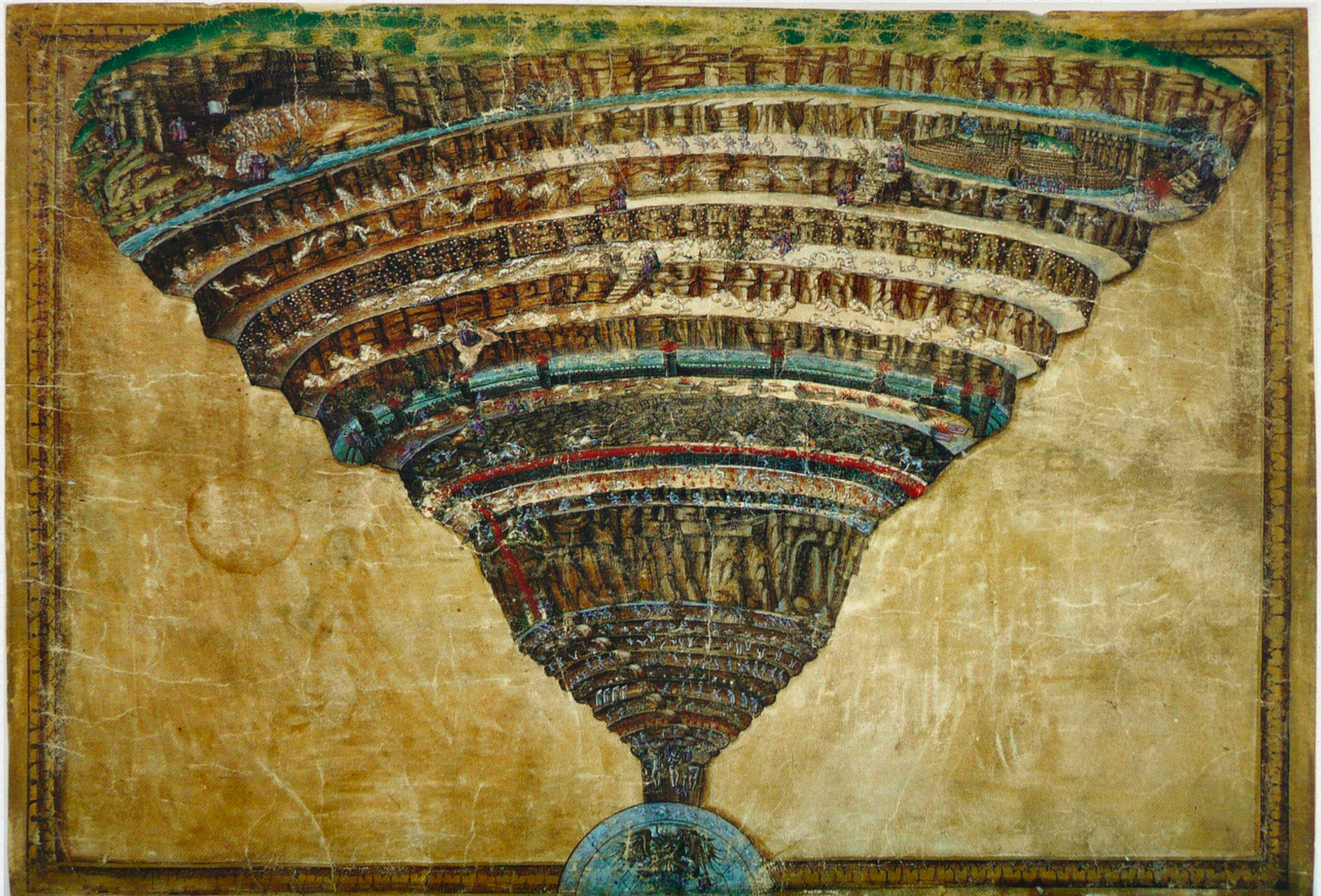
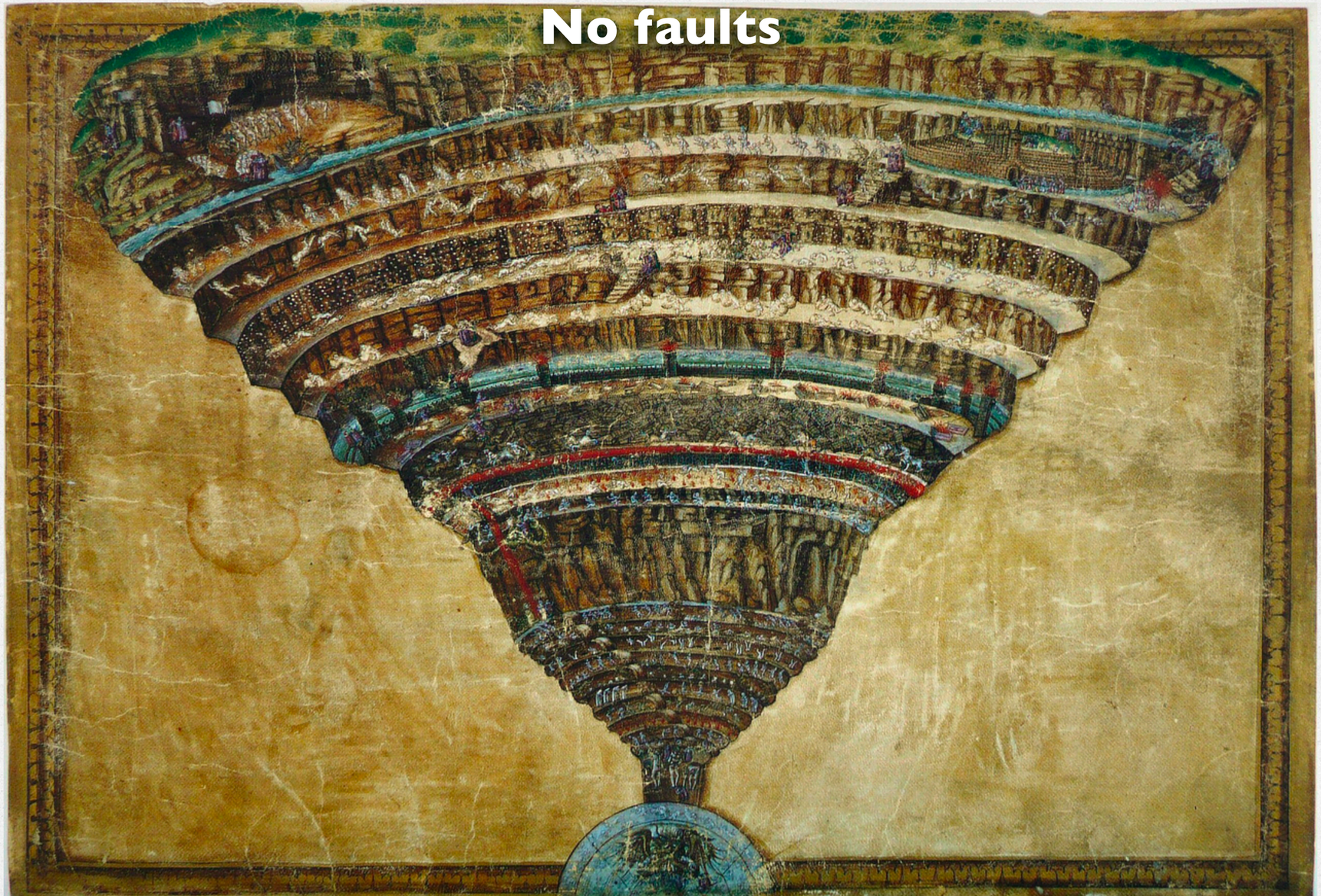
Byzantine Fault Tolerance

