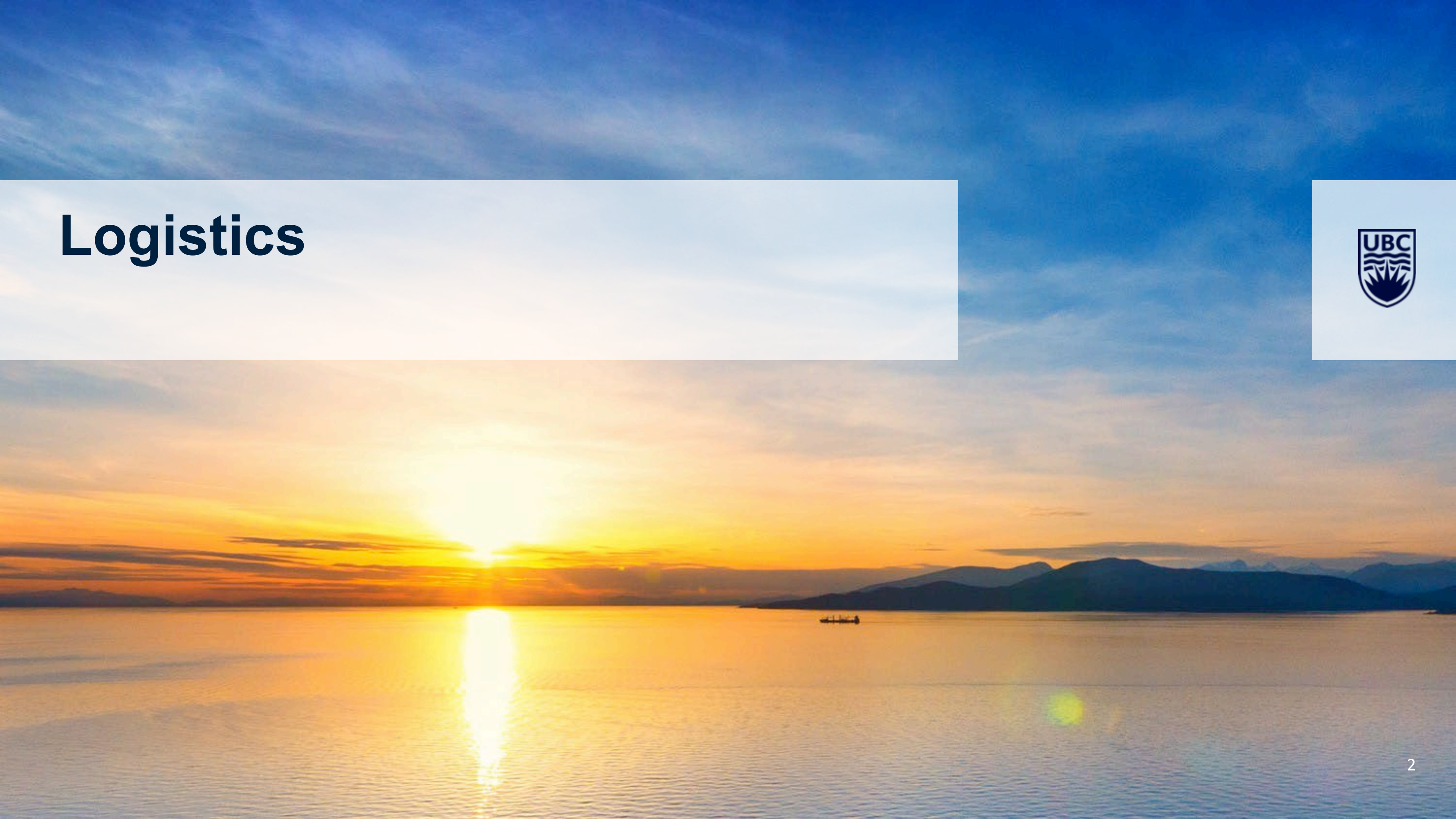
CPSC 416 Distributed Systems

Winter 2022 Term 2 (February 14, 2023)

Tony Mason (fsgeek@cs.ubc.ca), Lecturer



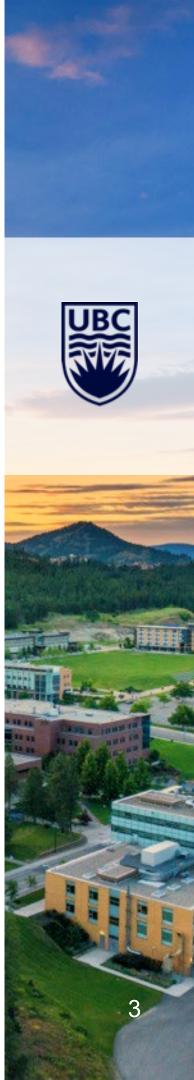
Logistics

Deadlines

Project 3 Released. Late Deadline: April 13, 2023 (except for academic concession cases). Grades pending.

Project 4 Released. Initially Due: March 13, 2023
Project 5 Released Due: April 13, 2023

All project work is due April 13, 2023. Late projects have a 75% score cap.



Deadlines

Alternate Path 1 & 2: Review in progress

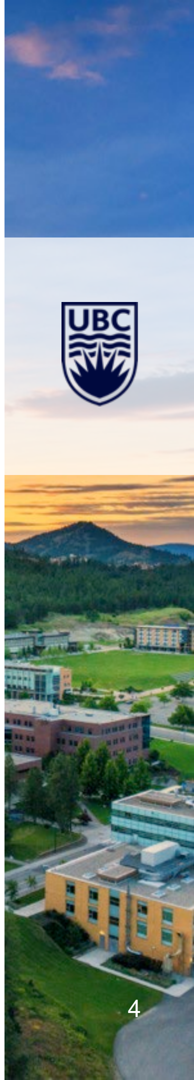
- Piazza private threads need TLC
  - **Weekly updates due each Monday @ 23:59 PT**

Instructor Office Hours:

- Zoom Office Hours (Tuesday) @ 13:00-14:00
- Discord (Casual) Office Hours (Thursday) @ 14:00-15:00

TA Office Hours:

- Eric: Friday 9-11 am (in-person and Zoom)
- Japraj: Wednesday 3-5 pm (Zoom)
- Yennis: Thursday 2-4 (Zoom), Friday 2-4 (in-person)

