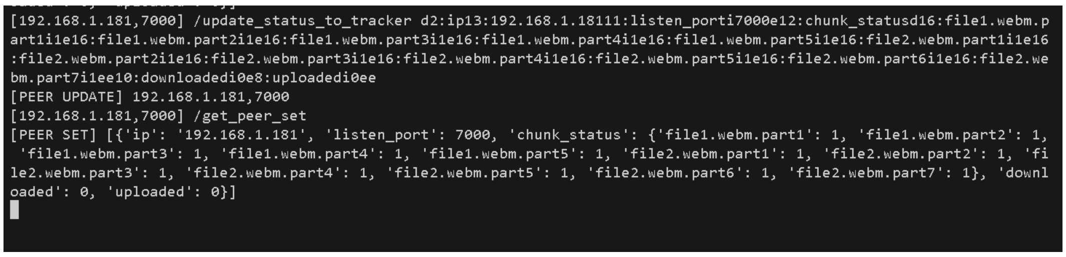
1. This command updates the status of this peer to the tracker.
2. This command runs automatically and periodically in each peers

##### 5.1.3.c Demonstration



```
[192.168.1.181,7000] /update_status_to_tracker
Tracker updated Peer[192.168.1.181,7000] status
[192.168.1.181,7000]
```

Hình 8: /update_status_to_tracker Demo



```
[192.168.1.181,7000] /update_status_to_tracker d2:ip13:192.168.1.18111:listen_porti7000e12:chunk_statusd16:file1.webm.p
art1i1e16:file1.webm.part2i1e16:file1.webm.part3i1e16:file1.webm.part4i1e16:file1.webm.part5i1e16:file2.webm.part1i1e16
:file2.webm.part2i1e16:file2.webm.part3i1e16:file2.webm.part4i1e16:file2.webm.part5i1e16:file2.webm.part6i1e16:file2.we
bm.part7i1ee10:downloadedi0e8:uploadedi0ee
[PEER UPDATE] 192.168.1.181,7000
[192.168.1.181,7000] /get_peer_set
[PEER SET] [{'ip': '192.168.1.181', 'listen_port': 7000, 'chunk_status': {'file1.webm.part1': 1, 'file1.webm.part2': 1,
 'file1.webm.part3': 1, 'file1.webm.part4': 1, 'file1.webm.part5': 1, 'file2.webm.part1': 1, 'file2.webm.part2': 1, 'fi
le2.webm.part3': 1, 'file2.webm.part4': 1, 'file2.webm.part5': 1, 'file2.webm.part6': 1, 'file2.webm.part7': 1}, 'downl
oaded': 0, 'uploaded': 0}]
```

Hình 9: /update_status_to_tracker Demo, tracker's POV

#### 5.1.4 /disconnect_tracker

##### 5.1.4.a Command syntax

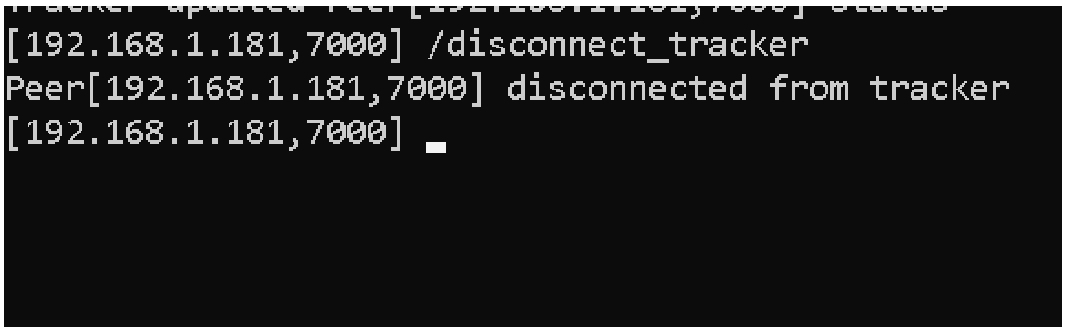
**/disconnect_tracker**

##### 5.1.4.b Explanation

1. Basically, this command can be sent from current peer to disconnect with tracker( not necessarily quit the torrent, this peer still exchange file with other peers).
2. Tracker will response the quitting message by printing on the screen an announce like "Peer[peer_ip,peer_port] disconnected from tracker"

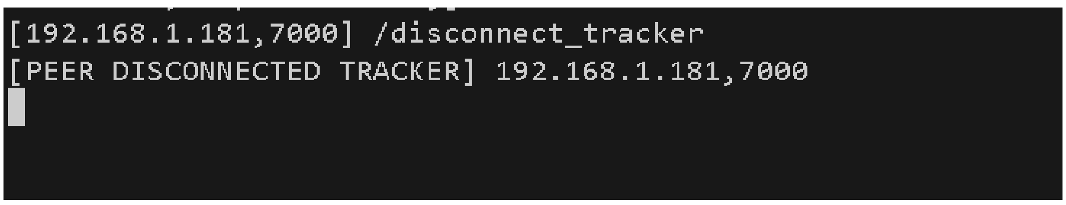
Note: peer disconnect from tracker but can still upload/download chunks to/from other peers ( tracker stops "listening peer cmds" thread)

