

Consensus and Paxos



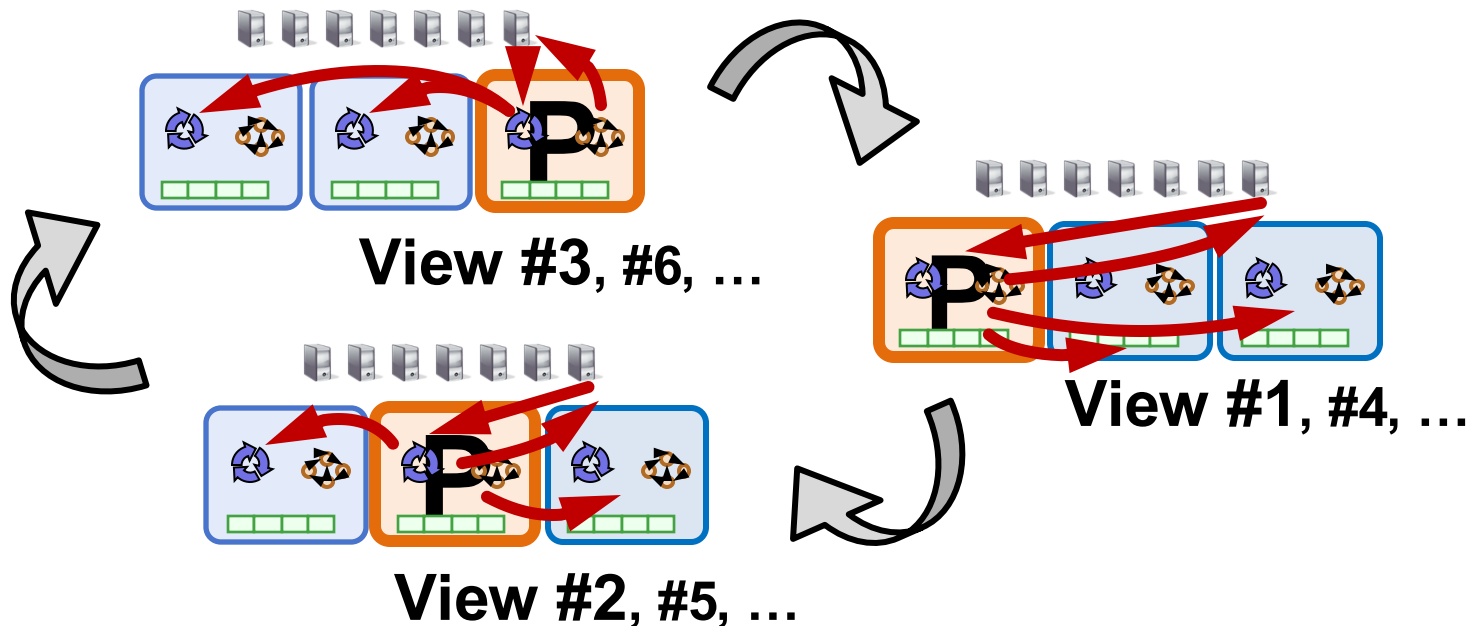
جامعة الملك عبد الله
للعلوم والتقنية
King Abdullah University of
Science and Technology

CS 240: Computing Systems and Concurrency
Lecture 7

Marco Canini

Recall the use of Views

- Let different replicas assume role of primary over time
- **How do the nodes agree on view / primary?**
- **What if both backup nodes attempt to become the new primary simultaneously?**



Consensus

Definition:

1. A general agreement about something
2. An idea or opinion that is shared by all the people in a group

Consensus used in systems

Group of servers attempting:

- Make sure all servers in group receive the same updates in the same order as each other
- Maintain own lists (views) on who is a current member of the group, and update lists when somebody leaves/fails
- Elect a leader in group, and inform everybody
- Ensure mutually exclusive (one process at a time only) access to a critical resource like a file

Consensus

Given a set of processes, each with an initial value:

- **Termination:** All non-faulty processes eventually decide on a value
- **Agreement:** All processes that decide do so on the same value
- **Validity:** The value that has been decided must have been proposed by some process

Recall: Safety vs liveness properties

Safety (bad things never happen)

Liveness (good things eventually happen)

Consensus

Given a set of processes, each with an initial value:

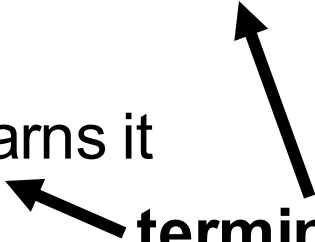
- **Termination:** All non-faulty processes eventually decide on a value ← Good thing that eventually should happen
- **Agreement:** All processes that decide do so on the same value ← Bad thing that should never happen
- **Validity:** The value that has been decided must have been proposed by some process ← Bad thing that should never happen

