Paxos

Paxos

- Basis of consensus protocol at the heart of many replicated systems
  - Apache Zookeeper, Google Chubby lock service,…
- Submitted for publication by Leslie Lamport first in 1989, finally published in ACM TOCS in 1998
  - Presented in terms of the functioning of a parliament in ancient Greek island
    - Most people found it hard to read and understand
  - Presented in a simpler form in 2003 by Lamport
    - Ok for overall understanding of the protocol. Should read the original paper for full understanding and implementation reference.
  - Interesting history behind its publication
    - See Paxos under https://lamport.azurewebsites.net/pubs/pubs.html (Interesting site to read for other works of Lamport also)

Consensus Problem

- A set of processes, each proposes a value
- Requirements:
  - All correct processes must choose a single value (Agrement)
  - The value chosen must be the value proposed by one of the correct processes (Validity)
- All processes must be able to learn the value chosen
  - No process should learn a value unless it is actually chosen

- Somewhat different than the consensus problem we studied in the context of Byzantine General's problems earlier
  - Lets refer to it as Agreement problem just to distinguish
- Which one is harder?
  - If you are given solution to this version of consensus, agreement can be solved
  - If you are given solution to agreement, this version of consensus may not be solved
    - If the values are different, agreement can agree on any value (not necessarily one of the values proposed, which is a requirement for this version of consensus)

Model

- Asynchronous
- Messages can be lost, duplicated, delayed, but not corrupted
- Nodes may crash and recover. Stable storage available to store data needed for recovery.

Agents in Paxos

- Three types of agents
  - Proposer – proposes value
  - Acceptor – accepts a value
  - Learner – learns the chosen value
- "Accept" and "Choosing" are different
  - A value is **chosen** if a majority of the acceptors accept it
- A node can be only proposer, only acceptor, only learner, or any combination of the three
  - Not important for the correctness of the protocol, or for understanding it
  - In practice, usually the same node will play the role of all three

Proposal

- A **proposal** is a two tuple
  - A proposal number
  - A value
- Proposal numbers are unique across the system
  - No two proposals are issued with the same proposal number
- Proposal numbers can be totally ordered

Paxos Protocol

- Two phases
- Phase 1
  - A proposer sends a Prepare(n) message to a set of acceptors (≥ majority) with a proposal number n
  - An acceptor, on receiving a Prepare(n) message, if it has not already replied to any Prepare message with a proposal number higher than n, replies with a Promise(n, (k, v)) message where k is the proposal number and v is the value in the highest number proposal the acceptor has accepted so far, if any (else just send n)
    - By a Promise message, the acceptor is promising the proposer that it will from now not respond to any proposal with a lower proposal number
      - But it can respond to one with a higher proposal number

- Phase 2:
  - If the proposer receives a Promise(n,…) message from a majority of the acceptors, it
    - Creates a proposal with proposal number n and value v in the highest numbered proposal accepted by any of the acceptors (received in the Promise message from that acceptor)
      - Can use any value if no acceptor has accepted any proposal so far
    - Sends an Accept message with this proposal to all acceptors
  - When the acceptor receives the Accept message, it accepts the proposal if and only if it has not responded to any Prepare message with a higher proposal number
    - Can send back an Accepted message to inform the proposer, though not needed for correctness

Learning the Chosen Value

- Easier, many possibilities
- Acceptors can inform Learners what value they accepted
  - All learners or a special one which can inform others
- Learners can ask Acceptors what value they accepted
- Learners know a value is chosen when a majority of the acceptors report they have accepted that value
- But what if a majority of acceptors accept and then one or more fails such that no majority of chosen value among live acceptors?

Need for Stable Storage

- An acceptor should store
  - The highest numbered proposal it has accepted so far
  - The highest proposal number it has responded to in a Promise() message
- A proposer must store
  - The highest proposal number it has sent out so far
    - Avoids sending two proposals with the same number

Paxos Guarantees

- Once a proposal with proposal number n and value v is chosen, all subsequent proposals with number > n issued by any proposer has value v
  - Also implies that all subsequent proposals accepted by any acceptor has value v
- Ensures that if a value is chosen
  - Only a single value is chosen
  - It is obvious that the value chosen will be one of the values proposed by a proposer
  - So satisfies both safety conditions
- But does it guarantee that a value will be chosen (liveness)?

Possible Livelock

- Proposer p sends Prepare(n) to all acceptors
- All acceptors respond with Promise(n,…) to p
- p sends Accept(n,…) to all acceptors
- Proposer q sends Prepare(m) to all acceptors with m > n
- All of q's Prepare() messages reach the acceptors before any of p's Accept() messages
- All acceptors respond with Promise(m,…) to q
- All acceptors reject p's Accept() message
- p times out and sends a new Prepare(r) with r > m
- This repeats so that no acceptor accepts either p or q's proposal

- Can be prevented from reaching consensus in other scenarios with fault also
  - All Prepare messages are always lost
  - Proposer gets Promise message from a majority, sends Accept, but some of the Accept messages are lost such that no majority accepts the value proposed
  - Other cases possible

- So Paxos does not guarantee consensus will be reached
- Only says consensus will be reached eventually if too many bad things do not happen (bad timings, loss of critical messages at critical times,….)
- This is expected, as otherwise Paxos would have solved consensus for asynchronous systems with unreliable links and crash faults
  - Would have contradicted the FLP (Fisher-Lynch-Patterson) Impossibility result that consensus is impossible to achieve in an asynchronous system even in the presence of a single crash fault

How to stop Livelock?

- Elect one proposer as a distinguished proposer
  - Leader Election
- Only that proposer can propose values
- But what if it fails?
  - Need to elect leaders in the presence of fault
  - Another agreement problem
    - All nodes need to agree on the leader

But why study Paxos?

- Basis of implementing State Machine Replication
  - Replicas are state machines
  - Replicas maintain logs of requests
  - Order of requests in logs of all replicas are kept same (total order)
  - Replicas apply the requests in order from log, ensures all replicas eventually have the same state

- Using Paxos for State Machine Replication
  - Clients issue request to a replica (say a write request)
  - Each replica can play the roles of all of proposer, acceptor, learner
  - Goal: For each client request, choose a unique position in the log of all nodes for that request
    - Log position i = client request to be executed as the i-th command
    - Paxos invoked to decide what should go in each log position of all replicas
    - One invocation of Paxos for each log position
- But we saw this may cause a livelock!

- Paxos solution: elect a leader
  - All client requests come to the leader (directly from client, or forwarded)
  - Leader decides the log position (say k) of the client command
  - Leader runs consensus to have the command chosen as the k-th command in the k-th invocation
- What if there is no leader, or more than one leader temporarily?
  - May not make progress, but safety is not violated

- Paxos does not specify which algorithm to use for leader election
- Can also use Paxos to install a new "view" (set of participating nodes) in all nodes
  - The value proposed is a set: {"view number", set of live nodes}
  - All nodes agree on a view with Paxos
  - Once they learn the chosen value, lowest id node in the chosen view can declare itself the leader