Assignment Report

Introduction

This report provides an overview of the implementation of the client-server application using Java RMI. The application consists of two main components: the Server and the Client. The Server hosts both file operations and compute operations through separate RMI servers, while the Client can either interact with the file server for file system operations or the compute server for performing computations. This report explains the implementation, what was learned during the process, and the issues encountered.

Implementation

The implementation was divided into the following components:
1. Server
2. Client
3. Utility Classes (Util, DirectoryWatcher, etc.)

Server

The Server component initializes two separate RMI servers: one for file operations and one for computation. It reads configuration values (such as directory paths and ports) from a YAML configuration file and spawns separate threads for the file server and compute server. The file server listens for file operations such as upload, delete, and rename, while the compute server performs basic arithmetic operations and sorting tasks.

Client

The Client component reads from the same YAML configuration file as the Server and determines whether to connect to the file server or the compute server based on the user input. For file operations, it watches a specified directory and synchronizes any changes with the file server. For computation tasks, the client communicates with the compute server using either synchronous or asynchronous RMI calls.

What Was Learned

Throughout the implementation of this assignment, we gained valuable experience in:
1. Working with Java RMI (Remote Method Invocation) for distributed computing.
2. Using YAML for configuration management.
3. Implementing file system watchers to detect and synchronize changes in directories.
4. Working with ExecutorService to manage threads efficiently.
5. Handling synchronous and asynchronous tasks in a client-server architecture.

Issues Encountered

Some of the main challenges encountered during this assignment include:

1. Initial difficulties with setting up RMI and ensuring the proper binding of services on both the client and server sides.
2. Handling I/O exceptions, particularly when reading files or communicating over the network.
3. Managing concurrency issues when multiple clients are accessing the same server resources simultaneously.
4. Properly configuring the file watcher to detect and sync changes efficiently.
5. Debugging and resolving issues related to RemoteException and MalformedURLException.

Conclusion

In conclusion, the implementation of this client-server application provided us with a deeper understanding of distributed computing and multi-threaded Java applications. We successfully implemented both file and compute servers using Java RMI, and the client is capable of synchronizing with the file server or performing computations using the compute server. Despite the challenges faced, the experience was rewarding, and we gained important skills in concurrent programming, RMI, and system architecture.

Team Members

The following team members contributed to this project:

1. Gajjala Ram Prasad Reddy (ID: 1002157117) - Worked on the Server component and YAML configuration.
2. Gadiraju Sai Krishna (ID: 1002153378) - Worked on the Client component and RMI communication.