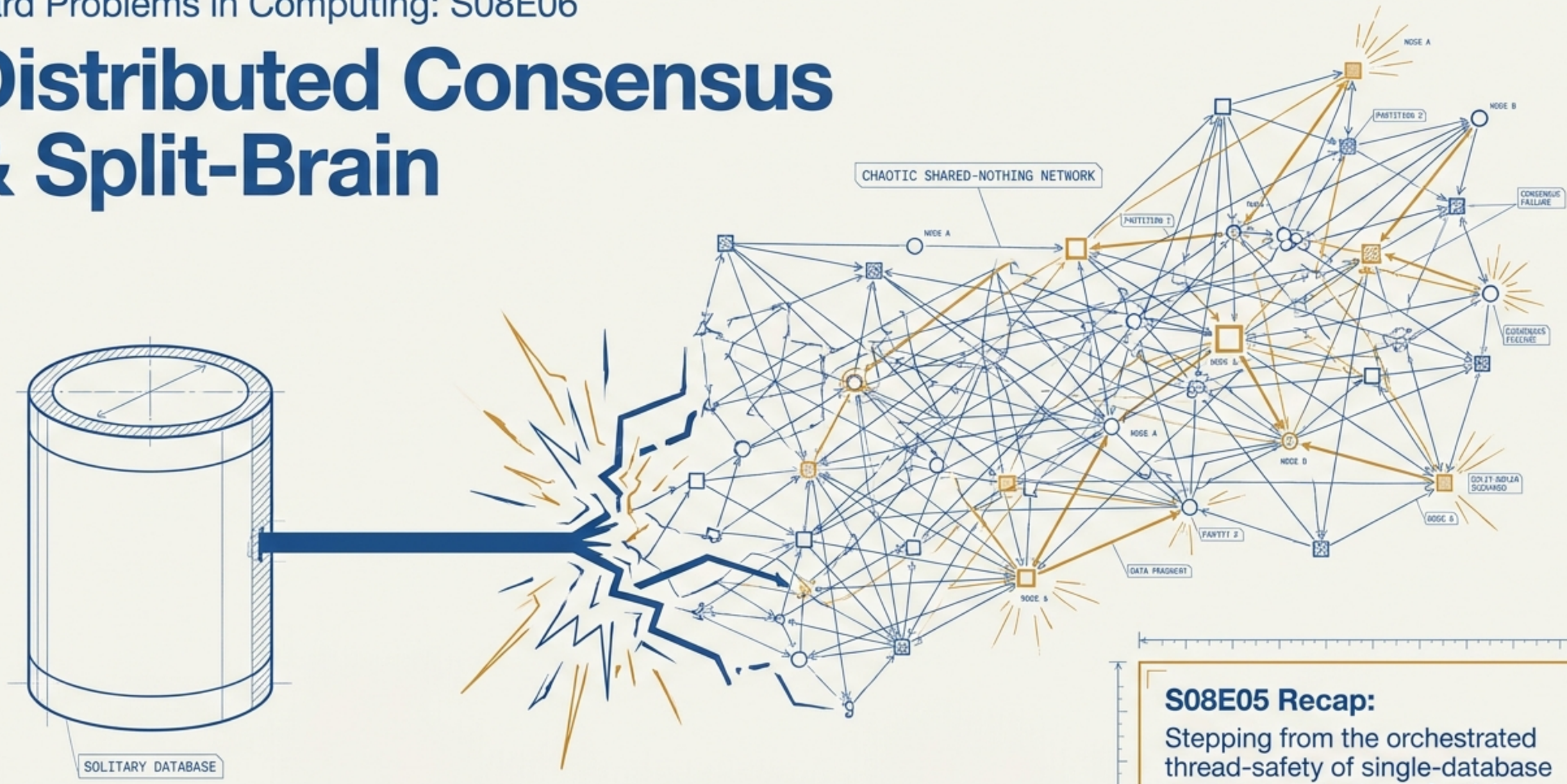
Hard Problems in Computing: S08E06

Distributed Consensus & Split-Brain



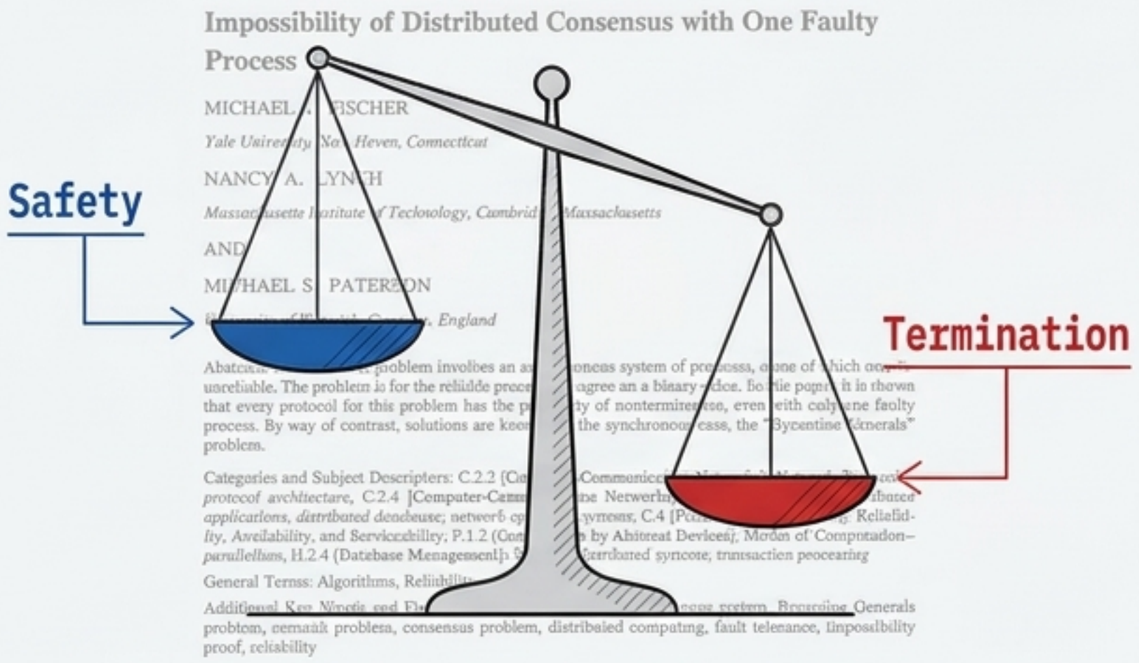
S08E05 Recap:

Stepping from the orchestrated thread-safety of single-database concurrency to the unforgiving chaos of shared-nothing networks.

Gemini Notebook

Section 1 / Theoretical Impossibility Boundaries

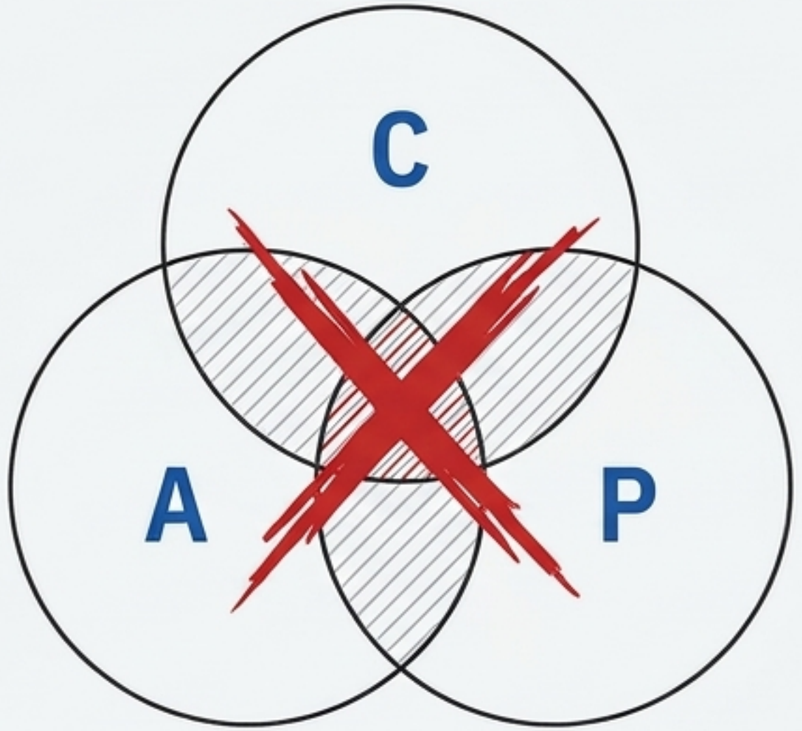
The Hard Limits of Asynchrony



FLP Impossibility (Fischer-Lynch-Paterson, 1985)

The Theorem: In an asynchronous network with indefinite delays, no consensus algorithm can guarantee both **safety** and **termination** if a single node crashes.

The Reality: We must rely on **randomized timeouts** or **partial synchrony** to force progress.



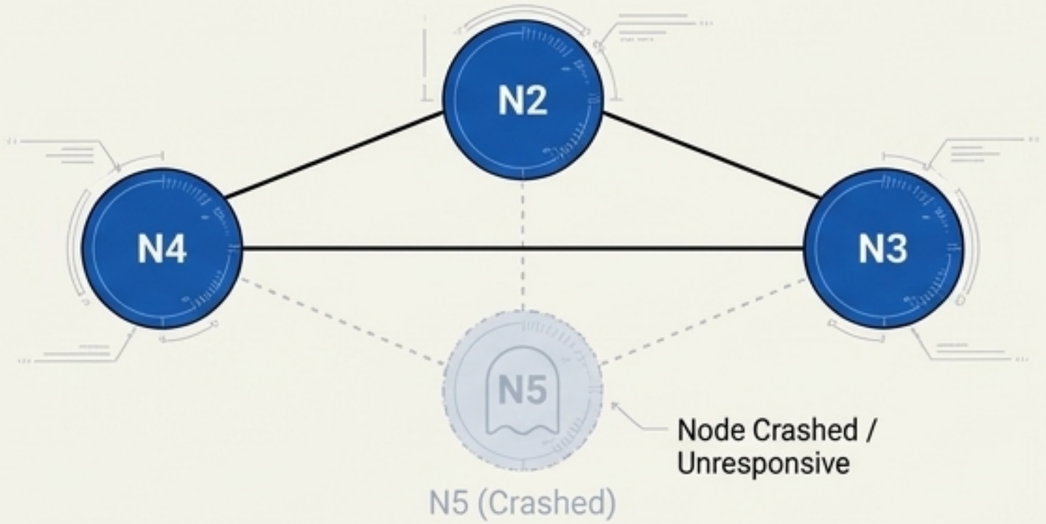
Gilbert-Lynch CAP Theorem

The Theorem: Under a network partition (P), you must choose **your poison**.

- **CP** (**Consistency**): Reject writes to prevent split-brain.
- **AP** (**Availability**): Accept writes on both sides, requiring complex conflict resolution later.

Gemini Notebook

Fault Tolerance Math: Crash vs. Malice

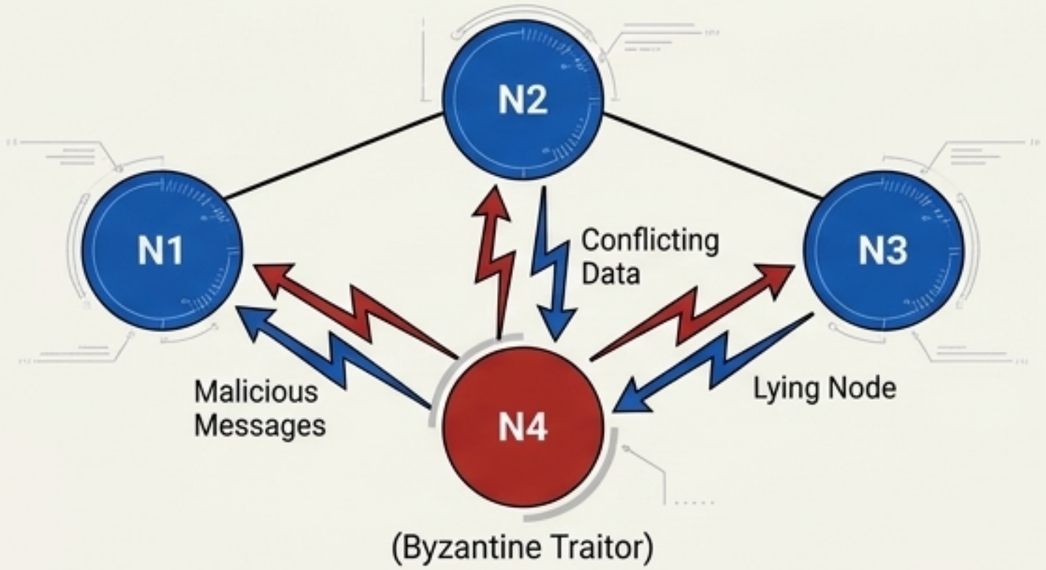


Crash Faults (2f + 1)

Definition: Nodes simply stop responding (crash, power loss).

Requirement: Reaching consensus demands a simple majority.