Ellis Michael

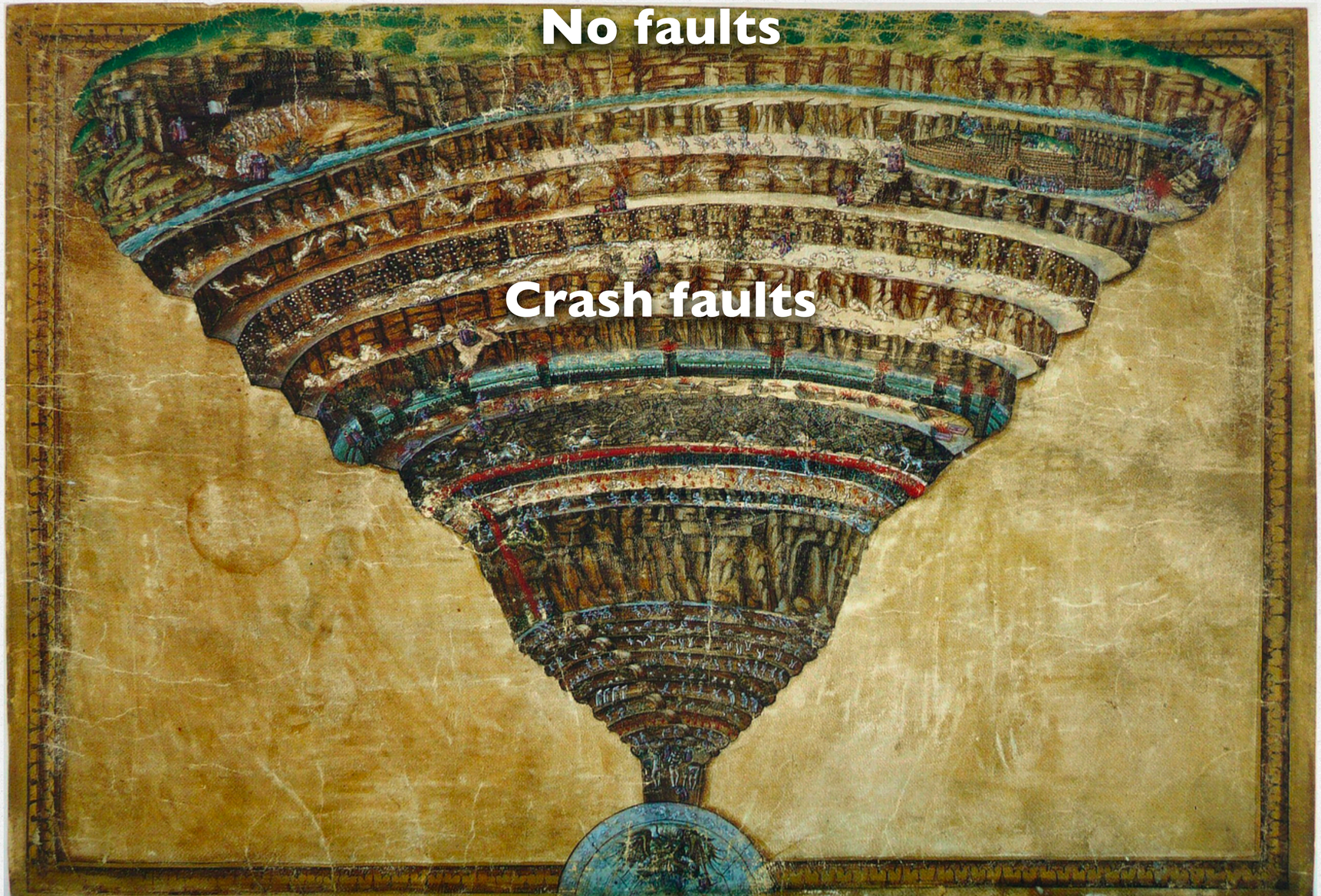
A Hierarchy of Fault Models



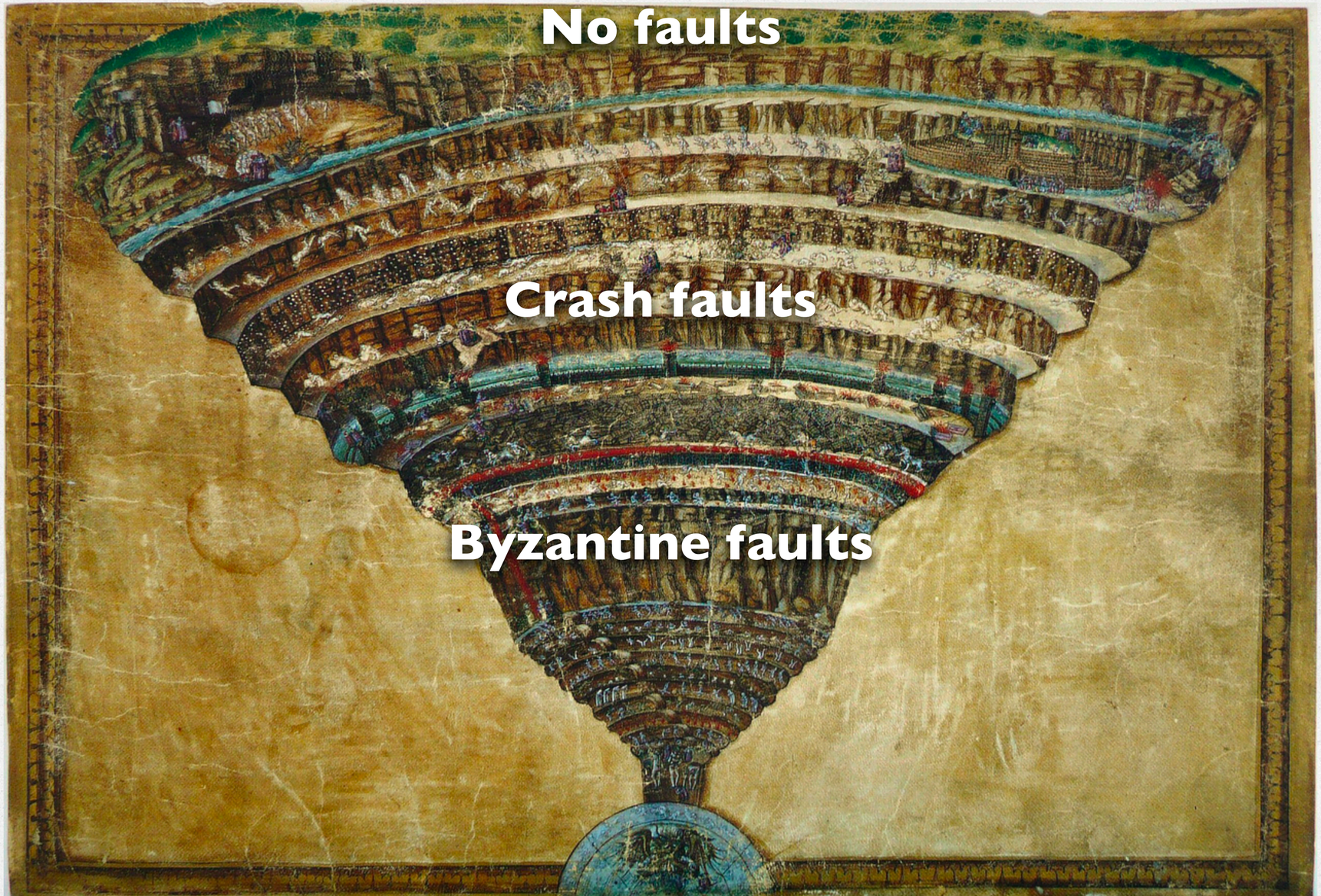
A Hierarchy of Fault Models



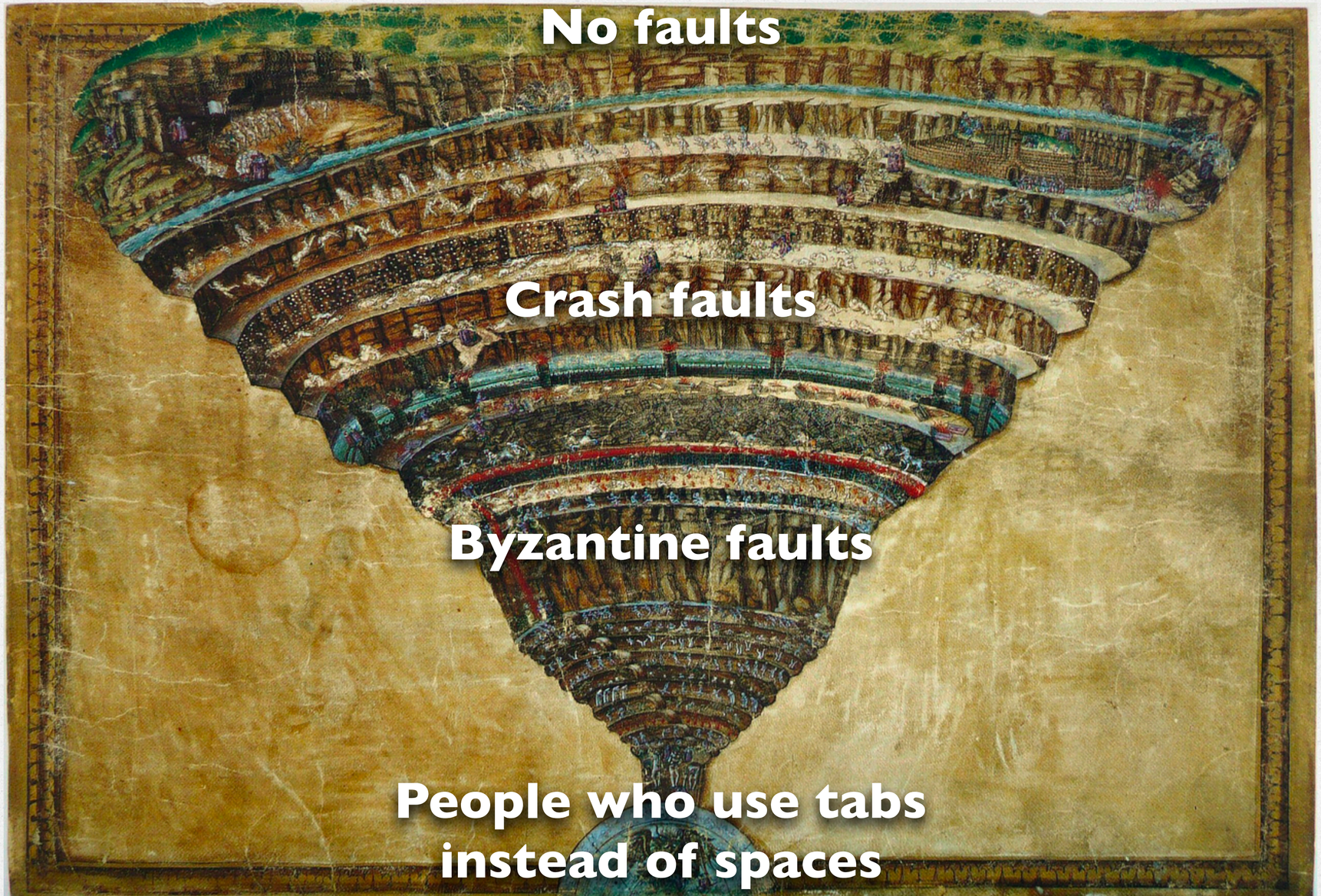
A Hierarchy of Fault Models



A Hierarchy of Fault Models



A Hierarchy of Fault Models



Byzantine Faults

- Also called "general" or "arbitrary" faults.
- Faulty nodes can take any actions. They can send any messages, collude with each other, etc. in an attempt to "trick" the non-faulty nodes and subvert the protocol.
- Why this model?

Strange Things Happen at Scale

- Hardware failures are real and can cause both crashes and aberrant behavior.
- Cosmic rays from outer space (!) can and will randomly flip bits in memory.
- Software bugs are all too common.
- Security vulnerabilities can let attackers into distributed systems.

Strange Things Happen at Scale

- Hardware failures are real and can cause both crashes and aberrant behavior.
- Cosmic rays from outer space (!) can and will randomly flip bits in memory.
- Software bugs are all too common.
- Security vulnerabilities can let attackers into distributed systems.

EmailMarketingDAILY

Gmail Down In Parts Of U.S., Google Says It Is Working On Problem

by **Ray Schultz** , May 6, 2019

Gmail is suffering service outages in the United States, largely in the Northeast and Southern California.

Downdetector has received numerous complaints, spiking at around 5:20 p.m. on Monday. The service problem is interfering with people doing their work.

"I can't attach files," writes Jerry Jordan. "I keep getting errors and it's causing me some serious issues on press deadlines for NASCAR coverage."

"is there an outage?" writes giaEP. "I cant seem to do anything. Keeps saying oops something went wrong. I need to do my work!!!!! Im working on safari."

One more complainant says: "Emails are no longer displaying in my Inbox."

Strange Things Happen at Scale

- Hardware failures are real and can cause both crashes and aberrant behavior.
- Cosmic rays from outer space (!) can and will randomly flip bits in memory.
- Software bugs are all too common.
- Security vulnerabilities can let attackers into distributed systems.

EmailMarketingDAILY

Gmail Down In Parts Of U.S., Google Says It Is Working On Problem

by Ray Schultz May 6, 2019

TECH & SCIENCE

MICROSOFT OUTAGE: AZURE, OFFICE 365 SHAREPOINT, ONEDRIVE AND MORE DOWN LEAVING USERS UNABLE TO ACCESS FEATURES

BY CAMMY HARBISON ON 5/2/19 AT 5:36 PM EDT

SHARE

TECH & SCIENCE MICROSOFT

Are Microsoft servers down or offline? Find out everything you need to know about the outages affecting Microsoft Azure, Office 365 Sharepoint and more.

Having trouble using Microsoft services? You aren't alone. A massive outage across Microsoft's services is affecting users around the world. Outages have been reported in Microsoft Azure and Office 365's Sharepoint and OneDrive along with several other services.

The issues were first reported around 3:45 p.m. ET and have been occurring intermittently throughout the afternoon and evening.

Microsoft has acknowledged the issues on Office 365's server status page. According to the company, affected services include SharePoint Online, OneDrive for Business, Microsoft Teams.

Northeast

5:20 p.m.
r work.

causing

saying
safari."

nbox."

Strange Things Happen at Scale

- Hardware failures are real and can cause both crashes and aberrant behavior.
- Cosmic rays from outer space (!) can and will randomly flip bits in memory.
- Software bugs are all too common.
- Security vulnerabilities can let attackers into distributed systems.

EmailMarketingDAILY

Gmail Down In Parts Of U.S., Google Says It Is Working On Problem

by Ray Schultz May 6, 2019

TECH & SCIENCE

MICROSOFT OUTAGE: AZURE, OFFICE 365 SHAREPOINT, ONEDRIVE AND MORE DOWN LEAVING USERS UNABLE TO ACCESS FEATURES

BY CAMMY HARBISON ON 5/2/19 AT 5:36 PM EDT

Northeast

5:20 p.m.

r work.

causing

SHARE

TECH & SCIENCE MICROSOFT

Are Microsoft servers down or offline the outages affecting Microsoft Azu

Having trouble using Microsoft services? Yo services is affecting users around the world. Office 365's Sharepoint and OneDrive along

The issues were first reported around 3:45 p throughout the afternoon and evening.

Microsoft has acknowledged the issues on company, affected services include SharePo

Apple's iPhone and Mac App Stores are down for some users

Jacob Siegal @JacobSiegal
May 8th, 2019 at 1:44 PM

Share Tweet

According to Apple's System Status page on its site, several of its major services are currently experiencing outages. The mobile App Store, Mac App Store, and iTunes Store are all affected, and Apple says that the issues arose at 4:40 ET this morning. That said, reports of the outages have only begun popping up recently, and the reports don't appear to be widespread, leading us to believe this is a relatively contained outage.

If the outage is regional, it isn't affecting the Northeast, as none of us here at BGR in New York or New Jersey have noticed any problems with any of Apple's services today. The incredibly small number of reports on *DownDetector* make it difficult to determine exactly where the outages are affecting users.

DON'T MISS

10 deals you don't want to miss on Saturday: AirPods 2, $35 Fire TV Stick 4K, $10 Philips Hue bulbs, more

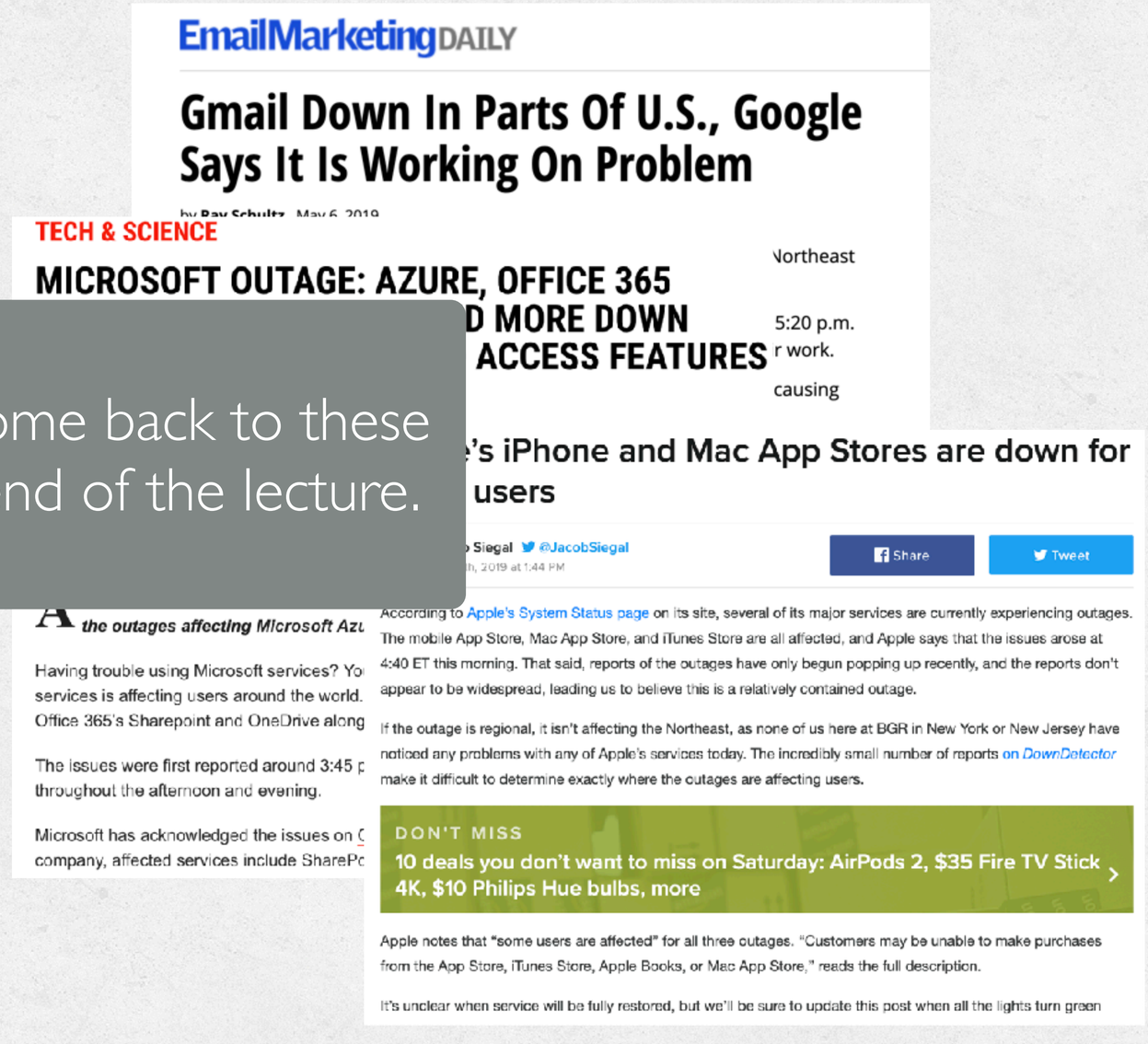
Apple notes that "some users are affected" for all three outages. "Customers may be unable to make purchases from the App Store, iTunes Store, Apple Books, or Mac App Store," reads the full description.

It's unclear when service will be fully restored, but we'll be sure to update this post when all the lights turn green

Strange Things Happen at Scale

- Hardware failures are real and can cause both crashes and aberrant behavior.
- Cosmic rays from outer space can and will randomly flip bits memory.
- Software bugs are all too common.
- Security vulnerabilities can let attackers into distributed systems.

We'll come back to these at the end of the lecture.



EmailMarketingDAILY

Gmail Down In Parts Of U.S., Google Says It Is Working On Problem

by Ray Schultz May 6, 2019

TECH & SCIENCE

MICROSOFT OUTAGE: AZURE, OFFICE 365 D MORE DOWN ACCESS FEATURES

Northeast

5:20 p.m.

r work.

causing

's iPhone and Mac App Stores are down for users

Siegal @JacobSiegal

h, 2019 at 1:44 PM

Share

Tweet

the outages affecting Microsoft Azu

Having trouble using Microsoft services? Yo services is affecting users around the world. Office 365's Sharepoint and OneDrive along

The issues were first reported around 3:45 p throughout the afternoon and evening.

Microsoft has acknowledged the issues on company, affected services include SharePo

According to Apple's System Status page on its site, several of its major services are currently experiencing outages. The mobile App Store, Mac App Store, and iTunes Store are all affected, and Apple says that the issues arose at 4:40 ET this morning. That said, reports of the outages have only begun popping up recently, and the reports don't appear to be widespread, leading us to believe this is a relatively contained outage.

If the outage is regional, it isn't affecting the Northeast, as none of us here at BGR in New York or New Jersey have noticed any problems with any of Apple's services today. The incredibly small number of reports on *DownDetector* make it difficult to determine exactly where the outages are affecting users.

DON'T MISS

10 deals you don't want to miss on Saturday: AirPods 2, $35 Fire TV Stick 4K, $10 Philips Hue bulbs, more

Apple notes that "some users are affected" for all three outages. "Customers may be unable to make purchases from the App Store, iTunes Store, Apple Books, or Mac App Store," reads the full description.

It's unclear when service will be fully restored, but we'll be sure to update this post when all the lights turn green

What About Paxos?

- Paxos tolerates a minority of processing failing by **crashing**.
- What could a malicious replica do to a Paxos deployment?

What About Paxos?

- Paxos tolerates a minority of processing failing by **crashing**.
- What could a malicious replica do to a Paxos deployment?
  - Stop processing requests.

What About Paxos?

- Paxos tolerates a minority of processing failing by **crashing**.
- What could a malicious replica do to a Paxos deployment?
  - Stop processing requests.
  - A leader could report incorrect results to a client.

What About Paxos?

- Paxos tolerates a minority of processing failing by **crashing**.
- What could a malicious replica do to a Paxos deployment?
  - Stop processing requests.
  - A leader could report incorrect results to a client.
  - A follower could acknowledge a proposal and then discard it.

What About Paxos?

- Paxos tolerates a minority of processing failing by **crashing**.
- What could a malicious replica do to a Paxos deployment?
  - Stop processing requests.
  - A leader could report incorrect results to a client.
  - A follower could acknowledge a proposal and then discard it.
  - A follower could respond to prepare messages without all previously acknowledged commands.

What About Paxos?