##### 5.1.4.c Demonstration



```
[192.168.1.181,7000] /disconnect_tracker
Peer[192.168.1.181,7000] disconnected from tracker
[192.168.1.181,7000]
```

Hình 10: /disconnect_tracker Demo



```
[192.168.1.181,7000] /disconnect_tracker
[PEER DISCONNECTED TRACKER] 192.168.1.181,7000
```

Hình 11: /disconnect_tracker Demo, tracker's POV

#### 5.1.5 /quit_torrent

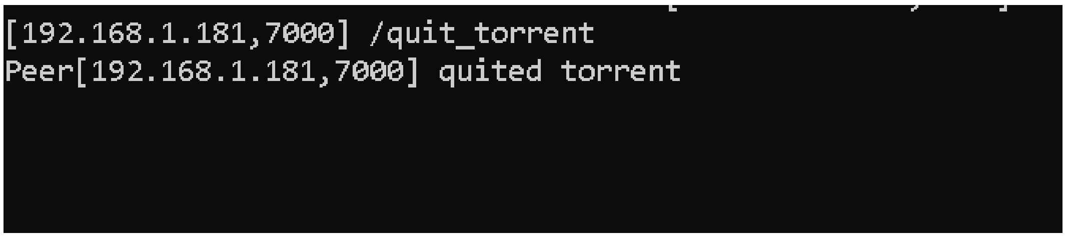
##### 5.1.5.a Command syntax

**/quit_torrent**

##### 5.1.5.b Explanation

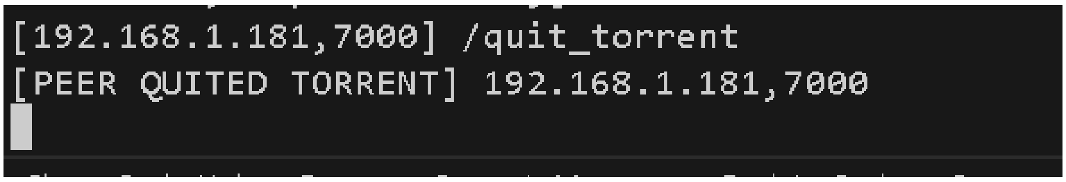
If a peer send this command to tracker, it mean that peer actively leave the torrent. That peer can not upload, download or any interation with the tracker and other peer.

##### 5.1.5.c Demonstration



```
[192.168.1.181,7000] /quit_torrent
Peer[192.168.1.181,7000] quited torrent
```

Hình 12: /quit_torrent Demo



```
[192.168.1.181,7000] /quit_torrent
[PEER QUITED TORRENT] 192.168.1.181,7000
```

Hình 13: /quit_torrent Demo, tracker's POV

## 5.2 Peer-Peer communication commands

### 5.2.1 /connect_peer

#### 5.2.1.a Command syntax

**/connect_peer [target_peer_ip] [target_peer_port]**

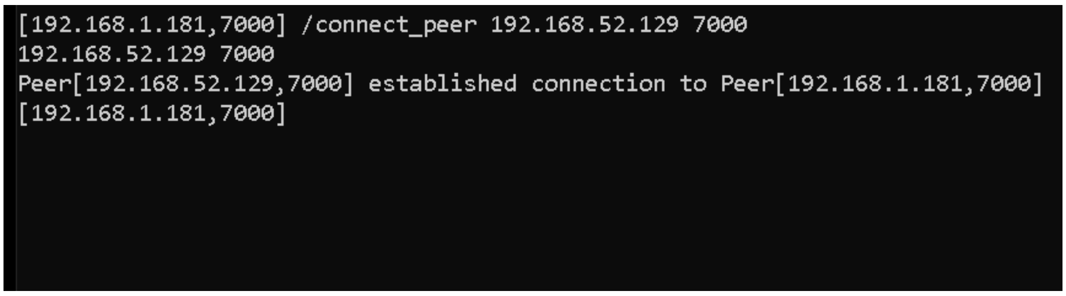
Where:

- [**target_peer_ip**]: IP address of the peer we want to connect to
- [**target_peer_port**]: Listening Port of the peer we want to connect to

#### 5.2.1.b Explanation

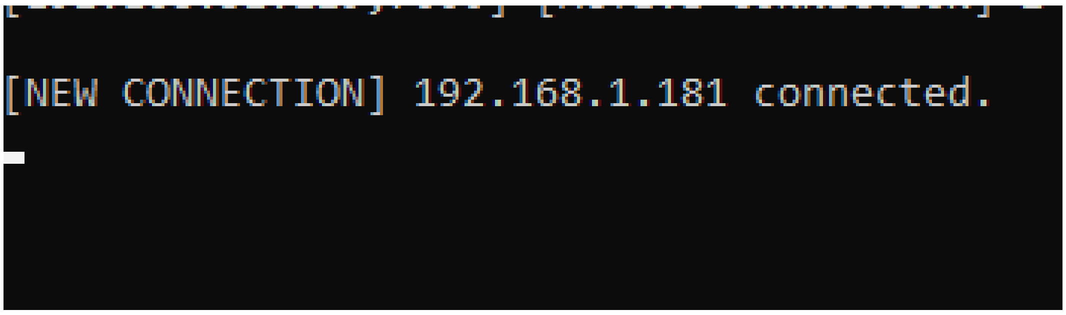
Like connecting tracker, this command use to establish connection between 2 peers by using TCP protocol. Connecting peers together is the very first step to help them trasnfer file in our torrent.