Paxos properties

Safety

- Only a single value is chosen ← **agreement**
- Only chosen values are learned by processes
- Only a proposed value can be chosen ← **validity**

Liveness

- Some proposed value eventually chosen if fewer than half of processes fail
 - If value is chosen, a process eventually learns it
- termination**
- 

Paxos' safety and liveness

- Paxos is always safe
- Paxos is very often live
 - But not **always**, more later

Roles of a process

- Three conceptual roles
 - **Proposers** propose values
 - **Acceptors** accept values, where chosen if majority accept
 - **Learners** learn the outcome (chosen value)
- In reality, a process can play any/all roles

Strawman

- 3 proposers, 1 acceptor
 - Acceptor accepts first value received
 - No liveness on failure
- 3 proposers (distinct proposals), 3 acceptors
 - Accept first value received, acceptors choose common value known by majority
 - But no such majority is guaranteed

Paxos

- Each acceptor accepts **multiple proposals**
 - Hopefully one of multiple accepted proposals will have a majority vote (and we determine that)
 - If not, rinse and repeat (more on this)
- How do we select among multiple proposals?
 - Ordering: proposal is tuple **(proposal #, value) = (n, v)**
 - Proposal # strictly increasing, globally unique
 - Globally unique?
 - Trick: set low-order bits to proposer's ID

Paxos Protocol Overview

- Proposers:

1. Choose a proposal number n
2. Ask acceptors if any accepted proposals with $n_a < n$
3. If existing proposal v_a returned, propose same value (n, v_a)
4. Otherwise, propose own value (n, v)

Note **altruism**: goal is to reach consensus, not “win”

- **Acceptors** try to accept value with highest proposal n
- **Learners** are passive and wait for the outcome

Paxos Phase 1

- Proposer:
 - Choose proposal number n , send $\langle \text{prepare}, n \rangle$ to acceptors
- Acceptors:
 - If $n > n_h$
 - $n_h = n$ ← promise not to accept any new proposals $n' < n$
 - If no prior proposal accepted
 - Reply $\langle \text{promise}, n, \emptyset \rangle$
 - Else
 - Reply $\langle \text{promise}, n, (n_a, v_a) \rangle$
 - Else
 - Reply $\langle \text{prepare-failed} \rangle$

Paxos Phase 2

- Proposer:

- If receive promise from **majority** of acceptors,
 - Determine v_a returned with highest n_a , if exists
 - Send $\langle \text{accept}, (n, v_a \parallel v) \rangle$ to acceptors

- Acceptors:

- Upon receiving $\langle \text{accept}, (n, v) \rangle$, if $n \geq n_h$,
 - Accept proposal and notify learner(s)

