Paxos Made Simple

wu

April 17, 2024

Contents

1 The Consensus Algorithm

1.1 The problem

Assume a collection of processes that can propose values. A consensus algorithm ensures that a single one among the proposed values is chosen. If no value is proposed, then no value should be chosen. If a value has been chosen, then processes should be able to learn the chosen value. The safety requirements for consensus are:

- Only a value that has been proposed may be chosen
- Only a single value is chosen
- A process never learns that a value has been chosen unless it actually has been

The **goal** is to ensure that some proposed value is eventually chosen and, if a value has been chosen, then a process can eventually learn the value.

We let the three roles in the consensus algorithm be performed by three classes of agents: **proposers**, **acceptors** and **learners**.

Assume that agents can communicate with one another by sending messages. We use the customary asynchronous, non-Byzantine model, where:

- Agents operate at arbitrary speed, may fail by stopping, and may restart. Since all agents may fail after a value is chosen and then restart, a solution is impossible unless some information can be remembered by an agent that has failed and restarted.
- Messages can take arbitrarily long to be delivered, can be duplicated and can be lost, but they are not corrupted.

1.2 Choosing a Value

Instead of a single acceptor, let's use multiple acceptor agents. A proposer sends a proposed value to a set of acceptors. An acceptor may **accept** the proposed value. The value is chosen when a large enough set of acceptors have accepted it.

In the absence of failure or message loss, we want a value to be chosen even if only one value is proposed by a single proposer. This suggests the requirement:

P1. An acceptor must accept the first proposal that it receives.

But this requirement raises a problem. Several values could be proposed by different proposers at about the same time, leading to a situation in which every acceptor has accepted a value, but no single value is accepted by a majority of them. Even with just two proposed values, if each is accepted by about half the acceptors, failure of a single acceptor could make it impossible to learn which of the values was chosen.

P1 and the requirement that a value is chosen only when it is accepted by a majority of acceptors imply that an acceptor must be allowed to accept more than one proposal. We keep track of the different proposals, so a proposal consists of a proposal number and a value. To prevent confusion, we require that different proposals have different numbers. A value is chosen when a single proposal with that value has been accepted by a majority of the acceptors. In that case, we say that the proposal has been **chosen**.

We can allow multiple proposals to be chosen, but we must guarantee that all chosen proposals have the same value. By induction on the proposal number, it suffices to guarantee:

P2. If a proposal with value v is chosen, then every higher-numbered proposal that chosen has value v.

P2 guarantees the crucial safety property that only a single value is chosen.

To be chosen, a proposal must be accepted by at least one acceptor. So, we can satisfy P2 by satisfying

P2^a. If a proposal with value v is chosen, then every higher-numbered proposal accpeted by any acceptor has value v.

Because communication is asynchronous, a proposal could be chosen with some particular acceptor c never having received any proposal. Suppose a new proposer "wakes up" and issues a higher-numbered proposal with different value. P1 requires c to accept this proposal, violating P2^a. Maintaining both P1 and P2^a requires strengthening $P2^a$ to:

P2^b. If a proposal with value v is chosen, then every higher-numbered proposal issued by an proposer has value v.

To discover