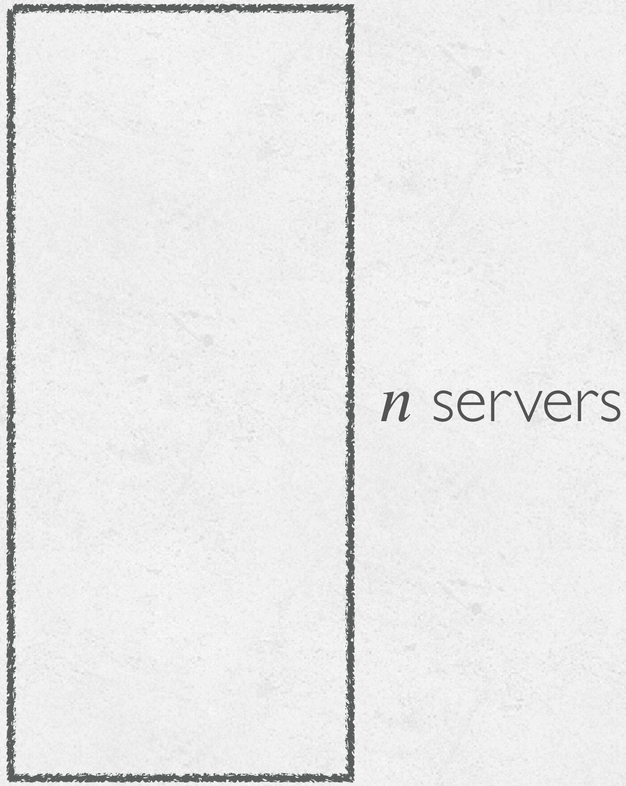
- Paxos tolerates a minority of processing failing by **crashing**.
- What could a malicious replica do to a Paxos deployment?
  - Stop processing requests.
  - A leader could report incorrect results to a client.
  - A follower could acknowledge a proposal and then discard it.
  - A follower could respond to prepare messages without all previously acknowledged commands.
  - A server could continually start new leader elections.

What About Paxos?

- Paxos tolerates a minority of processing failing by **crashing**.
- What could a malicious replica do to a Paxos deployment?
  - Stop processing requests.
  - A leader could report incorrect results to a client.
  - A follower could acknowledge a proposal and then discard it.
  - A follower could respond to prepare messages without all previously acknowledged commands.
  - A server could continually start new leader elections.
  - ...

Byzantine Quorums

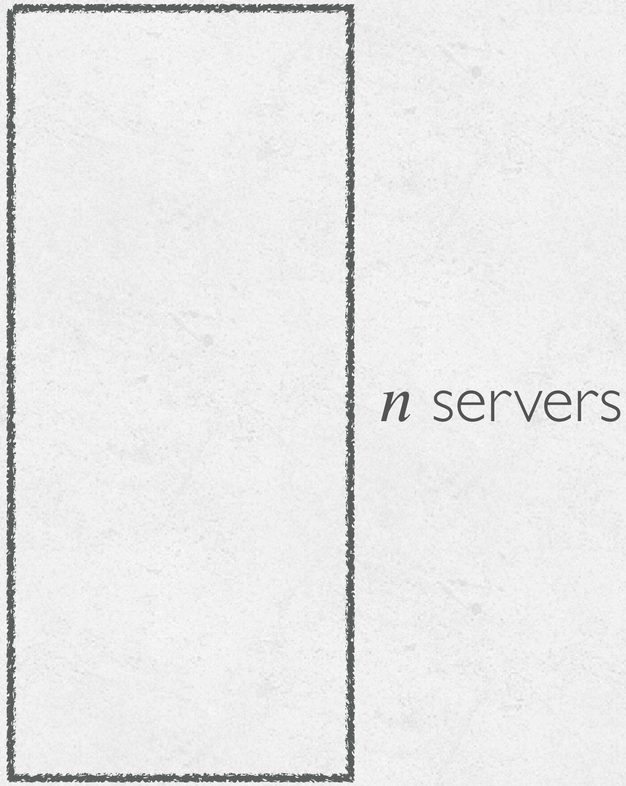
Obviously, if all servers are Byzantine, we can't guarantee anything. How many servers do we need to tolerate f faults?



Byzantine Quorums

Obviously, if all servers are Byzantine, we can't guarantee anything. How many servers do we need to tolerate f faults?

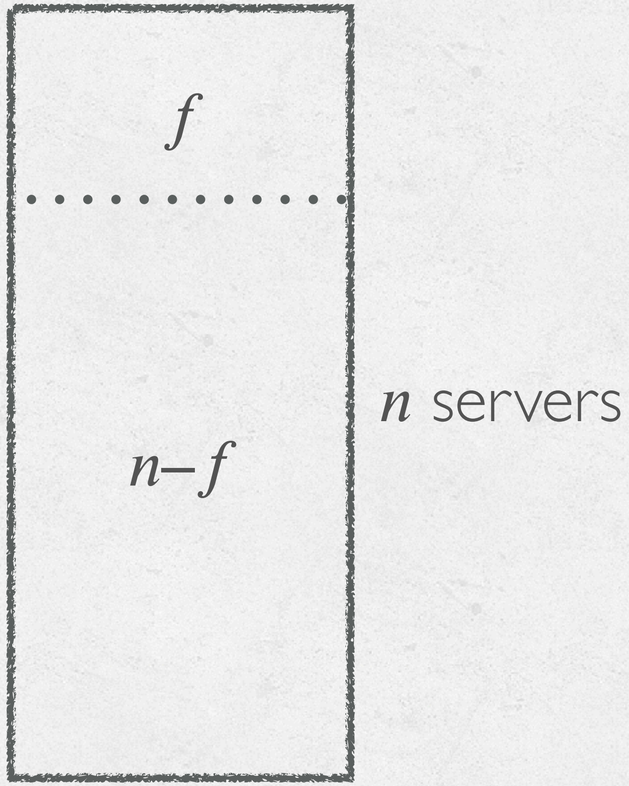
- In order to make progress, we can only wait for $n-f$ servers.



Byzantine Quorums

Obviously, if all servers are Byzantine, we can't guarantee anything. How many servers do we need to tolerate f faults?

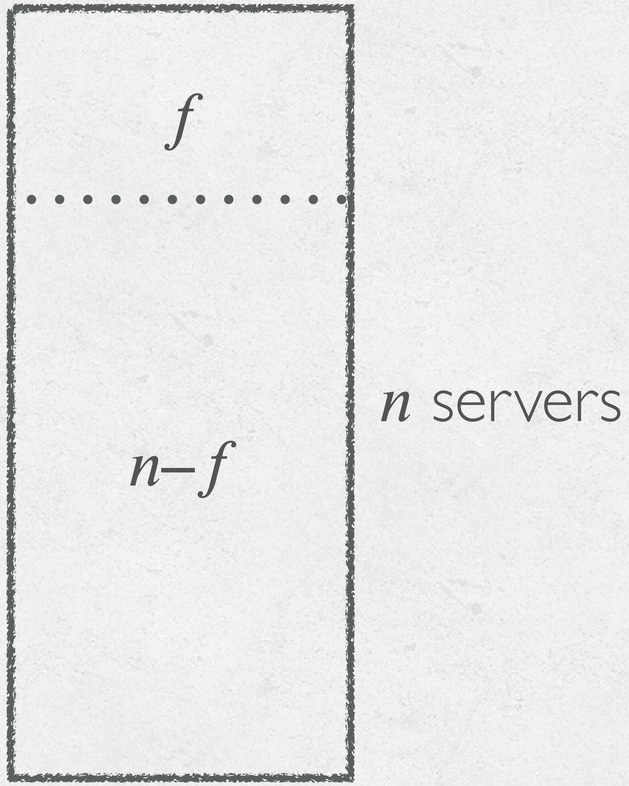
- In order to make progress, we can only wait for $n-f$ servers.



Byzantine Quorums

Obviously, if all servers are Byzantine, we can't guarantee anything. How many servers do we need to tolerate f faults?

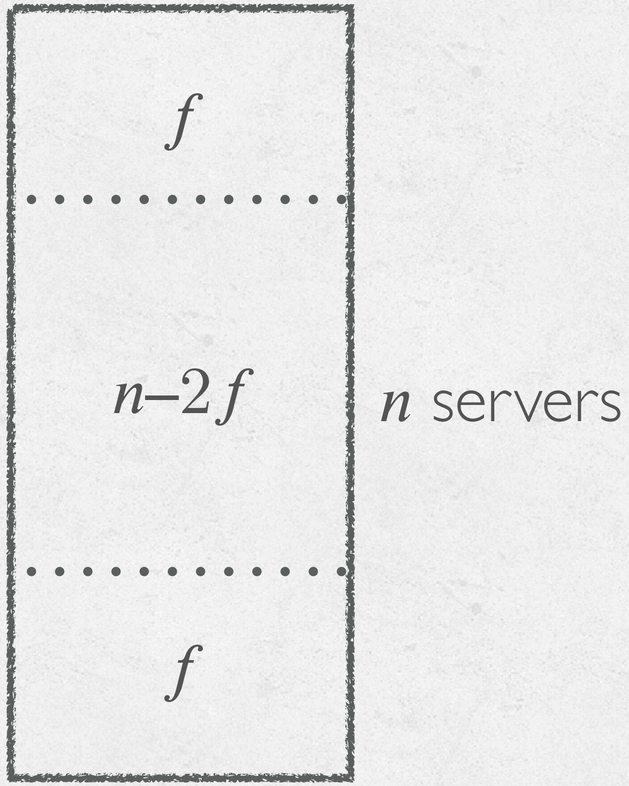
- In order to make progress, we can only wait for $n-f$ servers.
- What if two different servers contact $n-f$ quorums? If they intersect at f or fewer servers, that's not good.



Byzantine Quorums

Obviously, if all servers are Byzantine, we can't guarantee anything. How many servers do we need to tolerate f faults?

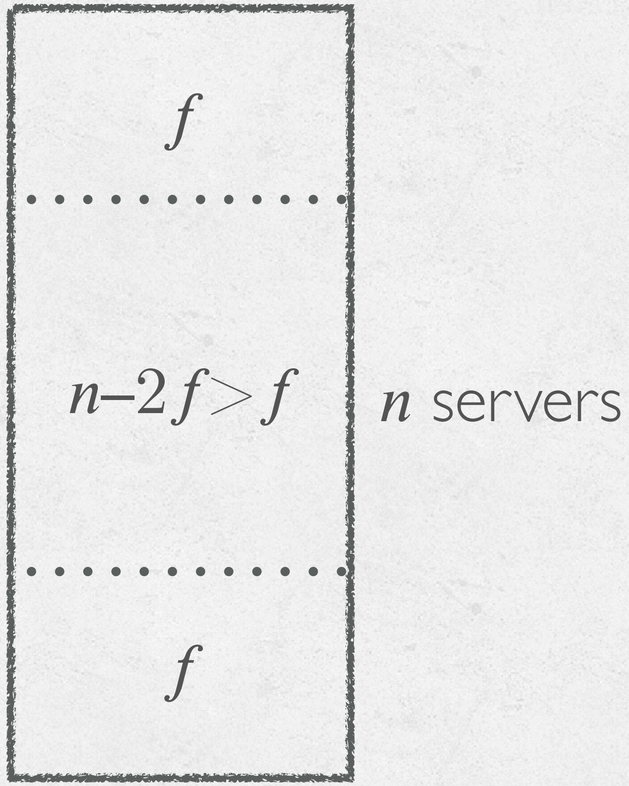
- In order to make progress, we can only wait for $n-f$ servers.
- What if two different servers contact $n-f$ quorums? If they intersect at f or fewer servers, that's not good.



Byzantine Quorums

Obviously, if all servers are Byzantine, we can't guarantee anything. How many servers do we need to tolerate f faults?

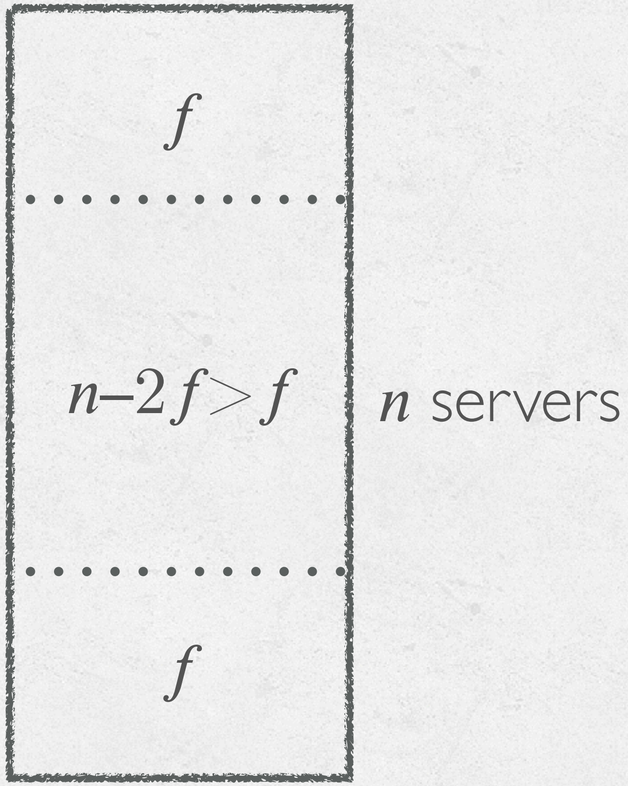
- In order to make progress, we can only wait for $n-f$ servers.
- What if two different servers contact $n-f$ quorums? If they intersect at f or fewer servers, that's not good.



Byzantine Quorums

Obviously, if all servers are Byzantine, we can't guarantee anything. How many servers do we need to tolerate f faults?

- In order to make progress, we can only wait for $n-f$ servers.
- What if two different servers contact $n-f$ quorums? If they intersect at f or fewer servers, that's not good.
- Therefore, we need at least $3f+1$ servers. Any two quorums of $2f+1 = n-f$ will intersect at at least one non-faulty server.

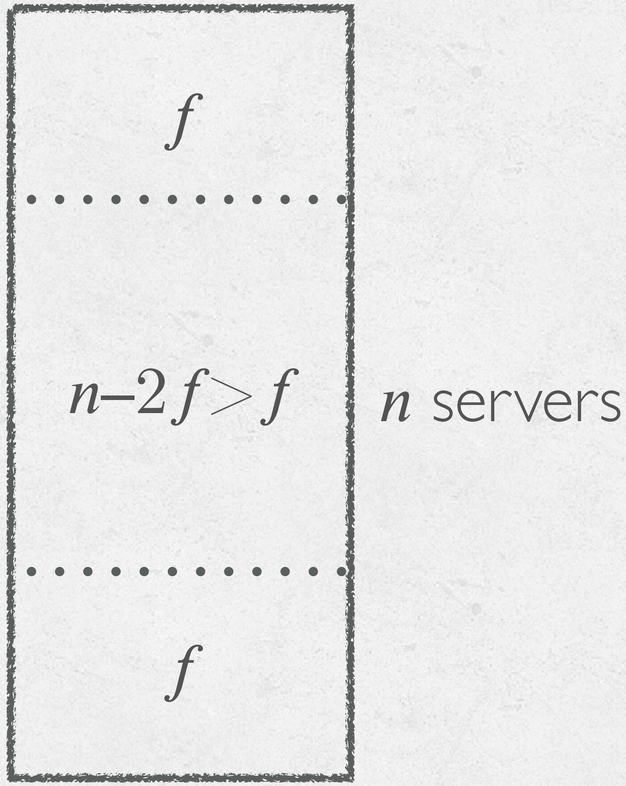


Byzantine Quorums

Obviously, if all servers are Byzantine, we can't guarantee anything. How many servers do we need to tolerate f faults?

- In order to make progress, we can wait for $n-f$ servers.
- What if two different servers, quorums? If they intersect at f or fewer servers, that's not good.
- Therefore, we need at least $3f+1$ servers. Any two quorums of $2f+1 = n-f$ will intersect at at least one non-faulty server.

Provable lower bound.



Setup

- $n=3f+1$ servers, f of which can be faulty. Unlimited clients.
- We assume public-key infrastructure. Servers and clients can sign messages and verify signatures. Signatures aren't forgeable.
  - We denote message m with $\langle m \rangle$, and message m signed by p as $\langle m \rangle_p$.
- Servers also have access to a digest function (cryptographic hash) on messages, $D(m)$, which we assume is collision-resistant.
- The attacker controls f faulty servers and knows the protocol the other servers are running. The attacker also has control over the network and can delay and reorder messages to all nodes.

Goal

The goal, as in Paxos, is state-machine replication.

