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

Computer Networks (CO3093)

Assignment 1

Simple Torrent-like Application (STA)

Instructor: Nguyễn Phương Duy

Students:	Nguyễn Trung Tín	2213500
	Trần Quốc Toàn	2213540
	Nguyễn Thị Bảo Trâm	2213574
	Trương Anh Tuấn	2213810

HO CHI MINH CITY, NOVEMBER 2024

Mục lục

1 Requirement elicitation **3**
1.1 Domain Context . . . 3
1.2 Functional Requirements . . . 3
1.3 Non-functional Requirements . . . 5
1.4 Use-case Diagram . . . 7

2 System Modeling **8**
2.1 System Architecture . . . 8
2.2 System functionalities and Protocols . . . 8
2.2.1 Peer Discovery and Management . . . 8
2.2.2 File Sharing . . . 9
2.2.3 File Downloading . . . 9
2.2.4 Network Communication . . . 9
2.2.5 Activity Monitoring . . . 9
2.3 Activity Diagram . . . 10

1 Requirement elicitation

1.1 Domain Context

Before the widespread adoption of the internet, sharing digital content relied on physical storage devices such as floppy disks and USB drives. For long distances, these storage devices were often transported through postal services. These methods were constrained by delays, physical risks to the storage medium, and limited scalability.

With the advent of the internet, Peer-to-Peer (P2P) systems introduced a more efficient and decentralized approach to file sharing. P2P protocols, such as those used by BitTorrent, optimized bandwidth utilization and eliminated the need for centralized servers by distributing file chunks across multiple peers. These systems allowed users to upload and download content directly from one another, providing flexibility and reliability.

This project, Simple Torrent-like Application (STA), seeks to implement a basic P2P system inspired by BitTorrent. The application will focus on key functionalities such as file chunking, peer discovery, and distributed file sharing to explore the foundational principles of decentralized systems.

1.2 Functional Requirements

The application includes the two types of hosts: tracker and node.

1. **Node**

   (a) **Client**

   - When require a file that does not belong to its repository, a request is sent to the tracker to get the peers and files information.
   - Inform the tracker as to what files are contained in its local repository but does not actually transmit file data to the tracker through tracker protocol.
   - Connect to an active peer at a specified IP and port.
   - Select files to download or upload.
   - Upload files to multiple peers simultaneously.
   - Download files from peers into a specified local directory (don't send a request for a piece the peer does not have, and don't send a request for a piece the node already have.).
   - Display download/upload progress.
   - Manage downloaded files (e.g., delete them).

- Support downloading multiple files from multiple source nodes at once, simultaneously.

(b) **Server**

- Maintain connections with active peers' clients
- Respond to client download requests for pieces.
- Enable multi-client file sharing using multithreading to handle concurrent transfers.

(c) **User Interface**

- Satisfy the minimum requirements of downloading and seeding a single torrent to multiple peers.
- Provide ability to see all the upload/download statistics.

2. **Tracker**

- Track file metadata, including file pieces and list of peers.
- Maintain the active/inactive status of peers.
- Manage the availability of shared files.
- **Tracker request parameters**: should include the compacted flag(s) or metainfo in your request, a bare minimum requirement is a magnet text of your torrent that is able to parse (to locate to a Metainfo File placed on the server).
- **Tracker response**: respond with "text/plain" document consisting of a dictionary with the following keys:
  - *Failure reason*: If present, then no other keys may be present. The value is a human-readable error message as to why the request failed (string).
  - *Warning message*: (new, optional) Similar to failure reason, but the response still gets processed normally. The warning message is shown just like an error.
  - *Tracker id*: A string that the client should send back on its next announcements. If absent and a previous announcement sent a tracker id, do not discard the old value; keep using it.
  - *Peers*: (dictionary model) the value is a list of dictionaries, each with the following keys:
    - peer id: peer's self-selected ID, as described above for the tracker request (string)

* ip: peer's IP address either IPv6 (hexed) or IPv4 (dotted quad) or DNS name (string)
* port: peer's port number (integer)

1.3 Non-functional Requirements

1. **Availability**

ID	Description
NF-AVAI1	Ensure system availability of 99.9% uptime, with minimal scheduled downtime for maintenance
NF-AVAI2	Enable seamless resumption of interrupted downloads after application restarts

2. **Performance**

ID	Description
NF-PERF1	Support a minimum of 10 concurrent download processes and up to 1,000 peer connections without significant performance degradation
NF-PERF2	Achieve a download speed of at least 80% of the user's available bandwidth under optimal conditions

3. **Usability**

ID	Description
NF-USAB1	Provide an intuitive and visually clear interface that displays progress and file size
NF-USAB2	Design a user-friendly interface for general users with accessible controls

