HotStuff: BFT Consensus in the Lens of Blockchain

Paper by: Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan Gueta, and Ittai Abraham

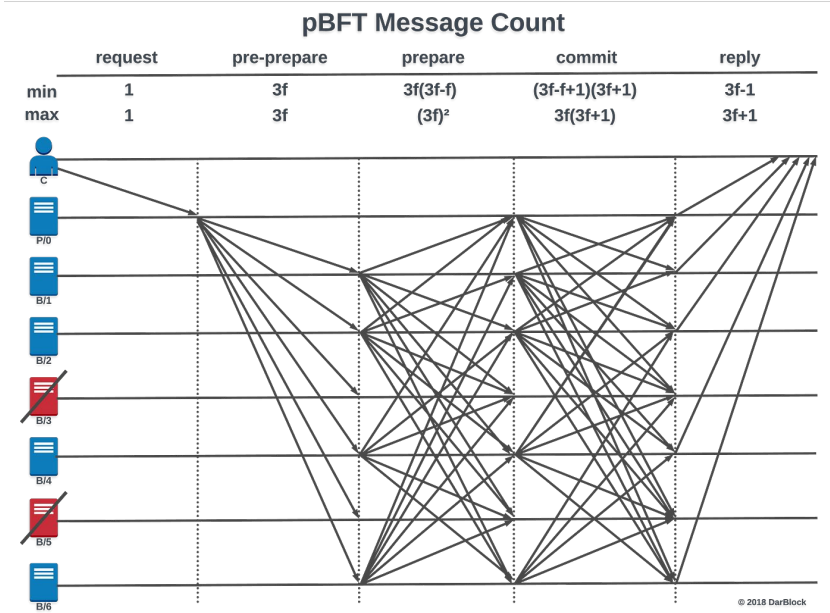
Presentation by: Saipranav Kotamreddy, Madhumitha Venkatesan, Nikita B. Emberi, Srikkanth R

Diagrams pulled from:
https://expolab.org/ecs265-fall-2021/slides/4_HotStuff.pdf by Xianda Hou, Oliver Shen, Ashwin Sekhari, Sheshavishnuprasad D, Mythreya K
https://expolab.org/ecs265-fall-2022/slides/HotStuff-presentation.pdf by Tong Zhu, Hongxiang Zhang, Siyuan Liu, Yifeng Shi, Junchao Chen

PBFT Overview

HotStuff is based off of a Byzantine Fault Tolerance protocol which allows the distributed system to achieve consensus via a Leader Node and can handle malicious nodes, for both the primary and replicas.

Protocol supports n>=3f+1, where n is total nodes and f is the number of faulty nodes for the system to still work



Problem with PBFT

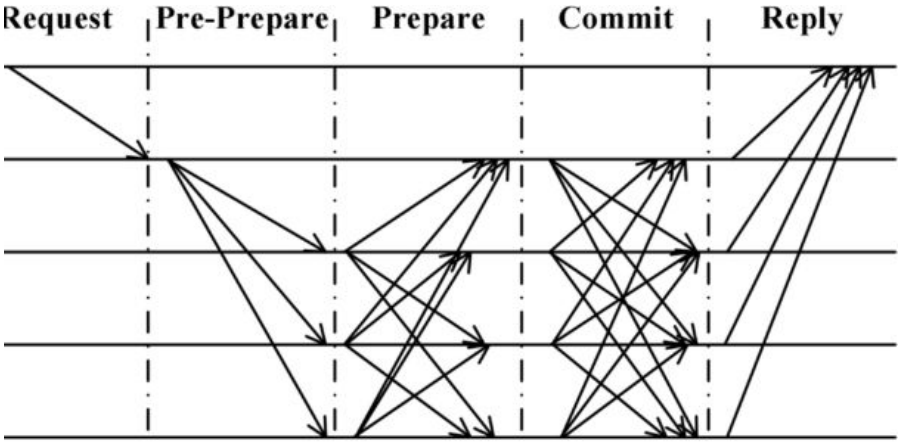
- BFT was designed around node counts between 4 and 7, however modern blockchain systems require this node count to scale to the thousands, which PBFT is unequipped to handle
- This is in large part due to communication costs in BFT, as BFT has O(n^2) communication complexity and if a view change is necessary, this raises to O(n^3) communication complexity

Protocol	Authenticator complexity			Responsiveness
	Correct leader	*Leader failure (view-change)*	*f leader failures*	
DLS [25]	$O(n^4)$	$O(n^4)$	$O(n^4)$	
PBFT [20]	$O(n^2)$	$O(n^3)$	$O(fn^3)$	✓
SBFT [30]	$O(n)$	$O(n^2)$	$O(fn^2)$	✓
Tendermint [15] / Casper [17]	$O(n^2)$	$O(n^2)$	$O(fn^2)$	
Tendermint* / Casper*	$O(n)$	$O(n)$	$O(fn)$	
HotStuff	$\boldsymbol{O(n)}$	$\boldsymbol{O(n)}$	$\boldsymbol{O(fn)}$	✓