Standard: Raft, Paxos, ZooKeeper. To survive f crashes, you need 2f + 1 total nodes. (Survive 2 crashes = 5 nodes).



Byzantine Faults (3f + 1)

Definition: Nodes actively lie, act maliciously, or suffer silent data corruption.

Requirement: Reaching consensus demands a supermajority.

Standard: Lamport / Castro-Liskov PBFT. To survive f malicious actors, you need 3f + 1 total nodes.
(Survive 1 traitor = 4 nodes).

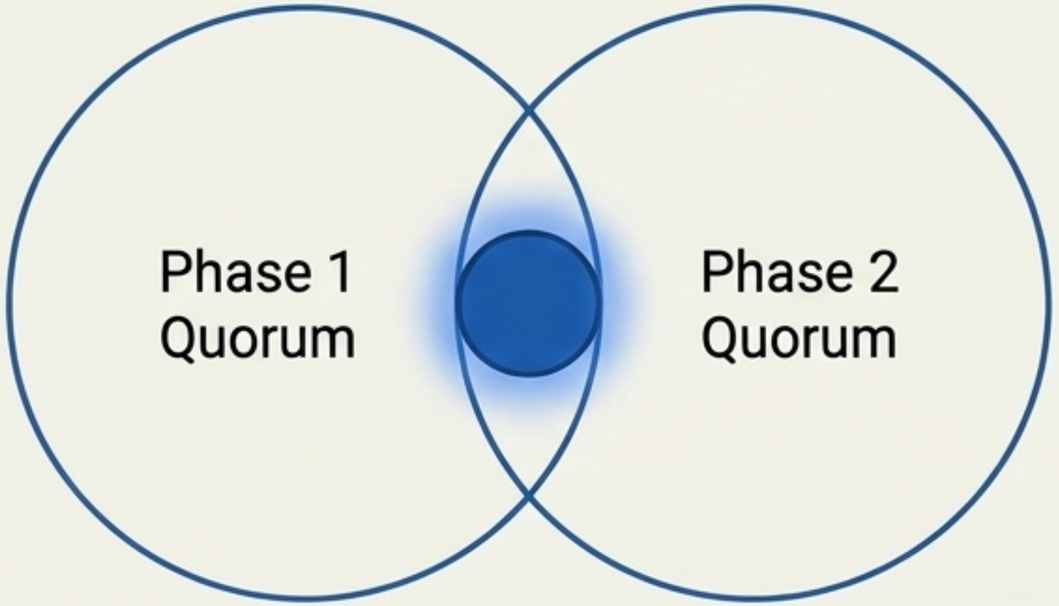
Gemini Notebook

Lamport's Paxos & Flexible Quorums

Phase 1 (Prepare/Promise) → **Phase 2 (Accept/Accepted)**

Proposer selects sequence n, asks Acceptors for a promise not to accept proposals < n.

Proposer sends value v. Acceptors commit unless they've promised a higher n.

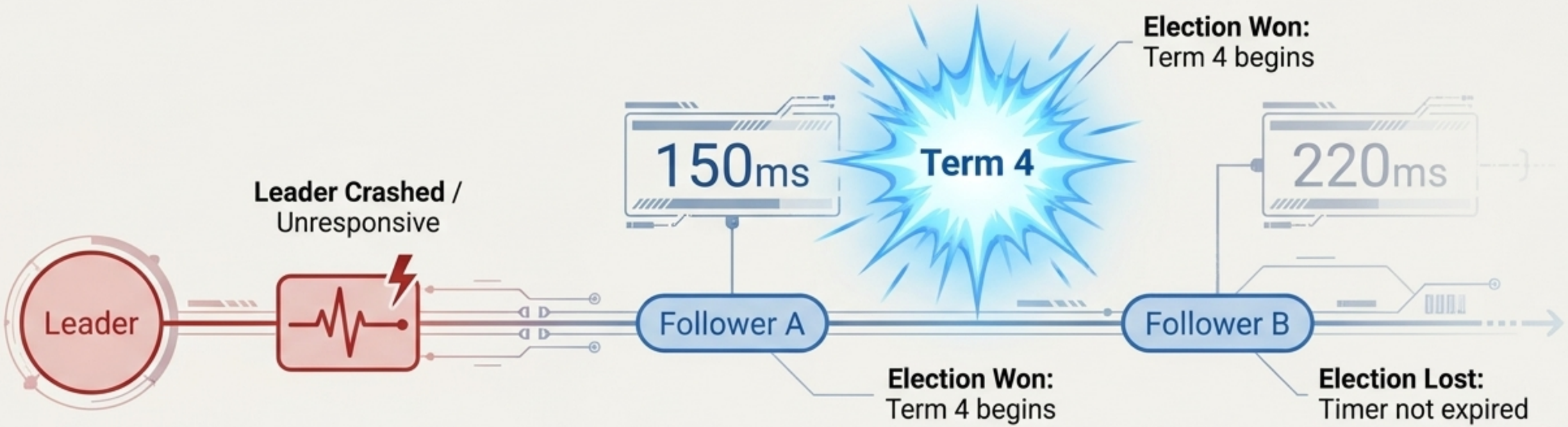


Flexible Paxos (Howard, Malkhi, Spiegelman)

- **The Myth:** Paxos requires strict majorities for both phases.
- **The Reality:** Phase 1 and Phase 2 quorums only need to intersect. A smaller Phase 2 quorum speeds up steady-state commits, provided Phase 1 quorums are large enough to guarantee that single node intersection.

Gemini Notebook

Ongaro & Ousterhout's Raft



Monotonic Terms as Logical Clocks

Time is fractured into arbitrary terms. Higher term numbers instantly invalidate older terms, cleanly severing split-brains.

Beating FLP with Randomization

To defeat the FLP split-vote livelock (endless election ties), Raft uses randomized election timeouts (e.g., 150–300ms). The first to wake up wins.

Majority Log Commitment

A log entry is only safely committed when successfully written to $\lfloor N/2 \rfloor + 1$ nodes.

