

CPSC 416 – Distributed Systems

Lecture 17: Byzantine Fault Tolerance

UBC Vancouver

2026-03-10

From Last Time

Turn to your neighbor. 2 minutes.

Every protocol we've studied so far assumes nodes are either correct or silent (crashed).

What specific property of crash failures makes Paxos work?

(Pairs: 2 minutes)

The Assumption We've Been Making

Crash-stop failure model:

- A node either follows the protocol correctly, or stops
- A crashed node sends *no messages*
- It never sends a wrong message, a contradictory message, or a lie

Every quorum argument in L10–L14 relied on this:

If a node is in a quorum, and it says “I accepted v at ballot b ”...

We believed it.

Failure-first question

Every protocol so far assumes nodes are either correct or silent.

What happens when a node sends different messages to different peers?

What happens when a node *lies*?

The Failure Model Changes

Crash failures (L10–L14):

- Correct or silent
- f failures out of n nodes $\rightarrow$ need $n \geq 2f + 1$ for majority quorums

Byzantine failures (today):

- Arbitrary behavior: wrong values, contradictory messages, selective silence
- Includes malicious behavior (worst case)
- Includes bugs that produce arbitrary outputs (realistic case)

A Byzantine node can say “I accepted X ” to you and “I accepted Y ” to someone else.

*This is called **equivocation**.*

Why “Byzantine”?

Lamport, Shostak, Pease (1982):

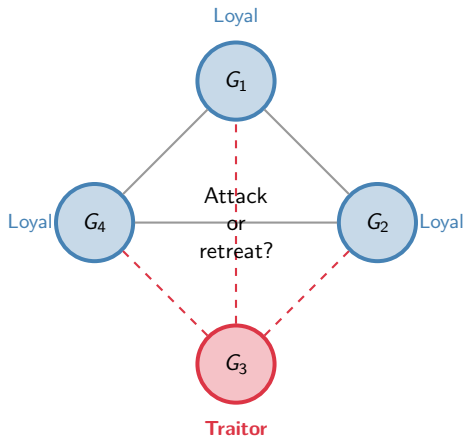
*Generals of the Byzantine army surround a city. They must coordinate: attack or retreat. Some generals are **traitors** who send different orders to different generals.*

The name is a metaphor for the hardest kind of distributed failure:

- You can't distinguish a traitor from a loyal general by looking at one message
- A traitor actively tries to cause disagreement
- The loyal generals must agree *despite* the traitors

The model doesn't require actual malice. It requires tolerance of *any* behavior.

The Byzantine Generals Problem



4 generals, 1 traitor. Must agree on a single plan.
The traitor can say anything to anyone.

Equivocation: The Core Problem



G_1 thinks:
“ G_3 said Attack”

G_2 thinks:
“ G_3 said Retreat”

G_1 and G_2 have contradictory information from the same source.

Neither can tell who is lying
without consulting each other.

Checkpoint 1

Articulate:

In crash-fault Paxos, why doesn't equivocation happen?

What property of crash failures prevents a node from saying different things to different peers?

(Pairs: 2 minutes)

Why $2f+1$ Is Not Enough

Recall L10: With crash failures, $n = 2f + 1$ nodes suffice.

Majority quorum = $f + 1$. Two majorities overlap by at least 1.

But with Byzantine failures:

The f faulty nodes can be in *both* quorums.

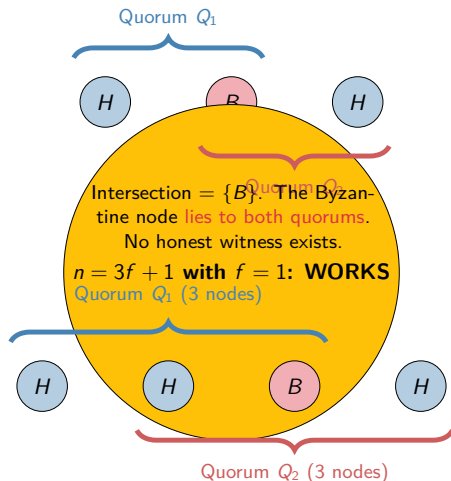
And they can **say different things to each quorum**.

The overlap node might be the liar.

With $n = 2f + 1$ and f Byzantine: the intersection could be entirely Byzantine.

The Counting Argument

$n = 2f + 1$ with $f = 1$: **FAILS**



Intersection = 2 nodes. Even if B is one of them,
at least 1 honest node witnesses both quorums.

The $3f + 1$ Bound: Why

We need quorum intersection to contain at least one honest node.