#### 5.2.1.c Demonstration



```
[192.168.1.181,7000] /connect_peer 192.168.52.129 7000
192.168.52.129 7000
Peer[192.168.52.129,7000] established connection to Peer[192.168.1.181,7000]
[192.168.1.181,7000]
```

Hình 14: /connect_peer Demo



```
[NEW CONNECTION] 192.168.1.181 connected.
```

Hình 15: /connect_peer Demo, target peer's POV

### 5.2.2 /ping

#### 5.2.2.a Command syntax

**/ping [target_peer_ip] [target_peer_port]**

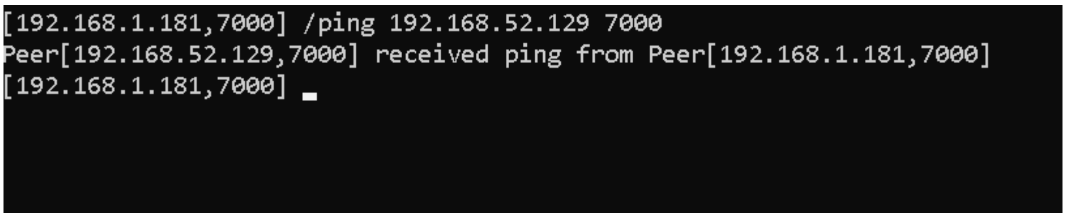
Where:

- **[target_peer_ip]**: IP address of the peer we want to connect to
- **[target_peer_port]**: Listening Port of the peer we want to connect to

#### 5.2.2.b Explanation

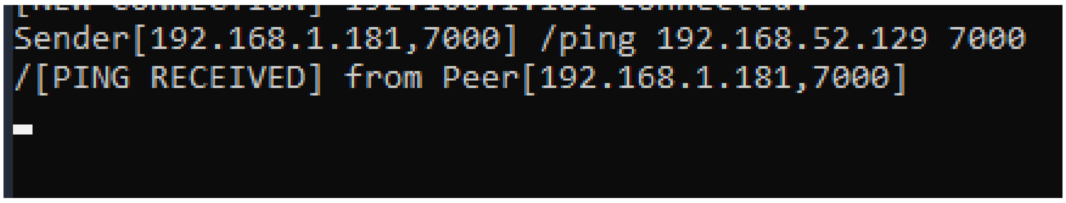
A peer that runs this command will send a dummy packet to its target. And if there is a connection between them, a message is received from the target. This command is used to check if there is a connection between the peers

#### 5.2.2.c Demonstration



```
[192.168.1.181,7000] /ping 192.168.52.129 7000
Peer[192.168.52.129,7000] received ping from Peer[192.168.1.181,7000]
[192.168.1.181,7000] _
```

Hình 16: /ping Demo



```
Sender[192.168.1.181,7000] /ping 192.168.52.129 7000
/[PING RECEIVED] from Peer[192.168.1.181,7000]
_
```

Hình 17: /ping Demo, target peer's POV

### 5.2.3 /request_download

#### 5.2.3.a Command syntax

**/download_request [target_peer_ip] [target_peer_port] [missing_chunk]**

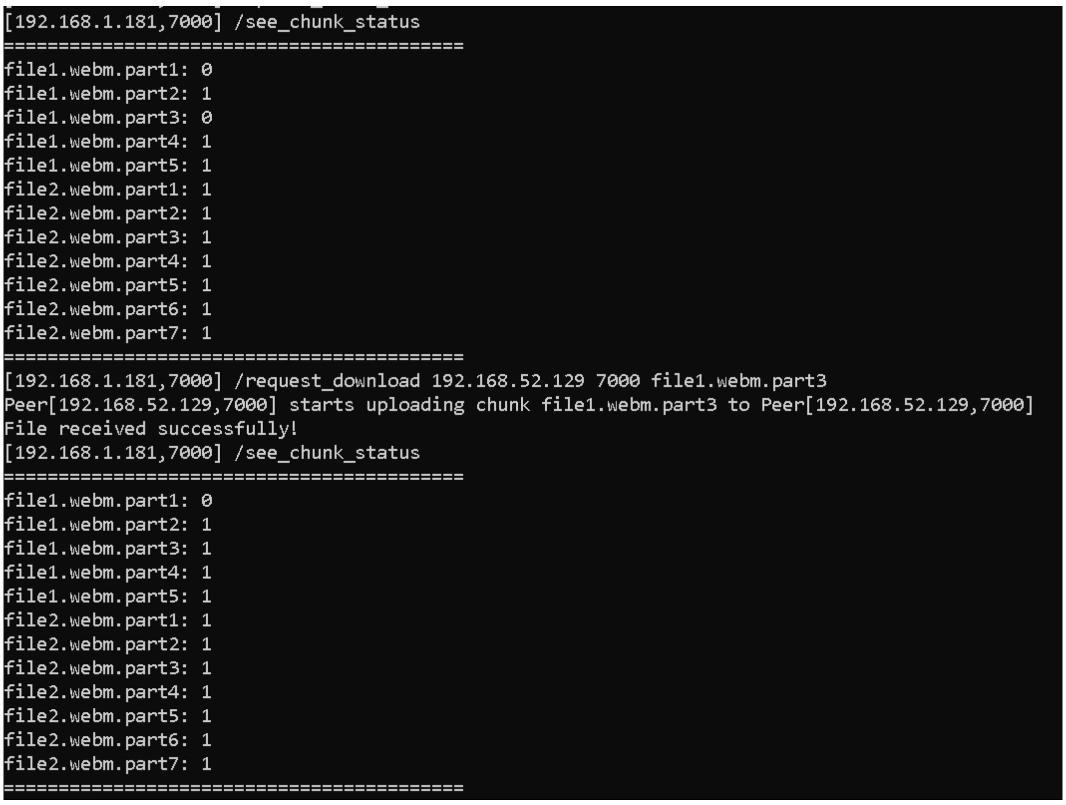
Where:

