Paxos Made Simple

Leslie Lamport

Siao-Ting Wang

Outline

- Introduction
- Basic Paxos
  - Achieve consensus on one value
- Multi-Paxos
  - Achieve consensus on a sequence of values

Introduction

- Paxos is a protocol for solving consensus in an unreliable network.
- Assumptions:
  - Processes communicate through messages.
  - Clients and servers are asynchronous, fail-stop failure mode(non-Byzantine model). Processes may fail or restart. Messages may be duplicate or lost, but not corrupted.
- Goal
  - Safety
    - Only a value that has been proposed may be chosen
    - Only a single value is chosen and a process never learns that a value has been chosen unless it actually has been.
  - Liveness
    - Some proposed value is eventually chosen and if a value has been chosen, then a process can eventually learn the value.
    - Paxos meets mostly liveness

Basic Paxos

- Problem:
  - Achieve consensus on one value
  - A collection of processes can proposes values. Ensure that there is only one among the proposed values is chosen.
- Three roles:
  - Proposers: Propose requests and try to convince the Acceptors to agree on them.
  - Acceptors: Accept or reject values they receive from the proposers
  - Learners: Learn the results of the consensus.

Basic Paxos

- Strawman 1
  - A proposer sends a proposed value to a set of acceptors.
  - An acceptor must accept the first proposal that it receives. The value that is accepted by a majority of acceptors will be chosen.
  - Problems
    - Several values are proposed by different proposers. If an acceptor can only accept one proposal, leading to no single value is accepted by a majority of them.
- Induction 1:
  - Each acceptors must be allowed to accept more than one proposal.
  - Each proposal must be identified by unique number(totally ordered).
  - Proposal = <unique sequence number, value>

Basic Paxos

- Induction 1
  - Each acceptor may accept more than one proposal. Therefore, multiple proposals may be chosen.
  - Problems:
    - If multiple proposals may be chosen, how to guarantee all chosen proposals have the same value?
- Induction 2:
  - If a proposal with value v is chosen, then every higher-numbered proposal issued by any proposer has value v.
  - Therefore, proposers need to learn the value v and the highest sequence number.

Basic Paxos

- From previous inductions, we can get the following Paxos Algorithm:
  - Two Phases for choosing one value
    - Phase 1(Prepare) [Proposers, Acceptors]
      - Propose proposals and find out the already chosen value if any
    - Phase 2(Accept) [Proposers, Acceptors]
      - Achieve consensus
  - Learn a chosen value [Acceptors, Learners]

Phase 1: Prepare

- Proposers
  - A proposer send a prepare request<n , v> to acceptors.
    - n is proposal sequence number. v is the value proposed by the proposer.
  - Expect to get the safe value from acceptors' response
- Acceptors
  - Acceptors need to keep
    - acceptedSeq. The highest accepted proposal sequence number.
    - acceptedValue. The value that proposed by the highest accepted proposal.
    - hightestSeq: The highest proposal sequence number it has seen.
  - After receiving a prepare request from a proposer with sequence number n, respond with a promise not to accept any more proposals numbered less than n.
    - If n > hightestSeq:
      - hightestSeq = n
      - Return <OK, acceptedSeq, acceptedValue>
    - Else:
      - Return <REJECT>

Phase 2: Accept

- Proposers
  - After receiving a response to its prepare requests (numbered n) from a majority of acceptors, then it sends an accept request to each of those acceptors.
  - A proposer send a accept request<n , v> to acceptors.
    - n is proposal sequence number
    - v is the value of the highest-numbered proposal among the responses, or is any value if the responses reported no proposals.
- Acceptors
  - If an acceptor receives an accept request for a proposal numbered n, it accepts the proposal unless it has already responded to a prepare request having a number greater than n.
    - If n >= hightestSeq:
      - hightestSeq = n
      - acceptedSeq = n, acceptedValue = v
      - Return <OK>
    - Else:
      - Return <REJECT>