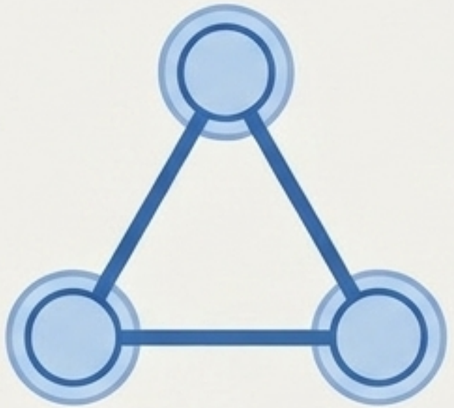
Gemini Notebook

The Mathematics of Cluster Topology



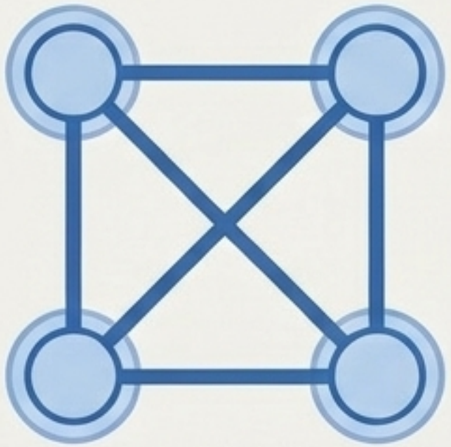
2-Node Cluster
0 Failures

A network partition looks mathematically identical to a node death. Auto-failover is impossible without split-brain.



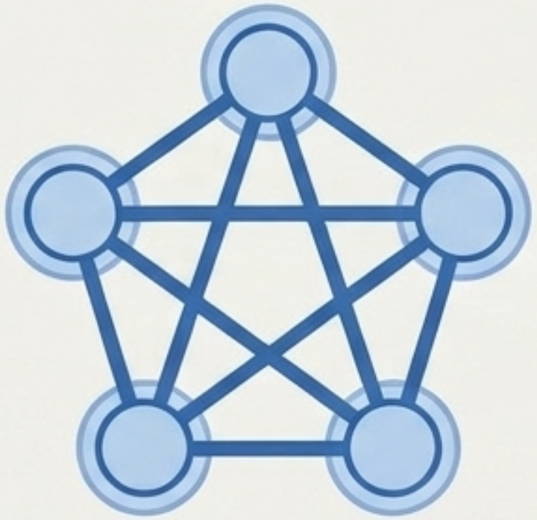
3-Node Cluster
1 Failure

Quorum = 2. Survives 1 death. The gold standard for minimum High Availability (HA).



4-Node Cluster
1 Failure

Quorum = 3. Still only survives 1 death, but increases the probability of a failure halting the cluster (since 3 of 4 must be up). Strictly worse than 3.



5-Node Cluster
2 Failures

Quorum = 3. Survives 2 deaths. Ideal for critical infrastructure.

Gemini Notebook

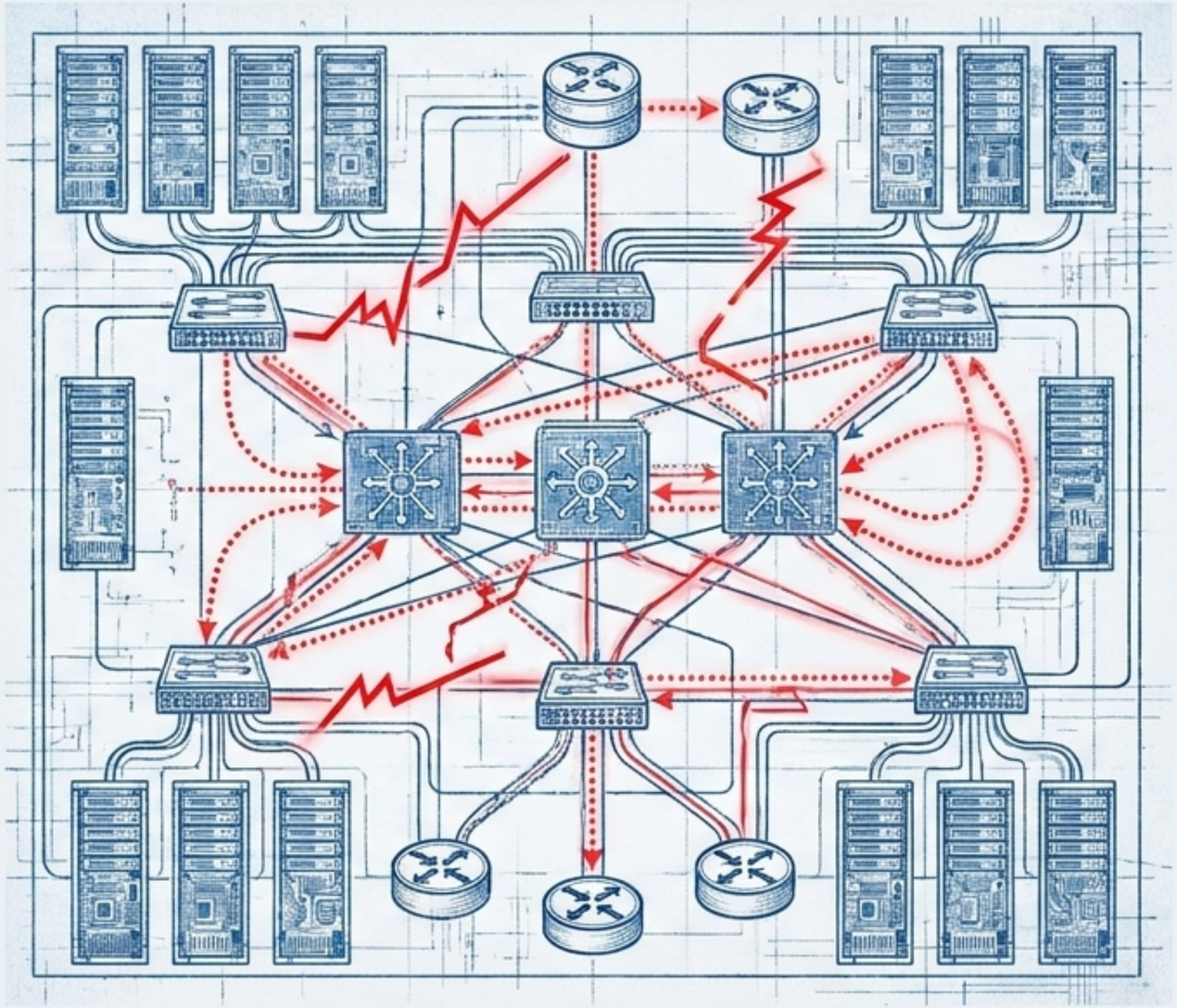
The Distributed Locking Matrix

Primitive	Safety (GC/Partitions)	Infra Cost	Throughput	Fencing Support
In-Process cron	❌ Broken on >1 pod	✅ Zero	✅ Very High	❌ None
PG pg_try_advisory_lock	⚠️ DB failover risk	✅ Low (reused)	⚠️ Medium	❌ None
PG SKIP LOCKED	✅ Safe (Row-level)	✅ Low (reused)	⚠️ Med (Dead-tuples)	⚠️ Implicit
Redis SET NX PX	❌ Efficiency lock only	✅ Low	✅ Extreme	❌ None
Redis Redlock (Multi)	❌ Vulnerable to clocks	⚠️ Medium	✅ High	❌ None
etcd / ZooKeeper Leases	✅ CP / Consensus	❌ High	⚠️ Medium	✅ Native (zxid)
Spanner / CockroachDB	✅ True Serializable	❌ Extreme	⚠️ Medium	✅ True MVCC