- [**target_peer_ip**]: IP address of the peer we want to connect to
- [**target_peer_port**]: Listening Port of the peer we want to connect to
- [**missing_chunk**]: name of the chunk the peer wants to download.

#### 5.2.3.b Explanation

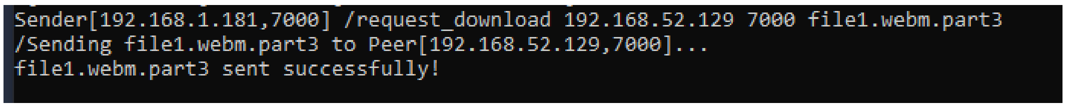
- This is a main functionality of our system. The peer must satisfy the previous step, which is to connect to the target peer before requesting download.
- The file is split into small chunk, each chunk has a size of 512KB.
- Each peer has its own folder (representing as a memory) to store its chunks. If the download request is received and handled successfully, the missing chunks will be sent from target peer to request peer
- A peer can send download request and upload chunks to other peers concurrently.

#### 5.2.3.c Demonstration



```
[192.168.1.181,7000] /see_chunk_status
==========================================
file1.webm.part1: 0
file1.webm.part2: 1
file1.webm.part3: 0
file1.webm.part4: 1
file1.webm.part5: 1
file2.webm.part1: 1
file2.webm.part2: 1
file2.webm.part3: 1
file2.webm.part4: 1
file2.webm.part5: 1
file2.webm.part6: 1
file2.webm.part7: 1
==========================================
[192.168.1.181,7000] /request_download 192.168.52.129 7000 file1.webm.part3
Peer[192.168.52.129,7000] starts uploading chunk file1.webm.part3 to Peer[192.168.52.129,7000]
File received successfully!
[192.168.1.181,7000] /see_chunk_status
==========================================
file1.webm.part1: 0
file1.webm.part2: 1
file1.webm.part3: 1
file1.webm.part4: 1
file1.webm.part5: 1
file2.webm.part1: 1
file2.webm.part2: 1
file2.webm.part3: 1
file2.webm.part4: 1
file2.webm.part5: 1
file2.webm.part6: 1
file2.webm.part7: 1
==========================================
```

Hình 18: /request_download Demo



```
Sender[192.168.1.181,7000] /request_download 192.168.52.129 7000 file1.webm.part3
/Sending file1.webm.part3 to Peer[192.168.52.129,7000]...
file1.webm.part3 sent successfully!
```

Hình 19: /request_download Demo, target peer's POV

### 5.2.4 /disconnect_peer

#### 5.2.4.a Command syntax

**/disconnect_peer [target_peer_ip] [target_peer_port]**

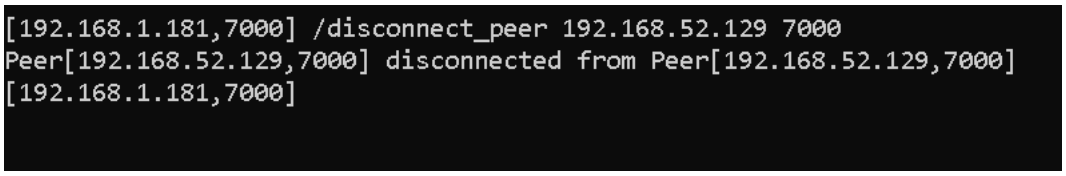
Where:

- [**target_peer_ip**]: IP address of the peer we want to disconnect
- [**target_peer_port**]: Listening Port of the peer we want to disconnect

#### 5.2.4.b Explanation

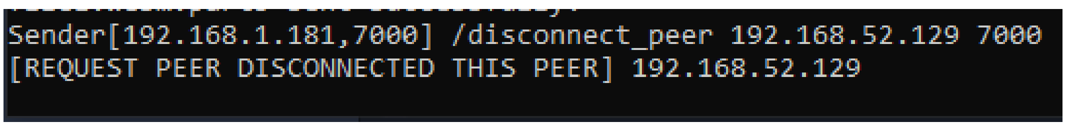
This command will terminate the connection between the current peer and the target peer. The current peer no longer can send or receive any messages or data to/from the target peer.