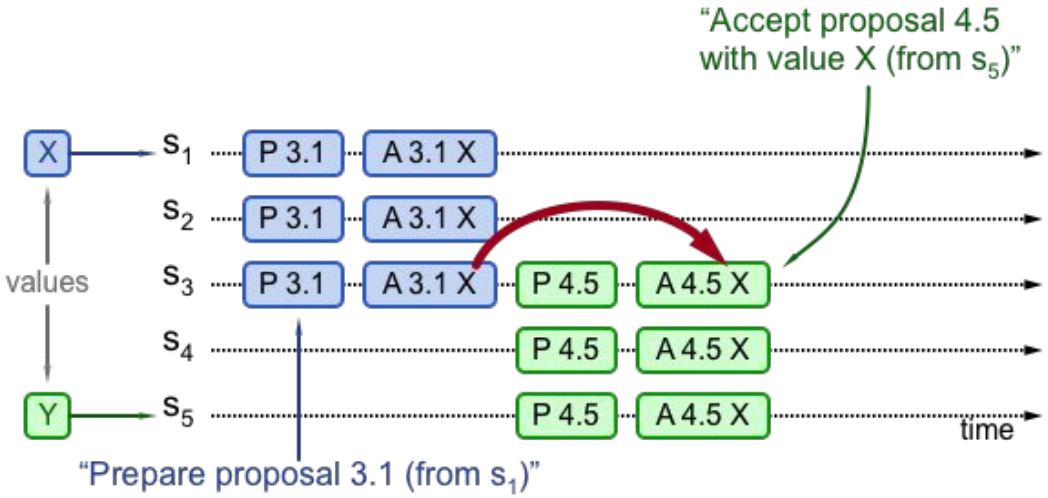
Learn a Chosen Value

- Acceptors responds their acceptance to all learners whenever any acceptor accepts a proposal.
  - The number of responses = (# of Acceptors) * (# of Learners)
- Reduce the number of responses from Acceptors
  - Choose some set of distinguished learners, which informs the other learners when a value is chosen. Acceptors responds their acceptance to those set of distinguished learners.
    - The number of responses = (# of Acceptors) + (# of Learners)

Basic Paxos Example

Three possibilities when later proposal prepares:

1. **Previous value already chosen:**
   - New proposer will find it and use it

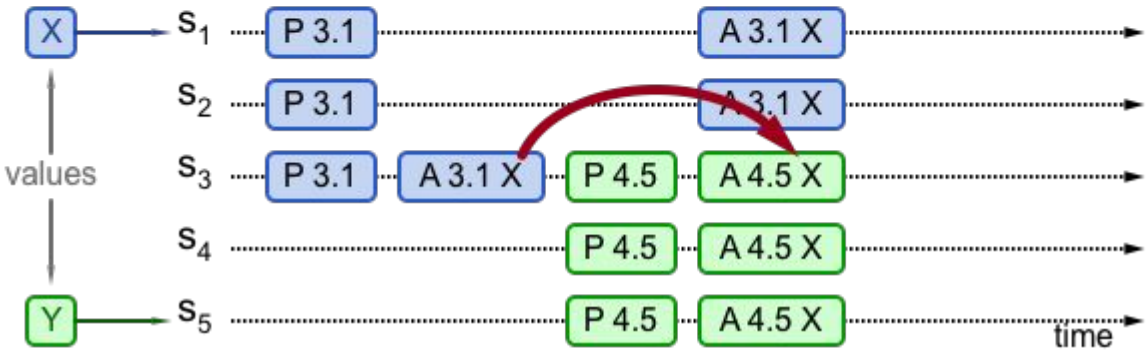


https://ramcloud.stanford.edu/~ongaro/userstudy/

Basic Paxos Example

Three possibilities when later proposal prepares:

2. **Previous value not chosen, but new proposer sees it:**
   - New proposer will use existing value
   - Both proposers can succeed

