CSE 486/586 Distributed Systems
Paxos --- 2

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Recap

- Paxos is a consensus algorithm.
  – It allows multiple acceptors accepting multiple proposals.
- A proposer always makes sure that,
  – If a value has been chosen, it always proposes the same value.
- Plan
  ✓ Brief history
  ✓ The protocol itself
  – How to "discover" the protocol
  – A real example: Google Chubby

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

What We'll Do Today

- Derive the requirements we want to satisfy.
- See how Paxos satisfies these requirements.
- This process shows you how to come up with a distributed protocol that has clearly stated correctness conditions.
  – No worries about corner cases!
  – We can learn what Paxos is covering and what it's not.

Review: Assumptions & Goals

- The network is *asynchronous* with message delays.
- The network can *lose or duplicate* messages, but *cannot corrupt* them.
- Processes can *crash and recover*.
- 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.

Review: Desired Properties

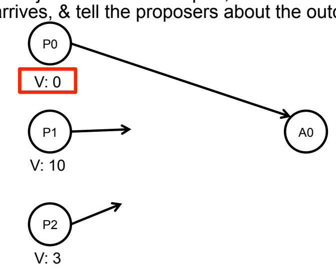
- Safety
  - Only a value that has been proposed can be chosen
  - Only a single value is chosen
  - A process never learns that a value has been chosen unless it has been
- Liveness
  - Some proposed value is eventually chosen
  - If a value is chosen, a process eventually learns it

Review: Roles of a Process

- Three roles
- Proposers: processes that propose values
- Acceptors: processes that accept values
  - Majority acceptance → choosing the value
- Learners: processes that learn the outcome (i.e., chosen value)
- In reality, a process can be any one, two, or all three.

Again, First Attempt

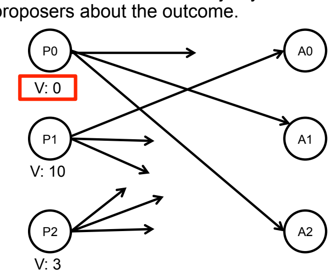
- Let's just have one acceptor, choose the first one that arrives, & tell the proposers about the outcome.



- Why pick the first msg?
  - It should work with one proposer proposing just one value.

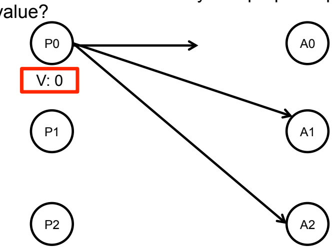
Again, Second Attempt

- Let's have multiple acceptors; each accepts the first one; then all choose the majority and tell the proposers about the outcome.



Again, Second Attempt

- What should we do if only one proposer proposes a value?

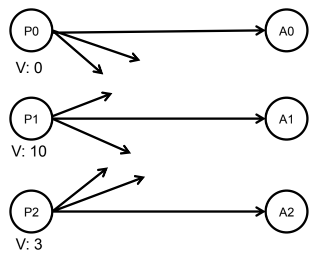


First Requirement

- In the absence of failure or msg loss, we want a value to be chosen even if only one value is proposed by a single proposer.
- This gives our first requirement.
- *P1. An acceptor must accept the first proposal that it receives.*

Problem with the Second Attempt

- One example, but many other possibilities



CSE 486/586 Administrivia

Paxos

- Let's have each acceptor accept *multiple proposals*.
  - Remember: a chosen value should be accepted by a majority of the acceptors.
  - Then we need to make sure that only one proposed value gets a majority vote.
  - Not multiple, not zero.
- Paxos: how do we select one value when there are multiple acceptors accepting multiple proposals?

Accepting Multiple Proposals

- There has to be a way to distinguish each proposal.
  - Let's use a globally-unique, strictly increasing sequence numbers, i.e., there should be no tie in any proposed values.
  - E.g., (per-process number).(process id) == 3.1, 3.2, 4.1, etc.
  - New proposal format: (proposal #, value)
- One issue
  - If acceptors accept multiple proposals, multiple proposals might each have a majority.
  - If each proposal has a different value, we can't reach consensus.

