

CPSC 416 – Distributed Systems

Lecture 10: From Failure to Consensus

UBC Vancouver

2026-02-05

Failure-first question

2PC requires unanimous consent. One permanent failure means abort.

Is there a way to make progress despite failures?

Recall: 2PC / 3PC

- Coordinator proposes, participants vote
- Any “no” or timeout → abort
- Coordinator fails after prepare → participants blocked (2PC) or abort (3PC)

Structural limitation: Unanimous consent required.

Checkpoint 1

Articulate:

What property would let us make progress despite some nodes being unavailable?

(Small groups: 2 minutes)

The Escape from Unanimity

We don't need *all* nodes.

We need *enough* nodes that any two operations see overlapping sets.

Definition: Quorum System

Quorum System

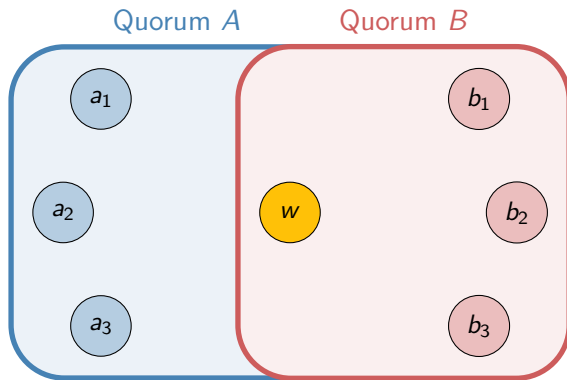
A **quorum system** $\mathcal{Q}$ over a set of nodes S is a collection of subsets of S such that:

$$\forall Q_1, Q_2 \in \mathcal{Q} : Q_1 \cap Q_2 \neq \emptyset$$

That's it. That's the whole invariant.

Everything else is *instantiation*.

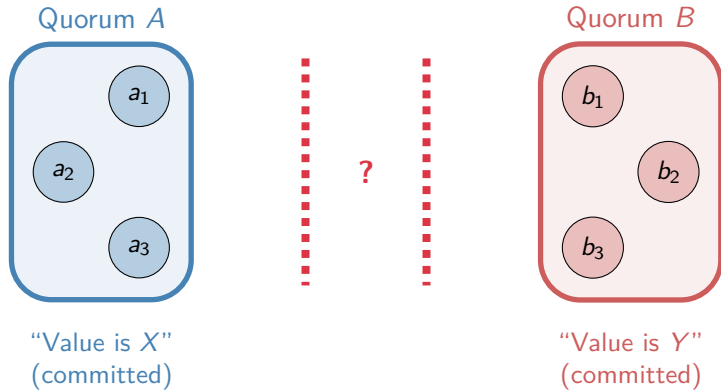
Why Consensus Works: The Intersection Property



Node w witnesses both operations.

If I write to A and you read from B ,
node w ensures you see my write.

Why Consensus Breaks: Intersection Failure



Checkpoint 2a

Articulate:

Why does intersection matter? What does it buy us?

(Pairs: 1 minute)

Example 1: The Grid

Arrange 9 nodes in a 3×3 matrix:

1	2	3
4	5	6
7	8	9

Define a quorum as: **any complete row + any complete column**

Quorum size: $\approx 2\sqrt{N}$

This is not a majority.