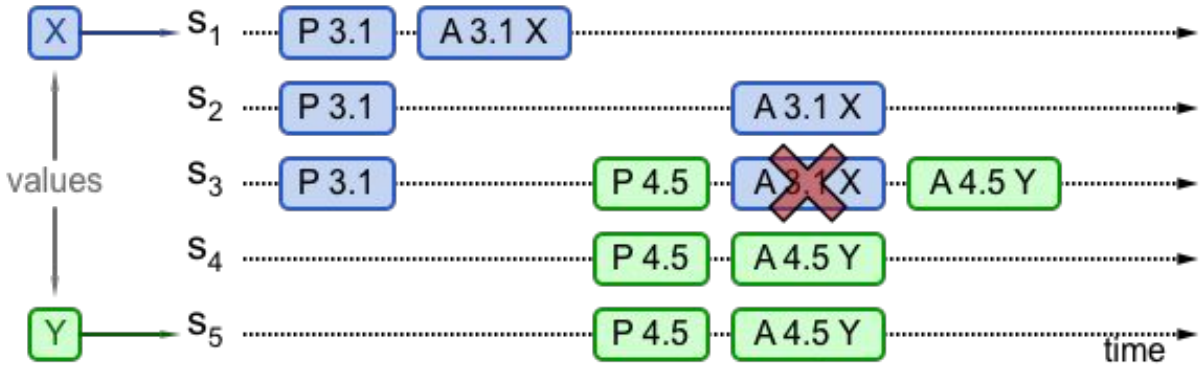


https://ramcloud.stanford.edu/~ongaro/userstudy/

Basic Paxos Example

Three possibilities when later proposal prepares:

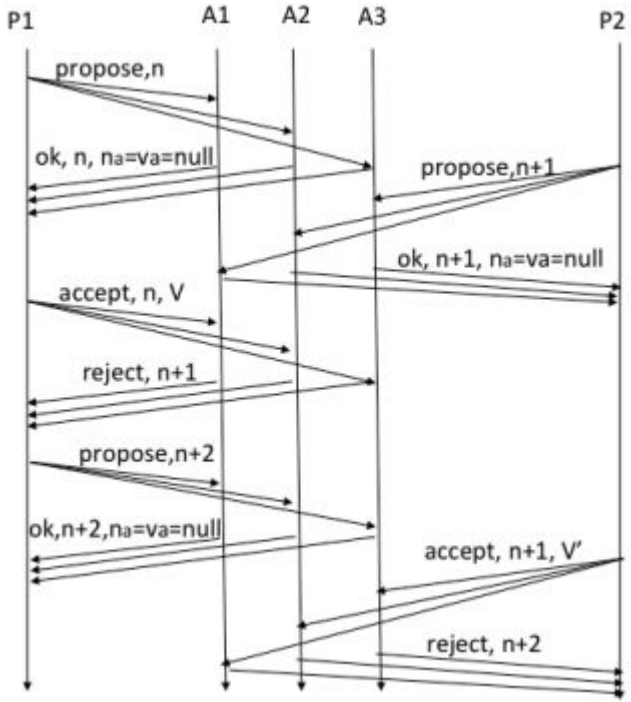
3. **Previous value not chosen, new proposer doesn't see it:**
   - New proposer chooses its own value
   - Older proposal blocked



https://ramcloud.stanford.edu/~ongaro/userstudy/

Enhance Liveness

- Prevent two proposers each keep issuing a sequence of proposals with increasing numbers, none of which are ever chosen.
- Solution
  - A distinguished proposer must be selected as the only one to try issuing proposals.
- Question
  - How to select the distinguished proposer?
    - The election requires using randomness or real time (timeouts)
    - Random backoffs.