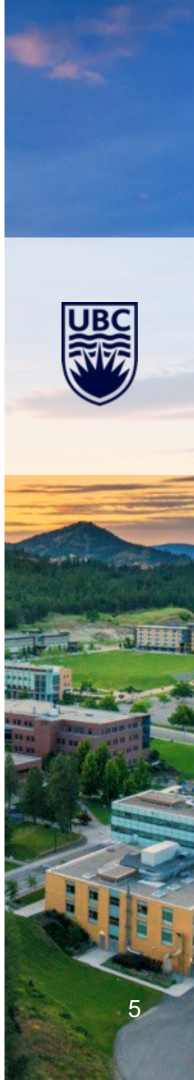


Readings

Required:

Recommended:

- The Part-Time Parliament
- Large-scale cluster management at Google with Borg
- Paxos & Computer Agreement (video)
- Understanding Paxos (Blog)
- Essential Paxos (Python/Java)

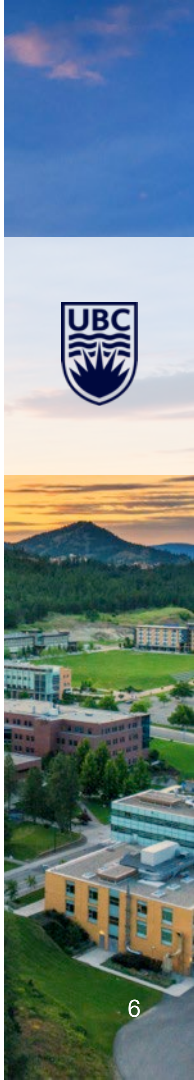


Questions?

Questions about the class?

Questions about the previous lecture?

Funny stories to share?



Today’s Failure

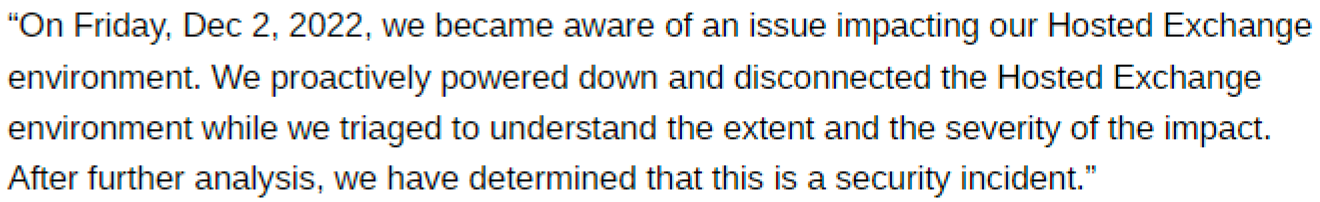
Rackspace Ransomware Outage

Event start: December 2, 2022

Event ends: ?

TL;DR Version

- Rackspace Experienced a ransomware attack (Exchange)
- Solution? Tell customers to *move to Microsoft Hosted Exchange*
- This effectively ended Rackspace’s Hosted Exchange business (US$30m/yr)



“On Friday, Dec 2, 2022, we became aware of an issue impacting our Hosted Exchange environment. We proactively powered down and disconnected the Hosted Exchange environment while we triaged to understand the extent and the severity of the impact. After further analysis, we have determined that this is a security incident.”

Lesson Goals

Quorum Consensus

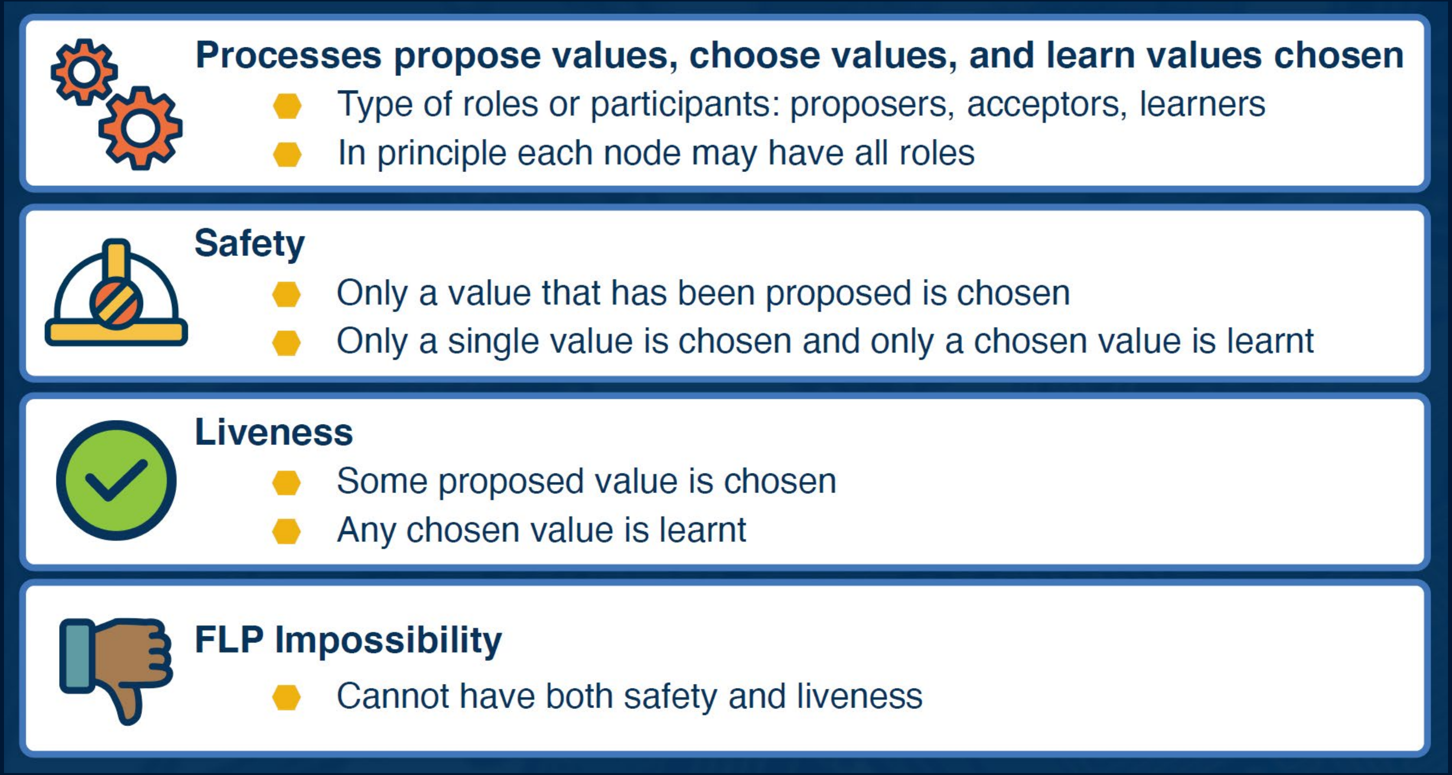
Goals of Distributed Consensus

Transactional Commit

- 2 Phase Commit
- 3 Phase Commit

Paxos

Consensus



Processes propose values, choose values, and learn values chosen

- Type of roles or participants: proposers, acceptors, learners
- In principle each node may have all roles

