CSN-361 (Computer Networks)
BitTorrent inspired P2P based file sharing system
Design Document

September 24, 2024

Version 1.0

Group-6

Name	Enrollment No
Meet Sindhav	22114053
Mohammed Haaziq	22114055
Aditya Mundada	22114058
Nayan Kakade	22114060
Sarvesh Baheti	22114087
Roopam Taneja	22125030

Summary

Our aim is to develop a BitTorrent-like client and tracker, which enable users to download files. It a peer-to-peer network where each peer has the ability to download and upload from each other without a need of a central server. Each client first sends a request to tracker about the file it wants to download. The tracker then provides the client with the set of IP address of peers who have the required file. The client can then establish a TCP or UDP connection with these peers and start the download. The file is itself divided into several pieces so that a peer can concurrently download different pieces of file from different peers, hence increasing the efficiency of the process. Tracker must store the metadata related to which peers have which piece available. Further improvements can be made by efficiently selecting the pieces to be downloaded in such a way that choking can be prevented.

High Level Design

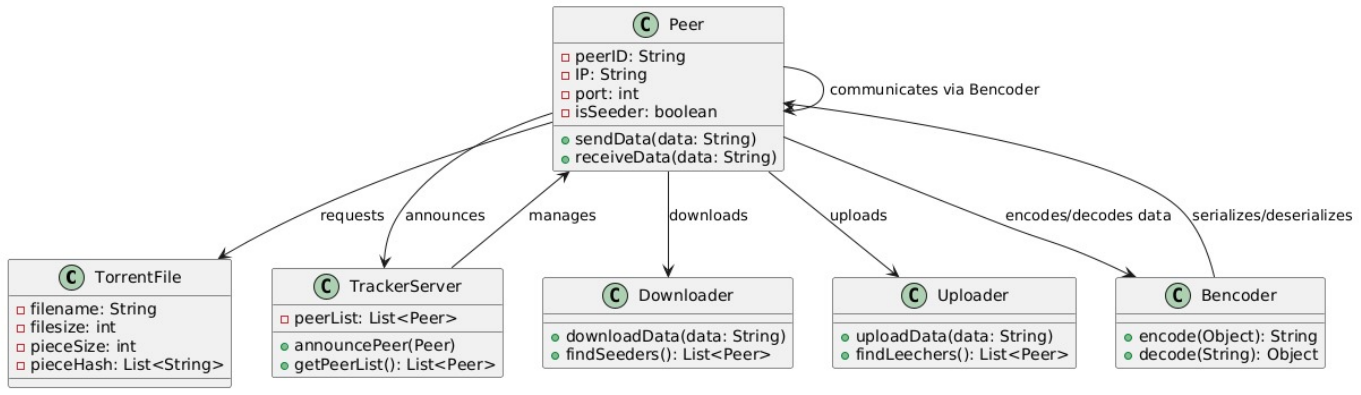

Figure 1: UML Diagram

The diagram outlines the structure and interactions between key components in a BitTorrent-based filesharing system, focusing on peers, a tracker, and a bencoder for data serialization.

1. **TorrentFile:**

   Role: Holds the metadata about the file being shared, such as piece size and hash values. Peers request this metadata to download and verify the file from other peers.

2. **Peer:**

   Methods:

   - `sendData(data: String):` Sends data (pieces of the file) to another peer.
   - `receiveData(data: String):` Receives data from another peer.

   Role: Peers are network participants that either download (leechers) or upload (seeders) file pieces. They interact directly with each other for data exchange.

3. **TrackerServer:**

   Methods:

   - `announcePeer(Peer):` Announces the peer's presence to the tracker.
   - `getPeerList():` Retrieves a list of peers currently sharing the file.

   Role: The tracker helps peers find each other by managing a list of active peers in the swarm. It coordinates connections between peers but does not host the file itself.

4. **Downloader:**

   Methods:

   - `downloadData(data: String):` Downloads file pieces from peers.
   - `findSeeders():` Finds seeders in the network to download data from.