https://columbia.github.io/ds1-class/lectures/06-paxos.pdf

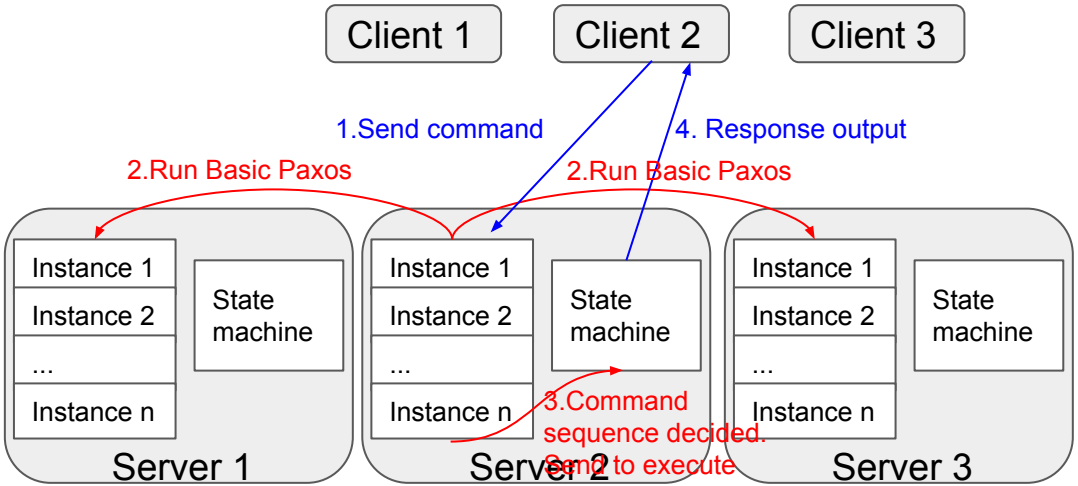
Multi-Paxos

- Achieve consensus on a sequence of values.
  - Base on basic paxos, which achieve consensus on one value.
- State machine example:
  - The system has a collections of severs. The servers are deterministic state machines that performs the client’s command and produce output in some sequence.
  - Clients issue commands to a server.
  - This system uses Multi-Paxos to guarantee that clients and servers see the same sequence of command and output.

Multi-Paxos

- Strawman
  - Run a sequence of separate instances of Basic Paxos to agree on the value of the command sequence. The value chosen by the 10^{th} instance is the the 10^{th} command in the sequence.
  - Each instance of Basic Paxos has its own copy of state.
    - highest proposal sequence number/accepted proposal number/accepted proposal value
  - Problems
    - What' wrong if we execute 10^{th} command when 10^{th} command is chosen without knowing 9^{th} command is chosen or not?
- Induction
  - Server can only execute i^{th} command if
    - $(i - 1)^{th}$ command has been executed.
    - i^{th} Paxos instance has chosen a value.

Multi-Paxos


Client 1
Client 2
Client 3
1.Send command
4. Response output
2.Run Basic Paxos
2.Run Basic Paxos
Instance 1
Instance 2
...
Instance n
State machine
Server 1
Instance 1
Instance 2
...
Instance n
State machine
3.Command sequence decided. Send to execute
Server 2
Instance 1
Instance 2
...
Instance n
State machine
Server 3

Multi-Paxos

- Improve efficiency and liveness
  - Using Basic Paxos to decide every command among all instances is inefficient. With multiple concurrent proposers, conflicts and restarts are likely
  - Leader election
    - A server elected to be a leader, which acts as the distinguished proposer.
    - Clients send commands to the leader and the leader decide the command sequence.
    - Normal operation with one leader
      - One leader server without failures
      - One previous failed leader server and a new leader is selected.
        - For those commands the new leader don’t know, just execute the Basic Paxos algorithm to decide the next client’s command or no-op command.
    - Challenges during leader changes
      - No server acts as leader.
        - No command will be proposed.
      - Multiple servers think they are all leaders. Then they all propose values in the same instance which could prevent any value to be chosen.

Thanks

Reference

- http://www.news.cs.nyu.edu/~jinyang/fa17-ds/notes/paxos.pdf
- https://raft.github.io/slides/raftuserstudy2013.pdf
- https://www.youtube.com/watch?v=JEpsBg0AO6o&t=1879s
- https://ramcloud.stanford.edu/~ongaro/userstudy/
- https://columbia.github.io/ds1-class/lectures/06-paxos.pdf