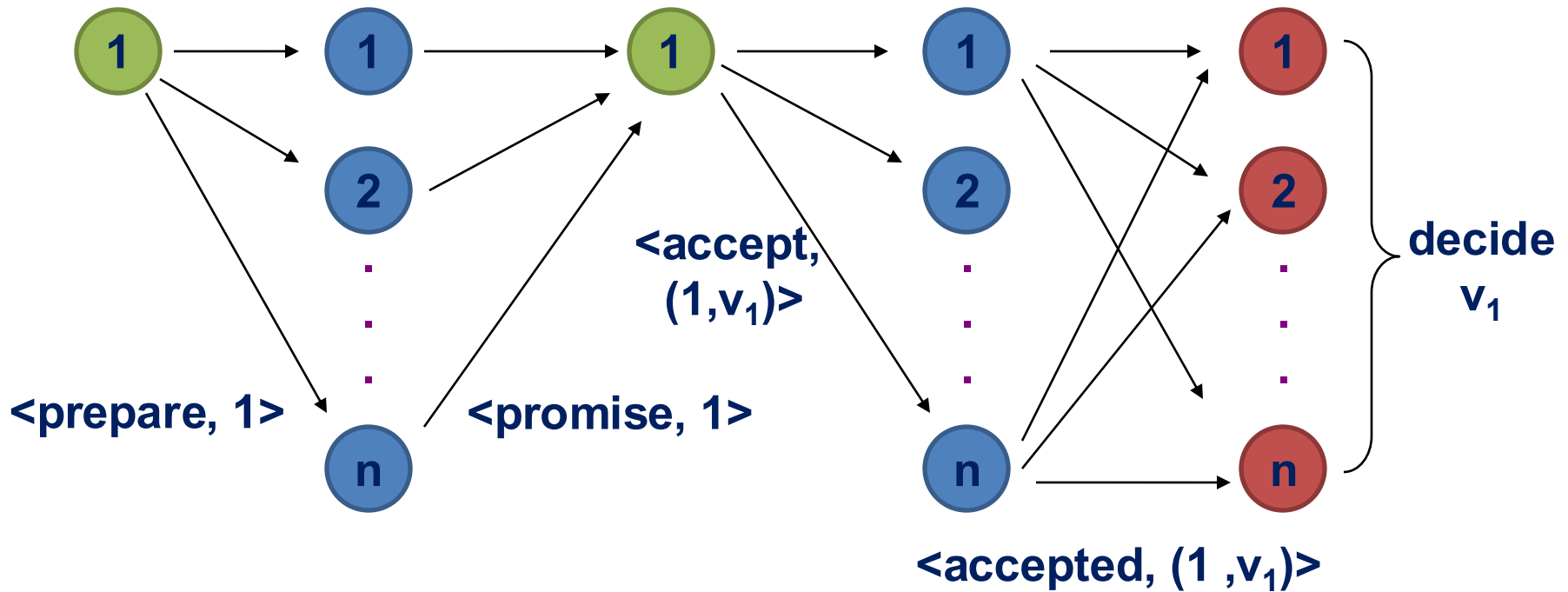
$$n_a = n_h = n$$

$$v_a = v$$

Paxos Phase 3

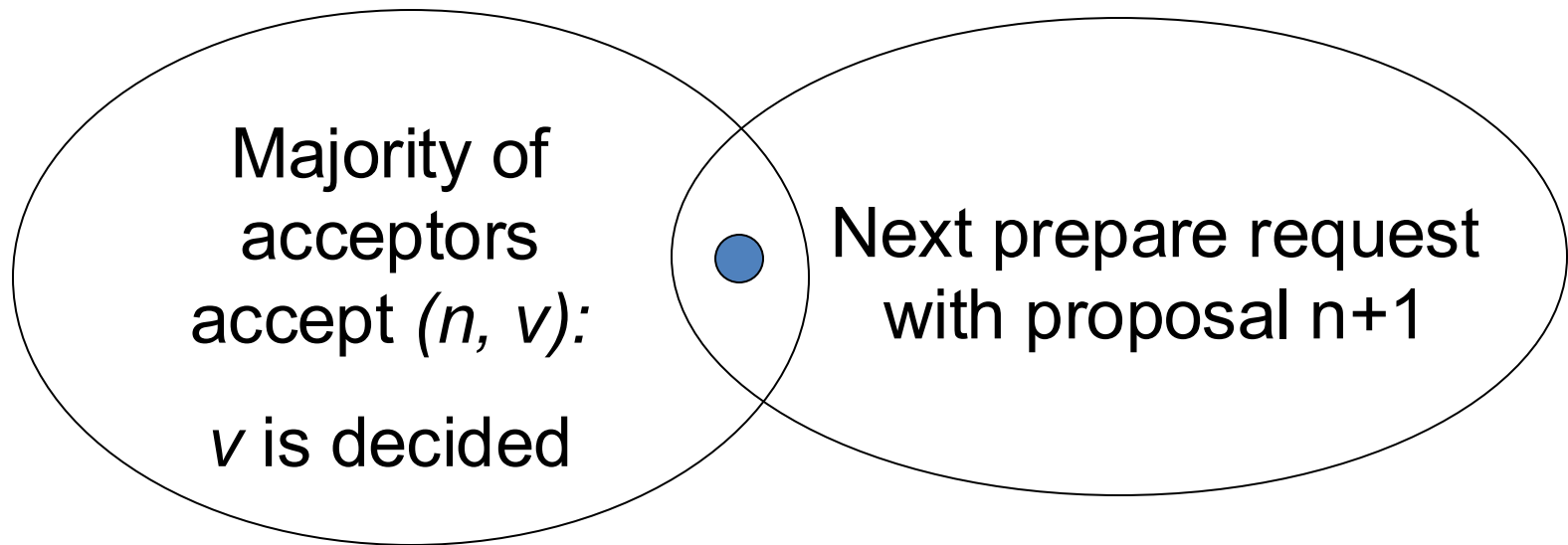
- **Learners** need to know which value chosen
- Approach #1
 - Each acceptor notifies all learners
 - More expensive
- Approach #2
 - Elect a “distinguished learner”
 - Acceptors notify elected learner, which informs others
 - Failure-prone

