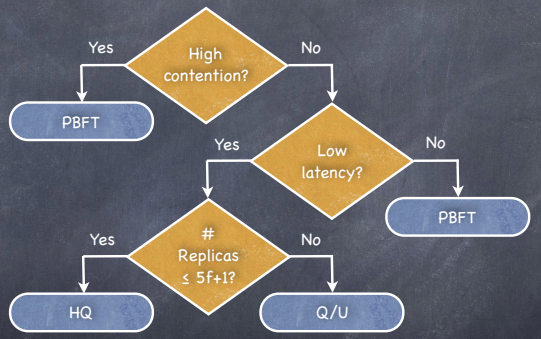
Zyzzyva

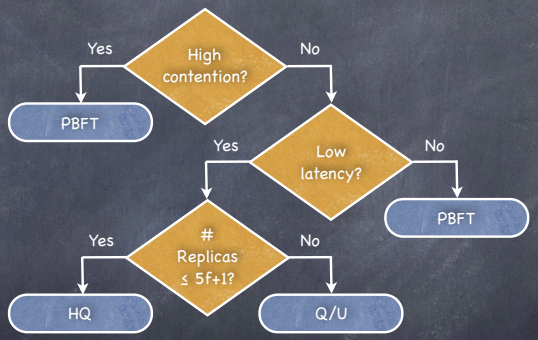
Why then another BFT protocol?



- Complex decision tree hampers BFT adoption

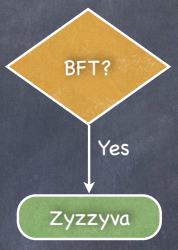
"Simplify, simplify"

H.D. Thoreau



"Simplify, simplify"

H.D. Thoreau

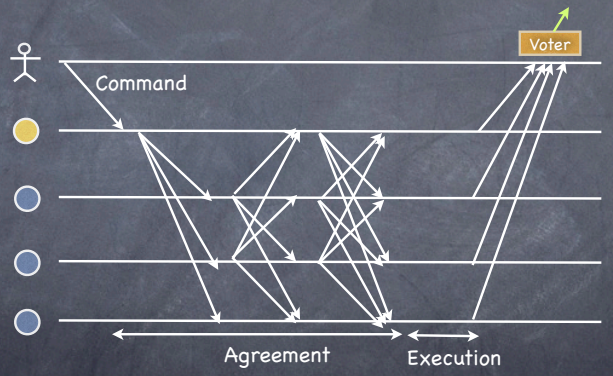


- One protocol that matches or tops its competitors in
  - ✓ latency
  - ✓ throughput
  - ✓ cost of replication

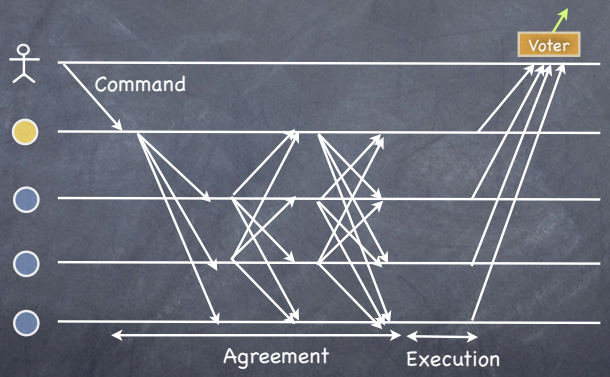
Replica coordination

- All correct replicas execute the same sequence of commands
- For each received command c, correct replicas:
  - Agree on c's position in the sequence
  - Execute c in the agreed upon order
  - Replies to the client

How it is done now



How Zyzzyva does it



Stability

- A command is stable at a replica once its position in the sequence cannot change

RSM Safety

Correct clients only process replies to stable commands

RSM Liveness

All commands issued by correct clients eventually become stable and elicit a reply

Enforcing safety

- RSM safety requires:
  - Correct clients only process replies to stable commands
- ...but RSM implementations enforce instead:
  - Correct replicas only execute and reply to commands that are stable
- Service performs an output commit with each reply

Speculative BFT: "Trust, but Verify"

- Insight: output commit at the client, not at the service!
- Replicas execute and reply to a command without knowing whether it is stable
  - trust order provided by primary
  - no explicit replica agreement!
- Correct client, before processing reply, verifies that it corresponds to stable command
  - if not, client takes action to ensure liveness