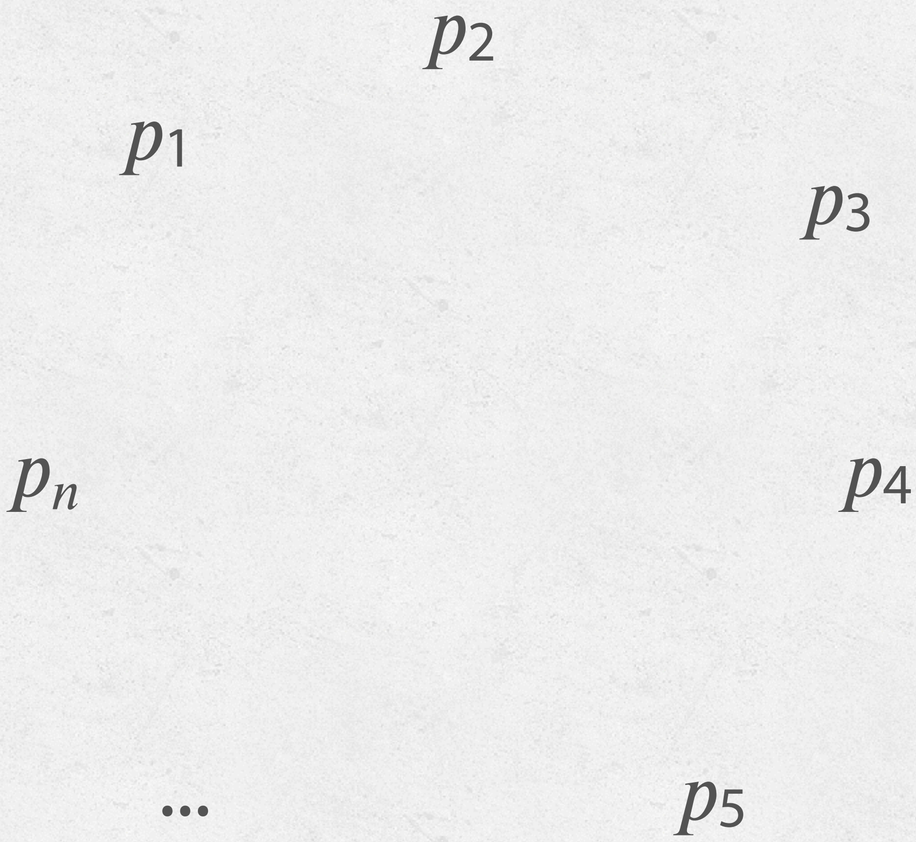
We want to guarantee safety when there are f or fewer failures (or an unlimited number of crash failures) and liveness during periods of synchrony.

Easy, right?

PBFT: The Basic Idea

Practical Byzantine Fault Tolerance (PBFT) is leader-based, just like Paxos. But it more closely resembles Viewstamped Replication [Oki and Liskov '88].

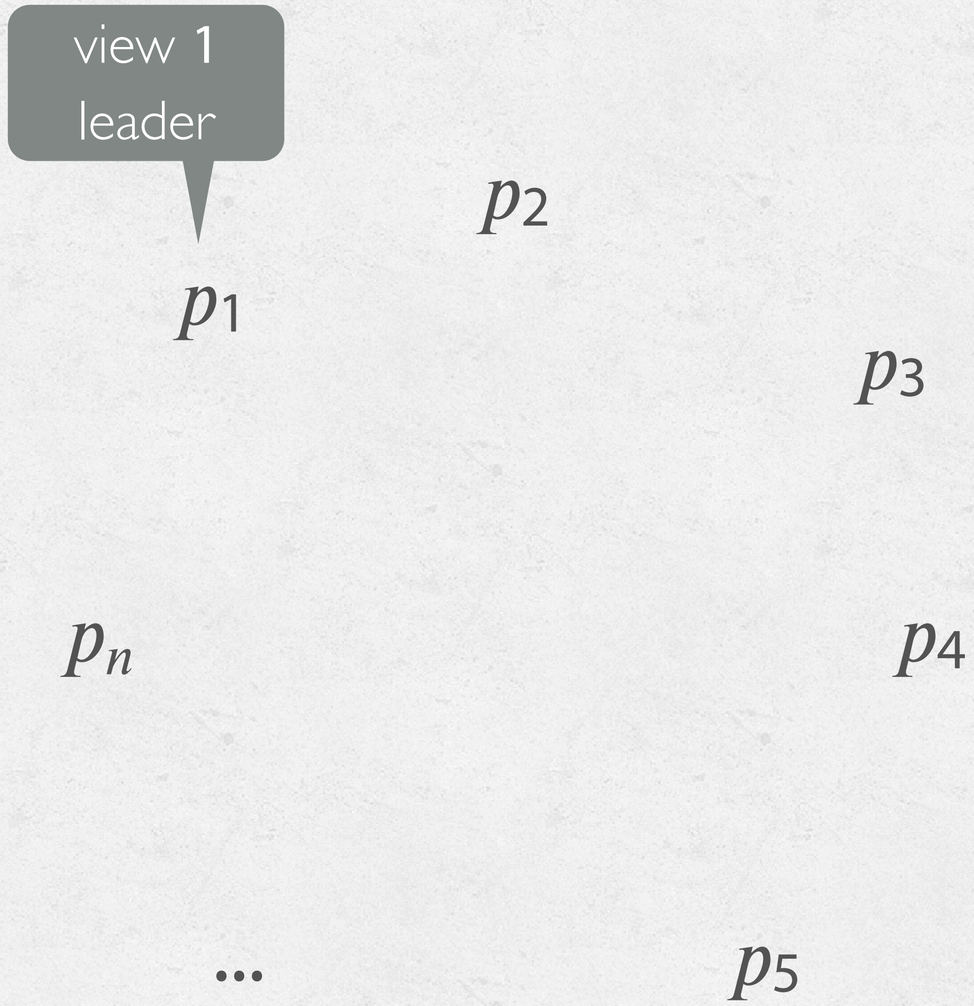
- The system progresses through a series of numbered **views**. There is a single leader associated with each view.
- The clients will send their commands to the leader.
- The leader assigns the command a sequence number (slot number) and forwards to the followers.
- The protocol ensures that this decision is permanently fixed; then they respond to the client.



PBFT: The Basic Idea

Practical Byzantine Fault Tolerance (PBFT) is leader-based, just like Paxos. But it more closely resembles Viewstamped Replication [Oki and Liskov '88].

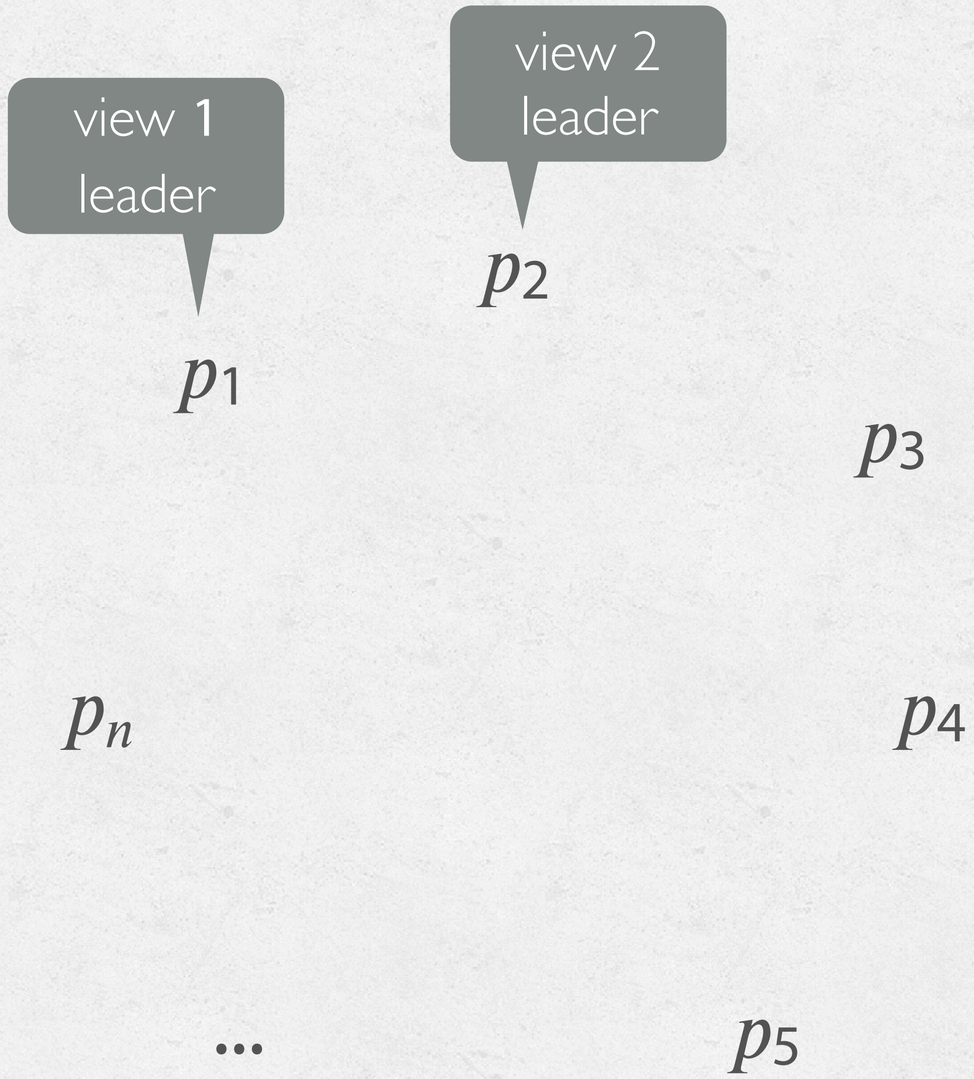
- The system progresses through a series of numbered **views**. There is a single leader associated with each view.
- The clients will send their commands to the leader.
- The leader assigns the command a sequence number (slot number) and forwards to the followers.
- The protocol ensures that this decision is permanently fixed; then they respond to the client.



PBFT: The Basic Idea

Practical Byzantine Fault Tolerance (PBFT) is leader-based, just like Paxos. But it more closely resembles Viewstamped Replication [Oki and Liskov '88].

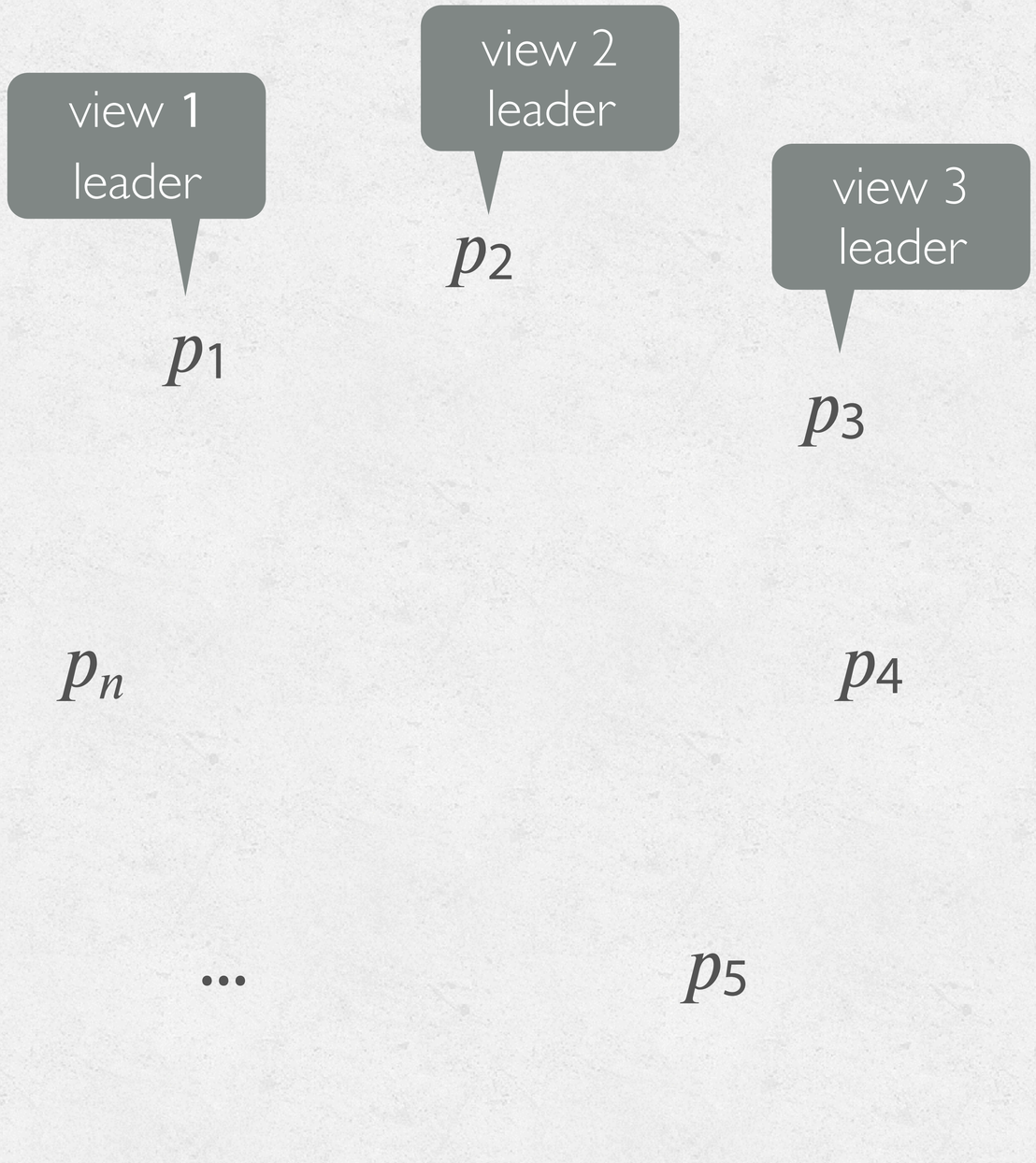
- The system progresses through a series of numbered **views**. There is a single leader associated with each view.
- The clients will send their commands to the leader.
- The leader assigns the command a sequence number (slot number) and forwards to the followers.
- The protocol ensures that this decision is permanently fixed; then they respond to the client.



PBFT: The Basic Idea

Practical Byzantine Fault Tolerance (PBFT) is leader-based, just like Paxos. But it more closely resembles Viewstamped Replication [Oki and Liskov '88].