   Role: The Downloader class manages the process of downloading pieces of the file from other peers, focusing on peers who are seeders.

5. **Uploader:**

   Methods:

- `uploadData(data: String)`: Uploads file pieces to other peers.
- `findLeechers()`: Finds leechers in the network to upload data to.

Role: The Uploader class manages the process of uploading file pieces to other peers, especially leechers who are downloading the file.

6. **Bencoder:**

   Methods:

   - `encode(Object)`: Serializes an object into a bencoded string.
   - `decode(String)`: Deserializes a bencoded string back into an object.

   Role: Bencoder is used to serialize and deserialize data between peers. It ensures the data being exchanged is formatted correctly for transmission.

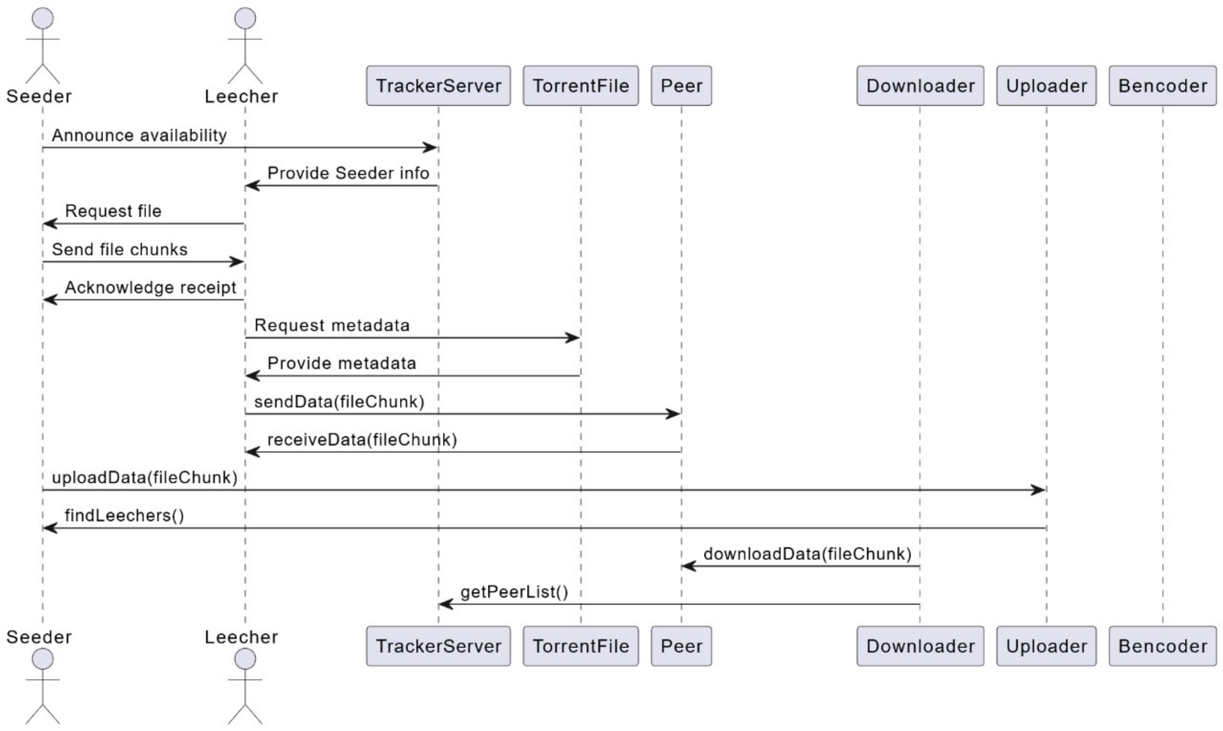

Figure 2: Sequence Diagram

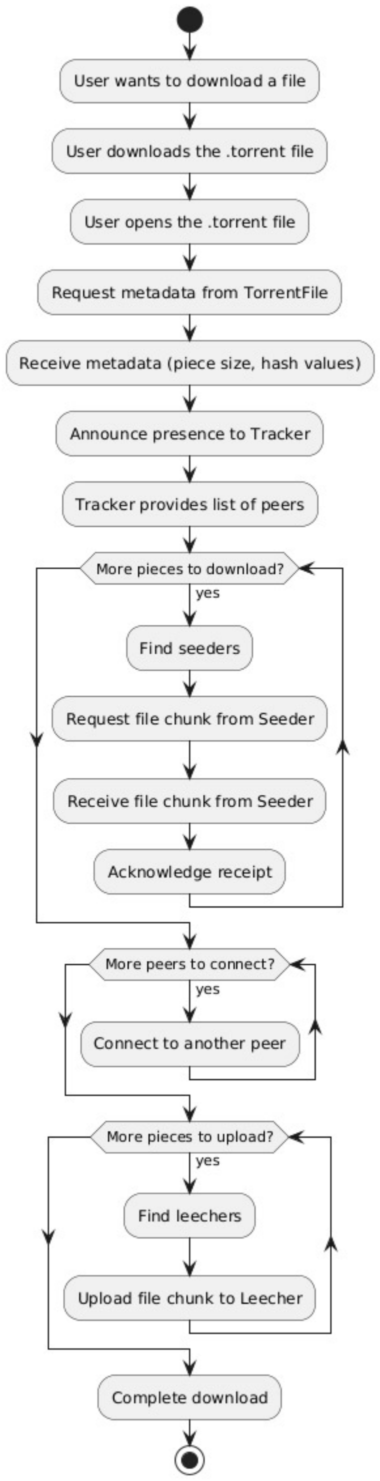
User wants to download a file
User downloads the .torrent file
User opens the .torrent file
Request metadata from TorrentFile
Receive metadata (piece size, hash values)
Announce presence to Tracker
Tracker provides list of peers
More pieces to download?
yes
Find seeders
Request file chunk from Seeder
Receive file chunk from Seeder
Acknowledge receipt
More peers to connect?
yes
Connect to another peer
More pieces to upload?
yes
Find leechers
Upload file chunk to Leecher
Complete download

Figure 3: Flowchart

Low Level Design

1. **Data Serialization:** .torrent files are encoded using bencoding. This file contains the URL of tracker, information about files, size of pieces etc. This requires an algorithm for encoding and eventually decoding this data, which will be developed by us in this project.
2. **Asynchronous Socket Handling:** Each peer can be connected to multiple peers when downloading a file and hence these multiple connections need to handled asynchronously. This can be implemented using help of threads or *epoll*, which is an efficient I/O event notification mechanism available in Linux systems, used for managing large numbers of file descriptors.