Verifying stability

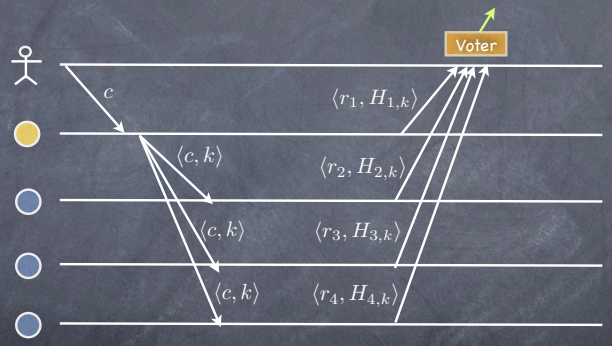
- Necessary condition for stability in Zyzzyva:

  A command c can become stable only if a majority of correct replicas agree on its position in the sequence
- Client can process a response for c iff:
  - a majority of correct replicas agrees on c's position
  - the set of replies is incompatible, for all possible future executions, with a majority of correct replicas agreeing on a different command holding c's current position

Command History

- $H_{i,k}$ = a hash of the sequence of the first k commands executed by replica i
- On receipt of a command c from the primary, replica appends c to its command history
- Replica reply for c includes:
  - the application-level response
  - the corresponding command history

Case 1: Unanimity



- Client processes response if all replies match:

$$r_1 = \ldots = r_4 \wedge H_{1,k} = \ldots = H_{4,k}$$

Safe?

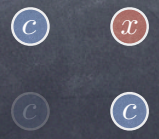
- ✓ A majority of correct replicas agrees on c's position (all do!)
- If primary fails
  - New primary determines k-th command by asking $n-f$ replicas for their H

Safe?

- ✓ A majority of correct replicas agrees on c's position (all do!)
- If primary fails
  - New primary determines k-th command by asking $n-f$ replicas for their H

Safe?

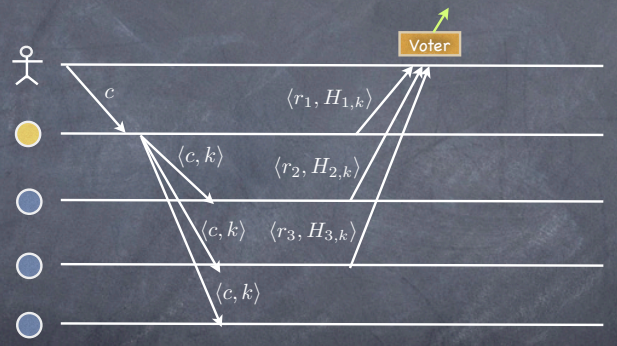
- ✓ A majority of correct replicas agrees on c's position (all do!)
- If primary fails
  - New primary determines k-th command by asking $n-f$ replicas for their H



Safe?

- ✓ A majority of correct replicas agrees on c's position (all do!)
- If primary fails
  - New primary determines c's position by asking $n-f$ replicas for their H
- ✓ It is impossible for a majority of correct replicas to agree on a different command for c's position

Case 2: A majority of correct replicas agree



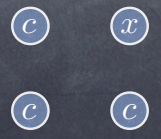
- At least $2f+1$ replies match

Safe?

- ✓ A majority of correct replicas agrees on c's position
- If primary fails
  - New primary determines k-th command by asking $n-f$ replicas for their H

Safe?

- ✓ A majority of correct replicas agrees on c's position
- If primary fails
  - New primary determines k-th command by asking $n-f$ replicas for their H



Safe?

- ✓ A majority of correct replicas agrees on c's position
- If primary fails
  - New primary determines k-th command by asking $n-f$ replicas for their H

Safe?

- ✓ A majority of correct replicas agrees on c's position
- If primary fails
  - New primary determines k-th command by asking $n-f$ replicas for their H

Safe?

- ✓ A majority of correct replicas agrees on c's position
- If primary fails
  - New primary determines k-th command by asking $n-f$ replicas for their H

Safe?

- ✓ A majority of correct replicas agrees on c's position
- If primary fails
  - New primary determines k-th command by asking $n-f$ replicas for their H

Safe?

