COMPUTER NETWORK

Assigment 1 – Network Application

Lecturer: Dr. Nguyen Le Duy Lai

Member:

Nguyễn Di Tân - 2053422

Lê Minh Gia Bảo - 2052396

Nguyễn Anh Hào - 2052971

Hữu Huy - 1952266

APRIL 28, 2024

HO CHI MINH UNIVERSITY OF TECHNOLOGY

I. Introduction

SimpleTorrent is a Java-based application designed to simulate the fundamental operations of a BitTorrent-like peer-to-peer (P2P) file sharing system for educational purposes. This project allows users to engage directly in the sharing and requesting of file pieces across a network using a centralized tracker system to manage connections and file metadata. By implementing core features of the BitTorrent protocol with simplified logic and user interfaces, SimpleTorrent provides an accessible platform to explore network programming concepts, TCP/IP protocol stack usage, and multithreaded application development in a practical setting.

II. Application

The SimpleTorrent app has several functions as follows:

- Peer Discovery and Management:

Centralized Tracker: SimpleTorrent employs a centralized server, known as a tracker, which keeps a registry of all peers and the files they share. This server facilitates the discovery process by allowing peers to find each other and the files available for download.

- File Sharing and Requesting:

File Uploads: Users can share files by uploading them to their local repository. The metadata of these files (but not the files themselves) is then registered with the tracker, making them available to other peers.

- File Downloads:

Peers can request files listed in the tracker. The application manages these requests and retrieves file pieces from multiple sources simultaneously, enhancing download efficiency and speed.

- File Handling:

Segmentation: Files are automatically divided into manageable pieces, which are then shared and downloaded piece by piece. This not only optimizes the transfer process but also allows for the resumption of downloads without starting over if interrupted.

Checksum Verification:

Each piece of the file comes with a checksum to ensure the integrity of the data received, thereby preventing corruption during transmission.

- User Interface:

Interactive CLI: SimpleTorrent provides a command-line interface (CLI) that guides users through the process of sharing and requesting files, displaying available files, and other peer-related activities.

- Network Communication:

TCP/IP Protocol Stack: The application leverages TCP/IP protocols to ensure reliable data transfer across the network. This includes establishing and maintaining connections, handling data transmission, and ensuring data integrity and order.

- Multithreading:

Concurrent Operations: SimpleTorrent is designed to handle multiple download and upload operations concurrently. This is achieved through multithreading, which allows the application to perform network communications in parallel, thus maximizing resource utilization and responsiveness.

III. Systems on the network

1. **Tracker Server**

Purpose: Manages and facilitates the discovery of files within the network by maintaining a current list of peers and their shared files.

Functionality: Responds to peer requests to register new shared files and queries for existing files. It updates its records accordingly to ensure all peers can discover available files efficiently.

Communication: Utilizes TCP/IP protocols to ensure reliable communication, handling incoming connections on a dedicated port, with threads managed by an ExecutorService for scalability.

2. **Client (Peer)**

Purpose: Each peer serves dual functions, requesting files from others and responding to requests for its shared files.

Functionality: Registers its shared files with the TrackerServer upon initialization and responds to direct file requests from other peers using a server socket.

Concurrency: Employs multithreading to manage multiple simultaneous incoming and outgoing connections, ensuring robust handling of file transfers.

3. **TorrentCLI**

   Purpose: Facilitates user interaction with the SimpleTorrent network, allowing for straightforward operations management.

   Functionality: Users can issue commands to share new files, request files from other peers, or terminate sessions, with immediate feedback provided through the CLI.

4. **Communication Protocols**

   TCP/IP: Chosen for its reliability, ensuring that all data transfers occur without corruption or loss. Peers use TCP connections both to communicate with the TrackerServer and to transfer files directly among themselves.

   Data Exchange Formats: The system uses Java's built-in serialization capabilities (ObjectOutputStream and ObjectInputStream) to transmit data such as file metadata and peer lists, ensuring that complex data structures are communicated intact..

5. **P2P**

- Peer-to-peer (P2P) computing or networking is a distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the network. They are said to form a peer-to-peer network of nodes.[1]
- Peers make a portion of their resources, such as processing power, disk storage or network bandwidth, directly available to other network participants, without the need for central coordination by servers or stable hosts.[2] Peers are both suppliers and consumers of resources, in contrast to the traditional client–server model in which the consumption and supply of resources are divided.[3]
- While P2P systems had previously been used in many application domains,[4] the architecture was popularized by the file sharing system Napster, originally released in 1999.[5] The concept has inspired new structures and philosophies in many areas of human interaction. In such social contexts, peer-to-peer as a meme refers to the egalitarian social

networking that has emerged throughout society, enabled by Internet technologies in general.