#### 5.2.4.c Demonstration



```
[192.168.1.181,7000] /disconnect_peer 192.168.52.129 7000
Peer[192.168.52.129,7000] disconnected from Peer[192.168.52.129,7000]
[192.168.1.181,7000]
```

Hình 20: /disconnect_peer Demo



```
Sender[192.168.1.181,7000] /disconnect_peer 192.168.52.129 7000
[REQUEST PEER DISCONNECTED THIS PEER] 192.168.52.129
```

Hình 21: /disconnect_peer Demo, target peer's POV

## 5.3 Additional commands

### 5.3.1 /check_tracker_connected

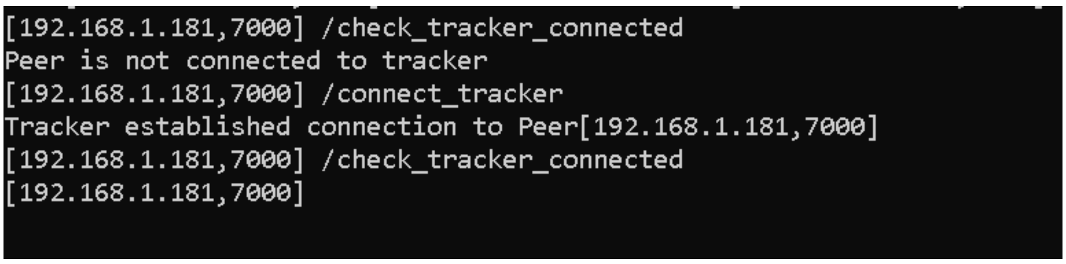
#### 5.3.1.a Command syntax

**/check_tracker_connected**

#### 5.3.1.b Explanation

- This function help the peer check whether it connect the tracker or not. Connecting with the tracker is a prerequisite to process any further commands.

#### 5.3.1.c Demonstration



```
[192.168.1.181,7000] /check_tracker_connected
Peer is not connected to tracker
[192.168.1.181,7000] /connect_tracker
Tracker established connection to Peer[192.168.1.181,7000]
[192.168.1.181,7000] /check_tracker_connected
[192.168.1.181,7000]
```

Hình 22: /check_tracker_connected Demo

### 5.3.2 /check_target_peer_connected

#### 5.3.2.a Command syntax

**/check_target_peer_connected [target_peer_ip] [target_peer_port]**

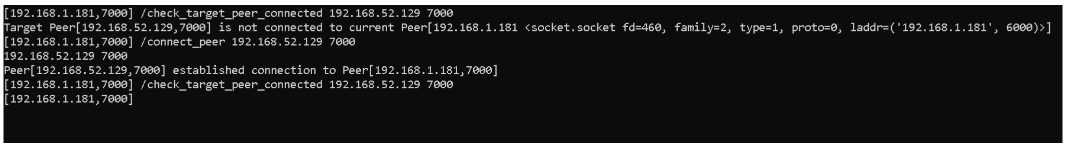
Where:

- [**target_peer_ip**]: IP address of the peer we want to check connection
- [**target_peer_port**]: Listening Port of the peer we want to check connection

#### 5.3.2.b Explanation

This function help the peer check whether it connect successfully with other peers or not

#### 5.3.2.c Demonstration



```
[192.168.1.181,7000] /check_target_peer_connected 192.168.52.129 7000
Target Peer[192.168.52.129,7000] is not connected to current Peer[192.168.1.181 <socket.socket fd=460, family=2, type=1, proto=0, laddr=('192.168.1.181', 6000)>]
[192.168.1.181,7000] /connect_peer 192.168.52.129 7000
192.168.52.129 7000
Peer[192.168.52.129,7000] established connection to Peer[192.168.1.181,7000]
[192.168.1.181,7000] /check_target_peer_connected 192.168.52.129 7000
[192.168.1.181,7000]
```

Hình 23: /check_target_peer_connected Demo

#### 5.3.3 /see_this_peer_info

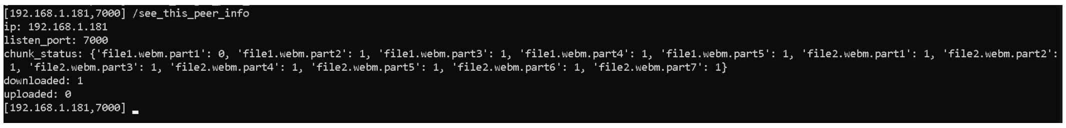
##### 5.3.3.a Command syntax

**/see_this_peer_info**

##### 5.3.3.b Explanation

- This command will show the peer information including its IP address, port number, current chunks, missing chunks.