Gemini Notebook

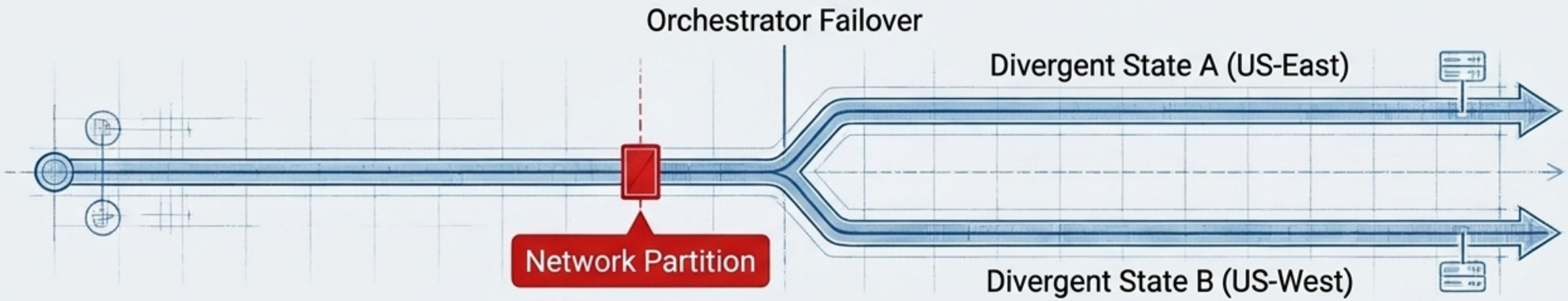
The Network is Reliable (It Isn’t)

- **The Fallacy:** Engineering assumes networks drop packets uniformly or simply fail completely (clean wire cuts).
- **The Reality:** Peter Bailis & Kyle Kingsbury’s research proves that NICs, switches, and WANs routinely cause asymmetric partitions, zombie connections, and non-transitive routing.
- **Jepsen Audits**: Kingsbury’s linearizability tests actively break enterprise databases by simulating realistic network partitions, uncovering stale reads, lost updates, and split-brains in systems that claimed CP safety.



Gemini Notebook

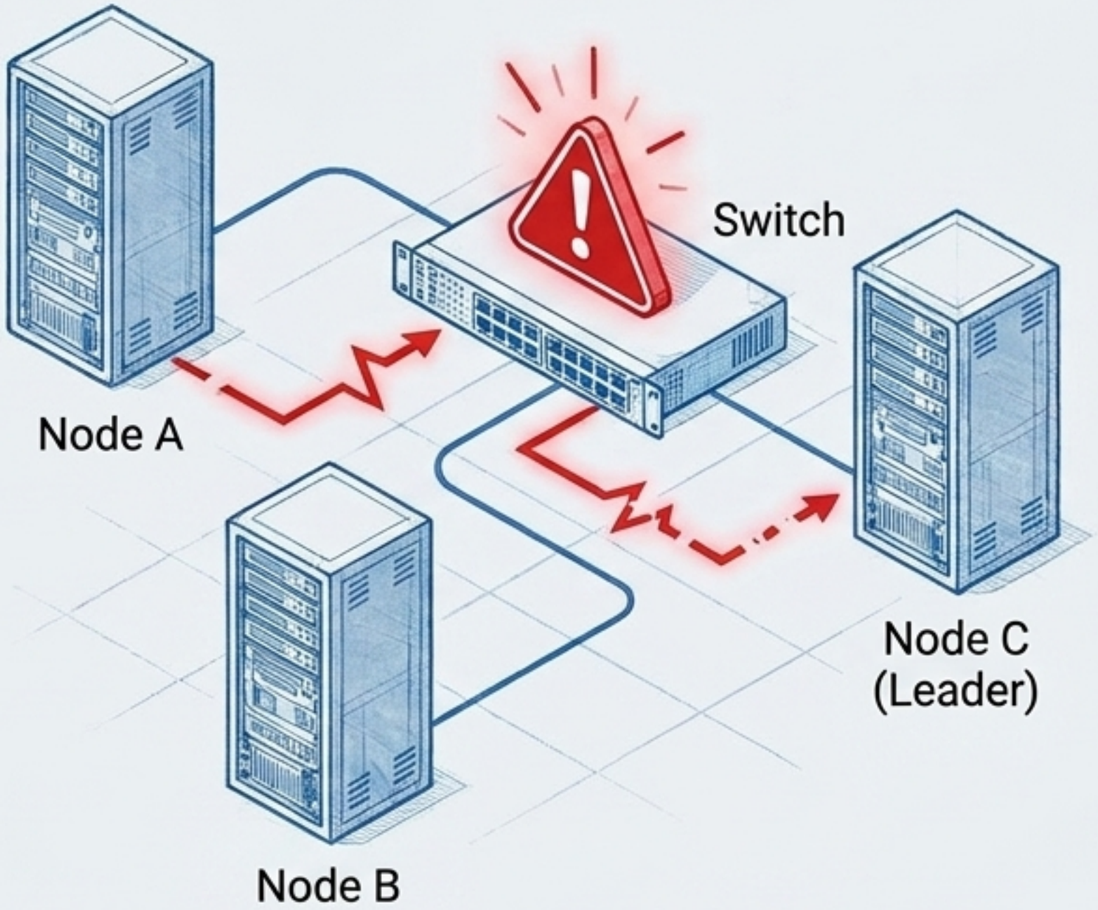
GitHub (Oct 21, 2018): 43 Seconds of Chaos



- **The Trigger:** A brief, 43-second network partition between two data centers.
- **The Failure:** The automated orchestrator failed to properly fence the old MySQL primary. A new primary was promoted while the old one continued accepting writes.
- **The Consequence:** True split-brain. Two divergent timelines of database writes occurring simultaneously.
- **The Recovery:** It required 24 hours of degraded performance and agonizing manual data reconciliation to merge the corrupted state safely.