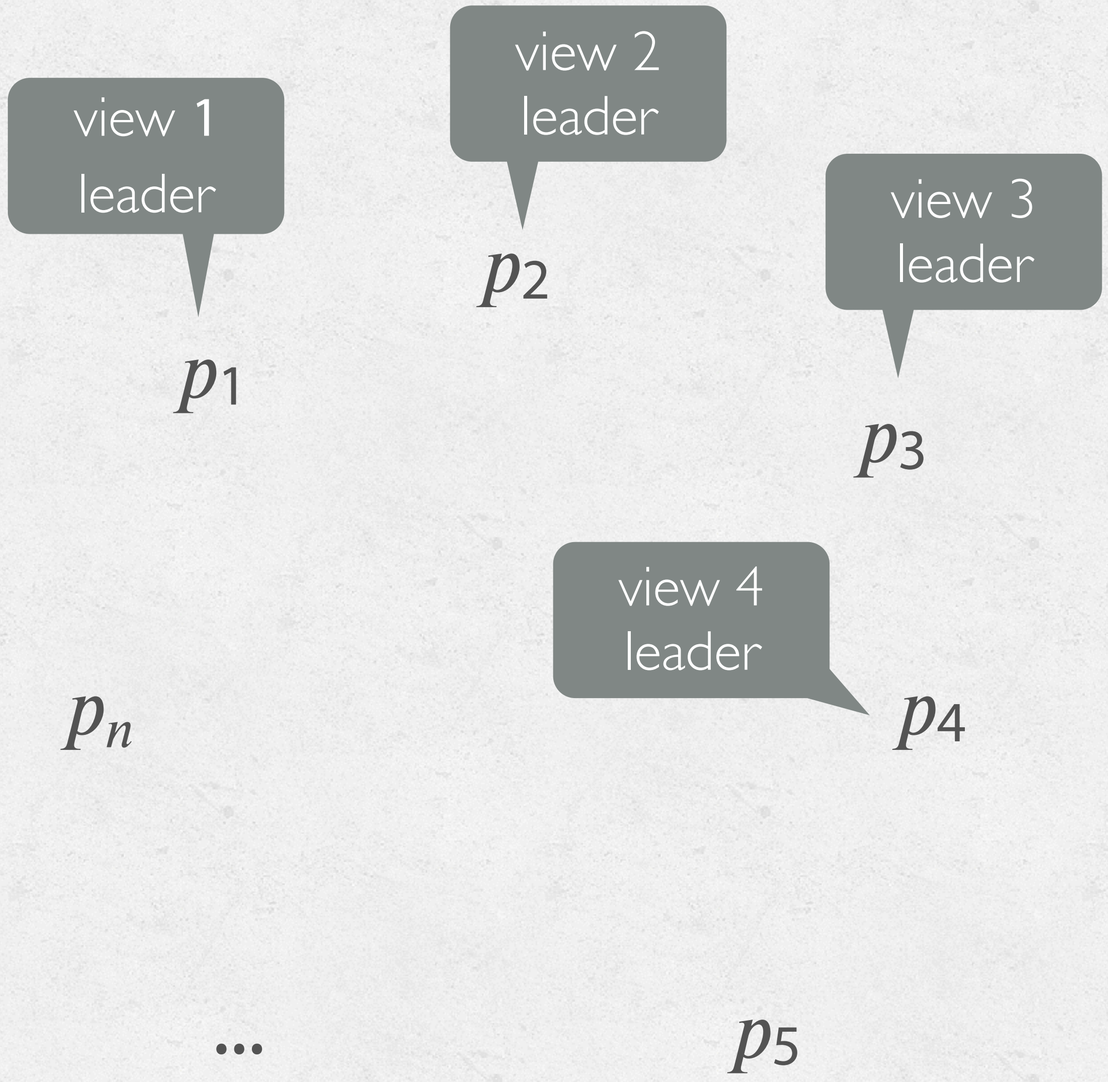
- The system progresses through a series of numbered **views**. There is a single leader associated with each view.
- The clients will send their commands to the leader.
- The leader assigns the command a sequence number (slot number) and forwards to the followers.
- The protocol ensures that this decision is permanently fixed; then they respond to the client.



PBFT: The Basic Idea

Practical Byzantine Fault Tolerance (PBFT) is leader-based, just like Paxos. But it more closely resembles Viewstamped Replication [Oki and Liskov '88].

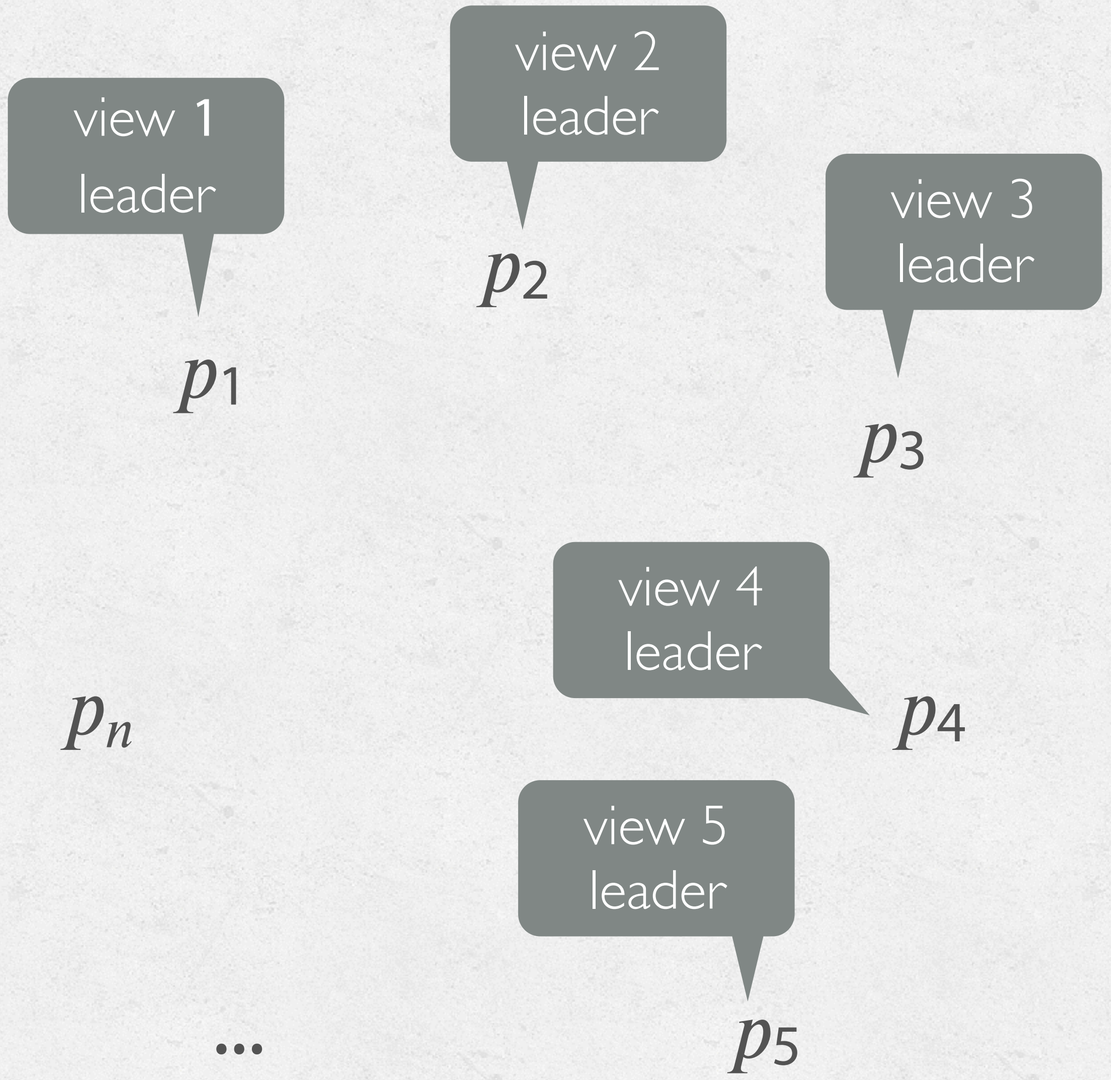
- The system progresses through a series of numbered **views**. There is a single leader associated with each view.
- The clients will send their commands to the leader.
- The leader assigns the command a sequence number (slot number) and forwards to the followers.
- The protocol ensures that this decision is permanently fixed; then they respond to the client.



PBFT: The Basic Idea

Practical Byzantine Fault Tolerance (PBFT) is leader-based, just like Paxos. But it more closely resembles Viewstamped Replication [Oki and Liskov '88].

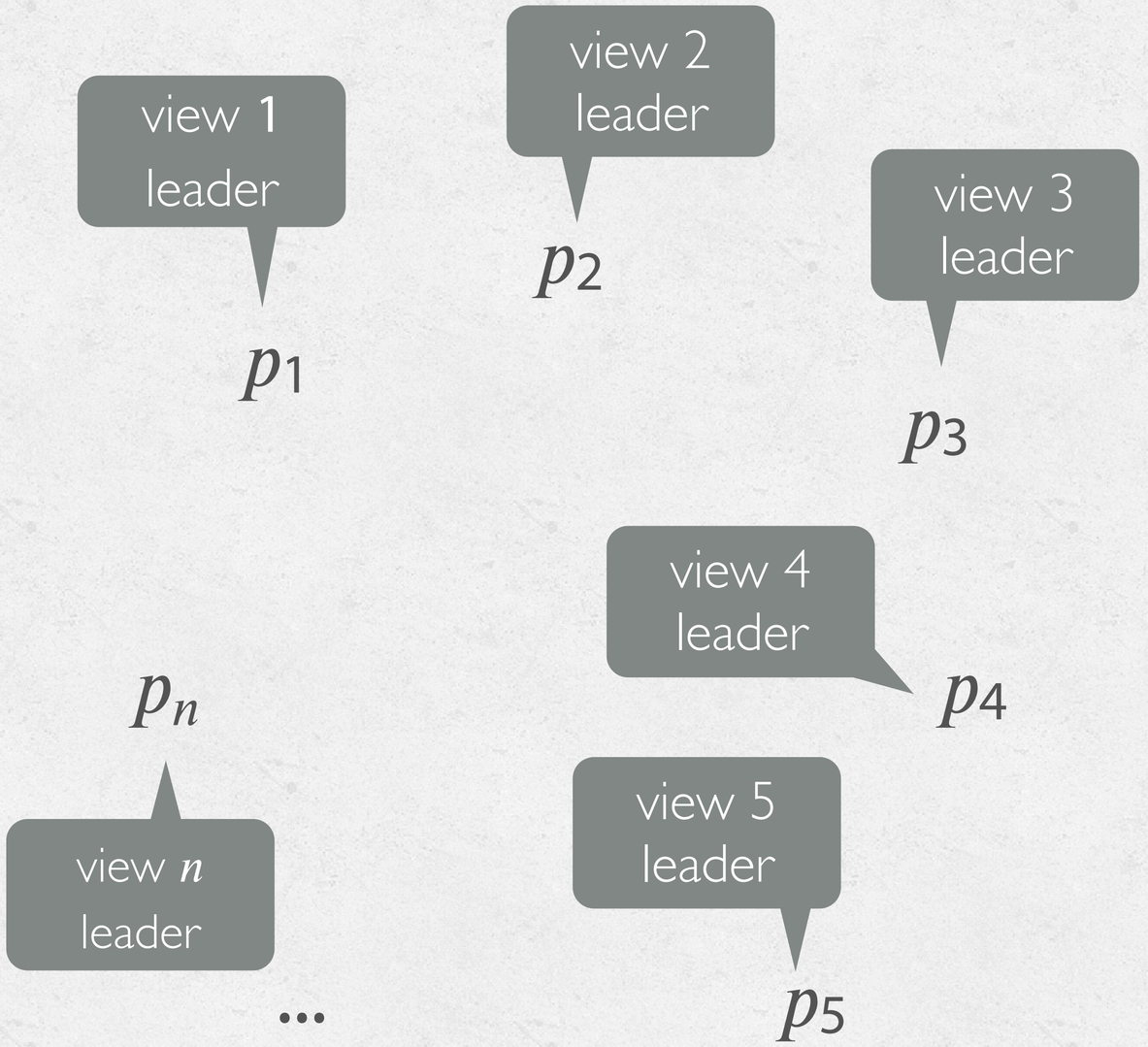
- The system progresses through a series of numbered **views**. There is a single leader associated with each view.
- The clients will send their commands to the leader.
- The leader assigns the command a sequence number (slot number) and forwards to the followers.
- The protocol ensures that this decision is permanently fixed; then they respond to the client.



PBFT: The Basic Idea

Practical Byzantine Fault Tolerance (PBFT) is leader-based, just like Paxos. But it more closely resembles Viewstamped Replication [Oki and Liskov '88].

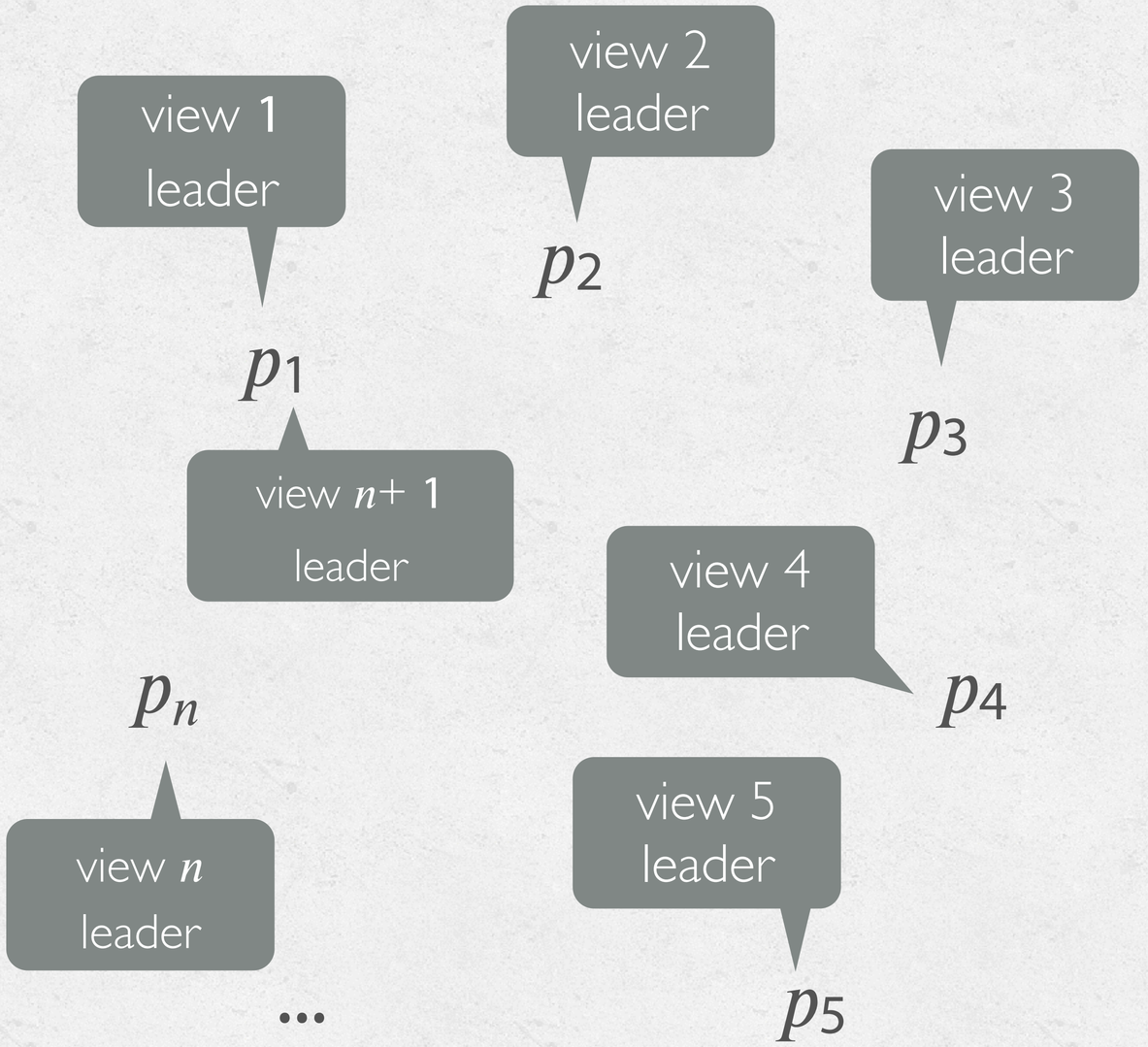
- The system progresses through a series of numbered **views**. There is a single leader associated with each view.
- The clients will send their commands to the leader.
- The leader assigns the command a sequence number (slot number) and forwards to the followers.
- The protocol ensures that this decision is permanently fixed; then they respond to the client.



PBFT: The Basic Idea

Practical Byzantine Fault Tolerance (PBFT) is leader-based, just like Paxos. But it more closely resembles Viewstamped Replication [Oki and Liskov '88].

- The system progresses through a series of numbered **views**. There is a single leader associated with each view.
- The clients will send their commands to the leader.
- The leader assigns the command a sequence number (slot number) and forwards to the followers.
- The protocol ensures that this decision is permanently fixed; then they respond to the client.



What's The Worst That Could Happen?

- The leader could be faulty.
  - It could assign different commands to the same sequence number.
  - It could try to send the wrong result to the client.
  - It could ignore the clients altogether.
- The followers could also be faulty and lie about the commands they received.

What's The Worst That Could Happen?