Safety

- Only a value that has been proposed is chosen
- Only a single value is chosen and only a chosen value is learnt

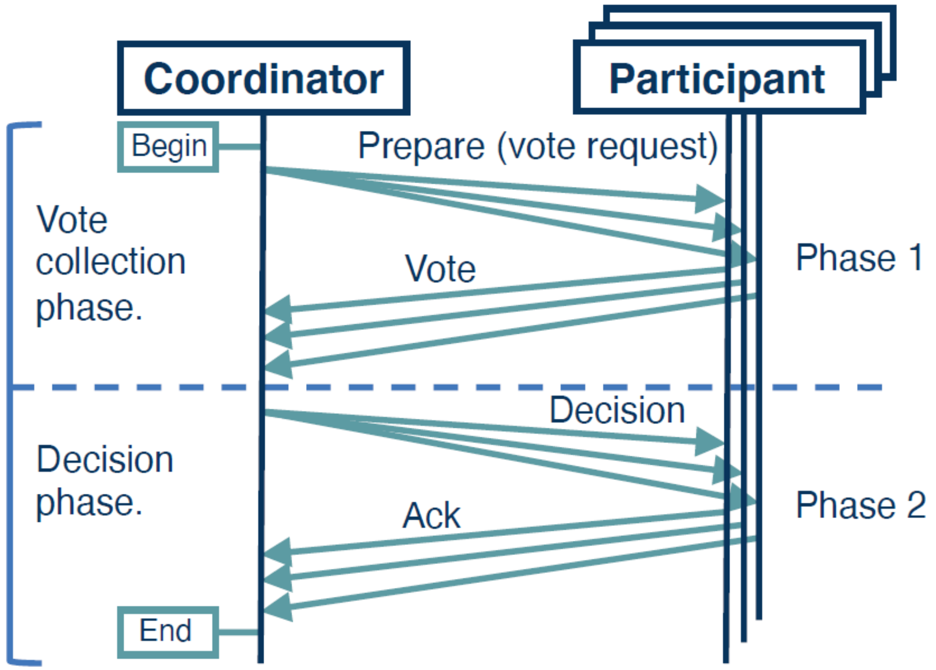
Liveness

- Some proposed value is chosen
- Any chosen value is learnt

FLP Impossibility

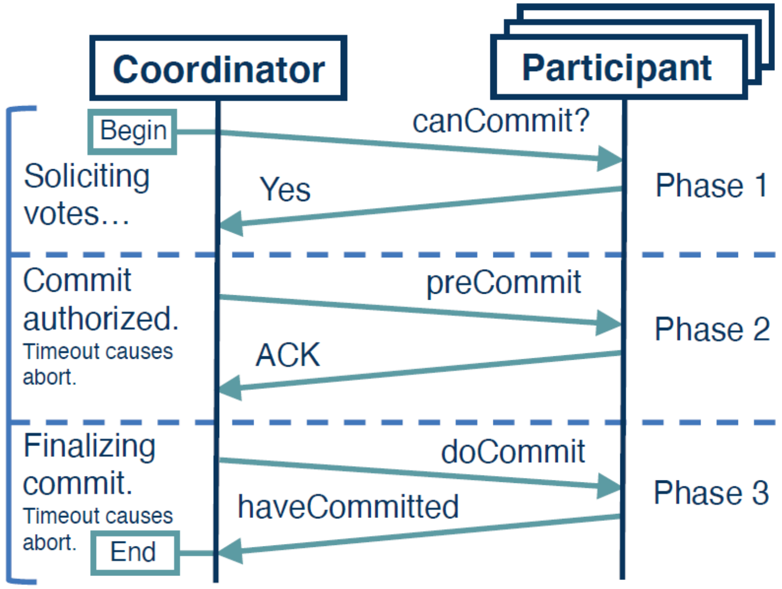
- Cannot have both safety and liveness

2 Phase Commit



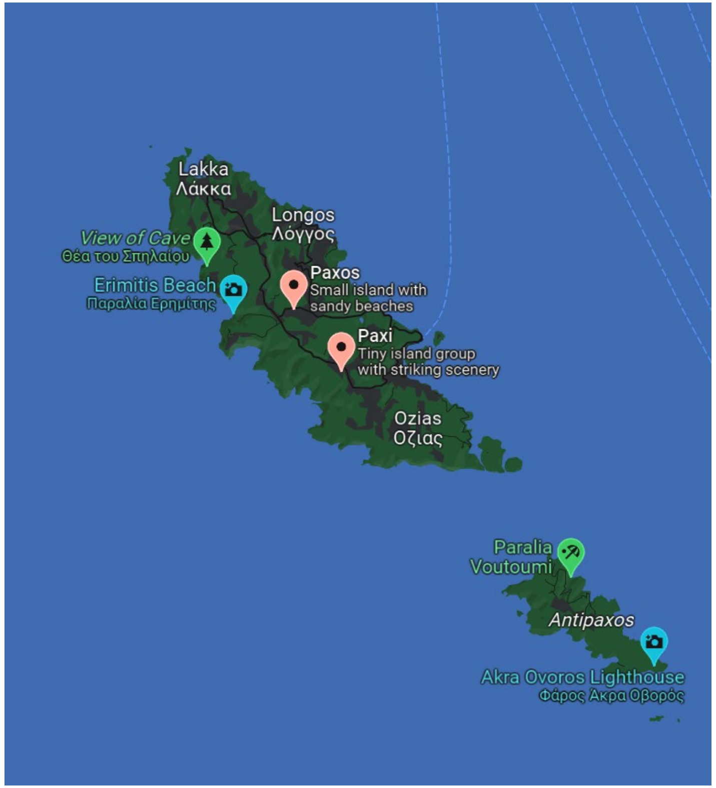
- Blocking, no liveness

3 Phase Commit



- Assumes fail-stop, safety issues on fail-restart

Paxos


Lakka
Λάκκα
Longos
Λόγγος
View of Cave
Θέα του Σπηλαίου
Erimitis Beach
Παραλία Ερημίτης
Paxos
Small island with sandy beaches
Paxi
Tiny island group with striking scenery
Ozias
Οζιας
Paralia Voutoumi
Antipaxos
Akra Ovoros Lighthouse
Φάρος Άκρα Οβορός

