

CPSC 416 – Distributed Systems

Lecture 22: The Failure Model as Political Choice

UBC Vancouver

2026-03-26

From Last Time

Turn to your neighbor. 2 minutes.

L17 introduced Byzantine faults. L18 showed impossibility results.

Name one system we've studied that assumes crash-fault only. *Why* does it make that assumption?

Name one that assumes Byzantine faults. What does it pay for that assumption?

(Pairs: 2 minutes)

Failure-First Question

Your cloud provider's SLA promises 99.99% availability.

What consistency model does that imply?

Who bears the cost when the SLA is violated?

Hold that question. We'll return to it.

Every System Made a Choice

Look back at the systems we've studied:

- **Paxos:** Crash-fault, strong consistency, leader-based
- **Chain Replication:** Crash-fault, linearizability, ordered topology
- **Dynamo:** Crash-fault, eventual consistency, leaderless
- **PBFT:** Byzantine, strong consistency, $3f + 1$ nodes

Each chose a failure model, a consistency guarantee, and a coordination structure.

These aren't just engineering decisions. They have winners and losers.

The Anatomy of a Technical Choice

Every failure model encodes three implicit commitments:

- ❶ **What can go wrong?** (failure model)
- ❷ **Who pays when it does?** (cost allocation)
- ❸ **Who decided?** (governance)

We've spent the course on (1). Today is about (2) and (3).

The technical choice is also a political choice.

Case Study 1: Eventual Consistency

Setup: A cloud provider offers a key-value store.

They choose **eventual consistency**.

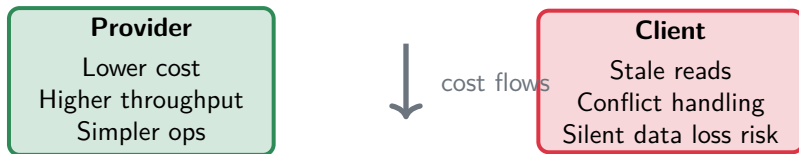
What does the provider gain?

- Cheaper to serve: no coordination on the write path
- Higher availability: any replica can serve reads
- Better tail latency: no quorum waiting

What does the client bear?

- Stale reads (may not see own writes)
- Conflict resolution (application must handle divergence)
- Debugging burden (“why did my data disappear?”)

Who Benefits?



The provider chose eventual consistency.
The client absorbs the consequences.

The choice is rational. The cost allocation is not neutral.

Checkpoint 1

Articulate:

Dynamo chose eventual consistency with vector clocks and client-side reconciliation. Who benefits from that design? The shopping cart user? The developer? Amazon's operations team?

(Pairs: 2 minutes)

Case Study 2: Crash-Fault, Not Byzantine

Every cloud consensus system you've seen uses crash-fault Paxos (or Raft).

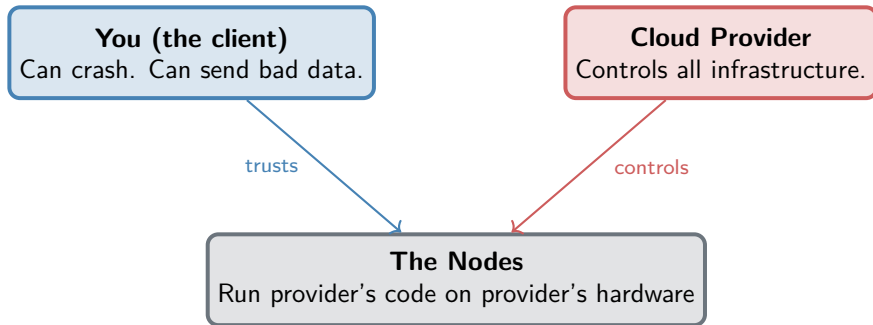
The crash-fault assumption: Nodes are honest but may fail by stopping.

What this rules out:

- Nodes lying about their state
- Nodes equivocating (sending different values to different peers)
- Nodes selectively dropping or reordering messages

Question: Who is the party most capable of doing these things?

The Trust Asymmetry



The crash-fault model says: "Nodes are honest."

The provider controls the nodes. The client chose to trust them.

Actually: the provider chose, and the client accepted.

What Would Byzantine Mean Here?

BFT for cloud databases would mean:

- $3f + 1$ nodes instead of $2f + 1$ (50% more hardware)
- $O(n^2)$ message complexity per round
- Cryptographic verification on every message
- Significantly higher latency

The provider's argument: "We're not Byzantine. Trust us. Here's our SLA."

The client's situation: Cannot verify. Can only observe outcomes.

The crash-fault assumption is an efficiency optimization *and* a trust declaration.

Case Study 3: The SLA

“99.99% available”

Sounds like a guarantee. Let's read it as an engineer.

- 99.99% of what? (requests? minutes? per-region?)
- Measured how? (provider's monitoring, not yours)
- Remedy: typically service credits (10–30% of monthly bill)
- Not remedy: your revenue loss, your users' trust, your data

The SLA is not a safety property. It's a business contract.

52 Minutes

99.99% availability = 52.6 minutes of downtime per year.

- What happens during those 52 minutes?
- What if they all occur during your peak traffic?
- What if “available” means “responding” but with stale data?

Who defines “available”?

The SLA does. The provider wrote the SLA.

An SLA is a technical-political document. Read the definitions, not just the number.

Checkpoint 2

Articulate:

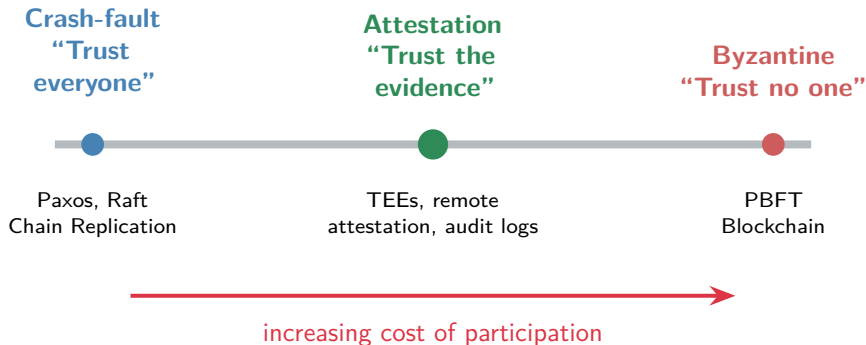
An SLA promises 99.99% availability for a strongly-consistent database.

During the 0.01% downtime, the system switches to eventually-consistent reads to maintain “availability.”

Is the SLA honest? Who benefits from this definition?

(Small groups: 3 minutes)

The Trust Spectrum



Crash-fault is cheap but assumes honesty.
Byzantine is expensive but assumes nothing.

Is there a structural middle path?

Attestation: Trust the Evidence

Idea: Don't trust the node. Trust what the node can *prove*.

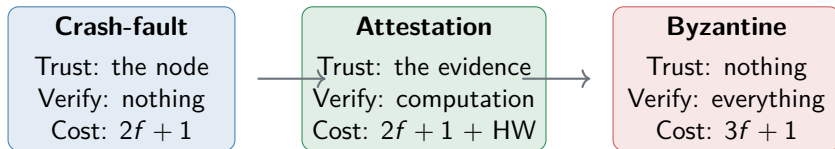
Trusted Execution Environments (TEEs):

- Hardware-enforced isolation (Intel SGX, ARM TrustZone, AMD SEV)
- Code runs in an enclave the host OS cannot inspect or modify
- The hardware signs a measurement of what's running

Remote Attestation:

- A node proves it's running specific code on genuine hardware
- The verifier doesn't trust the node — it trusts the *attestation*
- If the code is correct, the output is trustworthy

What Attestation Changes



Attestation keeps the crash-fault quorum size ($2f + 1$) but adds verifiable computation.

The provider can still crash. But they can't lie about *what ran*.

This is the structural middle path: not “trust me,” but “check my work.”

Testimony vs Telemetry

Testimony: “Here is what happened” (a claim, controllable by the speaker).

Telemetry: “Here is the computation trace” (observable, not easily fabricated).