- The leader could be faulty.
  - It could assign c… the same sequence numb…
  - It could try to send the wrong result to the client.
  - It could ignore the clients altogether.
- The followers could also be faulty and lie about the commands they received.

Clients wait for $f+1$ matching replies.

What's The Worst That Could Happen?

- The leader could be faulty.
  - It could assign c… the same sequence numb…
  - It could try to send the wrong result to the client.
  - It could ignore the clients altogether.
- The followers co… about the commands they…

Clients wait for $f+1$ matching replies.

Followers can replace a misbehaving leader with a **view change**.

What About Faulty Clients?

- We assume that there is some existing way for clients to authenticate themselves with the system.
- Access controls can be used to restrict what each client is allowed to do.
- System administrators (or the system itself) can revoke access for faulty clients.

Papers, Please

- Servers don't take each others' word for anything. They require proof.
- In order to verify that a client's command is legitimate, they need the signed message from the client (or proof thereof).
- All other steps in the system are taken only after receiving signed messages from a **quorum of $2f+1$** servers. Servers can also collect these messages into **certificates** they can use to prove to each other the legitimacy of certain steps.

Papers, Please

- Servers don't take each others' word for anything. They require proof.
- In order to verify that a client's command is legitimate, they need the signed message from the client (or proof thereof).
- All other steps in the system are taken only after receiving signed messages from a **quorum of $2f+1$** servers. Servers can also collect these messages into **certificates** they can use to prove to each other the legitimacy of certain steps.

Papers, Please

- Servers don't take each others' word for anything. They require proof.
- In order to verify that a client's command is legitimate, they need the signed message from the client (or proof thereof).
- All other steps in the system are taken only after receiving signed messages from a **quorum of $2f+1$** servers. Servers can also collect these messages into **certificates** they can use to prove to each other the legitimacy of certain steps.

Certificate

Protocol Overview

Three sub-protocols:

1. **Normal operations**

   Phase 1: Pre-prepare

   Phase 2: Prepare

   Phase 3: Commit

2. **View change**

3. **Garbage collection**

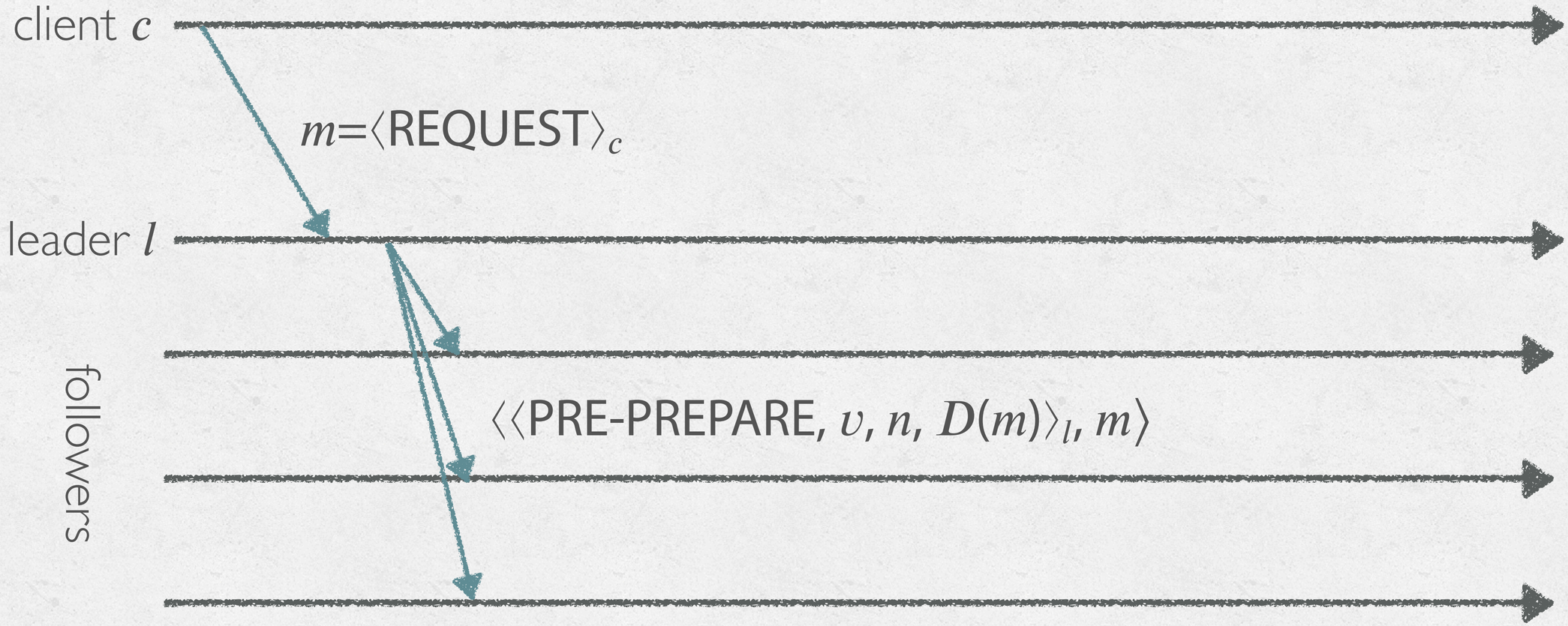
Server state:

- Current view
- State machine checkpoint
- Current state machine state
- Log of all not garbage collected messages

Normal Operations (I)

Normal Operations (I)

Normal Operations (I)



Accepting Pre-Prepares

The leader sends $\langle\langle$PRE-PREPARE, v, n, $D(m)\rangle_l$, $m\rangle$ to the followers.

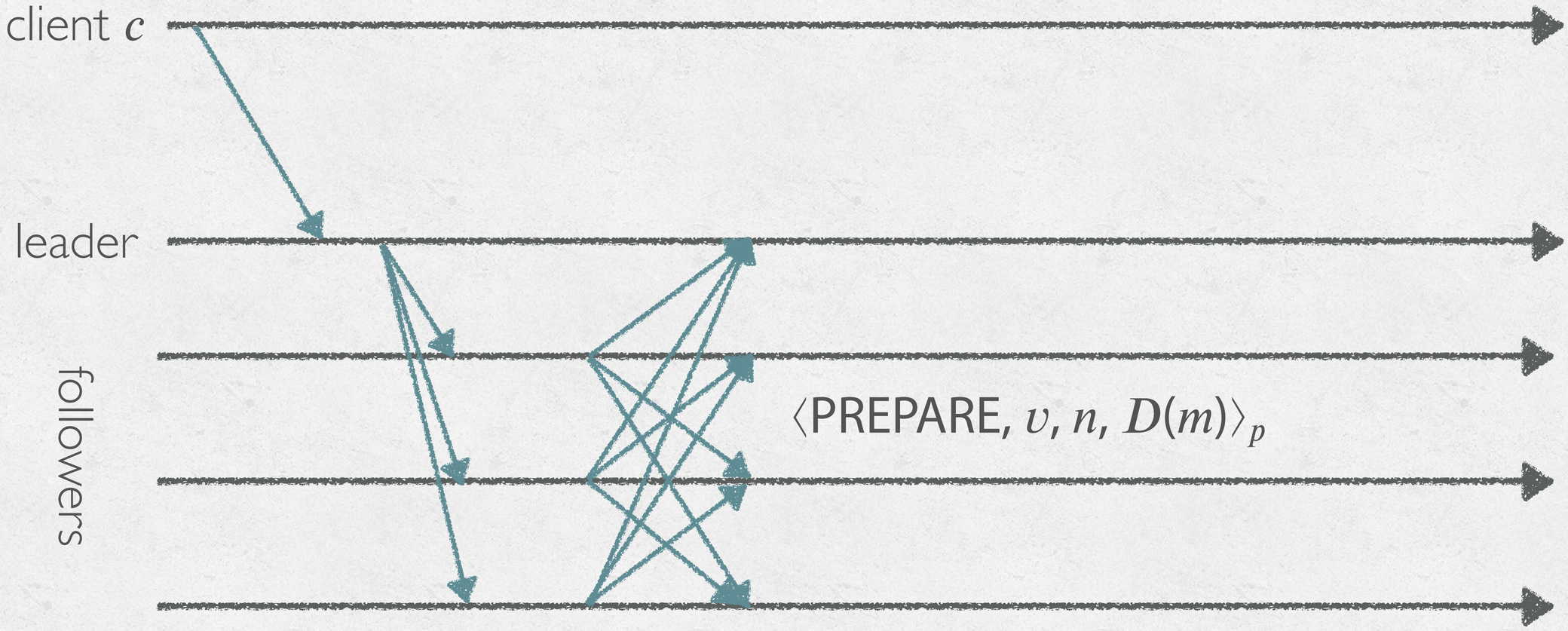
- v is the view number.
- n is the sequence number assigned by the leader.
- $D(m)$ is a digest of the message (to reduce amount of public key crypto).

A follower accepts the PRE-PREPARE if:

- The client request is valid.
- The follower is in view v.
- The follower hasn't accepted a different PRE-PREPARE for the same sequence number in the same view.
- The sequence number isn't too far ahead (to prevent sequence numbers from getting unnecessarily large).

Normal Operations (II)

Normal Operations (II)

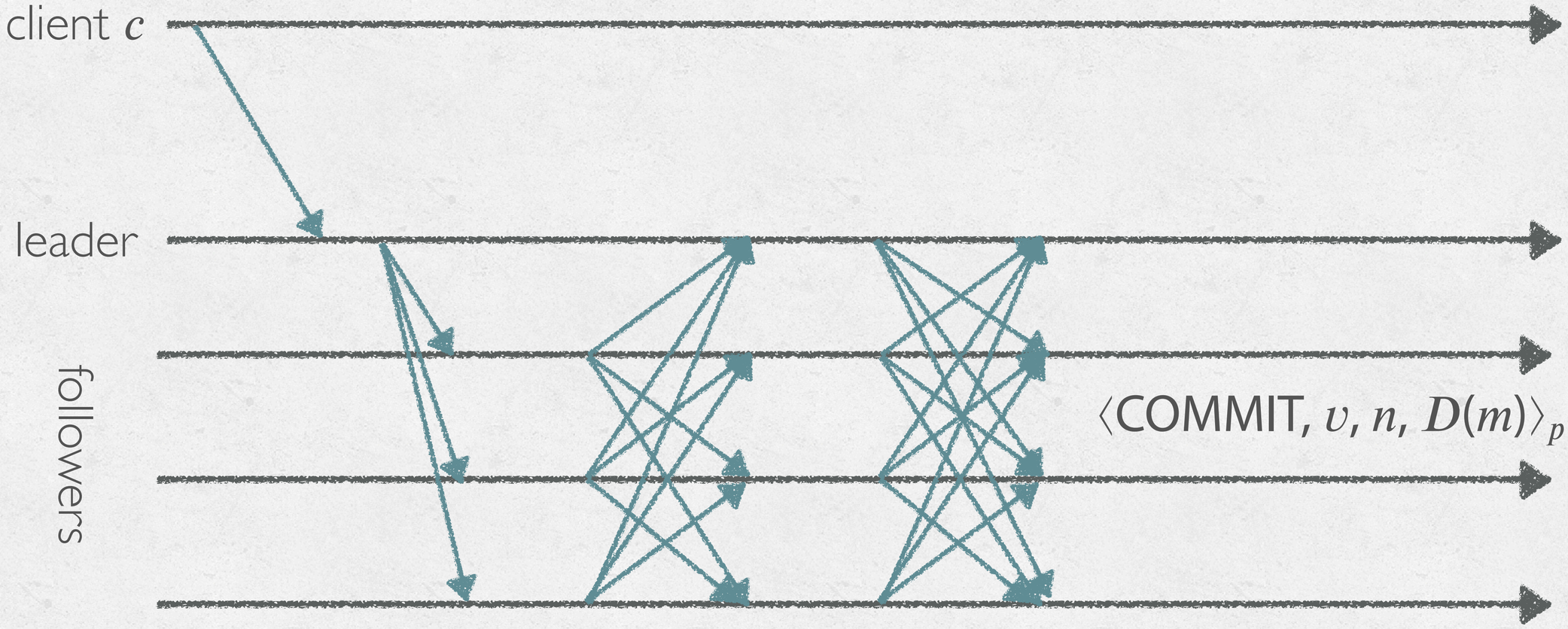


Prepare Certificates

- Once followers accept the PRE-PREPARE, they broadcast (signed) PREPARE messages.
- Once a server has received $2f$ matching PREPAREs and the associated PRE-PREPARE, it has a **Prepare Certificate**.
- Because quorums intersect at at least one honest server, and honest servers don't prepare different commands in the same slot, *no two prepare certificates ever exist for the same view and same sequence number and different commands*.
- However, a single server having a prepare certificate is not enough. What about view changes? The new leader might not get the Prepare Certificate, might not have enough information to pick the correct command in the new view.

Normal Operations (III)

Normal Operations (III)