With f Byzantine nodes:

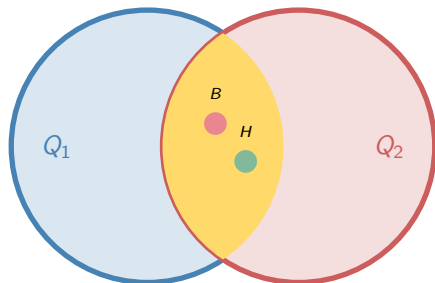
- A quorum must have size $> \frac{n+f}{2}$ (so intersection $> f$)
- Equivalently: quorum size $\geq 2f + 1$
- For two quorums of size $2f + 1$ from n nodes to intersect in $> f$ nodes:

$$|Q_1| + |Q_2| - n > f$$

$$(2f + 1) + (2f + 1) - n > f \implies n < 3f + 2 \implies n \geq 3f + 1$$

$3f + 1$ ensures that after removing f Byzantine nodes from the intersection, at least one honest witness remains.

Connect to L10: The Quorum Intersection Revisited



Failure model	Bound	Why
Crash	$n \geq 2f + 1$	Intersection ≥ 1 honest (none lie)
Byzantine	$n \geq 3f + 1$	Intersection $> f$ (outvote the liars)

Same quorum intersection argument. Different adversary. Different bound.

Checkpoint 2

Articulate:

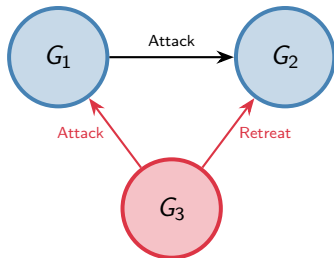
With 7 nodes and 2 Byzantine, what is the minimum quorum size?

How many nodes are in the intersection of two quorums? How many of those could be Byzantine?

(Pairs: 2 minutes)

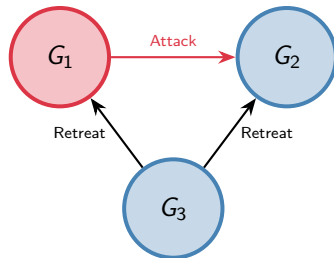
The Impossibility: 3 Generals, 1 Traitor

Scenario 1: G_3 is traitor



G_2 sees: 1 Attack, 1 Retreat.
Cannot decide.

Scenario 2: G_1 is traitor



G_2 sees: 1 Attack, 1 Retreat.
Cannot decide.

G_2 receives the same messages in both scenarios.
It cannot distinguish " G_3 is the traitor" from " G_1 is the traitor."

The Lower Bound Result

Theorem (Lamport, Shostak, Pease 1982)

No protocol can achieve Byzantine agreement among n processes if $n \leq 3f$, even with authenticated channels.

Intuition: With $n = 3f$, you can construct a scenario where no honest process can tell which f nodes are lying.

The proof uses a *simulation argument*: a set of f honest nodes running one scenario is indistinguishable from f Byzantine nodes running a different scenario.

$3f + 1$ is tight. It's both necessary and sufficient.

In-Class Activity: Detect the Traitor

Setup: 4 generals, 1 Byzantine traitor.

Generals G_1 , G_2 , G_4 are loyal. General G_3 is the traitor.

Round 1: Each general sends its value to all others.

- G_1 sends “Attack” to all
- G_2 sends “Attack” to all
- G_3 sends “Attack” to G_1 , “Retreat” to G_2 , “Retreat” to G_4
- G_4 sends “Attack” to all

Task: Walk through Round 2 (relay what you heard).

Show that the 3 honest generals can identify the inconsistency.

(Groups of 4: 10 minutes)

Activity Debrief: Round 2

Round 2: Each general relays what it heard from others.

	Heard from G_1	from G_2	from G_3	from G_4
G_1 sees	—	Atk	Atk	Atk
G_2 sees	Atk	—	Ret	Atk
G_4 sees	Atk	Atk	Ret	—

In Round 2, G_1 tells G_2 : “ G_3 told me Attack.”

G_2 and G_4 both heard “Retreat” from G_3 .

Majority vote on G_3 's value: 1 says Attack, 2 say Retreat.

→ Resolve to Retreat for G_3 's input. Use majority of all inputs: Attack wins 3–1.

With 4 nodes and 1 traitor, the 3 honest nodes always outvote the liar.

With 3 nodes and 1 traitor, they can't.

From Theory to Protocol: pBFT

Practical Byzantine Fault Tolerance (Castro & Liskov, 1999)

First protocol that made BFT viable for real systems.

Setting:

- $n = 3f + 1$ replicas
- One designated *primary* (like a Paxos leader)
- Clients send requests to the primary
- Replicas execute requests and send replies to the client
- Client accepts when $f + 1$ matching replies received

The structure looks like Multi-Paxos with extra rounds.

pBFT Message Pattern



3 phases of all-to-all communication (vs. 2 phases in Paxos).
The extra round is the cost of tolerating equivocation.

pBFT: Why Three Phases?