Paxos: Well-behaved Run

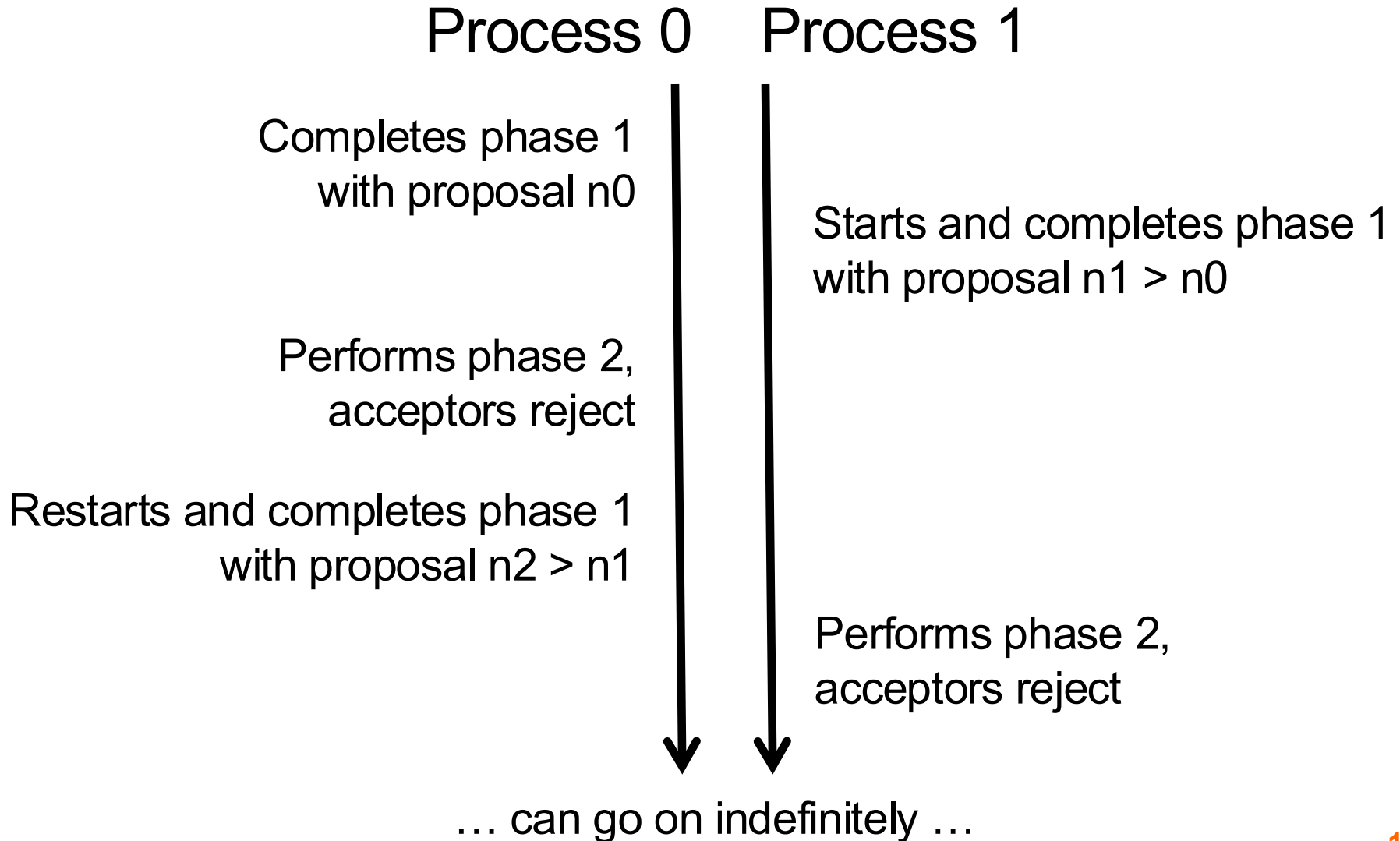


Paxos is safe

- Intuition: if proposal with value v decided, then every higher-numbered proposal issued by any proposer has value v .



Often, but not always, live



Paxos summary

- Described for a single round of consensus
- Proposer, Acceptors, Learners
 - Often implemented with nodes playing all roles
- Always safe: Quorum intersection
- Very often live
- Acceptors accept multiple values
 - But only one value is ultimately chosen
- Once a value is accepted by a majority it is chosen
- Can tolerate failures $f < N / 2$ (aka, **$2f+1$ nodes**)

Flavors of Paxos

- Terminology is a mess
- Paxos loosely, and confusingly defined...
- We'll stick with
 - Basic Paxos
 - Multi-Paxos

Flavors of Paxos: Basic Paxos

- Run the full protocol each time
 - e.g., for each slot in the command log
- Takes 2 rounds until a value is chosen

Flavors of Paxos: Multi-Paxos

- Elect a leader and have it run the 2nd phase directly
 - e.g., for each slot in the command log
 - Leader election uses Basic Paxos
- Takes 1 round until a value is chosen
 - Faster than Basic Paxos
- Used extensively in practice!
 - RAFT is similar to Multi Paxos