Paxos

The part-time parliament

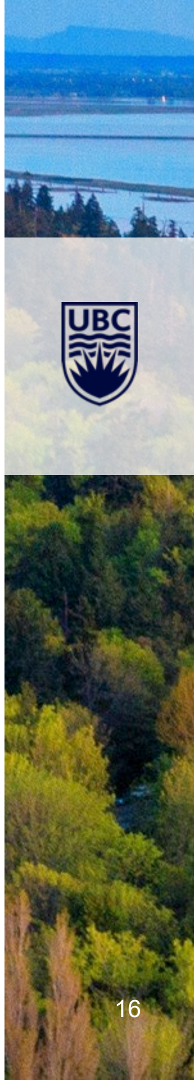
- Submitted to ACM *Transactions on Computer Systems* (TOCS) in 1990
- Accepted/Published in 1998

Describes a (mythical) Greek legislature and their system of asynchronous consensus

Paxon Parliament

Parliament members:

- Pass decrees (decisions)
- Work only part-time
- Communicate only via messages
- Messages are not guaranteed
    - Delay
    - Not sent (“abstain”)
- No malicious actors



Paxos Rules

Algorithm specifies a *state machine*

- Define states
- Define transitions based upon messages (“events”)

Consistency

- Only *proposed* decisions (“values”) can be chosen
- All participants (“nodes”) agree (“single value”)
- Participants only learn a value is chosen *if* it has been (“durability”)

Paper proves the parliament functions *properly*

- Decisions are consistent

Paxos Made Simple

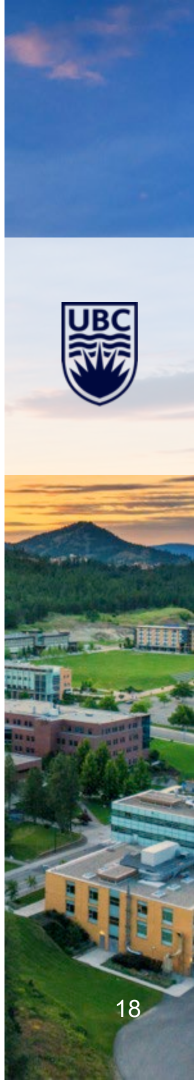
Lamport’s 2001 update

No Greek parliament

No olive tree price agreements

No synods

No fun



Paxos Made Simple

Same model

- Asynchronous
- Non-Byzantine
- Agents act at their own (arbitrary) speed
- Agents may fail (“stop”)
- Agents may recover from failure (“restart”)
- Messages are unreliable
    - Arbitrarily long delivery
    - Duplicated
    - Lost
    - No guaranteed order
    - No message corruption

Key Ideas

State Machine Replication:

- Each node is a replica of the same state machine (algorithm) following the same rules

Majority Quorum:

- Each decision is based on a majority quorum, two quorums are guaranteed to have intersecting members, so consensus decision can be disseminated
- Needed so it can tolerate fail-restart failures

Ordering:

- Everything is timestamped (so it can be ordered)
- Needed so it can tolerate arbitrary message delays

Paxos Phases

Prepare: Node proposes an “agreement” – propose a value/outcome

Accept: Collect votes to

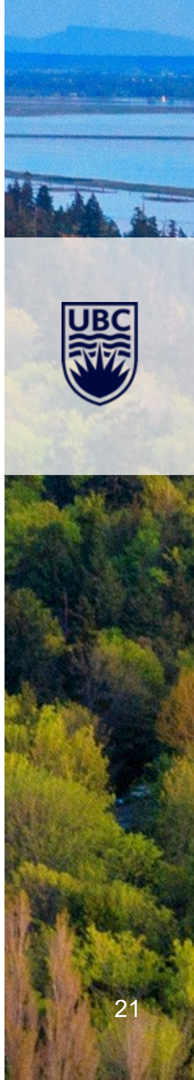
- Determine if an agreement is possible
- Determine a value/outcome