- ✓ A majority of correct replicas agrees on c's position
- If primary fails
  - New primary determines k-th command by asking $n-f$ replicas for their H

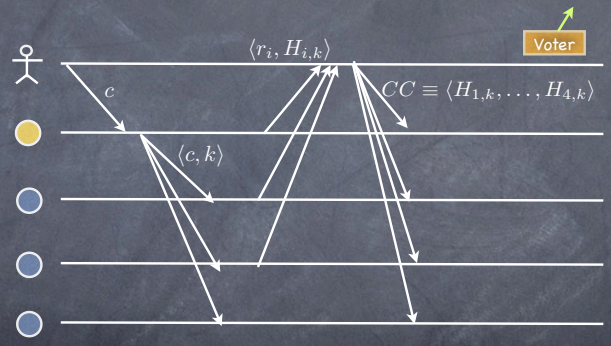
Safe?

- ✓ A majority of correct replicas agrees on c's position
- If primary fails
  - New primary determines k-th command by asking $n-f$ replicas for their H

Safe?

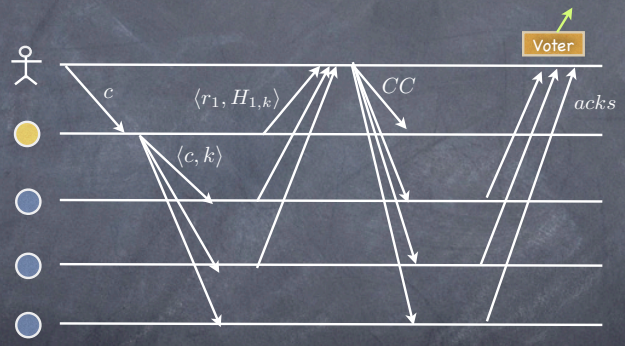
- ✓ A majority of correct replicas agrees on c's position
- If primary fails
  - New primary determines k-th command by asking $n-f$ replicas for their H
- Not safe!

Case 2: A majority of correct replicas agree



- Client sends to all a commit certificate containing $2f+1$ matching histories

Case 2: A majority of correct replicas agree



- Client processes response if it receives at least $2f+1$ acks

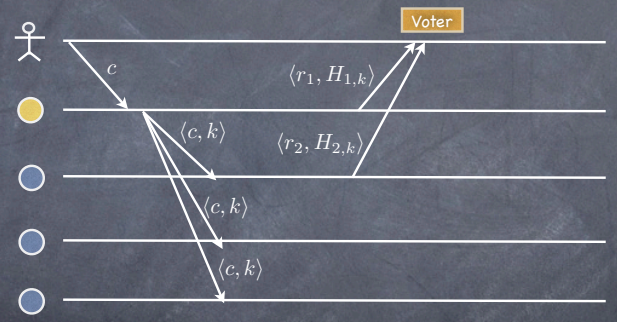
Safe?

- Certificate proves that a majority of correct replicas agreed on c's position
- If primary fails
  - New primary determines k-th command by contacting $n-f$ replicas
  - This set contains at least one correct replica with a copy of the certificate
- ✓ Incompatible with a majority backing a different command for that position

Stability and command histories

- Stability depends on matching command histories
- Stability is prefix-closed:
  - If a command with sequence number n is stable, then so is every command with sequence number $n' < n$

Case 3: None of the above

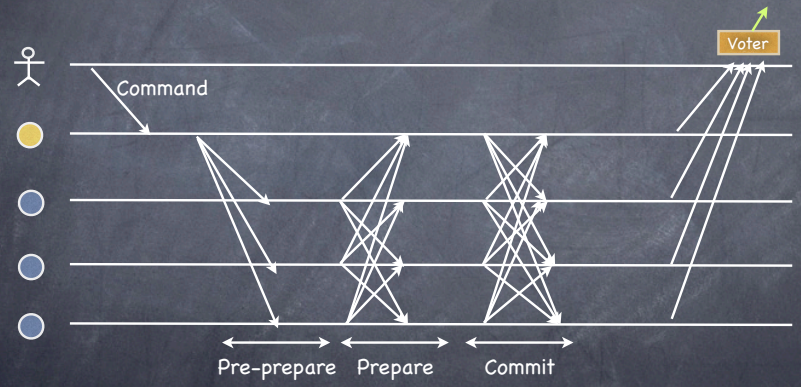


- Fewer than $2f+1$ replies match
- Clients retransmits c to all replicas-hinting primary may be faulty

