Addis Ababa Institute of Technology

Reflection Report

Fundamentals of Distributed Systems

Simon Dereje Woldearegay
UGR/0952/14
Software Stream

How RPC Simplifies Communication Compared to Socket Programming

RPC abstracts the complexities of low-level socket communication by allowing developers to call remote functions as if they were local. Unlike socket programming, which requires manually managing data serialization, protocol design, and message parsing, RPC automates these processes through frameworks. This reduces development effort and makes the code cleaner and easier to understand. For example, in this lab the client directly invokes methods on the Calculator service without worrying about networking details, enhancing developer productivity and reducing potential bugs.

Describe challenges encountered (e.g., handling timeouts, concurrent clients)

One of the challenges I encountered was understanding the distinction between using rpc.Accept(listener) and manually accepting connections in a loop with rpc.ServeConn(conn) handled concurrently via goroutines. Both methods successfully handled concurrent connections when I implemented both approaches and ran multiple clients simultaneously, likely because the methods were executed quickly, giving the illusion of concurrency. However, I understood that the explicit approach with goroutines is essential for true concurrency, especially with long-running or resource-intensive operations, as it prevents blocking and ensures the server can handle multiple requests efficiently.