CSN 361 (Computer Networks)
BitTorrent inspired P2P based file sharing system
Requirements and Specification 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

Topic: BitTorrent inspired P2P based file sharing system

Project Abstract

We aim to build a BitTorrent inspired P2P file sharing system which allows downloading files from distributed set of peers rather than a central server. BitTorrent, also referred to simply as torrent, is a communication protocol for peer-to-peer file sharing, which enables users to distribute data and electronic files over the Internet in a decentralized manner. It involves oversight of the peers, pieces (file is divided into parts called pieces), connections among peers, seeding, downloading etc.

We will use Bencoding serializing library to share metadata (.torrent file). Connection between peer and tracker will use UDP protocol. Connection between peers will use the services of TCP protocol. We will handle the multiple connections simultaneously with the help of asynchronous functions. We will benchmark the client performance based on several metrics viz. download speed, memory usage etc. After implementing the basic tracker, we will add extensions to it to support IPv6. After implementing the basic client features, we will try to add the feature of optimistic unchoking and rarest first mechanisms.

Motivation

We chose this topic as we felt that the volumes of file sharing is increasing day by day and there is a need to build a P2P file sharing system which tackles the drawbacks of existing P2P file sharing system.

The motive is not just to build a market product but also study and understand the underlying network protocols, programming languages from academic as well as practical perspective.

Also using it as a blackbox we can also build real world applications in the future.

Customer

Our dummy customers include our friends who like to share movie files or music files to each other in small group.

Competitive Landscape

Competitors:

1. **BitTorrent:** It is the most popular file sharing protocol known for efficient file transfer of large files. It has got various features like swarming (piece-based distribution), tit-for-tat strategy, choking and unchoking mechanism, and rarest first mechanisms, which make file transfer between peers efficient. It also uses Distributed Hash Table (Kademlia) to store details of the peers.
2. **IPFS (Interplanetary File System):** A distributed P2P protocol focused on making the web faster and more resilient by storing files and data in a decentralized way. This is more of a web service-niche protocol rather than a generic file sharing system.
3. **e-Mule:** Another P2P file sharing protocol that has a potential weakness of inability to identify fake or corrupted files.
4. **Gnuttela:** A P2P file sharing protocol that is completely decentralized and works on recursive "flooding" where each node forwards the request to its connected peers until the file is found. This flooding mechanism causes a serious issue as the number of peers increases.

Our Competitive Difference:

BitTorrent is our inspiration protocol. However, BitTorrent lacks in solving a few problems. It is not very suitable to use for a lesser number of peers, and it does not handle "leechers" too well. Leechers are nodes that download from the network but do not contribute back enough data.

In order to solve these problems, we want to customize the BitTorrent protocol based on recommendations in BitTorrent Enhancement Proposals (BEPs). These enhancements include:

1. Using a UDP tracker in the protocol that is lightweight and has fewer overheads, thus helping specifically address fewer peers when present in a network.
2. Implementing proposed optimizations in tit-for-tat, rarest first, and other strategies.
3. Trying to implement simple encryption at the sender and receiver by using concepts like public and private keys. The tracker may provide each peer with the public key of the other peers.

Thus overall, we aim to make an academic product customised to our needs. Our dummy customers include our friends who like to share movie files or music files to each other in small group. The major purpose of this project is not to make a market product but learn the nuances of a P2P file sharing system to gain knowledge of the complexities involved within.

System Requirements

Functional Requirements

- **BitTorrent Protocol Handling:**
  - Implement tracker communication with UDP trackers to announce, scrape, and obtain peer lists.
  - Implement peer handshake and message exchange using the BitTorrent protocol over TCP connections for data transfer between peers.

  - Support multi-peer download (piece selection, verification of pieces via SHA-1).
  - Optionally implement "Optimistic Unchoking" for faster peer selection.
  - Provide optional encryption/decryption for data sent between peers.
- **Asynchronous Socket Handling:**
  - Handle multiple connections with asynchronous TCP connections to multiple peers simultaneously.
  - Provide non-blocking I/O operations for efficient data handling.
- **Bencoding Library:**
  - Implement a bencoding serializer/deserializer to handle .torrent files and UDP tracker responses.
- **Performance Metrics:**
  - Compare the tracker to notable trackers like Chihaya, Opentracker, and Aquatic for benchmarking.
  - Measure metrics such as Download/Upload Speed (throughput), Memory Usage (bounded limits), and Concurrent Peer Limit (maximum limit on concurrent peer connections).

Non-Functional Requirements

- **Security and Privacy:**
  - Provide encryption via secure encrypted channels between peers.
  - Ensure data integrity by verifying pieces using hash functions (SHA-1).
  - Maintain privacy by avoiding unnecessary leaking of personal IP data.
- **Timing Constraints:**
  - **Tracker Response Timeout:** Handle timeouts for tracker communication (e.g., retries for failed UDP tracker requests).
  - **Connection Timeout:** Drop peer connections if no response is received within a predefined period.
- **Memory Requirements:**
  - Efficiently handle large torrents.
  - Optimize buffers for transferring file chunks without excessive memory usage.
- **Performance Requirements:**
  - Ensure high download speed by optimizing peer selection algorithms.
  - Minimize latency in tracker announcements and peer-to-peer communication.
- **Data Capacity:**
  - Support up to N simultaneous peers.
  - Handle torrents of varying file sizes, from small documents to large multimedia files.

Use Cases

- **Use Case 1: Downloading a File**
  - **Actors:** User, Tracker, Peers.
  - **Steps:**

        1. The user selects a .torrent file.
        2. The client communicates with a tracker via UDP to retrieve a list of peers.
        3. The client establishes TCP connections with peers.
        4. File pieces are downloaded from multiple peers and verified.
    - **Acceptance Test:** The user successfully downloads the file from multiple peers in a reasonable time, with encrypted communication.
    - **Priority:** Must have.
- **Use Case 2: Tracker Communication**
    - **Actors:** Client, UDP Tracker.
    - **Steps:**
        1. The client sends an announce message to the UDP tracker.
        2. The tracker responds with a peer list.
    - **Acceptance Test:** The client successfully communicates with the tracker and retrieves the peer list.
    - **Priority:** Must have.
- **Use Case 3: Peer-to-Peer Encryption**
    - **Actors:** Peers.
    - **Steps:**
        1. The client establishes an encrypted channel with another peer.
        2. The client and peer exchange encrypted messages.
    - **Acceptance Test:** The client can securely download pieces via encrypted peer communication.
    - **Priority:** Optional.

External Dependencies

- **Operating System:** Linux/Windows with POSIX socket support.
- **Libraries:**
    - OpenSSL (for encryption).
    - Bencoding parsing library (custom or third-party).
- **Trackers:** UDP trackers, tested with known trackers like Aquatic.
- **IPv6 Support:** Ensure compatibility with IPv6 networks.

User Interface Requirements

- **CLI:** The client should have a command-line interface where users can provide a .torrent file and view download progress.
- **Output:** Display detailed download stats (speed, number of connected peers, progress percentage).