Gemini Notebook

Cloudflare (Nov 2020): The Byzantine Switch



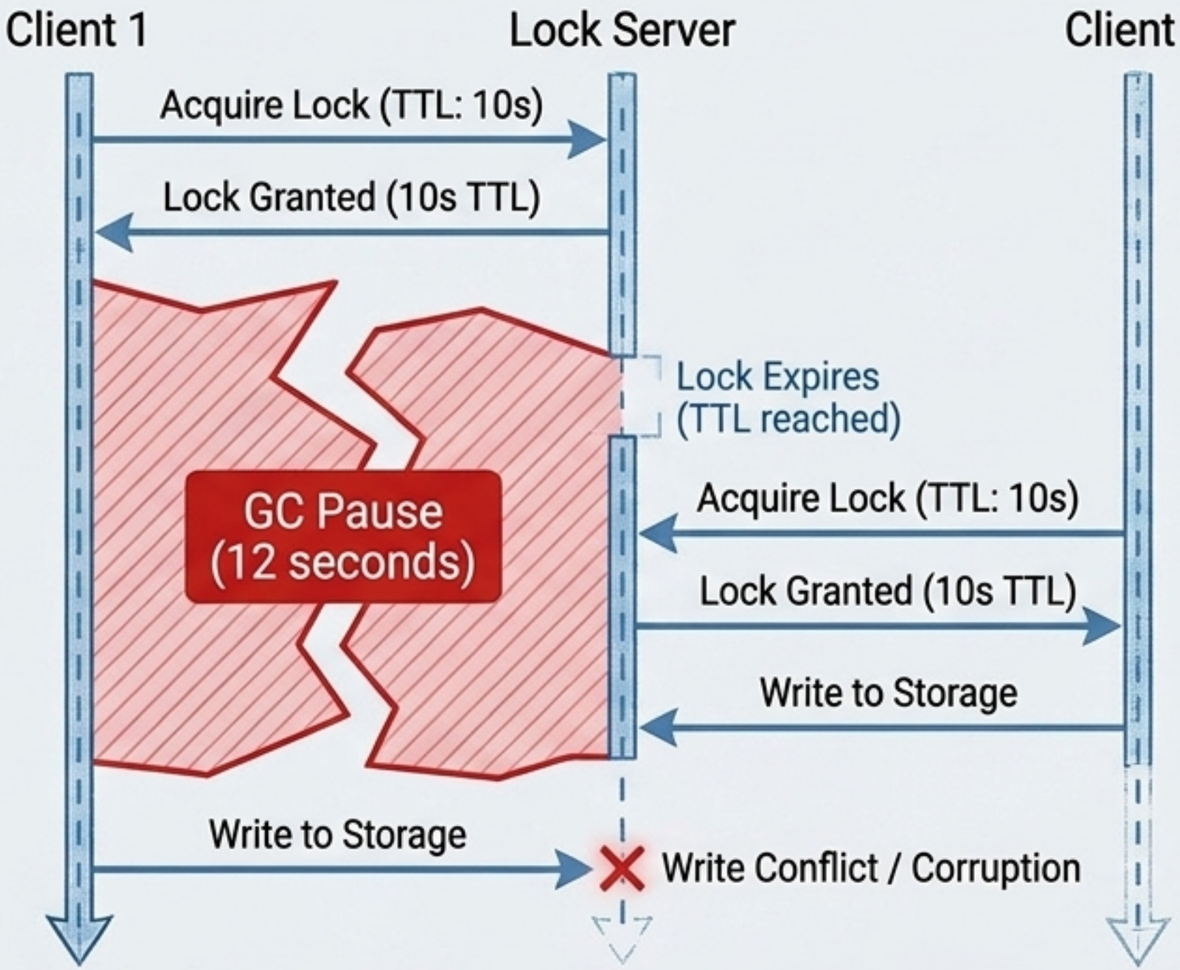
The Trigger: A degraded top-of-rack switch began dropping packets non-transitively.

The Byzantine Fault: etcd Node 1 could reach Node 2, but not Node 3 (the current leader). Node 1 believed the leader was dead; Node 2 knew the leader was fine.

The Consequence: Because of this conflicting reality, Node 1 endlessly called for new elections (Pre-Vote). The conflicting views caused etcd to lose quorum and freeze, triggering a massive control plane outage.

