Baron Alan - Van Dorssen Maud

Paxos Made Moderately Complex

Introduction

Paxos is a protocol widely used in the industry that implements a Replicated State Machine with asynchronous message passing that admits crashes. A state machine consist of a collection of states, a collection of transition between states, and a current state. The different replicas of a replicated state machine aren't necessarily in the same state at a given time, but execute commands in the same order, so they end up in the same state and produce the same sequence of output. State machines are replicated so they can support crashes; it is admitted that at least one replica never crashes, without knowing which one it is.

To achieve this goal, Paxos needs to go through different phases, and is divided in two parts called Single-Decree Paxos and Multi-Decree Paxos.

Single-decree Paxos

Single-decree Paxos achieves **Consensus** on an asynchronous message passing system. **Consensus** happens when processes agree on a value or an instruction. Different processes can propose different values or instructions at the same time. They then need to decide which value is accepted and which values are not. In order to work, Consensus has a few properties that need to be guaranteed. First of all, there are two **Safety** properties. The first says that each process must decide the same value. The second says that each decided value has to have been proposed by at least one process. And finally, there is a **Liveness** property that says that each correct process decides a value.

For this, **Single-decree Paxos** uses a few concepts. The Single-decree Paxos protocol works in rounds, which are represented by **ballot numbers**. Each round, a subset of processes, called **acceptors** receive a value for a certain ballot number from the **leader**. This leader is elected by the acceptors and sends values to the acceptors. Each acceptor then decides if it accepts the value or not. The leader, after receiving enough responses from the acceptors, knows whether value is accepted or not. If more than half of all acceptors accept the value, then the value is accepted. If not, no value is accepted and a new round is started.

The **CAP Theorem** says that it is impossible for any distributed system to guarantee Consistency (Consensus safety properties), Availability (Consensus liveness property) and Partition tolerance (t crashes, with $t \geq n/2$, n the number of processes). In order to guarantee the Consensus properties, the partition tolerance was sacrificed. Paxos supports t crashes, with $t < n/2$ and n the number of processes. Half of all processes need to stay correct for Paxos to work.

The **FLP Theorem** says that there is no deterministic algorithm that resolves Consensus in an asynchronous system and supports even one crashed process. In order to keep supporting crashes, the liveness property was weakened. If all processes manage to elect a common leader for a sufficiently long time (a round), then each correct process decides a value.

Multi-decree Paxos

Multi-decree Paxos uses the Consensus obtained thanks to Single-decree Paxos to create a **Replicated State Machine**. Each **replica** of the Replicated State Machine, when it receive a command, proposes that command for its lowest unused slot number, because each **slot** is indexed by a slot number. Slots with the same slot number for different replicas may not have the same command. The leader receives these proposals from the different replicas and uses Consensus to take a decision. The replicas await a decision from the leader before actually updating their state and computing a response to send back to the client. This way, the different replicas execute commands in the same order so they end up in the same state and produce the same sequence of output.

Conclusion

In conclusion, the replicas of a Replicated State Machine in a distributed system must agree in what order to make transitions. For that, Single-decree Paxos achieves Consensus using a leader and multiple acceptors. The Paxos protocol ensures that they never make inconsistent transitions, and as long as the communication works, progress is made. This is verified thanks to Multi-decree Paxos.