

CPSC 416 – Distributed Systems

Lecture 12: Breaking Paxos

UBC Vancouver

2026-02-12

From Last Time

Articulate:

Walk through the two phases of Paxos with a partner.

For each step, say *why* it exists: what failure would occur without it?

(Pairs: 3 minutes)

Paxos in One Slide

- ➊ **Prepare:** “I’m ballot b . What have you accepted?”
- ➋ **Promise:** “I won’t accept $< b$. Here’s what I have.”
- ➌ **Accept:** “Accept value v at ballot b .”
(where v = highest accepted from Phase 1, or proposer’s choice if none)
- ➍ **Accepted:** “Done.”
- ➎ **Chosen:** When a quorum has accepted.

The complexity is in understanding *why*, not in memorizing *what*.

Failure-first question

Paxos is safe: it never chooses two different values.

But does it always make progress? What can go wrong?

Recall from L11

Paxos guarantees:

$$\text{chosen}(v_1) \wedge \text{chosen}(v_2) \implies v_1 = v_2$$

Safety: Never choose two different values. ✓

Liveness: Eventually choose *some* value. ???

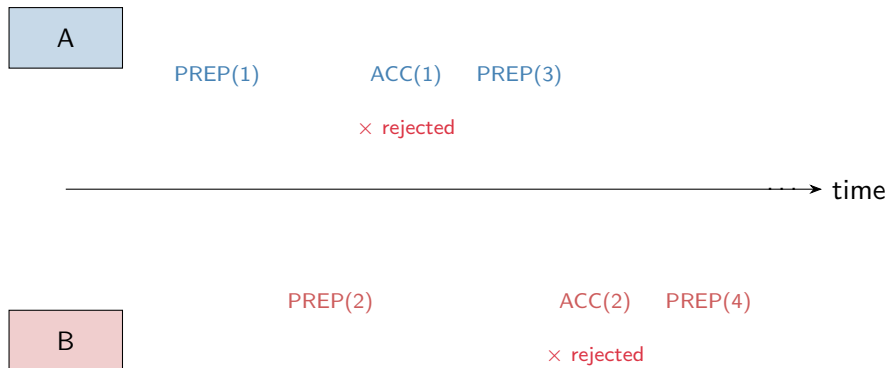
Failure Mode 1: Dueling Proposers

Scenario:

- 1 Proposer A sends `PREPARE(1)`
- 2 Quorum promises ballot 1
- 3 Proposer B sends `PREPARE(2)` — higher ballot!
- 4 Quorum promises ballot 2, invalidating A's promises
- 5 Proposer A's `ACCEPT(1, v)` is rejected
- 6 Proposer A retries with `PREPARE(3)`
- 7 Quorum promises ballot 3, invalidating B's promises
- 8 ...

Result: Infinite preemption. No value is ever chosen.

Dueling Proposers: Visualized



**Both proposers are correct. The protocol is safe.
But nothing gets done.**

Checkpoint 1

Articulate:

Paxos is safe but can fail to make progress.

What assumption would prevent dueling proposers?

(Pairs: 2 minutes)

The FLP Impossibility Result

Fischer, Lynch, Paterson (1985):

FLP Theorem

In an asynchronous system with even one faulty process,
no deterministic consensus protocol can guarantee termination.

Translation: You cannot have all three:

- Safety (never disagree)
- Liveness (eventually decide)
- Asynchrony (no timing assumptions)

FLP: Be the Adversary

Your goal: Prevent consensus. You control message timing.

- 1 Proposer A completes Phase 1. Has quorum promises at ballot 1.
- 2 A sends `ACCEPT(1, v)`.
- 3 You delay A's `ACCEPT` by one round-trip.
- 4 B's `PREPARE(2)` arrives first. Quorum re-promises.
- 5 A's `ACCEPT` arrives late. **Rejected.**
- 6 B completes Phase 1. Has quorum promises at ballot 2.
- 7 B sends `ACCEPT(2, w)`.
- 8 You delay B's `ACCEPT` by one round-trip.
- 9 A's `PREPARE(3)` arrives first. Quorum re-promises.
- 10 B's `ACCEPT` arrives late. **Rejected.**
- 11 ...