Pre-prepare (primary $\rightarrow$ all):

- Primary assigns sequence number to request
- “Here’s what to do next”

Prepare (all $\rightarrow$ all):

- Each replica confirms it received the same pre-prepare
- Wait for $2f$ matching prepares (plus your own = $2f + 1$ total)
- This detects a Byzantine primary that sent different messages

Commit (all $\rightarrow$ all):

- Each replica announces it’s ready to commit
- Wait for $2f + 1$ matching commits
- This ensures enough honest replicas agree, surviving view changes

Prepare catches equivocation. Commit ensures durability across view changes.

View Changes: When the Primary Is Byzantine

What if the primary is the traitor?

Replicas detect a faulty primary by timeout:

- If a replica doesn't receive a valid pre-prepare in time, it suspects the primary
- $2f + 1$ replicas send view-change messages
- New primary collects view-change messages and proves it's legitimate

Compare to Paxos:

- Paxos leader failure $\rightarrow$ new Phase 1 with higher ballot
- pBFT primary failure $\rightarrow$ view change with $2f + 1$ agreement

Same structure: the leader is replaced, but committed values survive.

But view changes in pBFT are much more expensive ($O(n^2)$ messages).

Checkpoint 3

Articulate:

Paxos has 2 phases. pBFT has 3 phases.

What does the extra phase buy you?

What specific attack does it prevent?

(Pairs: 2 minutes)

The Cost of Byzantine Tolerance

	Crash fault (Paxos)	Byzantine (pBFT)
Nodes for f faults	$2f + 1$	$3f + 1$
Phases per operation	2	3
Messages per phase	$O(n)$	$O(n^2)$
Total messages	$O(n)$	$O(n^2)$
View change	$O(n)$	$O(n^3)$

For $f = 1$: Paxos needs 3 nodes. pBFT needs 4.

For $f = 10$: Paxos needs 21 nodes. pBFT needs 31.

The quadratic message complexity is the real killer.

At 100 nodes, each operation generates $\sim 10,000$ messages.

When Is BFT Worth It?

Almost never in practice. But knowing *when* it is matters.

BFT is NOT needed when:

- You control all the nodes (your own data center)
- You trust the hardware and the operators
- Crash faults are the dominant failure mode

BFT IS needed when:

- Nodes are operated by mutually distrusting parties
- Software bugs can produce arbitrary outputs (not just crashes)
- Regulatory requirements demand tolerance of compromised nodes
- The system must survive insider threats

The question isn't "can nodes lie?" It's "do you need to tolerate it?"

What If You Can't Enumerate Participants?

pBFT requires $3f + 1$ *known, fixed* nodes.

What if anyone can join? How do you count f ?

Nakamoto (2008): Replace node identity with *computational work*.

- Byzantine tolerance without knowing who the participants are
- Sybil resistance: creating identities is cheap, but proof-of-work is not
- Trade-off: enormous energy cost, probabilistic finality, low throughput

Blockchain is one answer to:

“What if you can't enumerate participants?”

It's not the only answer, and it's rarely the best answer.

But the **question** is real, and understanding BFT is prerequisite to evaluating the answers.

Evidence Question

You deploy a 4-node pBFT system ($f = 1$).

One node starts sending different pre-prepare messages to different replicas.

How would the honest replicas detect this?

What would you see in the logs?

(Hint: which phase catches equivocation?)

What We Learned

- **Byzantine failure model:** nodes can behave arbitrarily, including lying
- **Equivocation** is the core new threat: different messages to different peers
- $3f + 1$ because quorum intersection must contain $> f$ nodes
- Same quorum intersection argument as L10, **different bound**
- **pBFT:** 3 all-to-all phases (the extra round catches equivocation)
- $O(n^2)$ **messages per operation** — BFT is expensive
- BFT matters when nodes are untrusted or bugs produce arbitrary outputs

The thread:

L10: Quorum intersection is the foundation.

L14: Flexible Paxos shows the intersection constraint is more general than majority.

L17: Byzantine faults show the intersection constraint is *stricter* when nodes can lie.

Next: The Impossibility Zoo — BFT, FLP, and CAP as a unified landscape.

- **Reading:** “The Byzantine Generals Problem” (Lamport, Shostak, Pease, 1982)
- **Optional:** “Practical Byzantine Fault Tolerance” (Castro & Liskov, 1999)
- **Self-assessment:** Opens 13:00, due next class 11:30 (Pacific)
- **Next lecture (L18):** The Impossibility Zoo — FLP, CAP, and BFT bounds
- Capstone checkpoint details on Canvas