##### 5.3.3.c Demonstration



```
[192.168.1.181,7000] /see_this_peer_info
ip: 192.168.1.181
listen_port: 7000
chunk_status: {'file1.webm.part1': 0, 'file1.webm.part2': 1, 'file1.webm.part3': 1, 'file1.webm.part4': 1, 'file1.webm.part5': 1, 'file2.webm.part1': 1, 'file2.webm.part2':
 1, 'file2.webm.part3': 1, 'file2.webm.part4': 1, 'file2.webm.part5': 1, 'file2.webm.part6': 1, 'file2.webm.part7': 1}
downloaded: 1
uploaded: 0
[192.168.1.181,7000] _
```

Hình 24: /see_this_peer_info Demo

#### 5.3.4 /see_peer_set

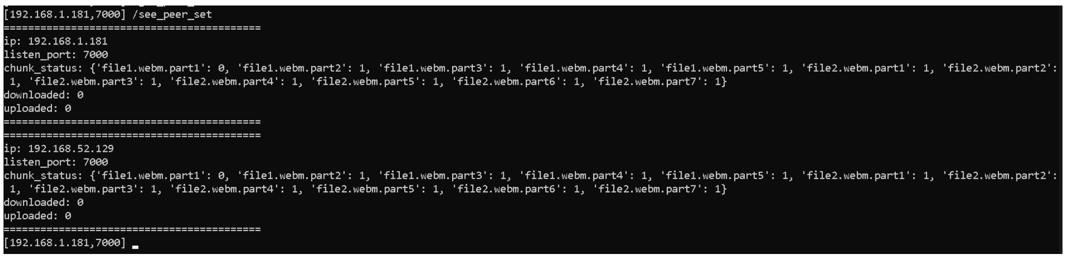
##### 5.3.4.a Command syntax

**/see_peer_set**

##### 5.3.4.b Explanation

- This function will show a list of peers existing in our torrent.

##### 5.3.4.c Demonstration



```
[192.168.1.181,7000] /see_peer_set
==========================================
ip: 192.168.1.181
listen_port: 7000
chunk_status: {'file1.webm.part1': 0, 'file1.webm.part2': 1, 'file1.webm.part3': 1, 'file1.webm.part4': 1, 'file1.webm.part5': 1, 'file2.webm.part1': 1, 'file2.webm.part2':
 1, 'file2.webm.part3': 1, 'file2.webm.part4': 1, 'file2.webm.part5': 1, 'file2.webm.part6': 1, 'file2.webm.part7': 1}
downloaded: 0
uploaded: 0
==========================================
==========================================
ip: 192.168.52.129
listen_port: 7000
chunk_status: {'file1.webm.part1': 0, 'file1.webm.part2': 1, 'file1.webm.part3': 1, 'file1.webm.part4': 1, 'file1.webm.part5': 1, 'file2.webm.part1': 1, 'file2.webm.part2':
 1, 'file2.webm.part3': 1, 'file2.webm.part4': 1, 'file2.webm.part5': 1, 'file2.webm.part6': 1, 'file2.webm.part7': 1}
downloaded: 0
uploaded: 0
==========================================
[192.168.1.181,7000] _
```

Hình 25: /see_peer_set Demo

#### 5.3.5 /see_connected

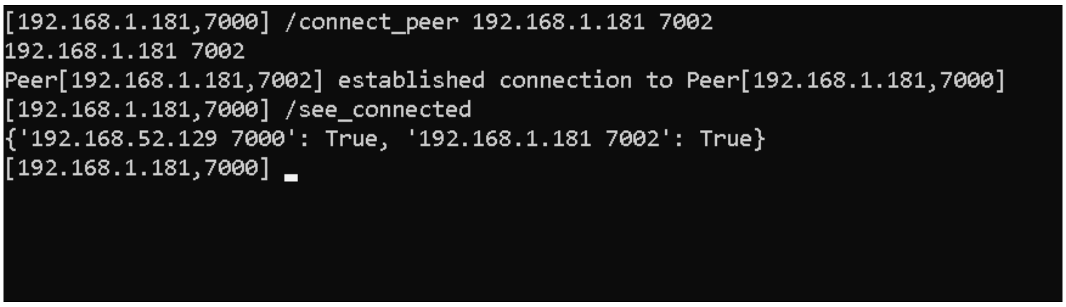
##### 5.3.5.a Command syntax

**/see_connected**

##### 5.3.5.b Explanation

- The command will print out all the connection status between the current peer and other peers

##### 5.3.5.c Demonstration



```
[192.168.1.181,7000] /connect_peer 192.168.1.181 7002
192.168.1.181 7002
Peer[192.168.1.181,7002] established connection to Peer[192.168.1.181,7000]
[192.168.1.181,7000] /see_connected
{'192.168.52.129 7000': True, '192.168.1.181 7002': True}
[192.168.1.181,7000] _
```