Grid Quorums: Why They Intersect

1	2	3
4	5	6
7	8	9

1	2	3
4	5	6
7	8	9

$$Q_1 = \{1, 2, 3, 5, 8\} \quad Q_2 = \{1, 4, 7, 8, 9\}$$

$$Q_1 \cap Q_2 = \{1, 8\} \neq \emptyset \quad \checkmark$$

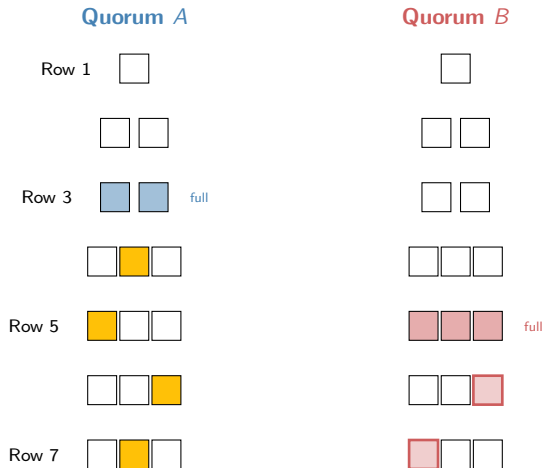
Checkpoint 2b

Articulate:

Why do *any* two row+column quorums always intersect?

(Pairs: 1 minute)

Crumbling Walls: Beyond the Simple Grid



Quorum A (row 3): full row + one rep from each row below

Quorum B (row 5): full row + one rep from each row below

Intersection: *B*'s full row is below *A*'s, so *A* has a rep in row 5 — *inside B*.

Example 2: Majority (as a Corollary)

Define:

$$\mathcal{Q} = \left\{ X \subseteq S : |X| > \frac{|S|}{2} \right\}$$

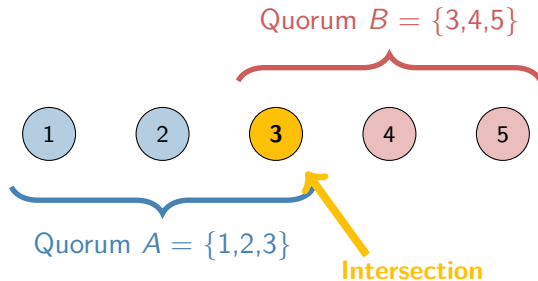
Proof of intersection:

If $|Q_1| > N/2$ and $|Q_2| > N/2$, then $|Q_1| + |Q_2| > N$.

By pigeonhole: $Q_1 \cap Q_2 \neq \emptyset$.

Majority is the simplest quorum system, not the definition of quorum.

Majority Quorums: Why $\lfloor n/2 \rfloor + 1$ Works



With $n = 5$ nodes, any majority quorum has ≥ 3 nodes.

Pigeonhole principle: Two sets of size 3 from 5 elements *must* share at least one element.

$$|A| + |B| > n \implies A \cap B \neq \emptyset$$

Example 3: Weighted Voting (Gifford 1979)

Assign *votes* to nodes.

A quorum is any subset whose votes exceed a threshold.

- Read quorum threshold: R
- Write quorum threshold: W
- Total votes: V

Intersection property: $R + W > V$

Same invariant, different instantiation.

Checkpoint 2c

Articulate:

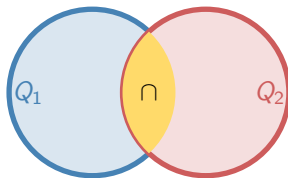
We've seen grid quorums, majority quorums, weighted quorums.

What do they all have in common? What's the thing that actually matters?

(Plenary: 2 minutes)

The Core Insight

A **quorum system** is any collection of sets where every pair shares at least one element.



This is why consensus works.
This is why consensus breaks when it fails.
This is what recovery requires.
Everything else is engineering.

Foreshadowing: Geo-Replication

	Replica A	Replica B	Replica C
DC-West	1	2	3
DC-Central	4	5	6
DC-East	7	8	9

Your failure domains aren't just “nodes.”
They're racks, power zones, data centers.

The quorum structure should match the failure structure.

We'll return to this.

Quorum Gives Us Visibility

If every write quorum intersects every read quorum...

...then every reader sees at least one node that witnessed the last write.

But what if two writers write concurrently?

The Ordering Problem

Scenario:

- Writer A writes $v = 1$ to $\{1, 2, 5\}$
- Writer B writes $v = 2$ to $\{5, 8, 9\}$
- Reader contacts $\{1, 8, 9\}$

Reader sees node 1 with $v = 1$ and nodes 8, 9 with $v = 2$.