Gemini Notebook

Why Pure TTL Leases Always Fail

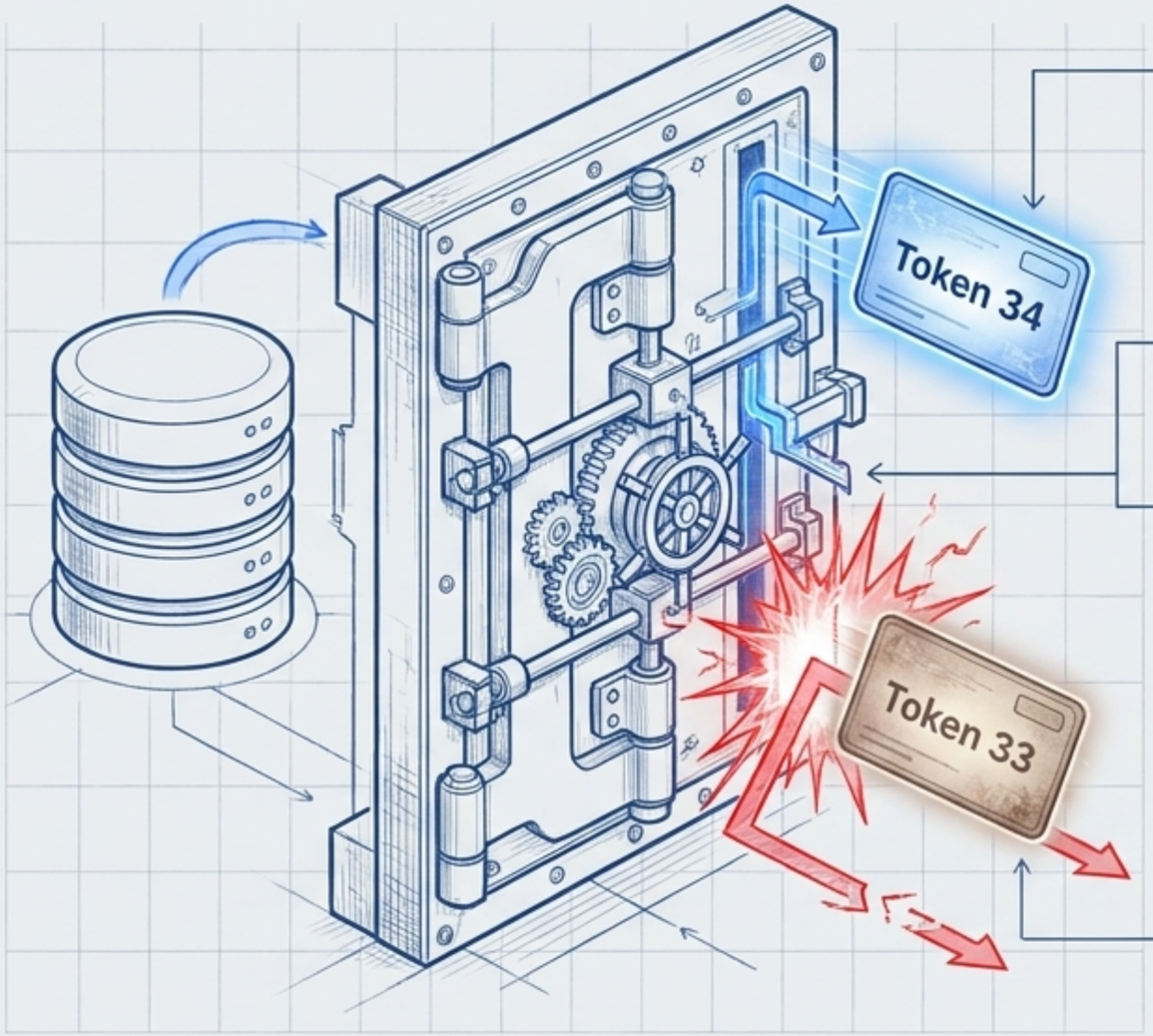


- **Step 1:** Client 1 acquires a Redlock lease with a 10-second TTL.
- **Step 2 (The Trap):** Client 1 suffers a 12-second Stop-The-World GC Pause (or VM live migration stall).
- **Step 3:** The lease silently expires. Client 2 acquires the same lease and writes to storage.
- **Step 4:** Client 1 wakes up. Unaware of the time skip, it believes it still holds the lock and writes to storage.
- **Result:** Silent data corruption. Redlock cannot guarantee correctness against clock jumps or process pauses.

Gemini Notebook

Section 5 / Kleppmann vs. antirez & Fencing Tokens

The Fix: Monotonic Fencing Tokens



The Mechanism: The consensus store issues a monotonically increasing token (e.g., Unmesh Joshi's Generation Clock) with every lock acquisition.

The Handshake: The client must pass this token to the final storage layer.

The Arbiter: The target storage layer acts as the final bouncer (Compare-and-Swap). If it has already accepted a write with Token 34 (from Client 2), it explicitly rejects any delayed writes bearing zombie Token 33 (from Client 1).

Real-World Example: Kubernetes resourceVersion CAS updates natively implement this pattern.

Gemini Notebook

Four Catastrophic LLM Patterns

1. The In-Container Cron: LLMs routinely suggest adding node-cron inside web containers.
Result: Scaling to 5 pods triggers the background job 5 times simultaneously.

2. Non-Atomic Redis Locks: LLMs write SETNX, followed by a separate EXPIRE.
Result: A crash between lines creates an infinite lock.
Worse: Using DEL without token checking deletes someone else's active lock.

3. DIY Heartbeat Scripts: Suggesting a 2-node ping script to auto-promote a DB.

Result: A tiny network blip creates an unrecoverable split-brain.

4. Hand-Rolled Active-Active: Syncing two databases via application webhooks.

Result: Guaranteed data divergence and race conditions.

Gemini Notebook

Section 7/AI Prompting & YAGNI Rubric

Default Strictly to Single-Primary Postgres

The YAGNI Rule: You Aren't Gonna Need It. Distributed consensus is a massive operational tax. Rely on Postgres until scale dictates otherwise.



Postgres as Coordinator: Use `pg_try_advisory_xact_lock`. It is bulletproof because locks are intrinsically bound to the transaction lifecycle and auto-release on commit or disconnect.

Queues: Use **FOR UPDATE SKIP LOCKED** for highly concurrent job polling.

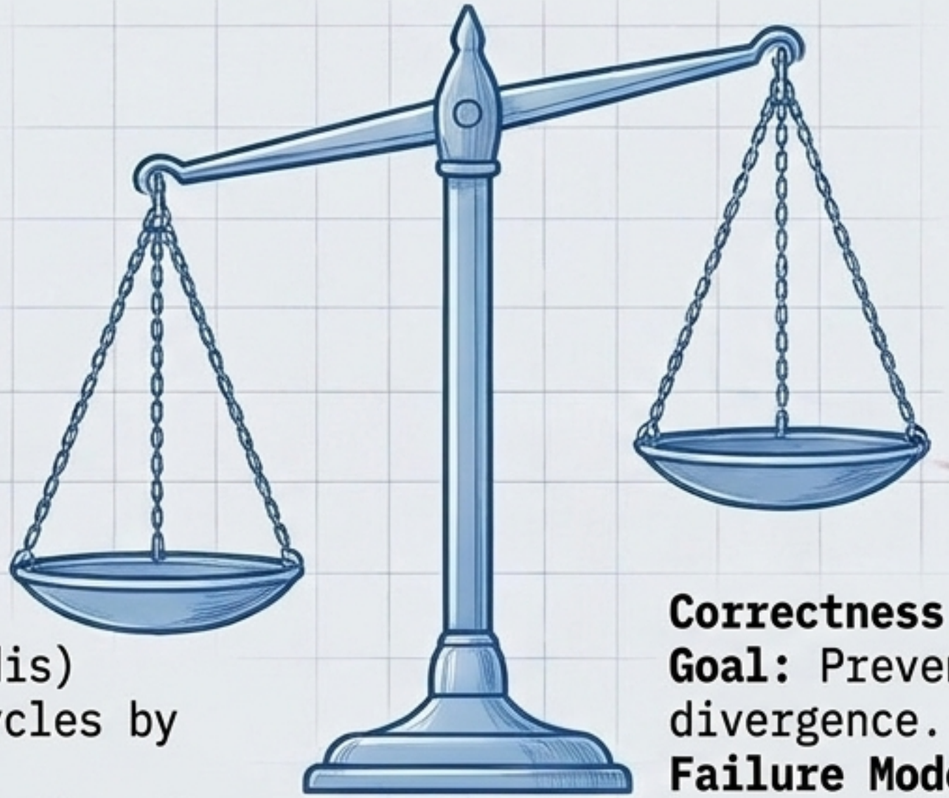
The Limit: Beware **Brandur's** MVCC limit—high-throughput queues will bloat Postgres with dead tuples. At extreme scale (>10k TPS), migrate to a dedicated queue.

Gemini Notebook

Section 7/AI Prompting & YAGNI Rubric

Separation of Concerns: Efficiency vs. Correctness







Efficiency Locks (e.g., Redis)
Goal: Save money and CPU cycles by preventing redundant work.
Failure Mode: If the lock fails, the work happens twice. Annoying, but safe.
Pattern: Use standard TTLs. Must pair with idempotent handlers.

