CS6650 Project2
Yahui Zhang

Executive Summary

The purpose of this assignment was to enhance client-server communication by implementing Remote Procedure Calls (RPC) using Java's Remote Method Invocation (RMI) framework. This transition from traditional socket communication to RMI allows for more abstract and efficient interaction between client and server, enabling clients to invoke methods on the server as if they were local calls.

The scope of the assignment included making the server multi-threaded, allowing it to handle multiple client requests concurrently and ensuring that the application can scale effectively. Multithreading can offer several advantages over single-threaded or sequential programs, such as increased throughput, better responsiveness, resource efficiency, and parallelism.

During the implementation of this project, I gained valuable insights into the workings of RMI. The transition from sockets to RPC simplified the communication model, allowing for cleaner and more maintainable code. I also learned how to create a remote object and bind it to a registry, which serves as a lookup service for clients. One of the significant technical skills I developed was the use of thread pools to make the server multi-threaded. By leveraging Java's ExecutorService, I was able to efficiently manage multiple threads, ensuring that the server could process multiple client requests simultaneously without the overhead of creating new threads for each request.

Furthermore, I discovered the advantages of using ConcurrentHashMap over synchronized collections. This data structure enabled safe and efficient management of shared data in a concurrent environment, reducing contention and improving overall application performance. By minimizing the complexity associated with manual locking mechanisms, ConcurrentHashMap streamlined the development process and enhanced code reliability.

Additionally, I learned two methods for creating a registry for the RMI server. The first method involved manually starting the registry in a separate terminal, while the second utilized the createRegistry() method to programmatically create and export the registry on the local host. I opted for the latter approach, which provided more flexibility and ease of deployment during development and testing.

Overall, this assignment deepened my understanding of concurrent programming, RMI, and the best practices for developing robust client-server applications.