Principles of Computer Systems - 2

Team members
Arnava Srivastava (B22AI009)
Samay Mehar (B22AI048)

File Distribution System

Introduction

The File Sharing System is a distributed application designed to facilitate file management, resource allocation, and collaboration in networked environments. This comprehensive report provides an in-depth analysis of the system's architecture, functionalities, implementation details, and real-world applications.

System Architecture

The File Sharing System follows a client-server architecture, where a central server manages file storage and resource allocation, while clients interact with the server to perform file operations and request resources. The system utilizes TCP/IP sockets for communication between clients and the server over the network.

Concepts incorporated :

1. **Socket Programming:** Utilized TCP/IP sockets for communication between clients and the server over the network.
2. **File Management:** Supported file operations such as uploading and downloading, simulating file system functionalities.
3. **Client-Server Architecture:** Implemented a client-server model where the server manages file storage and resource allocation, while clients interact with the server to perform file operations.
4. **Resource Allocation and Process Synchronisation:** Employed the Banker's algorithm to allocate and manage resources dynamically, preventing deadlock and ensuring safety.

Functionalities

The system offers the following key functionalities:

- **File Operations:** Clients can perform file-related operations, including listing files in a directory, uploading files to the server, and downloading files from the server.
- **Resource Allocation**: Clients can request resources (e.g., A, B, C) from the server to perform specific tasks and release previously allocated resources back to the server.
- **Process Synchronization:** Implements Peterson's algorithm to handle critical sections for concurrent access to shared resources, ensuring mutual exclusion and preventing race conditions.
- **Real-time Communication:** Facilitates real-time communication between clients and the server, enabling seamless interaction and data exchange.

How to Implement?

Server Side: The server script (server_new.py) initializes a server socket, listens for incoming client connections, and handles client requests using multithreading. It implements process synchronization and resource allocation algorithms to ensure secure and efficient file sharing among clients.

Client Side: The client script (client.py) establishes a connection to the server, sends requests for file operations and resource allocation, and receives responses from the server. It provides a user-friendly interface for clients to interact with the system and perform various tasks.

Real-World Applications

- **Cloud Storage Services**: Platforms like Dropbox, Google Drive, and OneDrive employ distributed systems to store and manage user files across multiple servers.
- **Collaborative Document Editing**: Like Google Docs and Microsoft Office Online.
- **Enterprise File Sharing**: Organizations can use the system to create internal file sharing solutions for employees to exchange documents, reports, and other resources within the organization securely.
- **Version Control Systems**: Like Git for collaboration in software development

Testing and Demonstration

1. Run the “server_new.py” file to initiate the server and start listening for requests on the host IP (here, 172.31.4.87) and port number : 9999

```
Arnava@Abhays-MacBook-Air server % python server_new.py

Server started and listening on 172.31.4.87:9999
```

2. Next run the “client.py” code file (in a separate terminal or computer) to connect to the server and use the implemented functionalities.

```
Arnava@Abhays-MacBook-Air client2 % python client2.py
Connected to server 172.31.4.87:9999
Welcome Client-2. What would you like to do?
Your options :
1 - List files in directory
2 - Upload text file
3 - Download text file
4 - Request resources
5 - Release resources
6 - EXIT
Input your option (1-6):
```

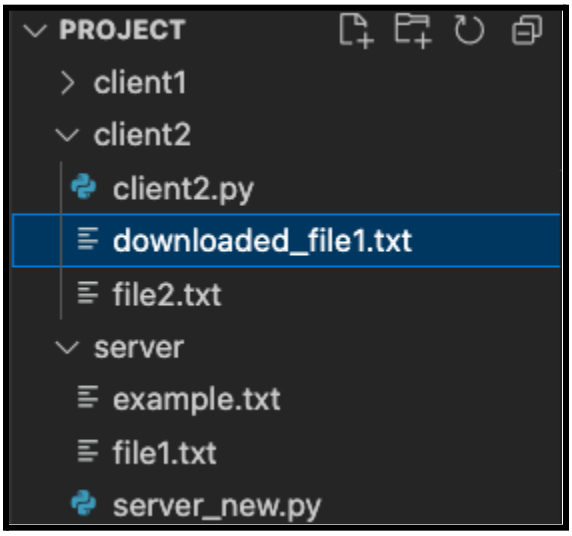
We get the options for File managing and Resource Allocation.

3. Let's say we Enter option 1. List of files on server directory are shown.

```
Arnava@Abhays-MacBook-Air client2 % python client2.py
Connected to server 172.31.4.87:9999
Welcome Client-2. What would you like to do?
Your options :
1 - List files in directory
2 - Upload text file
3 - Download text file
4 - Request resources
5 - Release resources
6 - EXIT
Input your option (1-6): 1
Files in current directory: ['file1.txt', 'example.txt', 'server_new.py']
```

4. Now, say we download "file1.txt" by choosing option 3 and specifying the filename. We can check the downloaded_file1.txt has been downloaded in the client directory.

```
Input your option (1-6): 3
Which file to be downloaded? file1.txt
File downloaded successfully!
```



'downloaded_file1.txt' added in the client directory.

5. Resource allocation through Banker's Algorithm

```
Input your option (1-6): 4
Server response: Resource allocation successful!
Input your option (1-6): 4
Server response: Resource allocation successful!
Input your option (1-6): 4
Server response: Resource allocation successful!
Input your option (1-6): 4
Server response: Resource allocation failed. Unsafe state detected!
Input your option (1-6):
```

6. The server also handles multiple clients from different IPs at a time.

```
Arnava@Abhays-MacBook-Air server % python server_new.py

Server started and listening on 172.31.4.87:9999
Connection established from 172.31.27.79:63482
Connection closed by 172.31.27.79:63482
Connection established from 172.31.4.87:62704
```

Conclusion

The File Sharing System offers a robust and efficient solution for distributed file management, resource allocation, and collaboration in networked environments. By implementing process synchronization and resource allocation algorithms, the system ensures secure and reliable file sharing among clients, making it suitable for a wide range of real-world applications.

References

- Socket Programming in Python - https://realpython.com/python-sockets/
- Previous assignments in the course CSL2090 PCS-2, for code templates for implementing socket objects, and banker's algorithm.