6. **Advantages of Peer to Peer Network - Cost:**
    - The overall cost of building and maintaining a peer to peer network is relatively inexpensive. The setup cost has been greatly reduced due to the fact that there is no central configuration. Moreover for the windows server, there is no payment required for each of the users on the network. The payment should be done only once.

    a. **Disadvantages**

7. **Decentralization**
    - Peer to Peer networking lacks the feature of centralization. There is no central server, thus files are stored on individual machines. The entire network accessibility is not in the hands of a single person. This makes it more challenging for the users to locate and find files. If the search is done through each database, the users could waste a lot of time.
8. **Performance**
    - Performance is another issue faced by a peer to peer network. Once the number of devices connecting the network increases, there will be a performance degrade since each computer is being accessed by other users. Hence, P2P network doesn't work well with growing networks.

9. **Security**
    - Security for individual files are comparatively less in peer to peer networking. There is no security other than assigning permissions. Even if the permissions are assigned, any person with the access to it will be able to log on. Some users don't even require to log on from their respective workstation.

IV. Explanation and Evaluation

1. **File Sharing and Requesting:**
   - Protocol: TCP/IP
   - Usage: Ensures reliable, ordered, and error-checked delivery of data between the TrackerServer and peers, and between peers themselves. TCP connections are used both for registering shared files with the TrackerServer and for transferring file data directly between peers.
2. **Data Integrity Checks:**
   - Protocol: SHA-256 (part of the Secure Hash Algorithms)
   - Usage: FileUtilities uses SHA-256 to generate checksums for each piece of the file, ensuring data integrity during transfers. Peers verify these checksums upon receipt of file pieces to detect and handle corrupt or incomplete data.
3. **Communication Between Peers and TrackerServer:**
   - Protocol: Serialized Java Objects over TCP
   - Usage: Uses Java's serialization mechanisms to send and receive structured data (like file metadata and lists of peers) over TCP connections. This protocol facilitates the easy transmission of complex data structures essential for the tracker's management operations.
4. **User Commands Processing:**
   - Protocol: Standard Console Input/Output
   - Usage: Processes commands entered by the user in the TorrentCLI via standard input (keyboard) and provides feedback through standard output (console). This simple protocol allows for real-time user interaction with the system.

Class Name	Purpose	Responsibilities	Interactions
TrackerServer	Acts as the central coordinator within the network.	Manages lists of shared files and peers, handles client connections, processes share and request operations.	Receives connections and requests from peers, sends back file and peer information.
Peer	Functions both as a server and a client for file sharing.	Shares files, requests files, manages network communications.	Connects to TrackerServer for file operations, communicates directly with other peers for file transfers.
TorrentCLI	Provides a command-line interface for user interaction.	Processes user commands, displays system responses, manages peer operations based on user input.	Takes user commands, invokes methods on Peer based on user input.

❖ **Uncompleted Aspects and Limitations:**

While our SimpleTorrent application has been designed to simulate a peer-to-peer file sharing system, certain components and functionalities remain incomplete. These include:

Checksum Verification: The implementation of checksum verification for ensuring data integrity during file transfers is not fully operational. This feature is crucial for preventing data corruption, and its absence may affect the reliability of file downloads.

Robust Error Handling: Comprehensive error handling mechanisms to manage network failures, data corruption, or user input errors are currently underdeveloped. This can lead to unexpected application behavior under edge cases or failure scenarios.

Limited Testing Scope: Currently, the SimpleTorrent system has only been tested in a local environment, with all peers running on the same machine. This limited testing scope restricts our understanding of the system's performance and behavior in a distributed network environment.

V. Demo Application

Step 1: Start the TrackerServer

java TrackerServer

```
PS C:\Users\Bao\Downloads\simpletorrent> java TrackerServer
Tracker Server initialized on port: 5000
TrackerServer started on port: 5000
```

Step 2: Launch Peer

And Follow the command I already defined on our Readme.md:

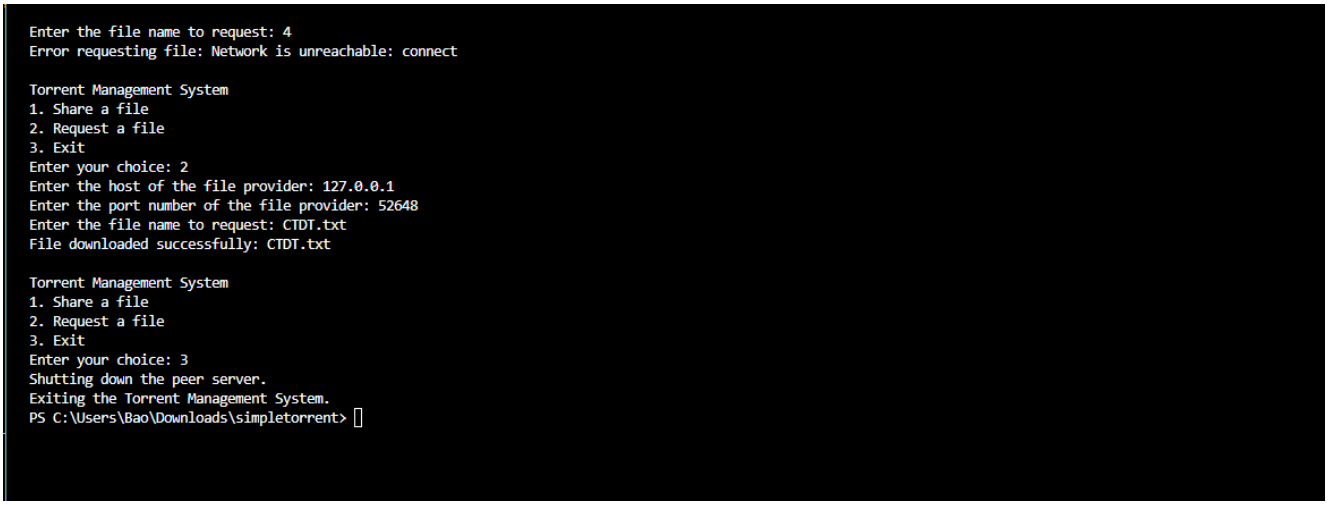
```
PS C:\Users\Bao\Downloads\simpletorrent> java Peer
Peer server initialized on port: 52631
Connected to Tracker at localhost:5000
Server listening on port: 52631
Enter command (share, request, exit):
share C:\Users\Bao\Downloads\P2PFiles\alice.txt
Added file for sharing: alice.txt
```

In this situation, we will try to download a .txt file

Step 3: Request to download a file in another instance

```
PS C:\Users\Bao\Downloads\simpletorrent> java Peer
Peer server initialized on port: 52639
Connected to Tracker at localhost:5000
Server listening on port: 52639
Enter command (share, request, exit):
request 127.0.0.1 52631 alice.txt
File downloaded successfully: alice.txt
```

Demo with CLI:



These demo we just testing in local host, if we want to run it in different situation, this is how we do it (For example, we have 2 laptop):

Step 1: Prepare the Environment

Install the Software: Ensure that both laptops have the Java Runtime Environment (JRE) installed and that your SimpleTorrent application (all necessary .java files) is present on both machines.

Step 2: Determine Network Configuration

Find IP Addresses:

On each laptop, open the command prompt (Windows) or terminal (macOS, Linux) and run ipconfig (Windows) or ifconfig (macOS, Linux) to find out the local IP address.

Check Network Connectivity:

Make sure both laptops are on the same network (e.g., connected to the same WiFi network). This simplifies the process by avoiding complex routing or firewall configurations.

Step 4: Run the TrackerServer

Choose a Laptop for the TrackerServer:

Decide which laptop will run the TrackerServer. Let's say Laptop A is chosen.

On Laptop A, run the TrackerServer. Make sure to specify the port if not using a default. Example:

java TrackerServer

Step 5: Configure and Run Peer Clients

Setup Peer on Laptop A:

Run the Peer application on Laptop A, ensuring it connects to its own TrackerServer. Use the IP address found earlier and the appropriate port.

Example:

java Peer localhost 5000

This tells the Peer that the TrackerServer is running on the same machine ('localhost').

Setup Peer on Laptop B:

On Laptop B, start the Peer application, pointing it to the IP address of Laptop A.

Example:

java Peer 192.168.1.2 5000

Replace 192.168.1.2 with the actual IP address of Laptop A where the TrackerServer is running.

Step 6: Test the Functionality

Share Files:

On either laptop, use the command-line interface to share files. Ensure the files are located in an accessible directory.

Request Files:

On the other laptop, request a file that has been shared by the first laptop. Watch the console output for messages about the file transfer process.

VI. Future work

Given the current limitations, the following steps are proposed to advance the project:

- **Complete Missing Features:** Prioritize the completion of all planned but unimplemented features, starting with those critical to the system's core functionality.

- **Expand Testing:** Implement a comprehensive testing strategy that includes unit tests, integration tests, and deployment in varied network environments.

- **Performance Optimization:** Once all features are implemented and initial tests are successful, focus on optimizing the code for better performance, especially regarding file handling and network communication.