Learn: All nodes can learn the value/outcome

Can be done in **two rounds of messages**

Proposal number is part of the messages

- Can handle fail-restart and delayed messages



Phase 1: Prepare

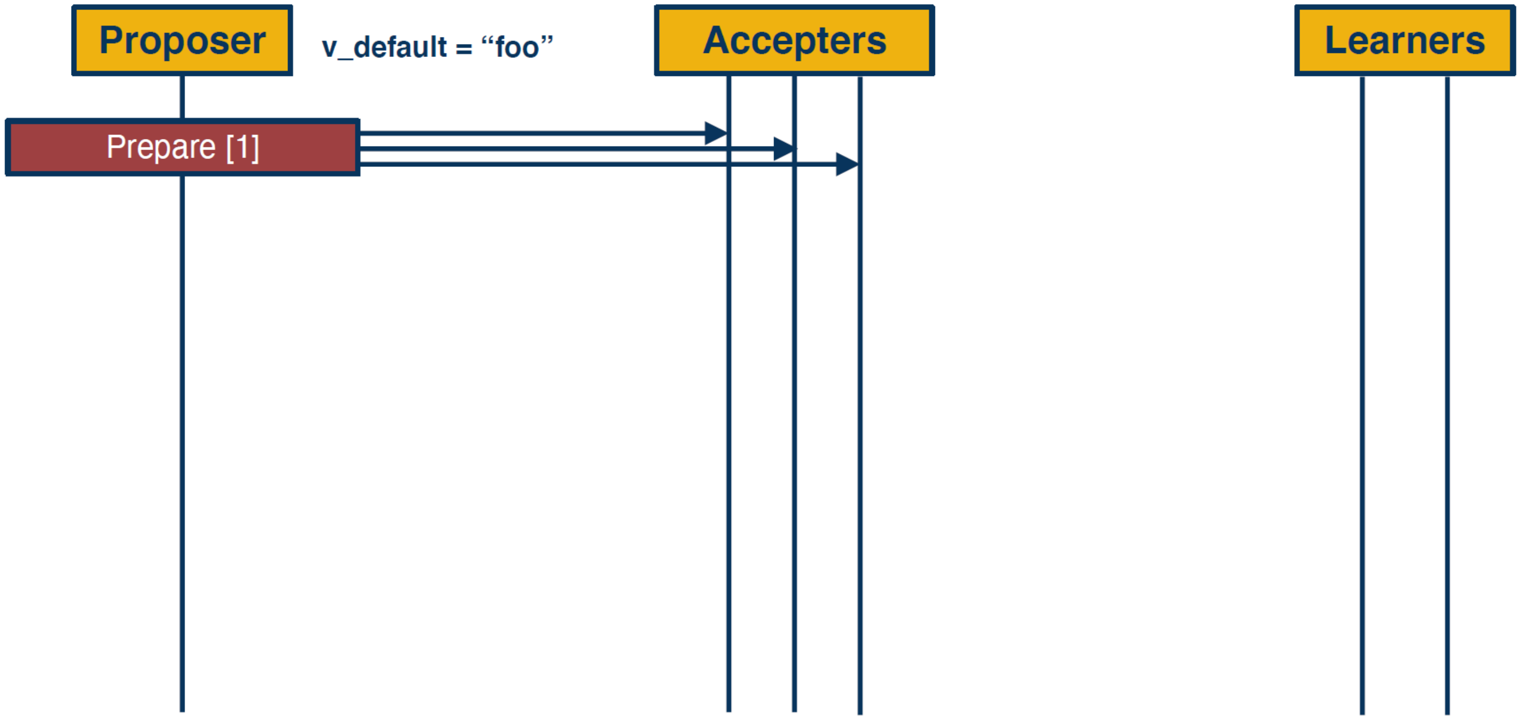
A node (the “leader”):

- Selects a proposal number n
  - n is from a totally ordered set of numbers across nodes
  - Example: There are m nodes, a node $\phi \in \{1, \dots, m\}$ uses $mi + \phi, i \in \mathbb{Z}$
  - No number is ever used twice
- Sends a **prepare request** with the number n to a majority of voting nodes (“acceptors”)

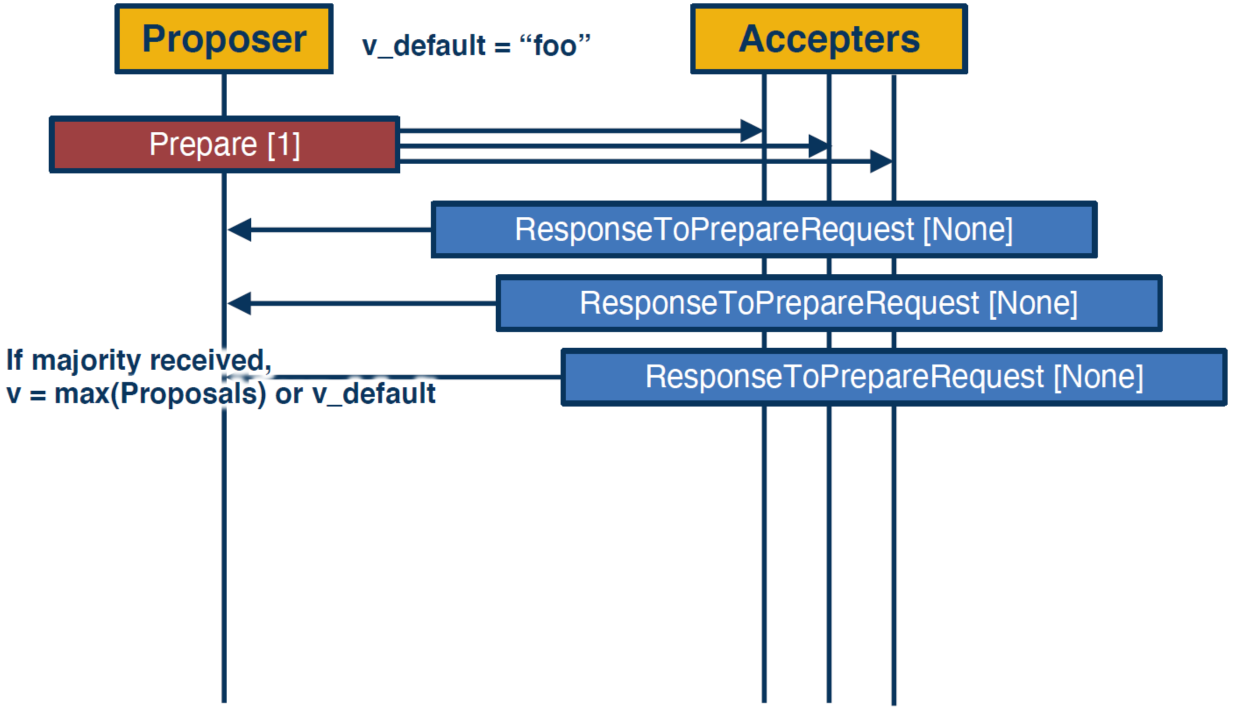
Acceptor:

- If a prepare request receives a proposal number n larger than any it has seen:
  - It responds with a *promise not to accept new proposals less than* n and the highest numbered proposal it has previously accepted.