Second Requirement

- We need to guarantee that once a majority chooses a value, all majorities should choose the same value.
  - I.e., all chosen proposals have the same value.
  - This guarantees only one value to be chosen.
  - This gives our next requirement.
- *P2. If a proposal with value V is chosen, then every higher-numbered proposal that is chosen has value V.*

Strengthening P2

- Let's see how a protocol can guarantee P2.
  - *P2. If a proposal with value V is chosen, then every higher-numbered proposal that is chosen has value V.*
- First, to be chosen, a proposal must be accepted by an acceptor.
- So we can strengthen P2:

- *P2a. If a proposal with value V is chosen, then every higher-numbered proposal accepted by any acceptor has value V.*

- By doing this, *we have change the requirement to be something that acceptors need to guarantee.*

Strengthening P2

- Guaranteeing P2a might be difficult because of P1:
  - *P1. An acceptor must accept the first proposal that it receives.*
  - *P2a. If a proposal with value V is chosen, then every higher-numbered proposal accepted by any acceptor has value V.*
- We might violate P2a if we guarantee P1.
  - A proposer might propose a different value with a higher proposal number.
- Scenario
  - A value V is chosen.
  - An acceptor C never receives any proposal (due to asynchrony).
  - A proposer fails, recovers, and issues a different proposal with a higher number and a different value.
  - C accepts it (violating P2a).

Combining P1 & P2a

- Guaranteeing P2a is not enough because of P1:
  - *P1. An acceptor must accept the first proposal that it receives.*
  - *P2a. If a proposal with value V is chosen, then every higher-numbered proposal accepted by any acceptor has value V.*

- *P2b. If a proposal with value V is chosen, then every higher-numbered proposal issued by any proposer has value V.*

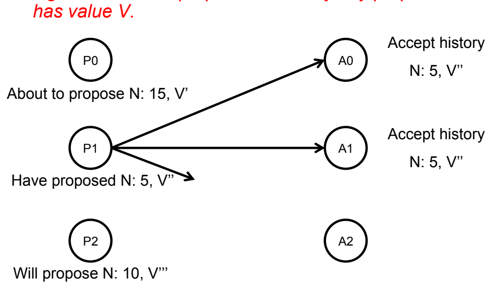
- Now we have changed the requirement P2 to *something that each proposer has to guarantee*.

How to Guarantee P2b

- *P2b. If a proposal with value v is chosen, then every higher-numbered proposal issued by any proposer has value V.*
- In order to guarantee this,
  - A proposer needs to know all proposal N' < N that has been accepted by a majority already.
- Two cases for a proposer
  - A proposer needs to know if there was any N' < N that has been accepted by a majority already. If this is the case, it should propose the same value.
  - A proposer also needs to know if there *will be* any N' < N that will be proposed by some other proposer that will be accepted by a majority.

How to Guarantee P2b

- *P2b. If a proposal with value v is chosen, then every higher-numbered proposal issued by any proposer has value V.*



"Invariant" to Maintain

- *P2c. For any V and N, if a proposal with value V and number N is issued, then there is a set S consisting of a majority of acceptors such that either*
  - *(A) no acceptor in S has accepted or will accept any proposal numbered less than N or,*
  - *(B) V is the value of the highest-numbered proposal among all proposals numbered less than N accepted by the acceptors in S.*

Paxos Phase 1

- A proposer chooses its proposal number N and sends a *prepare request* to acceptors.
- Maintains P2c:
  - P2c. For any V and N, if a proposal with value V and number N is issued, then there is a set S consisting of a majority of acceptors such that either (a) no acceptor in S has accepted or will accept any proposal numbered less than N or (b) V is the value of the highest-numbered proposal among all proposals numbered less than N accepted by the acceptors in S.
- Acceptors need to reply:
  - A promise to not accept any proposal numbered less than N any more (to make sure that the protocol doesn't deal with old proposals)
  - If there is, the accepted proposal with the highest number less than N

Paxos Phase 2

- If a proposer receives a reply from a majority, it sends an *accept request* with the proposal (N, V).
  - V: the highest 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, any value.
- Upon receiving (N, V), acceptors need to maintain P2c by either:
  - Accepting it
  - Or, rejecting 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)
- How to solve this?
  - Next slide

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

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

Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC).