Chapter 46: Distributed Systems

1. Introduction to Distributed Systems

- A **distributed system** is a collection of independent computers that appears to its users as a single coherent system.
- **The Central Challenge:** Building a reliable system from unreliable components. Failures (hardware, software, network) are the norm, not the exception.

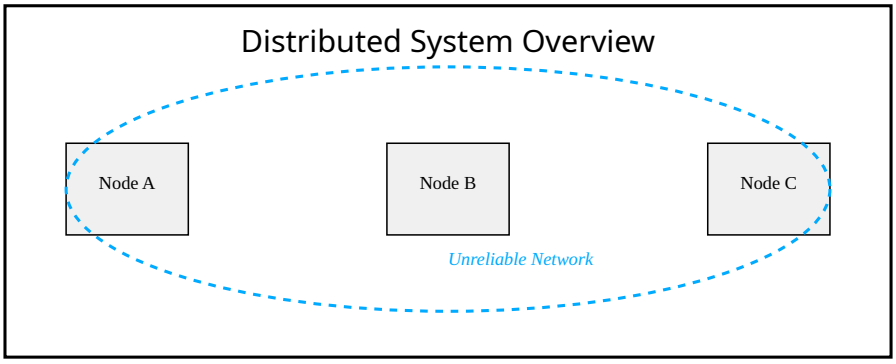


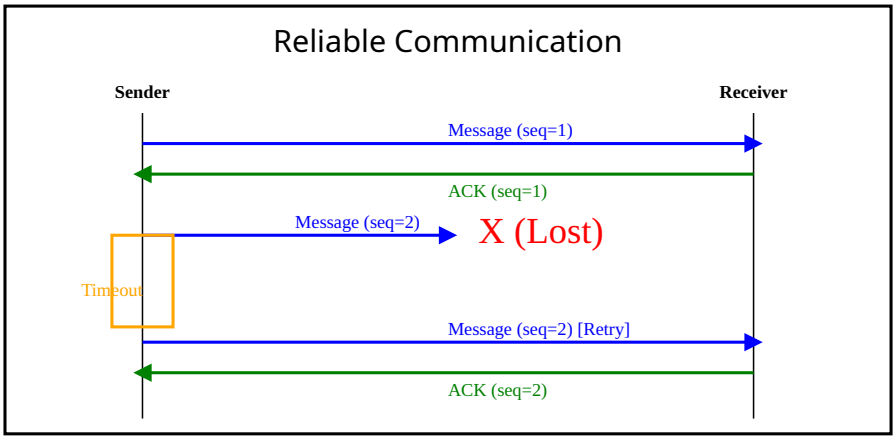
Figure 1: Distributed System Overview

2. Communication: The Foundation

- Communication in a distributed system is inherently unreliable. Packets can be lost, corrupted, or reordered.
- We need to build a reliable communication layer on top of an unreliable network.

2.1. Building Reliable Communication

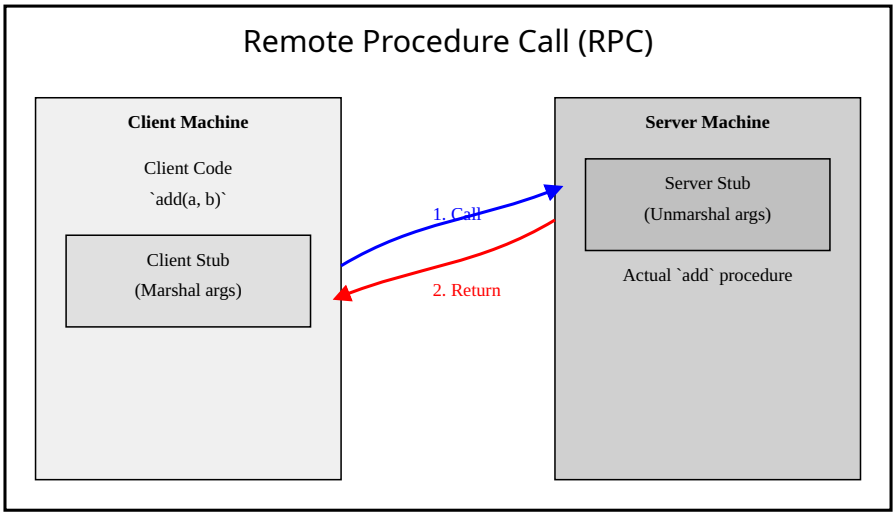
- **Acknowledgments (ACKs):** The receiver sends an ACK to confirm that a message was received.
- **Timeouts and Retries:** The sender starts a timer when it sends a message. If an ACK is not received before the timer expires, the sender re-sends the message.
- **Duplicate Detection:** Messages are given unique sequence numbers so the receiver can identify and discard duplicates.



3. Abstractions for Distributed Communication

3.1. Remote Procedure Call (RPC)

- **RPC** is the dominant abstraction for building distributed systems.
- It makes a function call on a remote machine look and feel just like a local function call.
- **How it works:**
  1. The client calls a **stub function**.
  2. The client stub **marshals** (packs) the arguments into a message and sends it to the server.
  3. The server stub **unmarshals** the arguments and calls the actual procedure.
  4. The server sends the result back to the client.



RPC Code Example

```
// Client code
int result = add(10, 20); // Looks like a normal function call

// Server code
int add(int a, int b) {
    return a + b;
}
```

3.2. Other Abstractions

- **Distributed Shared Memory (DSM):** An older abstraction that tried to make the memory of multiple machines look like a single, shared address space. It is not widely used today.

4. The End-to-End Argument

- This is a key design principle in distributed systems.
- It states that for a process to be truly reliable, the final check must be performed by the application at the "ends" of the system.
- Relying solely on lower-level reliability mechanisms is not enough.

5. Summary

- Distributed systems are essential for modern computing.
- The key challenge is building reliable systems from unreliable components.
- RPC is the most common abstraction for building distributed applications.
- The end-to-end argument is a crucial design principle.