Zyzzyva recap

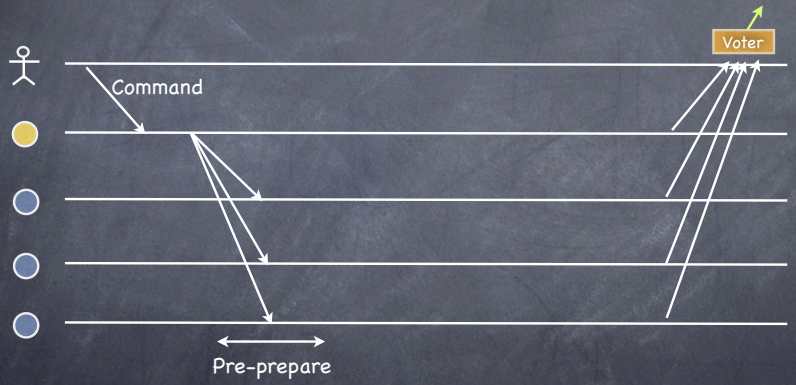
- Output commit at the client, not the service
- Replicas execute requests without explicit agreement
- Client verifies if response corresponds to stable command
- At most 2 phases within a view to make command stable

The Case of the Missing Phase



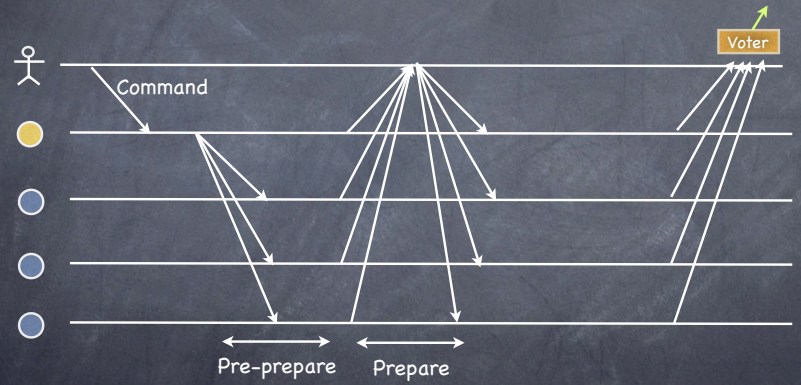
- Client processes response if it receives at least $f+1$ matching replies after commit phase

The Case of the Missing Phase



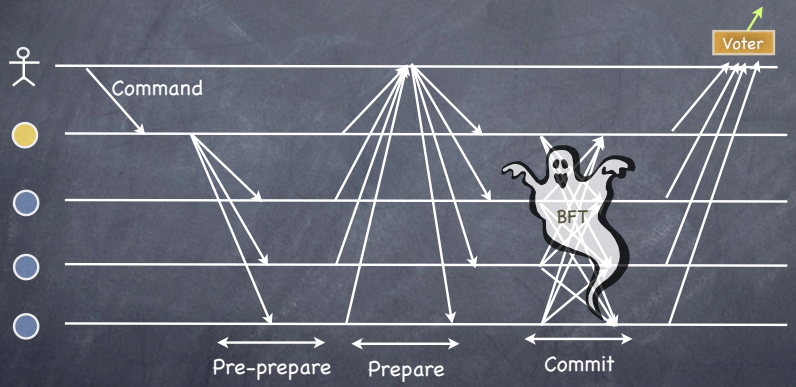
Unanimity

The Case of the Missing Phase



Majority

The Case of the Missing Phase



- Where did the third phase go?
- Why was it there to begin with?

View-Change: replacing the primary

- In PBFT, a replica that suspects primary is faulty goes unilaterally on strike
  - Stops processing messages in the view
  - Third "Commit" phase needed for liveness

View-Change: replacing the primary

- In PBFT, a replica that suspects primary is faulty goes unilaterally on strike
  - Stops processing messages in the view
  - Third "Commit" phase needed for liveness

- In Zyzzyva, the replica goes on "Technion strike"
  - Broadcasts "I hate the primary" and keeps on working
  - Stops when sees enough hate mail to ensure all correct replica will stop as well
- Extra phase is moved to the uncommon case

Faulty clients can't affect safety

- Faulty clients cannot create inconsistent commit certificates
  - Clients cannot fabricate command histories, as they are signed by replicas
  - It is impossible to generate a valid commit certificate that conflicts with the order of any stable request
    - Stability is prefix closed!

"Olly Olly Oxen Free!"

or, faulty clients can't affect liveness

"Olly Olly Oxen Free!"

or, faulty clients can't affect liveness

- Faulty client omits to send CC for c
- Replicas commit histories are unaffected!
- Later correct client who establishes $c' > c$ is stable "frees" c as well
  - Stability is prefix closed!

Throughput

Throughput

	Best case
PBFT	62K
QU	24K
HQ	15K
Zyzzyva	80K