Proposal: Example (1)



Proposal: Example (2)


Proposer
v_default = “foo”
Accepters
Learners
Prepare [1]
ResponseToPrepareRequest [None]
ResponseToPrepareRequest [None]
ResponseToPrepareRequest [None]
If majority received,
v = max(Proposals) or v_default

Phase 2: Accept

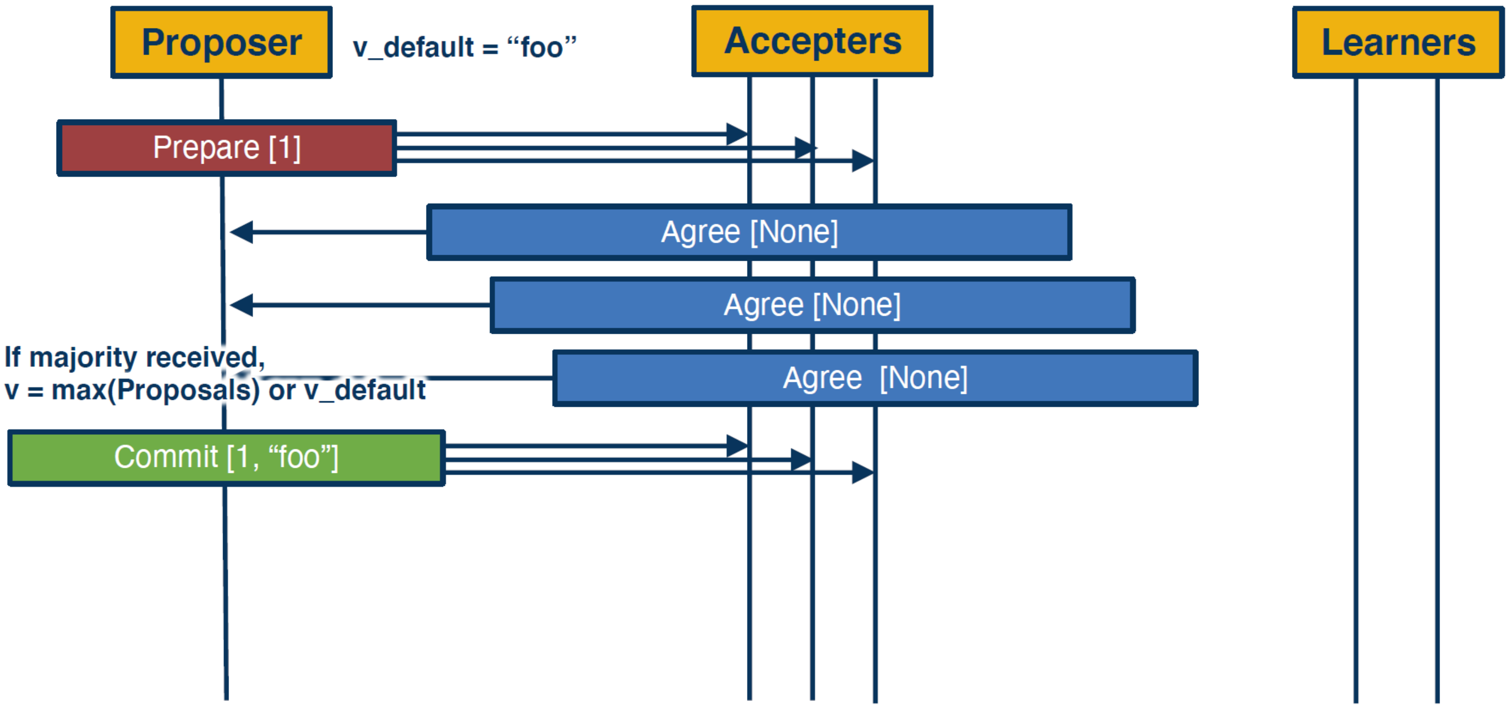
If the Proposer receives a response from a *majority* of acceptors:

- Sends an **accept request** to each acceptor
  - Proposal number (n)
  - Value/outcome (v)
    - The highest-numbered proposal in the *responses*
    - Distinguished value for “no proposal” (e.g., this is the first one)

An Acceptor receives an accept request for proposal n:

- If it has not seen a higher numbered proposal, it accepts
- If it has seen a higher numbered proposal, it doesn’t accept (“drop it”).

Acceptors: Example



Phase 3: Learn

Accepted value becomes **decided** value.