You never lie. You never corrupt. You just delay.

The Adversary Doesn't Exist

There is no adversary.

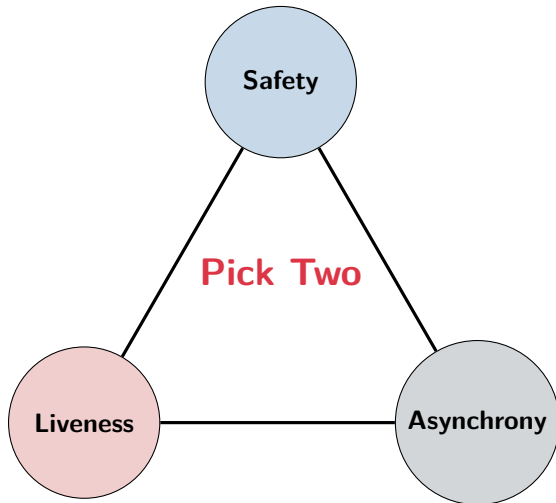
There's just a network where messages take variable time.

The system can't distinguish “slow” from “crashed.”

Every delay that “the adversary” exploits is something that *happens in real networks every day.*

FLP (1985): In any asynchronous system, this strategy always exists. No deterministic protocol can escape it.

The FLP Trade-off



Paxos picks Safety + Asynchrony. Liveness is not guaranteed.

What FLP Does NOT Say

FLP does **not** say:

- Consensus is impossible (it's not)
- Paxos is broken (it's not)
- We should give up (we shouldn't)

FLP **does** say:

- Any protocol that guarantees safety must make *assumptions* to guarantee liveness
- Those assumptions involve *timing* or *randomness*

FLP clarifies the design space. It doesn't close it.

Checkpoint 2

Articulate:

FLP says we need timing assumptions for liveness.

What timing assumptions could prevent dueling proposers?

(Pairs: 2 minutes)

Escaping FLP: Timing Assumptions

Option 1: Leader Election

- Elect a single proposer (the “leader”)
- Only the leader proposes
- No dueling → progress
- But: need to detect leader failure, elect new one

Option 2: Timeouts + Backoff

- If preempted, wait before retrying
- Randomized or exponential backoff
- Eventually one proposer “wins”

Both require partial synchrony: Messages eventually arrive within some bound.

Partial Synchrony

The Assumption:

There exists a Global Stabilization Time (GST) after which:

- All messages are delivered within a bounded time Δ
- The system behaves “synchronously enough”

Before GST: Anything can happen (messages lost, arbitrarily delayed).

After GST: The system eventually makes progress.

We don't know when GST occurs. We just assume it eventually does.

What We Learned

Paxos is safe but requires assumptions for liveness:

- **Dueling proposers:** Multiple proposers can prevent progress
- **FLP impossibility:** No way around this in pure async
- **Partial synchrony:** The practical escape hatch
- **Leader election:** The common optimization (more in L13)

The key trade-off:

Paxos trades liveness uncertainty for safety certainty.

FLP tells us this trade is unavoidable.

The Liveness Assumptions Paxos Requires

For Paxos to make progress:

- ① A quorum of acceptors must be reachable
- ② Eventually, a single proposer must run long enough to complete both phases
- ③ Messages must eventually be delivered

None of these are guaranteed in a truly asynchronous system.

But they're reasonable assumptions for most real deployments.

Evidence Question

You're debugging a Paxos deployment that seems “stuck.”

What would you look for in the logs?

How would you distinguish “dueling proposers” from “network partition”?

(This is a real operational challenge.)

Logistics

- **Capstone Team Declaration:** Due tonight 23:59
- **Reading break:** Feb 16–20 (no class Feb 17/19)
- **After break (L13):** Paxos Revisited + Multi-Paxos
- **Reading for L13:** “Paxos Made Live” (Google, 2007)
- **Self-assessment:** Opens 13:00, due Feb 24 11:30

Enjoy the break. Come back ready to explain Paxos and build sequences.