- An SLA is testimony: the provider’s claim about their behavior
- A TEE attestation is telemetry: hardware-signed evidence of execution
- A log entry is testimony if the writer controls the log
- A log entry is telemetry if it’s append-only, replicated, and signed

The failure model question becomes: what kind of evidence do you demand?

Checkpoint 3

Articulate:

A cloud provider runs your database on SGX enclaves with remote attestation.

What can the provider still do that harms you?

(Hint: what does attestation NOT cover?)

(Pairs: 2 minutes)

What Attestation Doesn't Cover

The provider can still:

- **Delay messages** (availability attack, not integrity)
- **Deny service** (pull the plug — crash-fault, not Byzantine)
- **Observe access patterns** (side channels, even with SGX)
- **Change the code between attestations** (time-of-check vs time-of-use)

Attestation narrows the attack surface. It doesn't eliminate it.

But it shifts the failure model from “trust me” to “verify this.”

That shift changes who bears the risk.

In-Class Activity

Scenario:

AWS offers you two database options for your capstone project:

	Option A	Option B
Consistency	Strongly consistent	Eventually consistent
Write latency	15ms	3ms
Monthly cost	\$X	\$X/3
SLA	99.9%	99.99%

Your task (groups of 3, 10 minutes):

- 1 Which do you choose? Justify with a failure-first argument.
- 2 What are you *accepting* by making this choice?
- 3 Who bears the cost of the failure mode you chose not to handle?

Activity Debrief

If you chose **Option A (strong consistency)**:

- You pay more money and higher latency
- The provider pays for coordination overhead
- Your users get correct reads, but lower availability (99.9%)
- Your users bear the availability cost

If you chose **Option B (eventual consistency)**:

- You pay less, get faster writes, higher availability
- The provider avoids coordination costs
- Your users may read stale data
- Your users bear the consistency cost

Either way, someone who didn't make the decision absorbs the consequence.

The Question Behind the Question

“Which database should I use?” is an engineering question.

“Who decided what failure modes are acceptable?” is a governance question.

In every system we've studied:

- The designer chose the failure model
- The operator chose the deployment configuration
- The user absorbed the residual risk

The user is rarely the one who chose.

Connecting Back

L17 (Byzantine faults):

We asked: what if nodes lie? The answer was BFT, at high cost.

Today's question: who decided we could assume they don't?

L18 (Impossibility results):

FLP and CAP constrain the design space.

Today's question: within those constraints, someone still chooses. Who?

L14 (Flexible Paxos):

Quorum sizes are a design choice, not a law of nature.

Same principle: the math constrains. The politics selects.

Looking Forward

L23 (CRDTs):

What if we design data structures that *never conflict*?

CRDTs are a technical escape from the coordination question — but they constrain what operations are possible. *Who decides which operations to give up?*

L24 (Self-Reporting Systems):

What if the system reports its own trustworthiness?

Attestation, audit trails, verifiable computation — building systems where trust is earned through evidence, not assumed by fiat.

Evidence Question

You're evaluating a cloud database for a system that handles financial transactions.

The provider says: "We use Raft for consensus. It's safe."

What evidence would you need to trust that claim?

What evidence is it impossible for them to provide under a crash-fault model?

(Consider: can you distinguish "correct but slow" from "Byzantine but hiding it"?)

What We Learned

- Every failure model is a **choice** with winners and losers
- Eventual consistency shifts cost to the client
- Crash-fault models assume the infrastructure operator is honest
- SLAs are business contracts, not safety properties
- **Attestation** offers a structural middle path: trust evidence, not nodes
- The testimony/telemetry distinction: claims vs verifiable computation
- Impossibility results constrain the design space; politics fills it

Technical choices encode values. The question is whether we make those values explicit.

- **Reading:** No new paper. Revisit a prior reading through the lens of “who decided?”
- **Self-assessment:** Opens 13:00, due next class 11:30 (Pacific)
- **Next lecture (L23):** CRDTs — escaping coordination through data structure design
- Capstone check-in: details on Canvas

For next time: Pick one system from the course. Identify the failure model choice, and ask: who benefits?