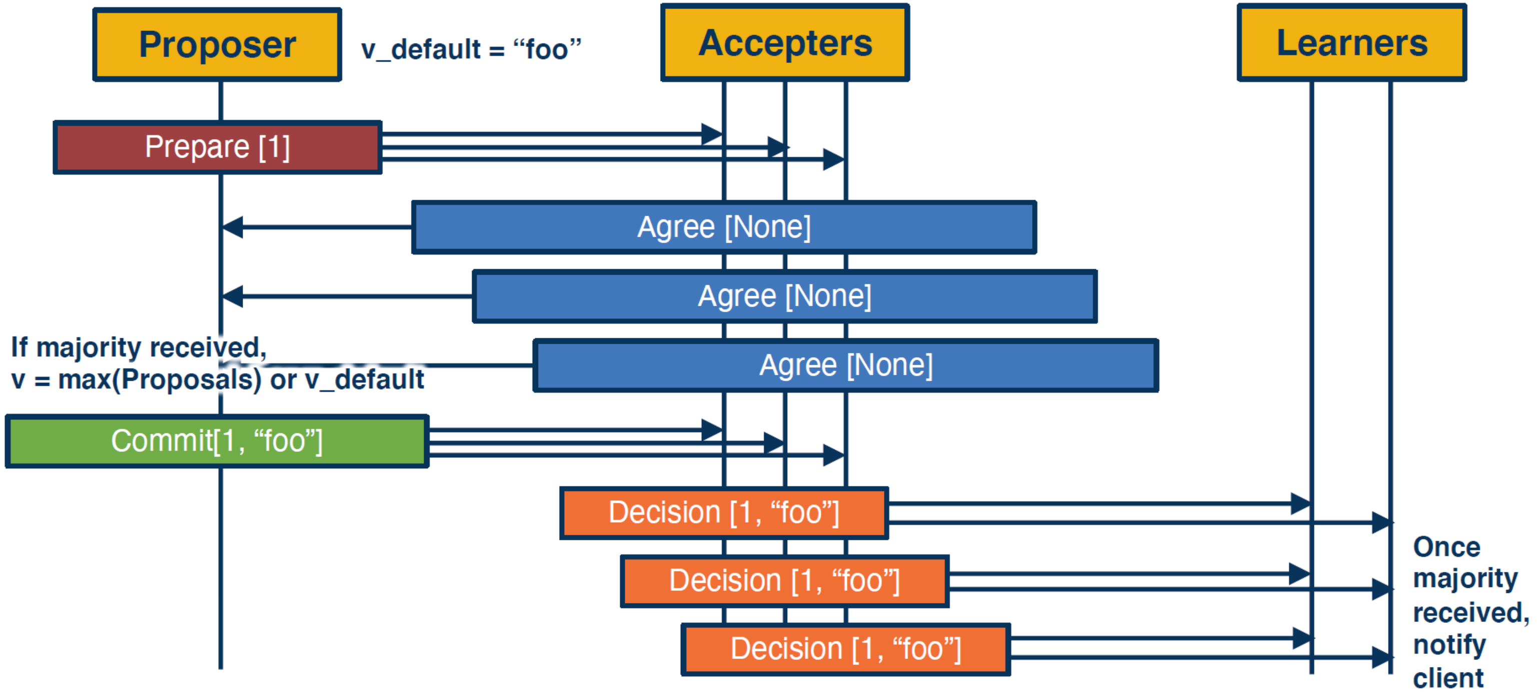
- Achieved quorum
- Visible to learners

Pick one or more *distinguished learners*

- Acceptors notify learners of accepted proposals
- Distinguished learners knows it is **decided** because it is visible to a majority of acceptors (“quorum”)
- Distinguished learners tell other learners

More distinguished learners = higher reliability at higher communications cost/complexity

