Paxos

Karthik Dantu and Steve Ko

Administrivia

- Assignment 3 is out, due next Monday.

Today: Paxos

- Resources:
  - Part-Time Parliament: http://research.microsoft.com/en-us/um/people/lamport/pubs/lamport-paxos.pdf
  - Paxos Made Moderately Complex: http://www.cs.cornell.edu/courses/cs7412/2011sp/paxos.pdf
  - My lectures: http://www.cse.buffalo.edu/~stevko/courses/cse486/spring14/

Paxos Assumptions & Goals

- The network is *asynchronous* with message delays.
- The network can *lose or duplicate* messages, but *cannot corrupt* them.
- Processes can *crash*.
- Processes are *non-Byzantine* (only crash-stop).
- Processes have *permanent storage*.
- Processes can *propose* values.

- The goal: every process agrees on a value out of the proposed values.

Desired Properties

Safety

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

Liveness

- Some proposed value is eventually chosen
- If a value is chosen, a process eventually learns it

Three Roles of a Process

Proposers: processes that propose values

Acceptors: processes that accept (i.e., consider) values

- "Considering a value": the value is a candidate for consensus.
- Majority acceptance → choosing the value

Learners: processes that learn the outcome (i.e., chosen value)

Three Roles of a Process

In reality, a process can be any one, two, or all three.

Important requirements

- The protocol should work under process failures and with delayed and lost messages.
- The consensus is reached via a majority (> ½).

Example: a replicated state machine

- All replicas agree on the order of execution for concurrent transactions
- All replica assume all roles, i.e., they can each propose, accept, and learn.

Paxos Protocol Overview

A proposal should have an ID.

- (proposal #, value) == (N, V)
- The proposal # strictly increasing and globally unique across all proposers

Three phases

- Prepare phase: a proposer learns previously-accepted proposals from the acceptors.
- Propose phase: a proposer sends out a proposal.
- Learn phase: learners learn the outcome.

Paxos Phase 1

A proposer chooses its proposal number N and sends a *prepare request* to acceptors.

- "Hey, have you accepted any proposal yet?"

An acceptor needs to reply:

- If it accepted anything, the accepted proposal and its value with the highest proposal number less than N
- A promise to not accept any proposal numbered less than N any more (to make sure that it doesn't alter the result of the reply).

Paxos Phase 2

If a proposer receives a reply from a majority, it sends an *accept request* with the proposal (N, V).

- V: the value from the highest proposal number N from the replies (i.e., the accepted proposals returned from acceptors in phase 1)
- Or, if no accepted proposal was returned in phase 1, a new value to propose.

Upon receiving (N, V), acceptors either:

- Accept it
- Or, reject it if there was another prepare request with N' higher than N, and it replied to it.

Paxos Phase 3

Learners need to know which value has been chosen.

Many possibilities

One way: have each acceptor respond to all learners

- Might be effective, but expensive

Another way: elect a “distinguished learner”

- Acceptors respond with their acceptances to this process
- This distinguished learner informs other learners.
- Failure-prone

Mixing the two: a set of distinguished learners

Problem: Progress (Liveness)

- *There's a race condition for proposals.*
- P0 completes phase 1 with a proposal number N0
- Before P0 starts phase 2, P1 starts and completes phase 1 with a proposal number N1 > N0.
- P0 performs phase 2, acceptors reject.
- Before P1 starts phase 2, P0 restarts and completes phase 1 with a proposal number N2 > N1.
- P1 performs phase 2, acceptors reject.
- …(this can go on forever)

Providing Liveness

- Solution: elect a distinguished proposer
  - I.e., have only one proposer
- If the distinguished proposer can successfully communicate with a majority, the protocol guarantees liveness.
  - I.e., if a process plays all three roles, Paxos can tolerate failures $f < 1/2 * N$.
- Still needs to get around FLP for the leader election, e.g., having a failure detector

Multi-Paxos

- In practice, single-decree Paxos is often not used.
- Multi-decree Paxos: Paxos for a sequence of values
  - One possibility: single Paxos instance for each value
  - Other possibilities exist.

Practical Application

- Scenario: Replicated Web servers
- How would you run Paxos to replicated Web servers?
  - What's the problem?
- One possibility
  - Each replica has "request slots" to fill.
  - A client communicates with one replica.
  - That replica becomes a proposer.
  - Run multi-Paxos to fill each request slot with a request.
  - Liveness?

Summary

- Paxos
  - A consensus algorithm
  - Handles crash-stop failures (f < 1/2 * N)
- Three phases
  - Phase 1: prepare request/reply
  - Phase 2: accept request/reply
  - Phase 3: learning of the chosen value