Which is “latest”?

Checkpoint 3

Articulate:

Quorum gives us visibility. What's still missing?

What do we need to resolve concurrent writes?

(Small groups: 2 minutes)

Lamport's Bakery Algorithm (1974)

Analogy: Bakery customers take numbers.

- Each customer picks a number $>$ all numbers they've seen
- Lowest number gets served first
- Ties broken by customer ID

Total ordering from local information.

From Bakery to Ballots

Translate to distributed setting:

- Proposer picks number $>$ all numbers it has seen
- Ties broken by proposer ID
- Pair (round, proposer_id) = **ballot**

This is the core of Paxos's ordering mechanism.

The ballot is not arbitrary: it's the *minimal structure* needed to totally order competing proposals.

Checkpoint 4

Articulate:

We have quorum for durability, ballots for ordering.

What's the shape of a protocol that uses both?

(Small groups: sketch on paper, 3 minutes)

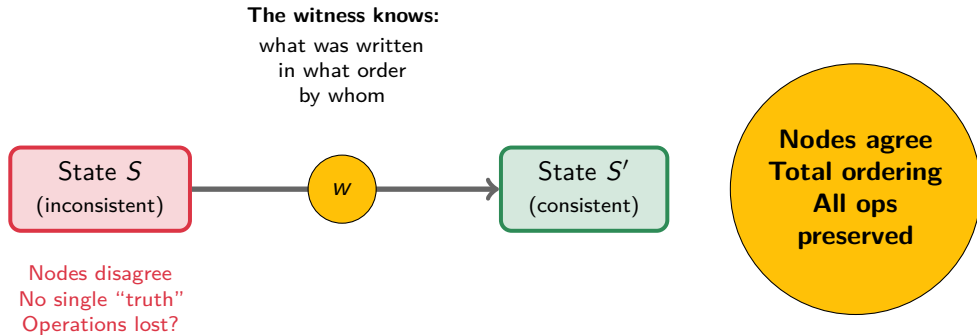
What You Just Reinvented

- ➊ Proposer picks a ballot
- ➋ Proposer contacts a quorum to “claim” that ballot
- ➌ If quorum agrees, proposer can write with that ballot
- ➍ Readers contact a quorum, take highest-ballot value

This is the prepare/accept structure of Paxos.

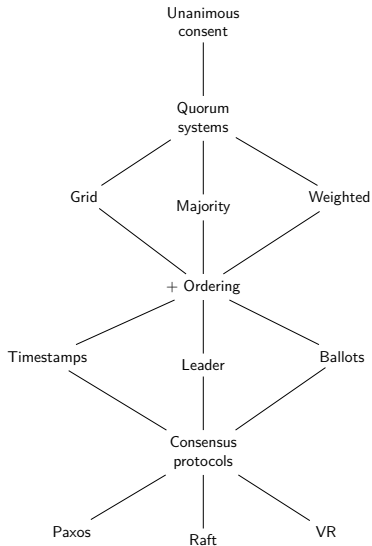
You derived it from the failure requirements.

Recovery: The Path from S to S'



Recovery is possible iff a node participated in *all* relevant quorums.
The intersection property guarantees such a node exists.

The Design Space



Many Paths, Same Destination

These paths are not identical, but they solve the same structural problem.

Next time: we walk the Paxos path in detail.

But you should know: the Raft path and the Viewstamped Replication path arrive at an isomorphic place.

Evidence Question

What failure would break our quorum system?

What would we log to debug it?

(Consider: What if the failure domain assumptions are wrong?)

Logistics

- **Reading:** “Paxos Made Simple” (Lamport, 2001)
- **Self-assessment:** Opens 13:00, due next class 11:30
- **Next lecture:** Walk the Paxos path, then break it