Learners: Example

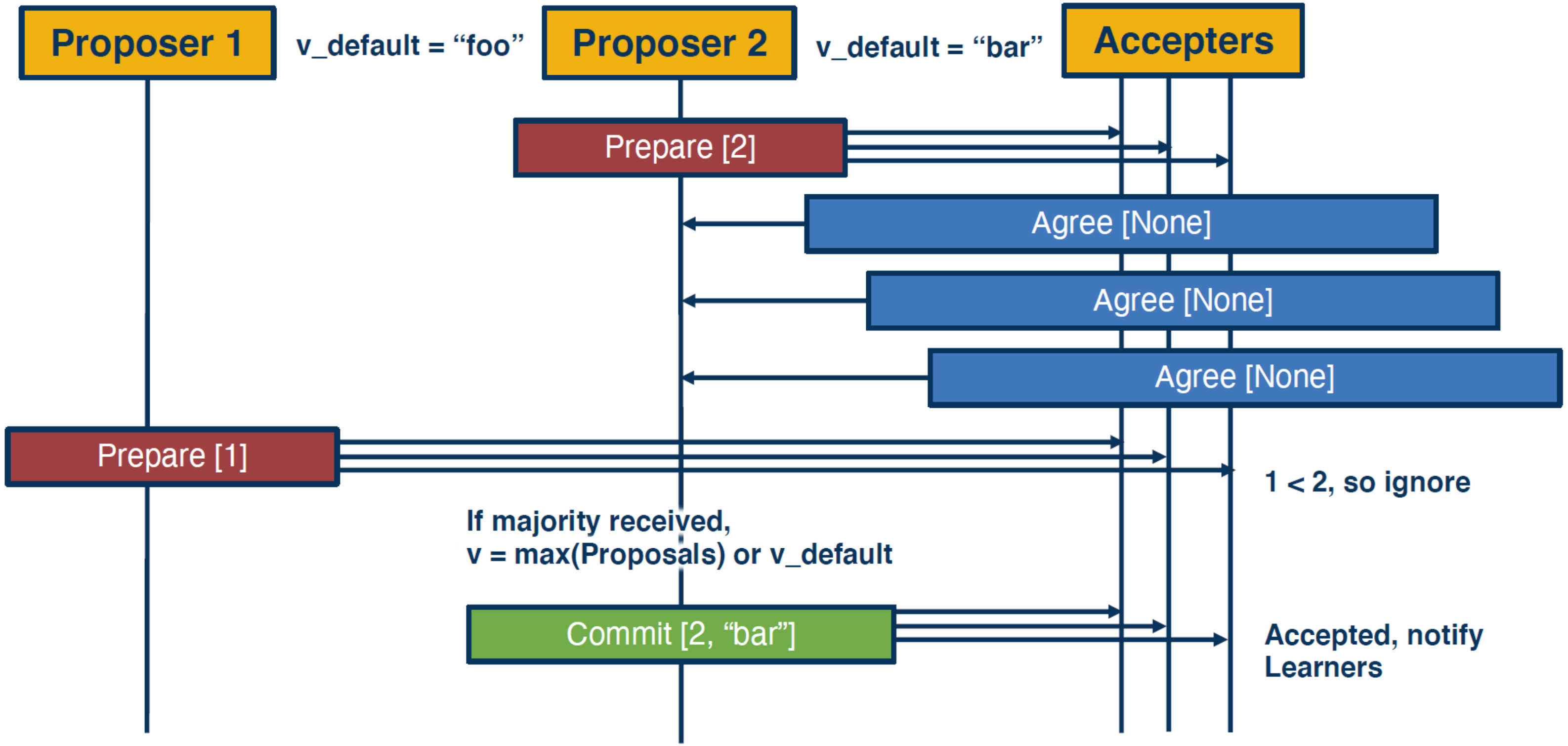


Edge Cases

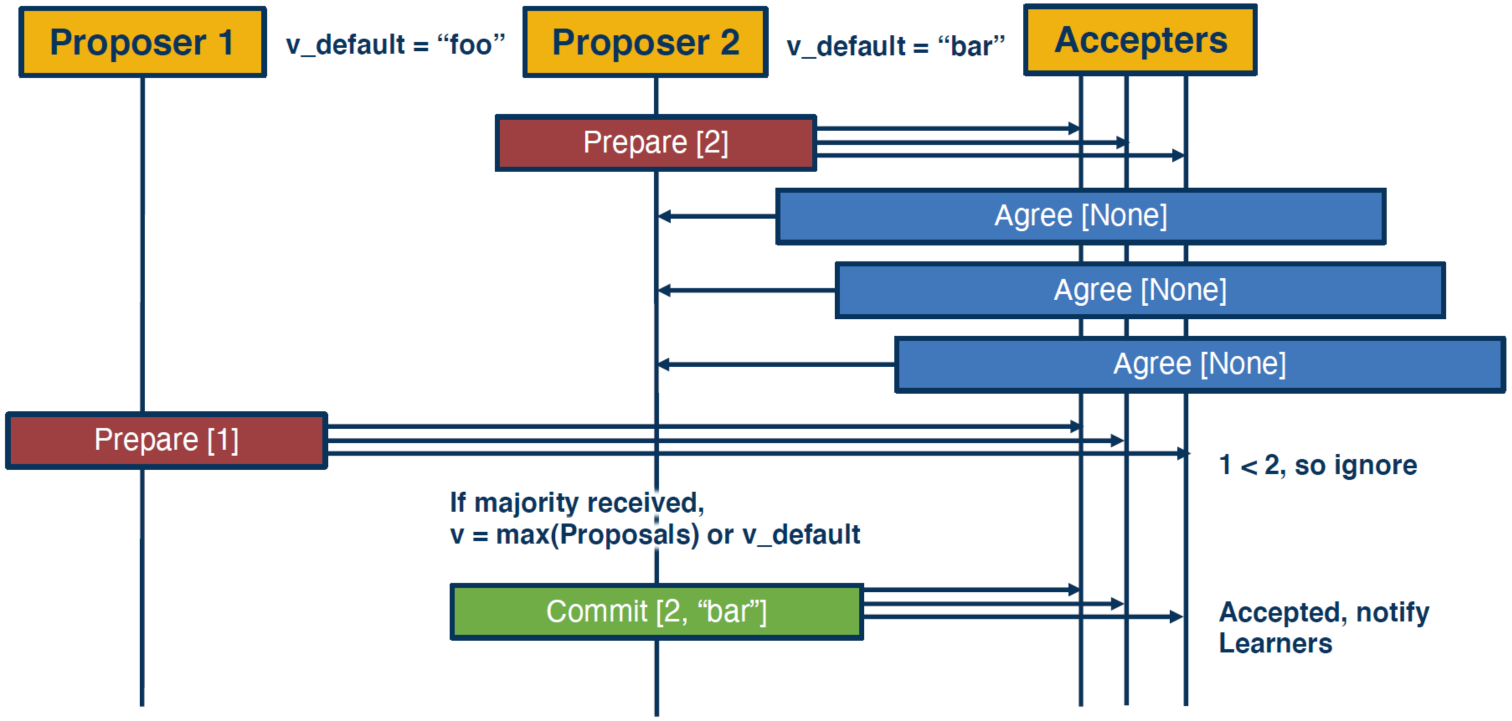
Proposers with *different* values

- Can only choose one!
- Ignore lower numbered proposal.

Challenging Cases: Different Values Proposed



Challenging Cases: Different Values Proposed



Liveness Challenges

Race between proposers

- Proposer 1 issues message 1: “prepare X”
- Proposer 2 issues message 2: “prepare Y”
- Keep sending higher numbered proposals
  - No guarantee a decision is reached
- Solutions:
  - Randomize delay time
  - Choose a leader (with timeouts for leader)
  - Heuristic solutions *might not work* (FLP)

Multi-Paxos

Single-Paxos agrees on a *single value*

- Example: ID of the lock owner (Google Chubby)

Multi-Paxos:

- Agrees on order of sequence of *multiple values*
- Part of *The Part-time Parliament*
- Multiple inflight exchanges, voting, etc.

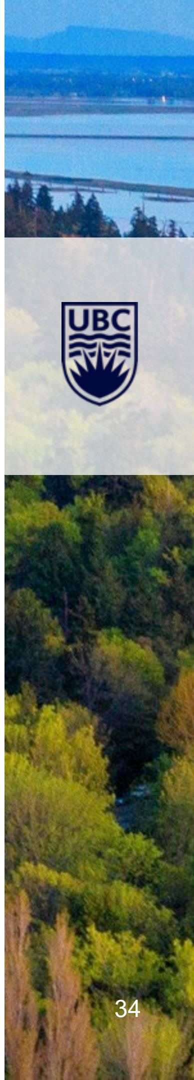
Multi-Paxos

Optimization:

- Select leader for a “view”
- All values in the view get accepted (and then learned) per Paxos
- Must detect and act upon view changes

Similar to Viewstamped Replication

See also Viewstamped Replication Revisited



Using Paxos

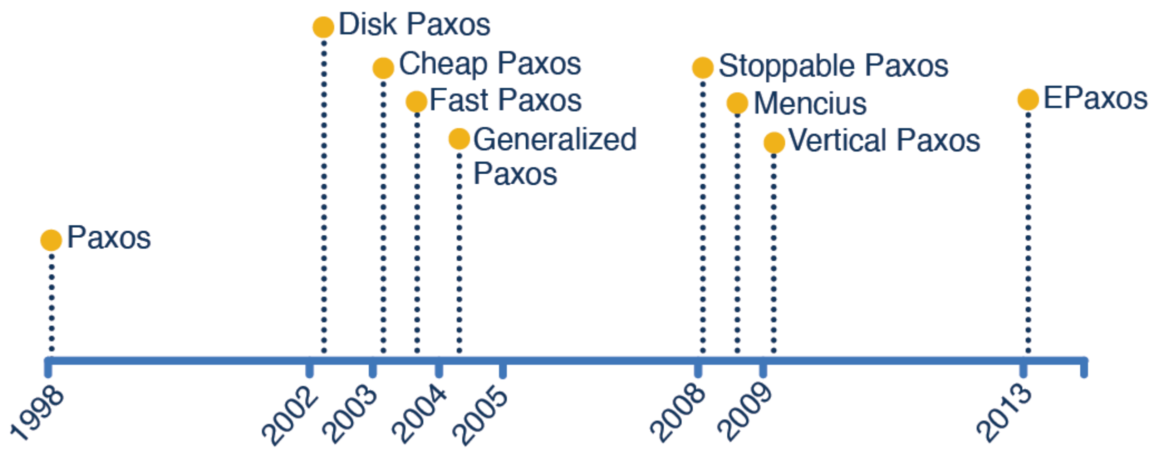
Digital Equipment Corporation Systems Research Center ("DEC SRC")

- Petal
- Frangipani
- Note: both used Paxos

Google Chubby

Zookeeper Atomic Broadcast (ZAB)

Paxos: Widely Used and Improved



Many research papers

- Paxos made live
- Paxos made transparent
- Paxos made moderately complex

Paxos in Action (animation)

Lesson Review

Paxos

Asynchronous Agreement Protocol

- Basic model: Proposers, Accepters, Learners
- Permits lossy environment, out of order operations
- Does not guarantee liveness
- Does not address Byzantine failure

Widely used and implemented protocol

Project 4: Build your own Paxos implementation.

- Does *not* require Project 3
- Uses AMO (Project 2)

Questions?

THE UNIVERSITY OF BRITISH COLUMBIA