4. **Reliability**

ID	Description
NF-RELI1	Automatically reconnect to peers in the event of a disconnection
NF-RELI2	Implement robust data recovery mechanisms to prevent data loss during unexpected shutdowns

5. **Efficiency**

ID	Description
NF-EFFI1	Optimize resource usage to minimize CPU and memory consumption, particularly on devices with limited hardware resources

6. **Security**

ID	Description
NF-SECU1	Support end-to-end encryption for all data transfers to ensure privacy and security

7. **Compatibility**

ID	Description
NF-COMP1	Ensure support for standard file-sharing formats such as .torrent and magnet links
NF-COMP2	Maintain compatibility with major operating systems, including Windows, macOS, and Linux

8. **Scalabilty**

ID	Description
NF-SCAL1	Design the system to handle increases in the number of users and peer connections without requiring substantial architectural changes

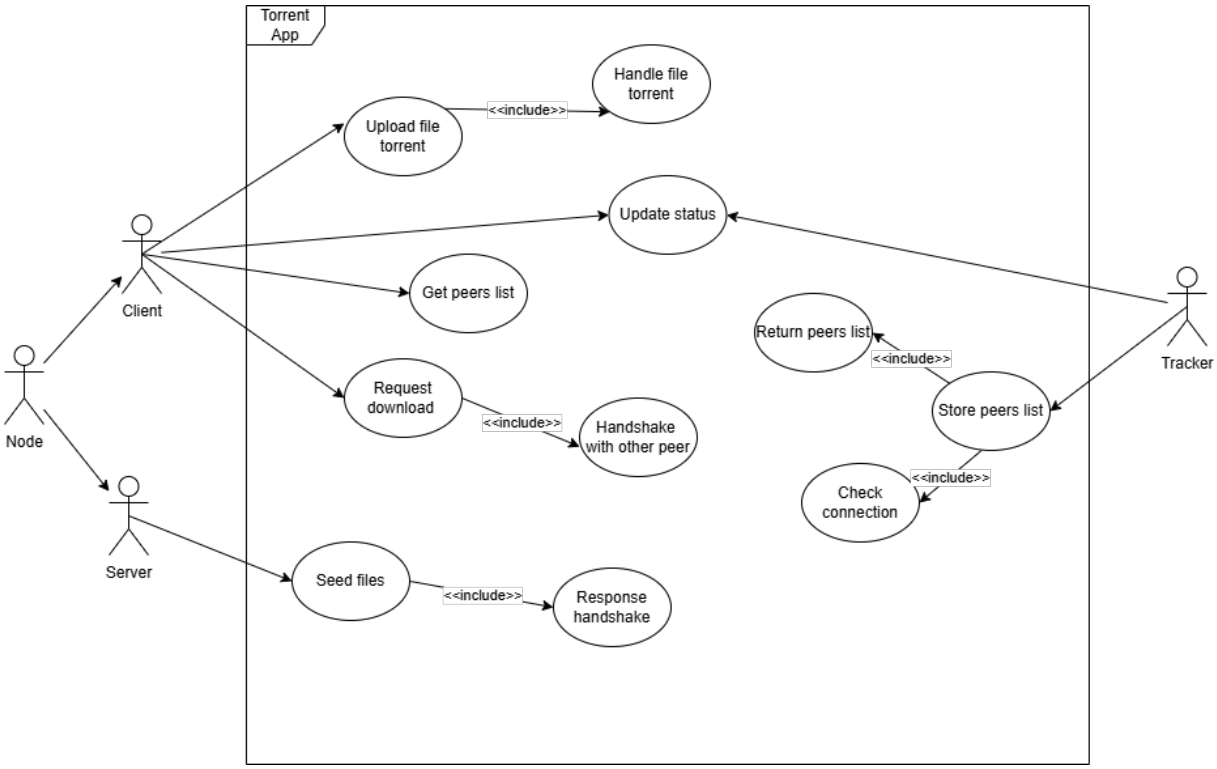
9. **Maintainability**

ID	Description
NF-MAIN1	Provide comprehensive and clear logging mechanisms to facilitate debugging and issue resolution