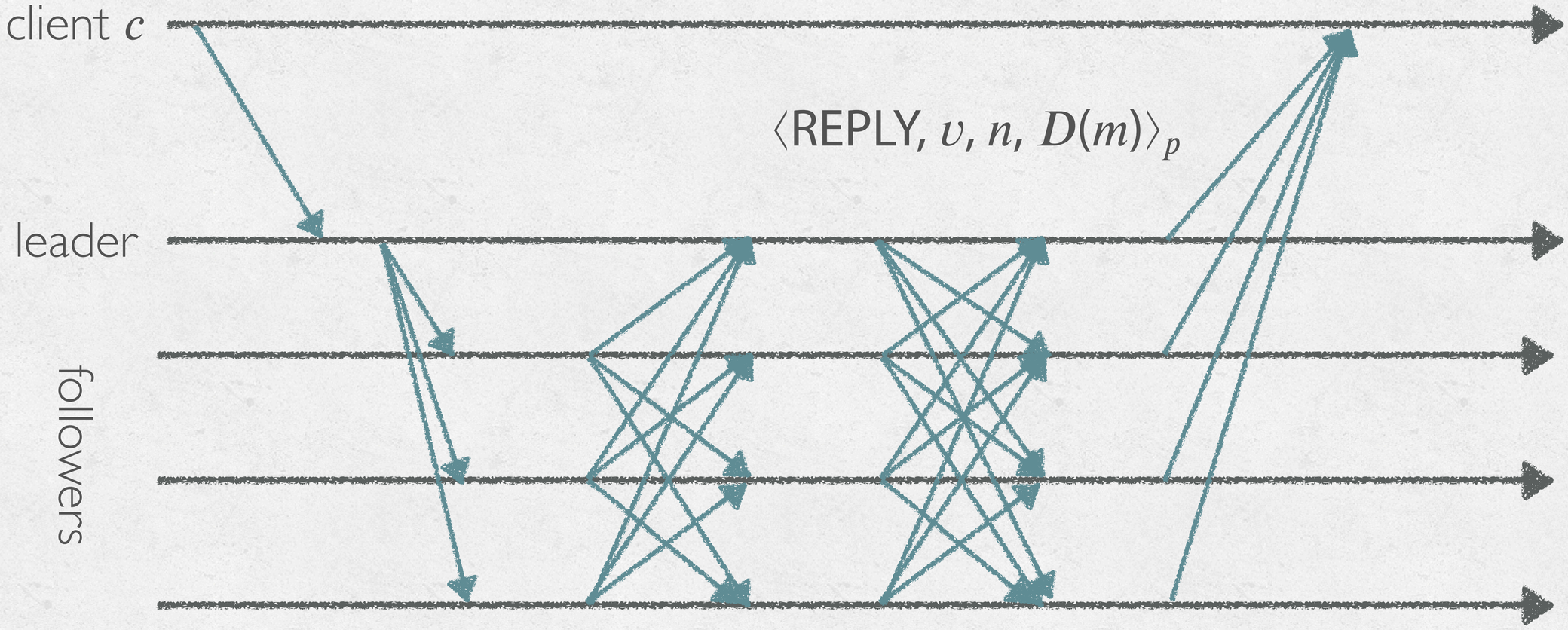


Commit Certificates

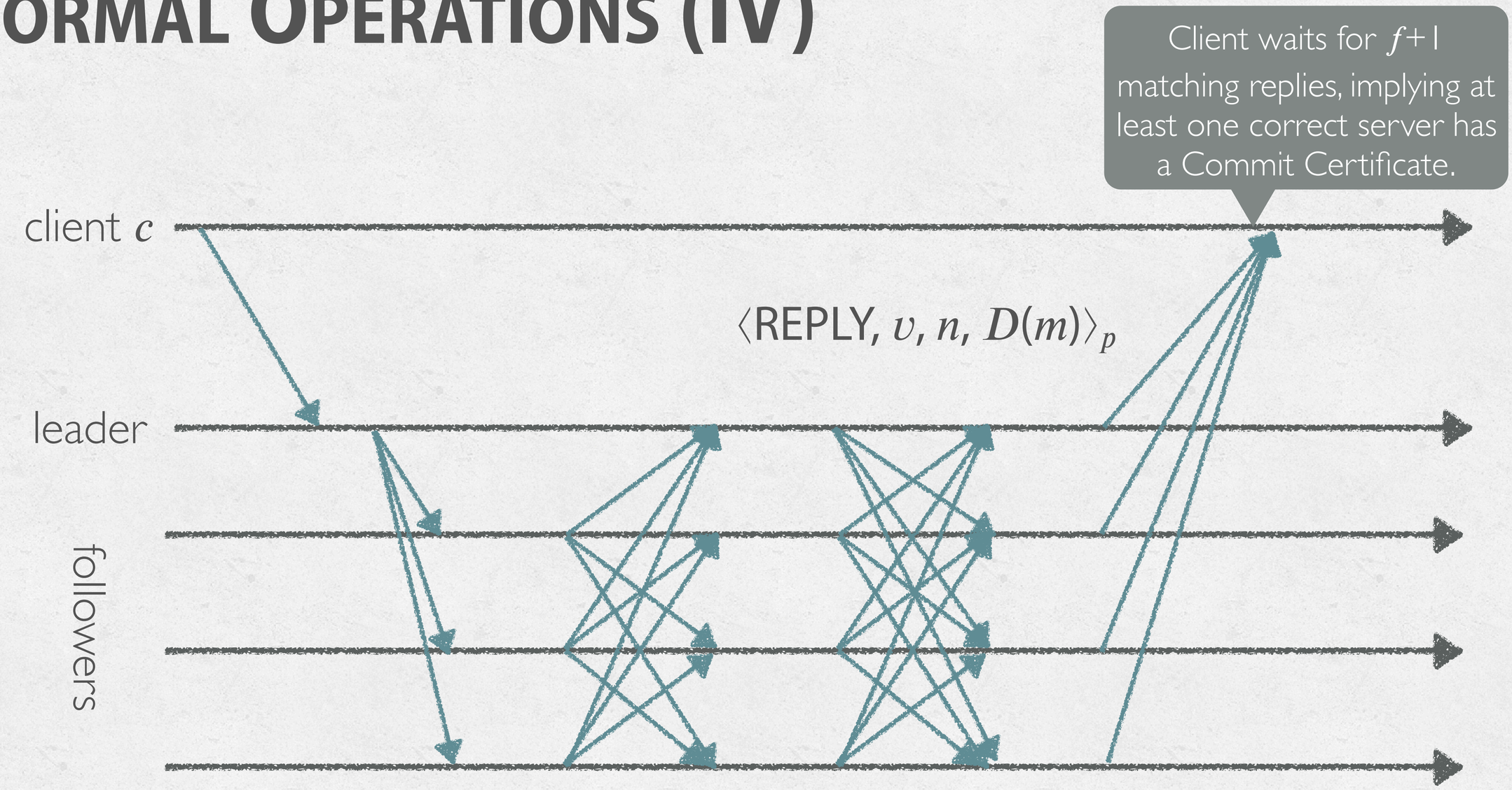
- Once a server has a Prepare Certificate, it broadcasts a COMMIT message.
- Once a server has $2f+1$ matching COMMITs (and the associated client message), it has a **Commit Certificate**.
- A commit certificate proves that every quorum of $2f+1$ servers has at least one non-faulty node with a Prepare Certificate. This command is now stable and will be fixed in the same slot future view changes.
- The server can then execute the command (provided it executed all previous commands) and reply to the client.

Normal Operations (IV)

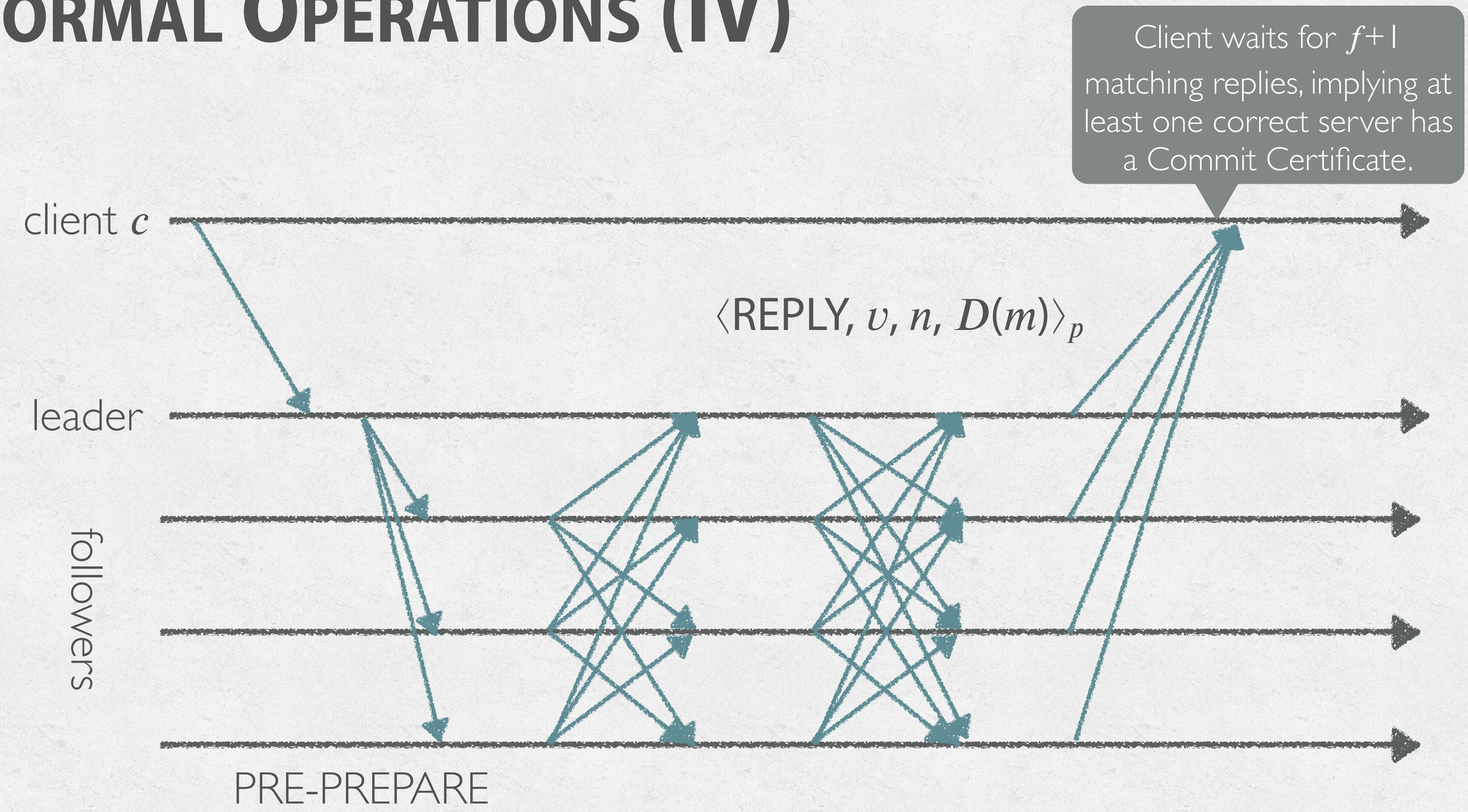
Normal Operations (IV)



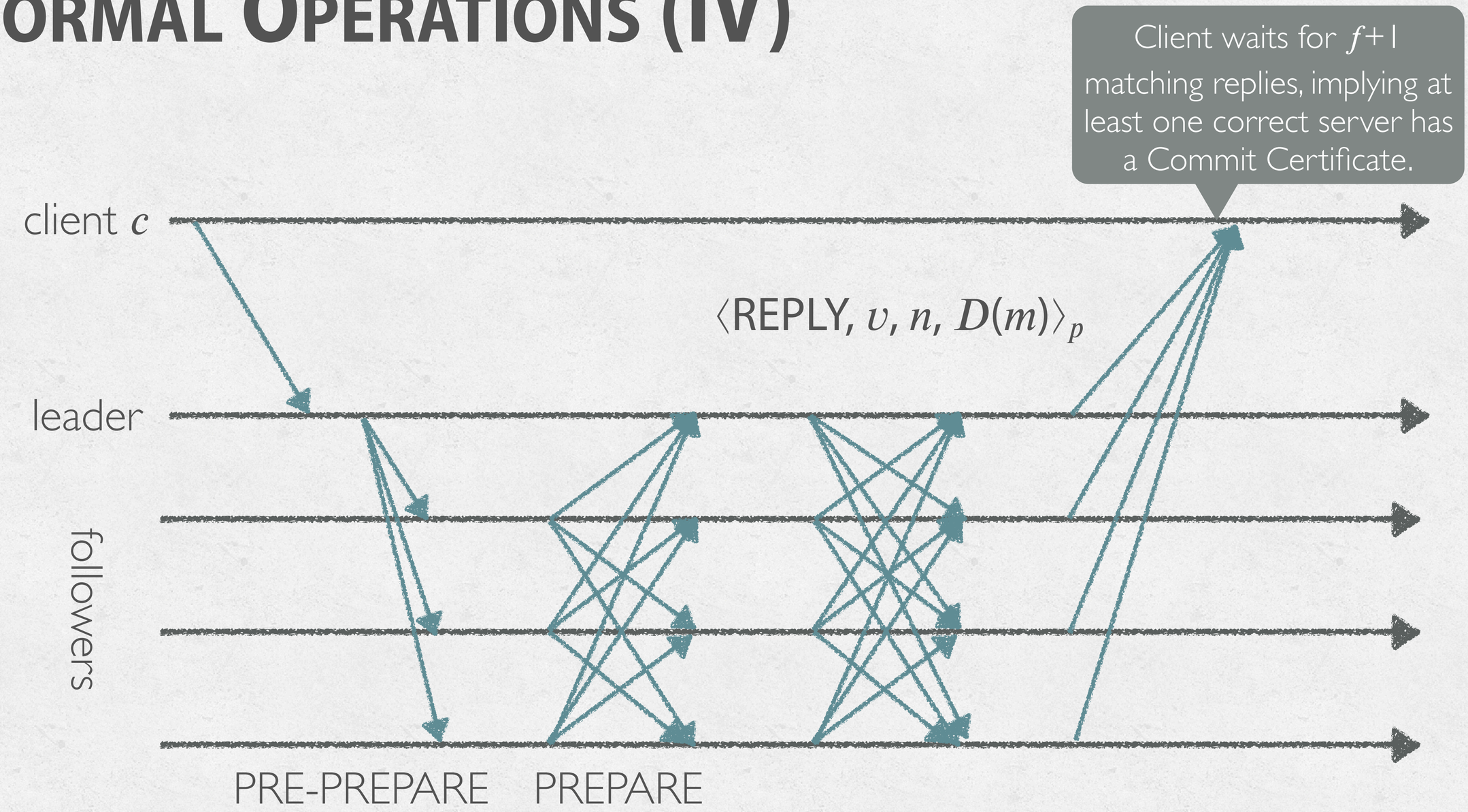
Normal Operations (IV)



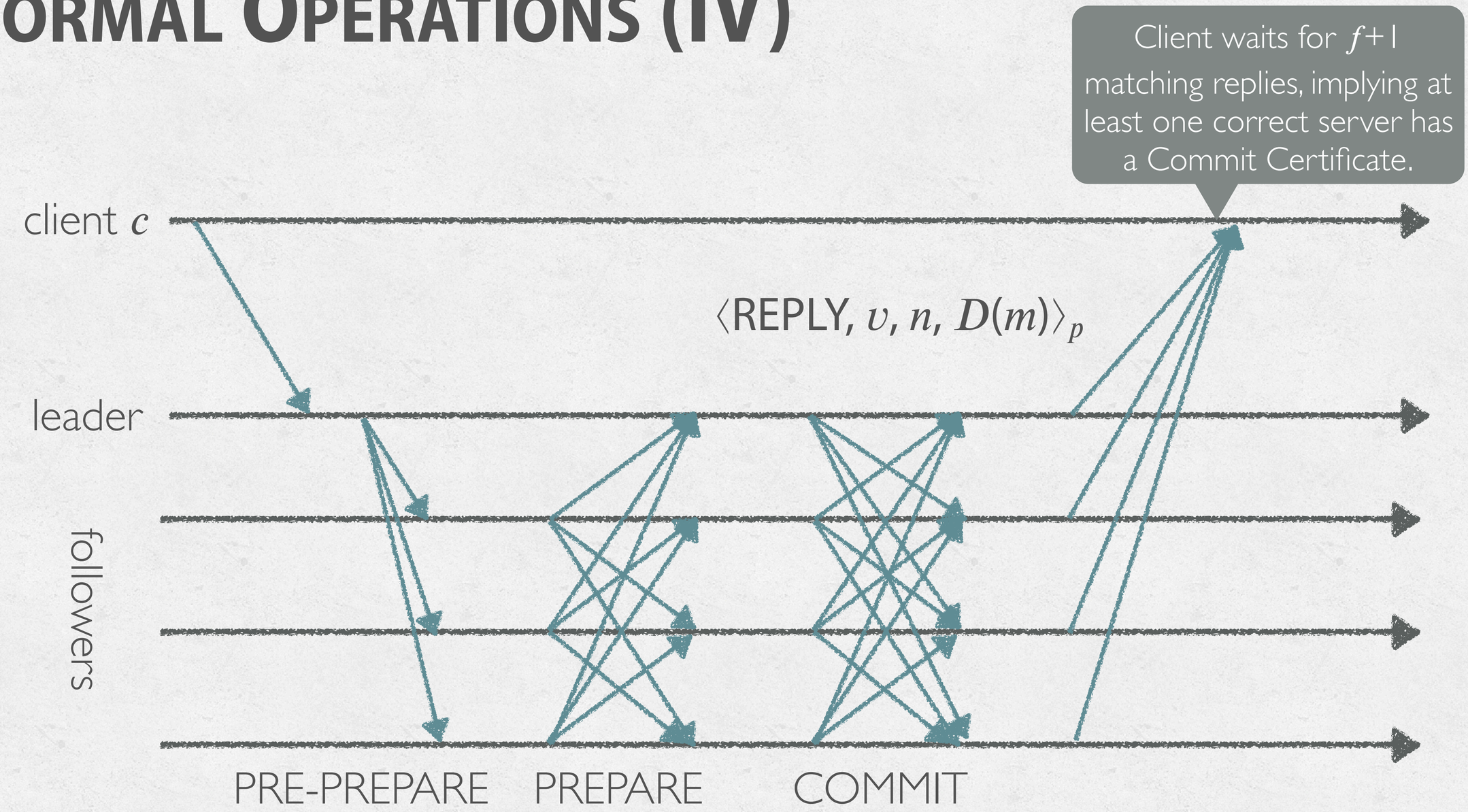
Normal Operations (IV)