Correctness Locks (e.g., etcd/ZooKeeper)
Goal: Prevent data corruption and state divergence.
Failure Mode: If the lock fails, the business dies.
Pattern: Requires an odd-node Paxos/Raft cluster + Monotonic Fencing Tokens verified via CAS at the storage layer.

Gemini Notebook

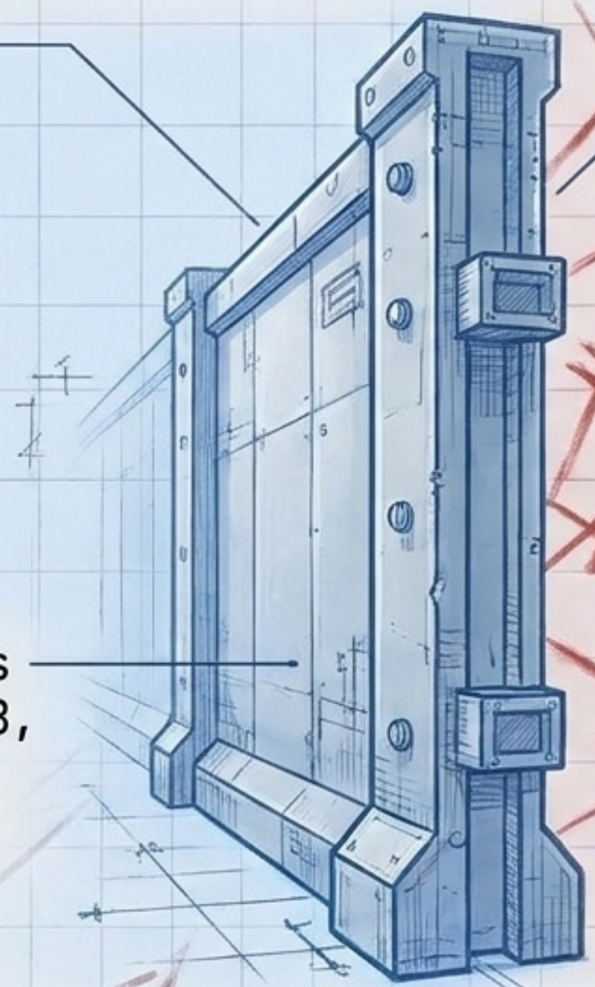
Section 7/AI Prompting & YAGNI Rubric

Define Your Trust Boundaries

Internal Trusted Network

Crash Faults are Enough: Inside your datacenter or VPC, Raft or Paxos (2f+1) is completely sufficient. Your nodes might die, but they aren't actively trying to deceive you.

Ban PBFT internally: AI agents often suggest blockchain, Web3, or Byzantine Fault Tolerant (3f+1) consensus for internal enterprise microservices.



Adversarial Web

The Rule: Unless you are operating across adversarial trust boundaries (where nodes are controlled by controlled external untrusted, external third parties), PBFT is a massive, unnecessary drag on latency, bandwidth, and compute.

Gemini Notebook

Section 7/AI Prompting & YAGNI Rubric

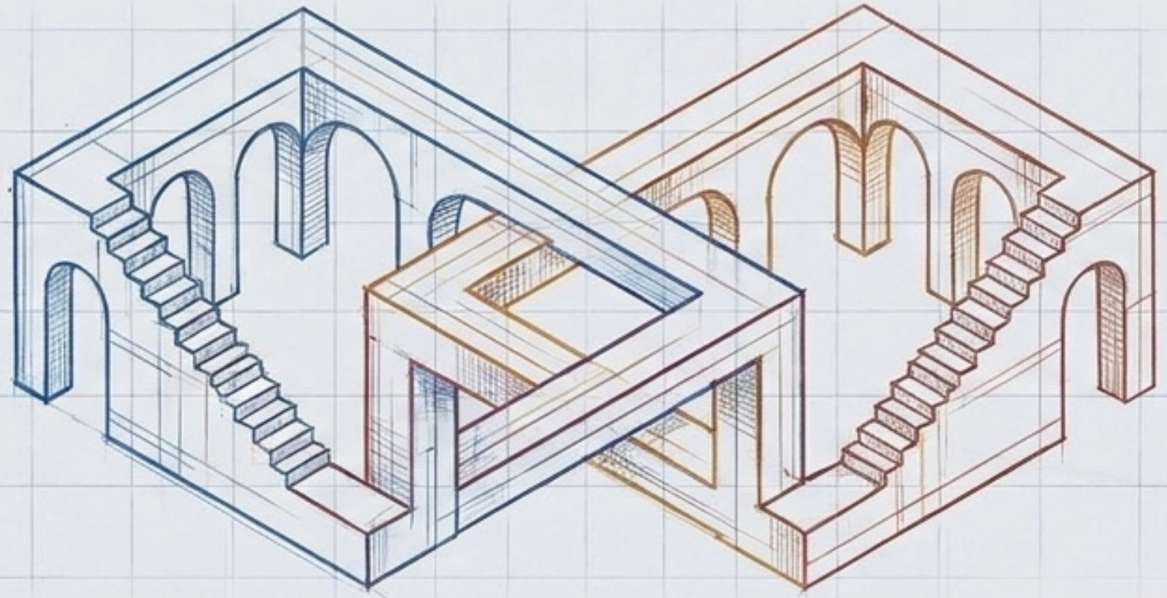
The Guardrails (Copy-Paste for CLAUDE.md / AGENTS.md)

```
# SYSTEM ARCHITECTURE GUARDRAILS
1. Default to Single-Primary Postgres for state.
2. Use pg_try_advisory_xact_lock for simple coordination; avoid Redis
   locks for strict correctness.
3. Queueing uses FOR UPDATE SKIP LOCKED unless dead-tuple bloat > 10k
   TPS.
4. Ban DIY failover scripts, 2-node clusters, and even-numbered
   quorums.
5. Never use TTLs alone for correctness. Always require Monotonic
   Fencing Tokens checked via CAS at the storage layer.
6. Assume the network is hostile, asymmetric, and will drop packets.
7. Ban PBFT/Blockchain consensus unless crossing organizational trust
   boundaries.
```

Gemini Notebook

Conclusion

The Network Will Lie to You.



Consensus is the art of extracting truth from untrusted systems. Build systems that assume hardware betrayal, rely on strict quorum math, and aggressively enforce fencing at the metal.

Arc 3 | S08E07

Next Up: Floating-Point Math (0.1 + 0.2 != 0.3) — Stepping out of the network and into the metal to explore the chaos of binary representation.

Gemini Notebook