10. **Compliance**

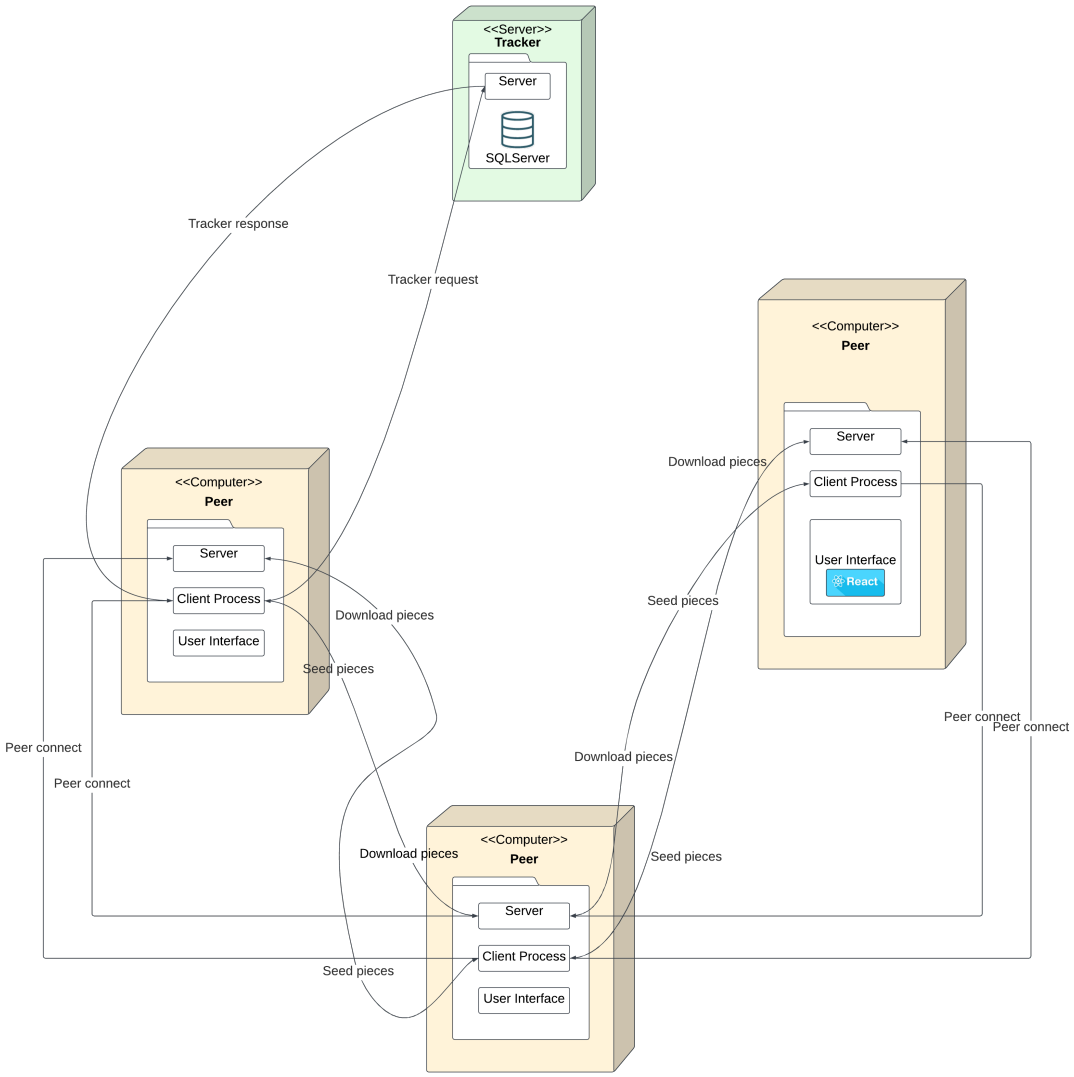
ID	Description
NF-COMP1	Adhere to BitTorrent protocol standards and comply with applicable regional copyright regulations

1.4 Use-case Diagram



2 System Modeling

2.1 System Architecture



2.2 System functionalities and Protocols

2.2.1 Peer Discovery and Management

- **Functionality**: Use a centralized server, called a tracker, to maintain a record of all peers and the files they share. The tracker facilitates peer discovery and identifies available files for download.

- **Protocol**: Communication with Tracker via **HTTP**: Peers connect to the tracker using announce requests to retrieve a list of available peers.

2.2.2 File Sharing

- **Functionality**: Users can share files by uploading them to their local repository. The file metadata is recorded in the tracker, making it accessible for other peers.
- **Protocol**: Communication with Tracker via **HTTP**.

2.2.3 File Downloading

- **Functionality**: Peers request the tracker for a list of available peers holding the desired file and download the file directly from them.
- **Protocol**: Communication with Tracker via **HTTP**: Safeguard file-related data through encrypted transmission.