3. **Logger:** For providing users with important information like number of connected peers, download speed and progress, a logger algorithm will be used. In case of errors, it will provide the user with valuable information and keep check of all the important updates.
4. **Database management:** For the tracker to manage a list of peers who contain a file chunk and their IP address we will implement sophisticated database management algorithms that use concepts like hashing.

Applications

1. **File Sharing:** BitTorrent is widely used for distributing large files such as videos, software, games, and operating system images. It's a popular method for distributing Linux distributions, like Ubuntu, because it reduces server load.
2. **Decentralized Backup Systems:** Some organizations and individuals use BitTorrent for decentralized backup systems, where files are shared across many peers for redundancy and fault tolerance. This approach ensures that a file remains accessible even if some nodes fail.
3. **Software Distribution:** Game companies like Blizzard (creators of World of Warcraft) use BitTorrent technology to distribute game updates and patches to players, reducing server strain. BitTorrent is sometimes used for distributing large software updates or packages, especially for open-source software. Blockchain systems like Bitcoin and Ethereum use a similar P2P architecture to distribute and synchronize node data.
4. **Video Streaming:** Some platforms (e.g., Popcorn Time) have adapted BitTorrent for streaming video content. Instead of downloading an entire file before watching, users stream content as they download it in chunks from other peers.

References

1. BitTorrent Specification - Theory Wiki
2. BEP 3 - BitTorrent Protocol Specification
3. BitTorrent Overview - Brown University