Hình 26: /see_connected Demo

### 5.3.6 /see_chunk_status

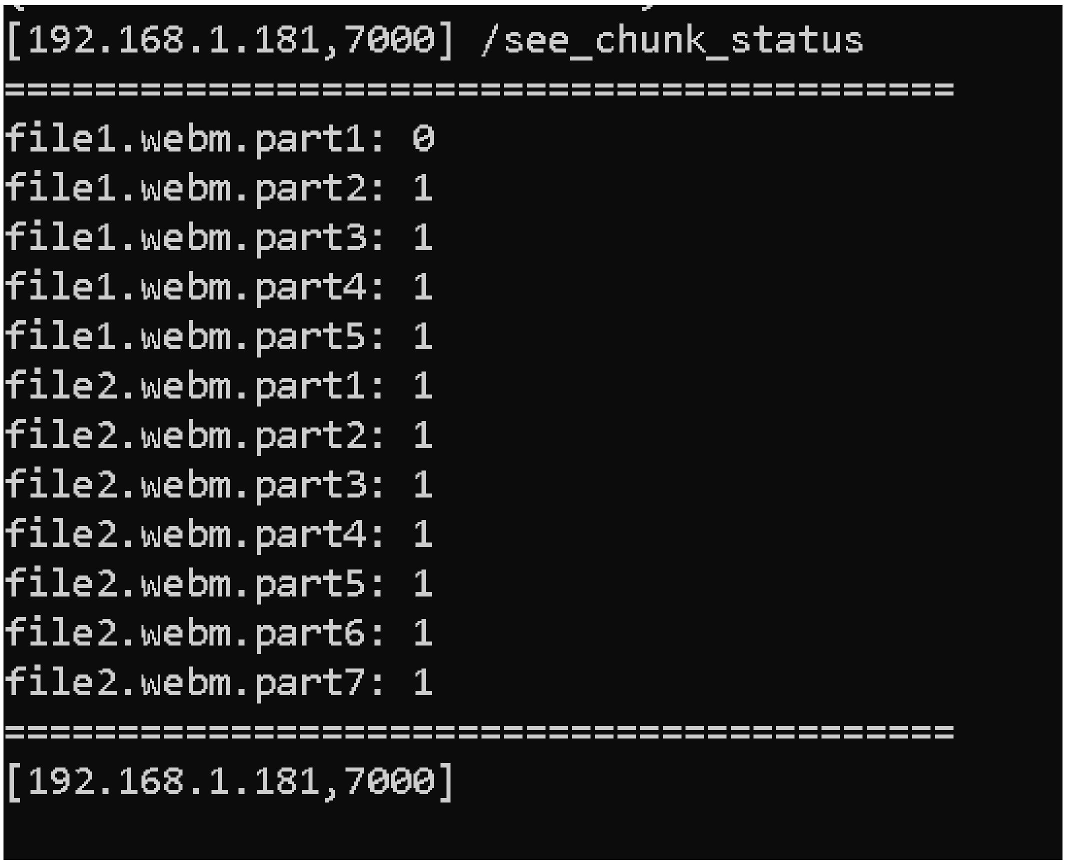
#### 5.3.6.a Command syntax

/see_chunk_status

#### 5.3.6.b Explanation

- This command will browse to the peer's memory (a folder) and check how many chunks that peer are having (labeled by 1) and missing (labeled by 0).

#### 5.3.6.c Demonstration



```
[192.168.1.181,7000] /see_chunk_status
==========================================
file1.webm.part1: 0
file1.webm.part2: 1
file1.webm.part3: 1
file1.webm.part4: 1
file1.webm.part5: 1
file2.webm.part1: 1
file2.webm.part2: 1
file2.webm.part3: 1
file2.webm.part4: 1
file2.webm.part5: 1
file2.webm.part6: 1
file2.webm.part7: 1
==========================================
[192.168.1.181,7000]
```

Hình 27: /see_chunk_status Demo

### 5.3.7 /merge_chunks

#### 5.3.7.a Command syntax

**/merge_chunks [file_name]**

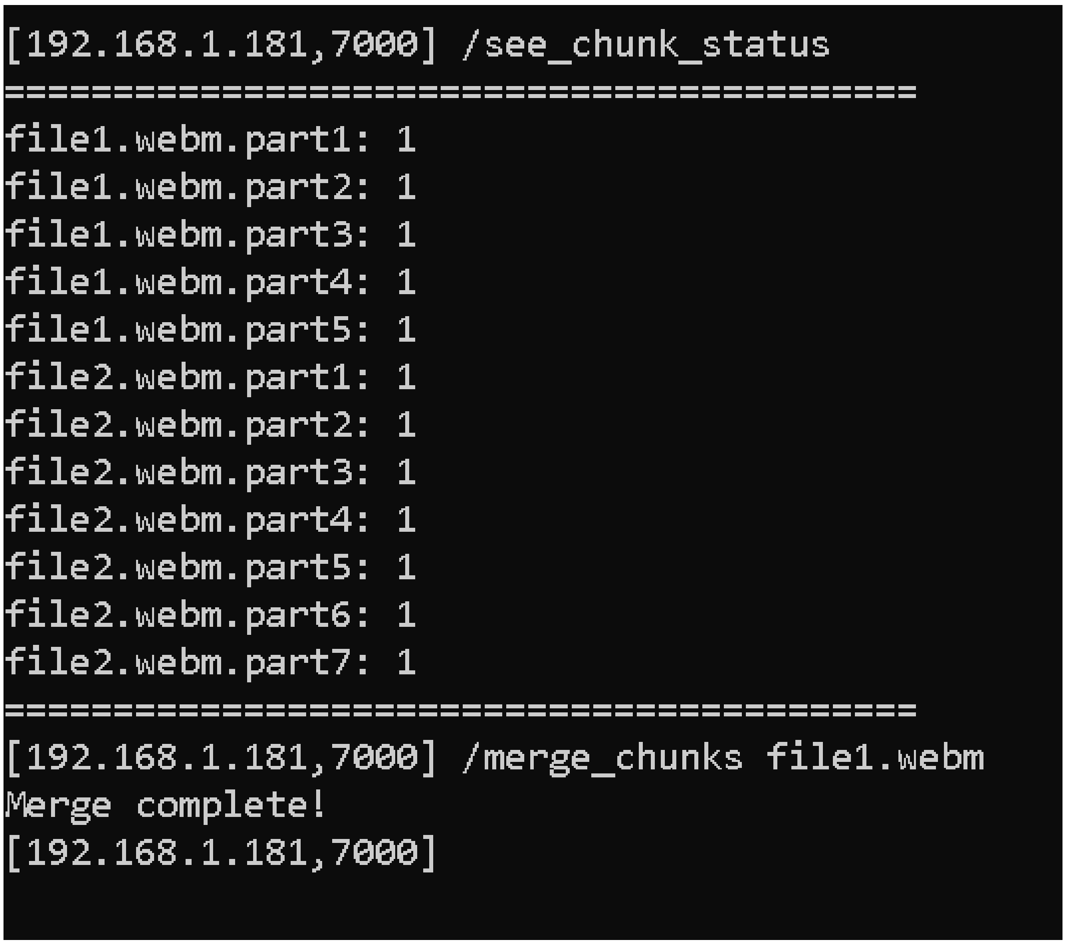
Where:

- [**file_name**]: name of the file the peer wishes to merge the correlated chunks into.

#### 5.3.7.b Explanation

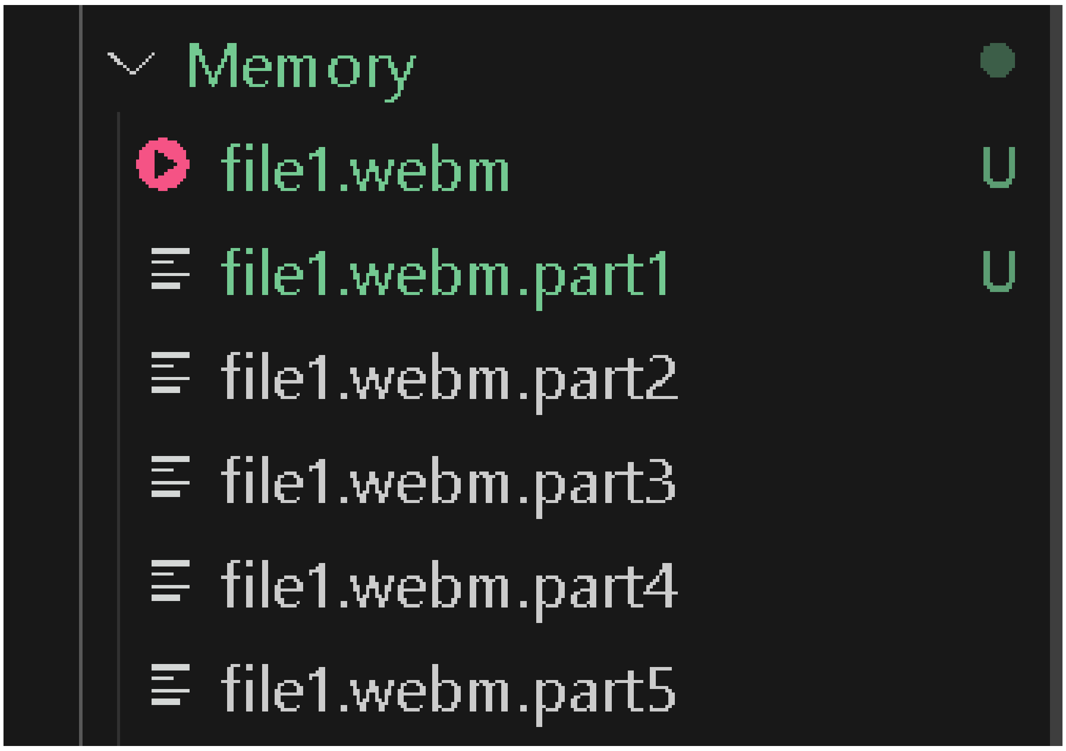
This command will initially check if all the chunks are fully downloaded, then the merging procedure of all chunks into a file will be executed.

#### 5.3.7.c Demonstration



```
[192.168.1.181,7000] /see_chunk_status
==========================================
file1.webm.part1: 1
file1.webm.part2: 1
file1.webm.part3: 1
file1.webm.part4: 1
file1.webm.part5: 1
file2.webm.part1: 1
file2.webm.part2: 1
file2.webm.part3: 1
file2.webm.part4: 1
file2.webm.part5: 1
file2.webm.part6: 1
file2.webm.part7: 1
==========================================
[192.168.1.181,7000] /merge_chunks file1.webm
Merge complete!
[192.168.1.181,7000]
```

Hình 28: /merge_chunks Demo



Hình 29: /merge_chunks Demo2

## 5.4 /auto_download command

### 5.4.1 Command syntax

**/auto_download**

### 5.4.2 Explanation

This command utilize all of the above commands to automate the file downloading process of a peer.

The command will do the following:

- Connect to tracker and get peer_set to see active seeders
- Loop for each chunks until all chunks that torrent can offer have been downloaded:
  - Pick a peer in the peer_set that has the chunk, and the lowest "uploaded" value
  - If can't find any peer, continue the loop to download other chunks
  - Call "/request_download" to the selected peer and receive the chunk.
- Merge all chunks in to files for personal use.