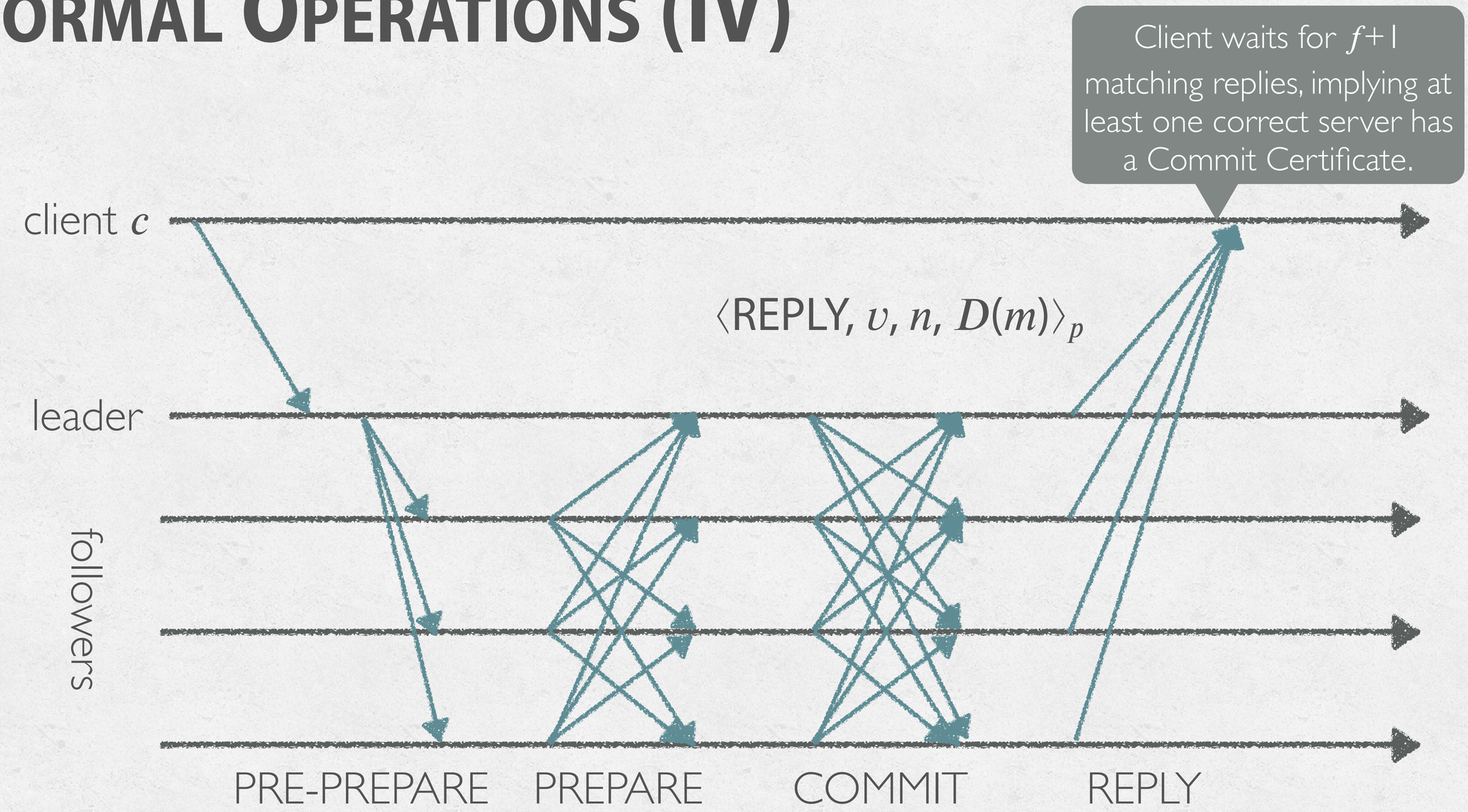
Normal Operations (IV)



Normal Operations (IV)



Normal Operations (IV)



View Change

View Change

- Followers monitor the leader. If the leader stops responding to pings or does anything shady, they start a view change.

View Change

- Followers monitor the leader. If the leader stops responding to pings or does anything shady, they start a view change.
- First, the follower sends $\langle$VIEW-CHANGE, v+1, $\mathscr{P}\rangle_p$ to the leader of view v+1 and $\langle$VIEW-CHANGE, v+1$\rangle_p$ to the other followers. The follower stops accepting messages for the old view.
  - $\mathscr{P}$ is the set of all Prepare Certificates (or Commit Certificates) the follower has received.

View Change

- Followers monitor the leader. If the leader stops responding to pings or does anything shady, they start a view change.
- First, the follower sends $\langle$VIEW-CHANGE, v+1, $\mathscr{P}\rangle_p$ to the leader of view v+1 and $\langle$VIEW-CHANGE, v+1$\rangle_p$ to the other followers. The follower stops accepting messages for the old view.
  - $\mathscr{P}$ is the set of all Prepare Certificates (or Commit Certificates) the follower has received.
- Other followers join in the view change when they receive f+1 VIEW-CHANGE messages.

Starting a New View

Starting a New View

Once the new leader receives $2f$ VIEW-CHANGE messages from the other servers, it broadcasts $\langle$NEW-VIEW, v+1, $\mathscr{V}$, $\mathscr{O}\rangle_p$

Starting a New View

Once the new leader receives $2f$ VIEW-CHANGE messages from the other servers, it broadcasts $\langle$NEW-VIEW, v+1, $\mathscr{V}$, $\mathscr{O}\rangle_p$

- $\mathscr{V}$ is the set of VIEW-CHANGE messages it received.

Starting a New View

Once the new leader receives $2f$ VIEW-CHANGE messages from the other servers, it broadcasts $\langle$NEW-VIEW, v+1, $\mathscr{V}$, $\mathscr{O}\rangle_p$

- $\mathscr{V}$ is the set of VIEW-CHANGE messages it received.
- $\mathscr{O}$ is a set of PRE-PREPARES in the new view, one for every sequence number less than or equal to the largest sequence number seen in a Prepare Certificate in a VIEW-CHANGE message. If there is a Prepare Certificate for that sequence number, the PRE-PREPARE is for that command. Otherwise, the leader pre-prepares a no-op.

Starting a New View

Once the new leader receives $2f$ VIEW-CHANGE messages from the other servers, it broadcasts $\langle$NEW-VIEW, v+1, $\mathscr{V}$, $\mathscr{O}\rangle_p$

- $\mathscr{V}$ is the set of VIEW-CHANGE messages it received.
- $\mathscr{O}$ is a set of PRE-PREPARES in the new view, one for every sequence number less than or equal to the largest sequence number seen in a Prepare Certificate in a VIEW-CHANGE message. If there is a Prepare Certificate for that sequence number, the PRE-PREPARE is for that command. Otherwise, the leader pre-prepares a no-op.

Followers can independently verify that the view was started correctly from the set $\mathscr{V}$.

If everything checks out, they start the new view and process the PRE-PREPARES in $\mathscr{O}$ as normal.


=committed
=prepared
⊥=no-op
Status in previous view
c_1 c_2 c_3 c_4 c_5 c_6
1 2 3 4 5 6 7 8 9 10
Possible new leader's log
c_1 ⊥ c_2 ⊥ ⊥ c_4 ⊥ c_5
1 2 3 4 5 6 7 8 9 10

Garbage Collection

- In the normal case, servers save their log of commands and all of the messages they receive.
- In the non-Byzantine case, servers can periodically compact their logs. They can bring out-of-date servers back up-to-date with a **state transfer**.
- In the Byzantine case, a server can't just accept a state transfer from another node. It needs proof.

Garbage collection (II)

- Servers periodically decide to take a **checkpoint**.
- Each server hashes the state of its state machine and broadcasts $\langle \text{CHECKPOINT}, n, D(S) \rangle_p$, where n is the sequence number of the last executed command and $D(S)$ is a hash of the state.
- Once a server has $f+1$ CHECKPOINT messages, it can compact its log and discard old protocol messages. These messages serve as a Checkpoint Certificate, proving the validity of the state.

But What Did That Buy Us?

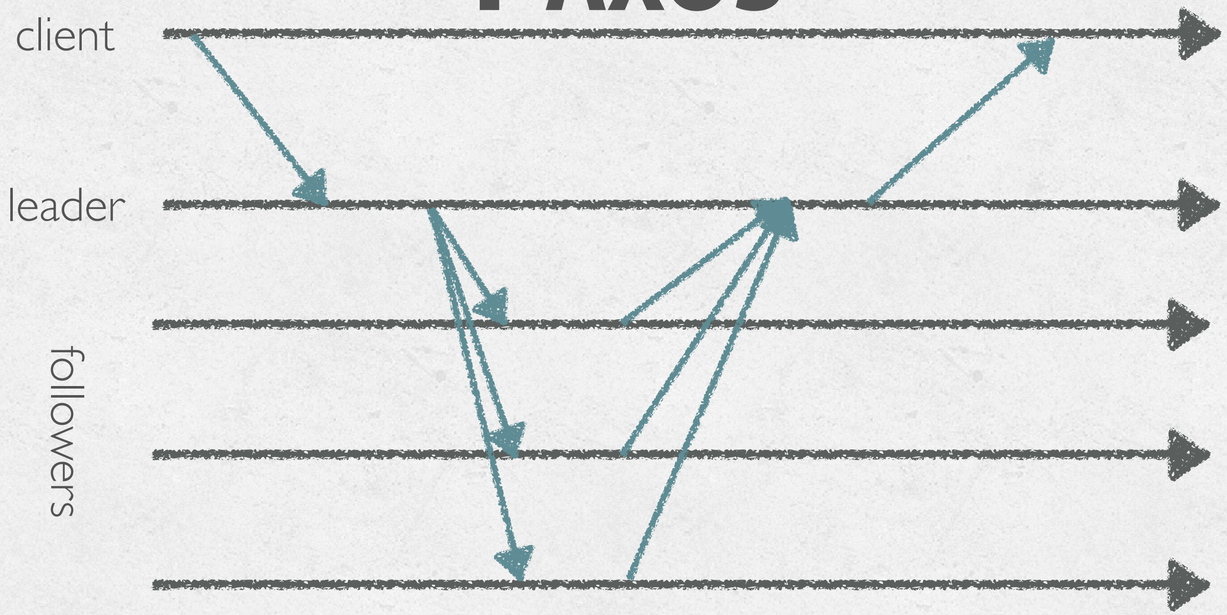
But What Did That Buy Us?

- Before, we could only tolerate crash failures.
- PBFT tolerates *any* failures, as long as only less than a third of the servers are faulty. (What happens if more are faulty?)
- However, as far as I know, PBFT and friends haven't seen wide adoption.

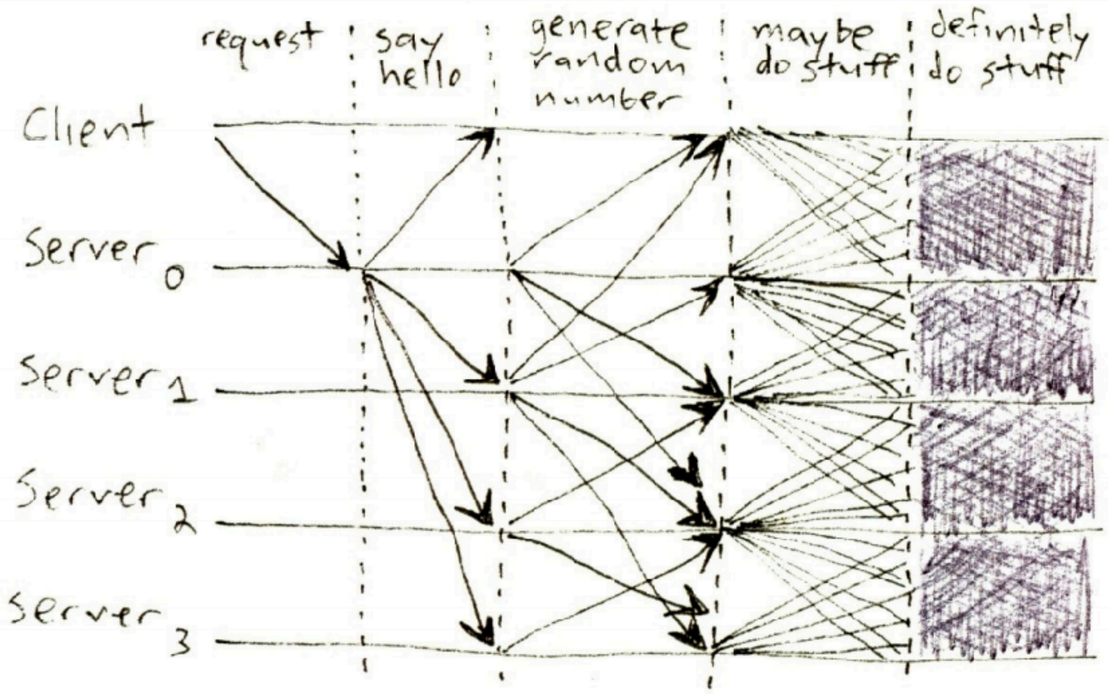
Performance

- Extra round of communication adds latency. (Can be avoided with speculative execution.)
- Committing a single operation requires $O(n^2)$ messages. (This can be improved, though at the cost of added latency.)
- Cryptography operations are slow! (Though the paper describes some strategies to speed them up using MACs.)

Paxos



PBFT



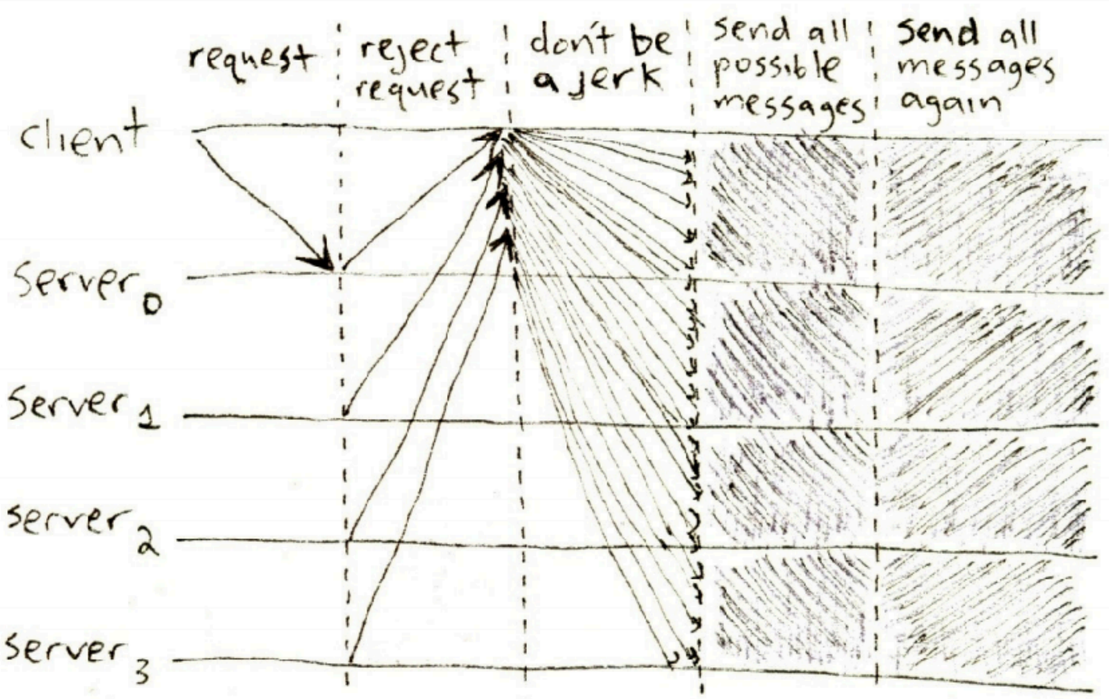


Figure 2: Our new protocol is clearly better.

[Mickens '13, *The Saddest Moment*]

How to Use BFT?

In order to use BFT, we need to have some reason to believe that the number of Byzantine failures is going to be limited, or at least that the failures will be independent and separated in time.

This probably holds true for hardware failures.

What about security flaws and software bugs?

One *possible* solution: n-version programming