Problem Statement: Transactions in Distributed Systems

Jonathan Shaw
January 31, 2020
shaw5@unbc.ca

1 Introduction

Committing a transaction in a distributed system can be a tricky matter. Before we commit a transaction, we want to be sure that all nodes are able to commit the transaction, and that the transaction will be performed atomically.

2 Definitions and Assumptions

We will model our distributed system as a connected graph, where each node represents a host that may run processes.

- A **transaction** is some action performed by multiple processes in a distributed system.
- Our system consists of a collection of processes called **resource managers (RMs)**, each of which executes on a different host.

We will model our system according to the following assumptions and simplifications:

- The system is modeled by a connected graph of RMs. Each node in the graph, then, can be seen as a host executing a single RM process. The solution should work for an arbitrary number of RMs.
- All RMs in the system participate in every transaction. Thus, the set of RMs participating in the transaction is known in advance.
- No assumptions are made about network topology other than connectedness.
- The edges of the graph represent channels through which connected RMs can communicate. The channel is reliable; messages between nodes are never lost.
- No node in the system is faulty; every node will respond to messages as intended.

3 Problem Statement

Any RM in the system should be able to initiate a transaction. The problem is to develop an algorithm whereby, given enough time, every RM agrees to either commit to or abort the transaction. When this occurs, we say the system has reached **consensus**.

We wish for our solution to have the following characteristics [1]:

- **Safety:** We want to be sure that we'll never reach an unresolvable situation.
- **Non-triviality:** Our algorithm should result in all RMs agreeing to commit to a transaction whenever possible. A solution in which transactions are always aborted is unacceptable.
- **Non-blocking:** Once an RM initiates a transaction, the system should reach consensus within some reasonable amount of time.

4 Importance of Problem

The problem of transaction consensus is fundamental to the study of distributed computing. Since distributed systems are geographically separated, it is necessary, before committing a transaction, to verify with *every* node that the transaction is legal. This ensures that every transaction in a distributed system remains *atomic* and *consistent* across nodes.

5 Applications of Transaction Consensus

Transaction consensus is applied in the following situations, among many more [2]:

- **High-availability systems:** If a system state is replicated across several nodes, transaction consensus can be used to ensure that any changes affect *all* nodes.
- **Cryptocurrencies:** Decentralized currencies require the network to come to consensus as to whether a transaction has occurred, circumventing the need to trust financial institutions.
- **Leader election:** It is often convenient for distributed systems to be coordinated by one node. Transaction consensus provides a mechanism through which a leader can be accepted or rejected by all nodes.

References

[1] J. Gray and L. Lamport, “Consensus on transaction commit,” ACM Trans. Database Syst., vol. 31, no. 1, pp. 133–160, Mar. 2006, doi: 10.1145/1132863.1132867.

[2] P. Krzyzanowski, “Consensus.” [Online]. Available: https://www.cs.rutgers.edu/ pxk/417/notes/content/consensus.html. [Accessed: 31-Jan-2020].