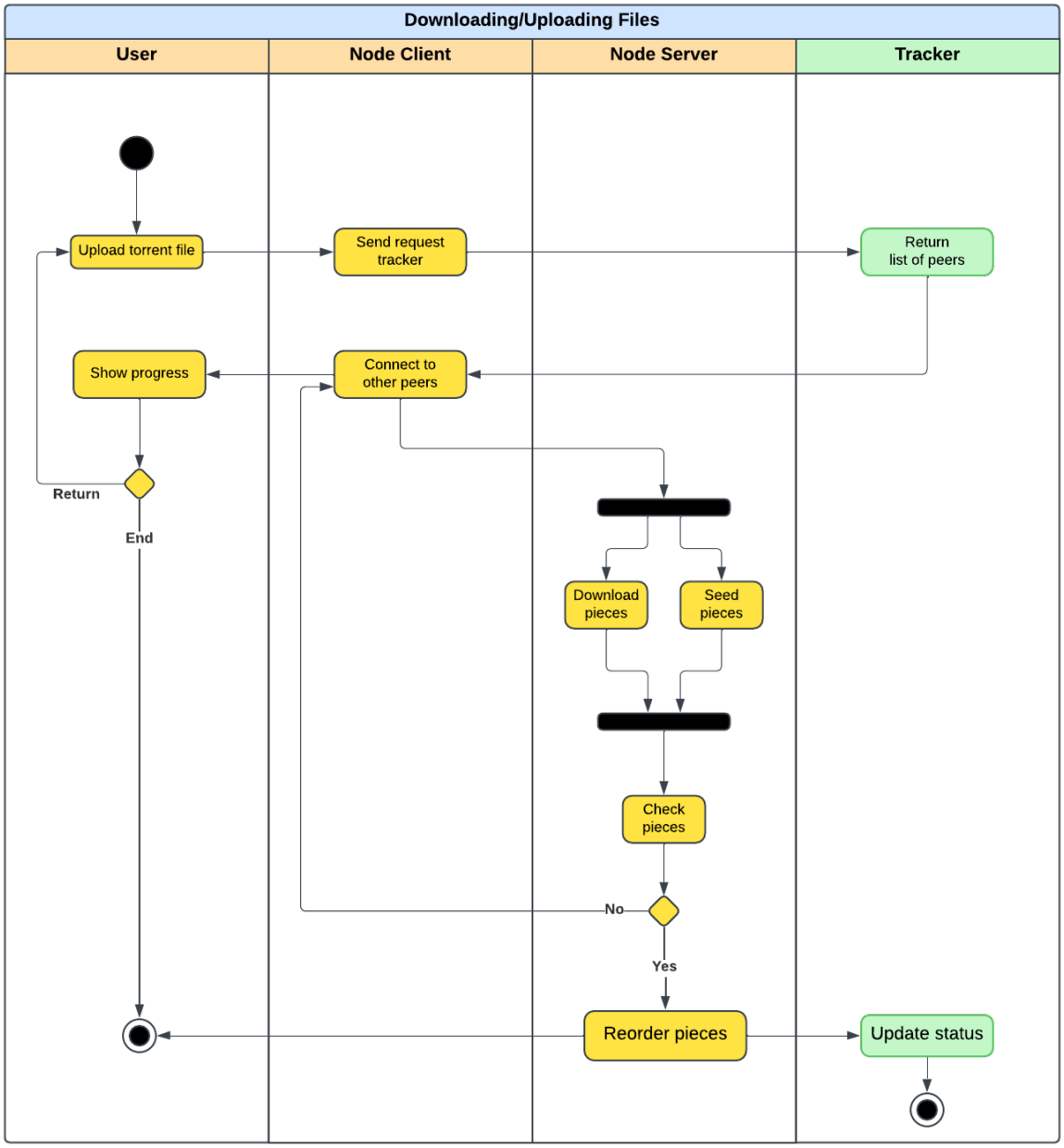
2.2.4 Network Communication

- **Functionality**:
  - Direct peer-to-peer connections for file transfers.
  - Secure communication with the tracker for metadata and peer information.
- **Protocol**: **TCP/IP** for direct peer connections: Ensure reliable and sequential delivery of data packets.

2.2.5 Activity Monitoring

- **Functionality**: Monitor upload/download status and provide real-time statistics.
- **Protocol**: **HTTP** for periodic updates:
  - Peers send periodic status updates (e.g., upload/download progress) to the tracker.
  - The tracker stores and responds with aggregated data for monitoring purposes.

2.3 Activity Diagram



This diagram describes a workflow where the user initiates a file sharing session, the peer client connects to other peers to download file pieces, the peer server facilitates file seeding and piece verification, and the tracker coordinates the peer network by managing and updating peer lists.

The user upload file torrent to retrieve the file's metadata. Upon starting a download, the peer client sends a request to the tracker for a list of peers that hold the file pieces.

Using the list from the tracker, the peer client connects to other peers in the network to begin the download. If any issues occur during the download process, such as a missing piece or failed connection, the peer client alerts the user. As the upload or download process proceeds, progress is displayed to the user. Once pieces are downloaded, the peer server seeds them to other clients, maintaining the file's availability. The server verifies the integrity of downloaded pieces to ensure they match the original file. If pieces are out of order or missing, the server reorders or requests missing parts to complete the file.

The tracker receives requests from peer clients and returns a list of available peers with the requested file pieces. Also, the tracker updates the availability status of pieces and peer connections to help maintain efficient file sharing.