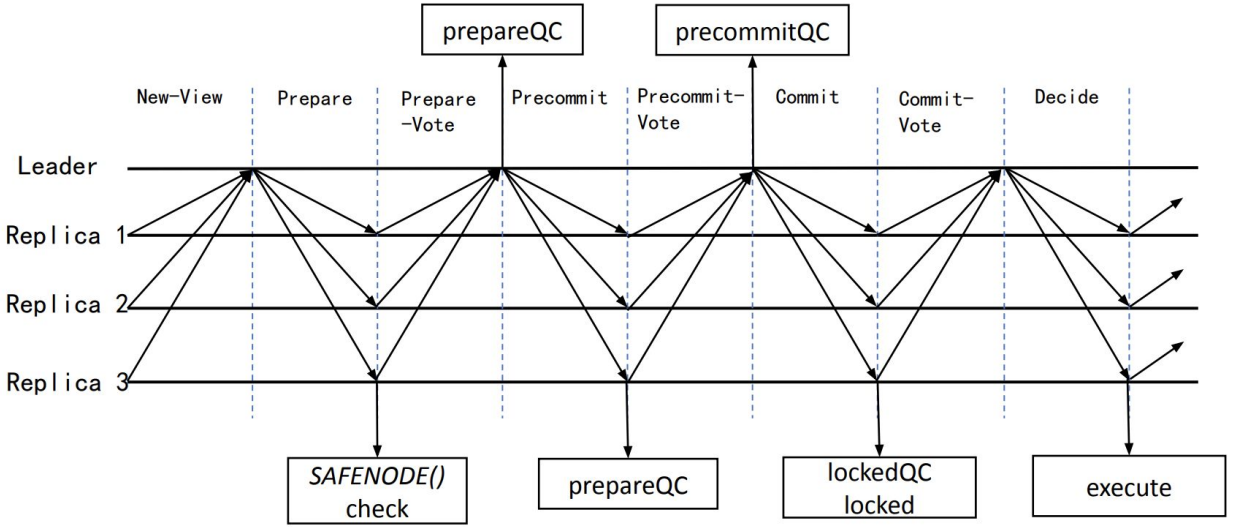
HotStuff's Proposed Solution

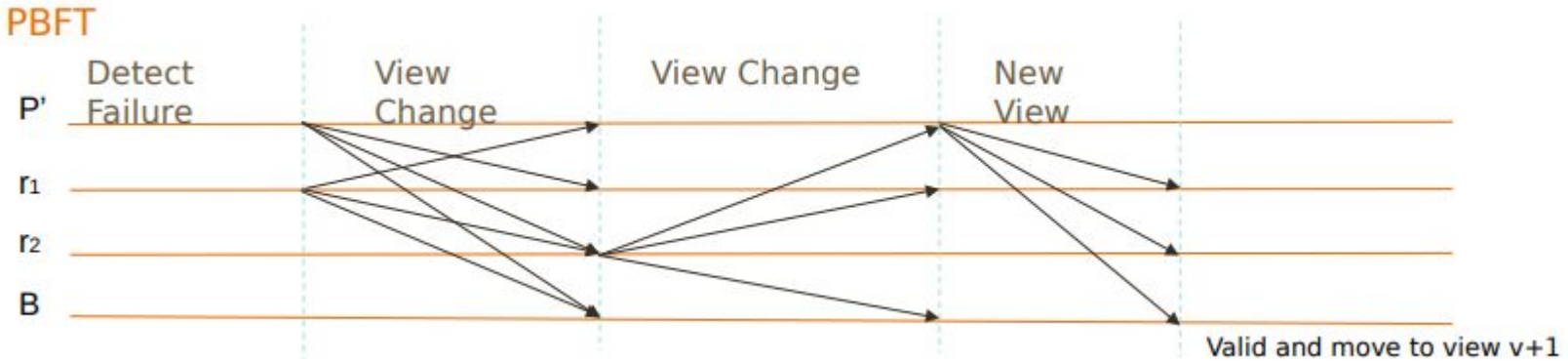
- HotStuff makes use of a one-to-all message structure to address this scaling problem, where the leader alone communicates to n replicas rather than every replica communicating with each other
- This structure places a large amount of power in the leader's hands, so HotStuff utilizes frequent view changes to avoid excess power being given to a single node
- Linear view changes to support frequent view changes
- Threshold Signature to allow for efficient combining of votes from non-primary replicas and for replicas to verify the leader actually received n-f votes

PBFT:

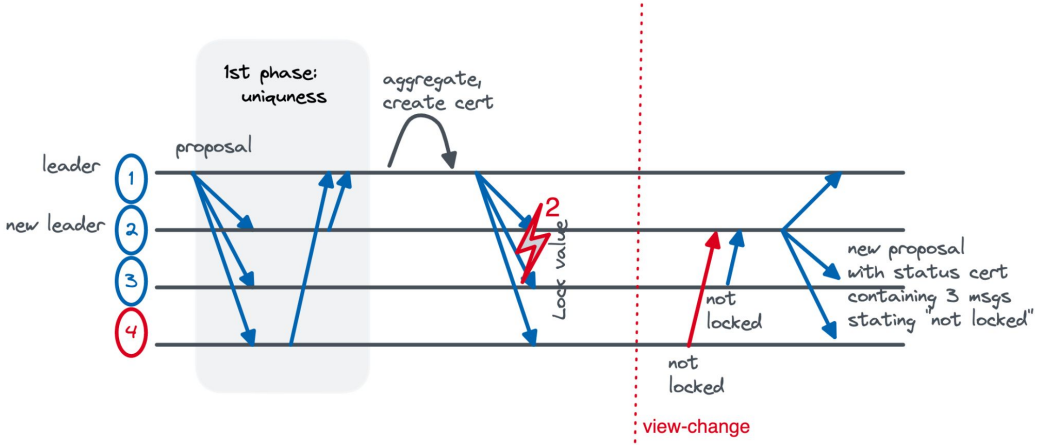


Hotstuff:



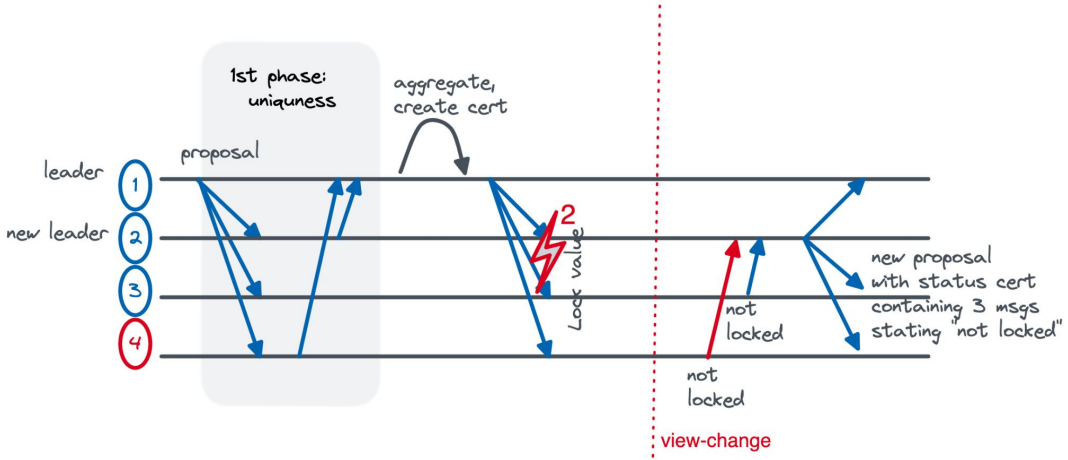

PBFT
Detect Failure
View Change
View Change
New View
P'
r1
r2
B
Valid and move to view v+1

Hotstuff:


1st phase: uniquness
aggregate, create cert
proposal
leader
new leader
1
2
3
4
2
Lock value
not locked
not locked
new proposal with status cert containing 3 msgs stating "not locked"
view-change

Linear View Change

- Linear View Change means that the cost for a new primary to raise a proposal and conduct consensus is equal to the cost for the original primary to do so
- A NextView Interrupt causes the replicas to send their status to the new primary and can happen during



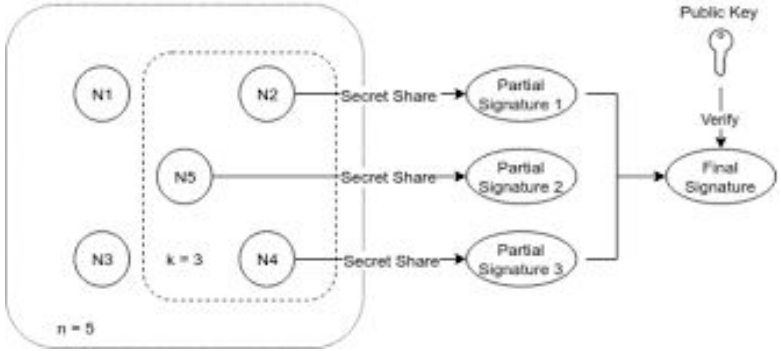
Threshold Signature

With a (k,n)-threshold signature scheme:

- Each replica has its own unique and distinct private key
- Each replica also possesses a common public key
- When a replica sends a message, it signs it using its private key as an ID
- At least k partial signatures can be combined into a final/complete signature
- If a replica receives a complete signature, it can verify it using the public key

Benefits:

- Reduces the number of signatures in consensus
- Reduces the size of the messages being sent
- Results in far less communication per node
- Less verification complexity



HotStuff Protocol Advancements and Key Features

- Linear View Change
- Optimistic Responsiveness
- Expenses incurred by a new leader to drive the protocol towards consensus are comparable to those of the present leader.
- Solves a liveness problem, the hidden lock problem, with the view change protocol by adding a lock-precursor phase.

Mitigating Liveness Issues with Precursor-Lock Round:

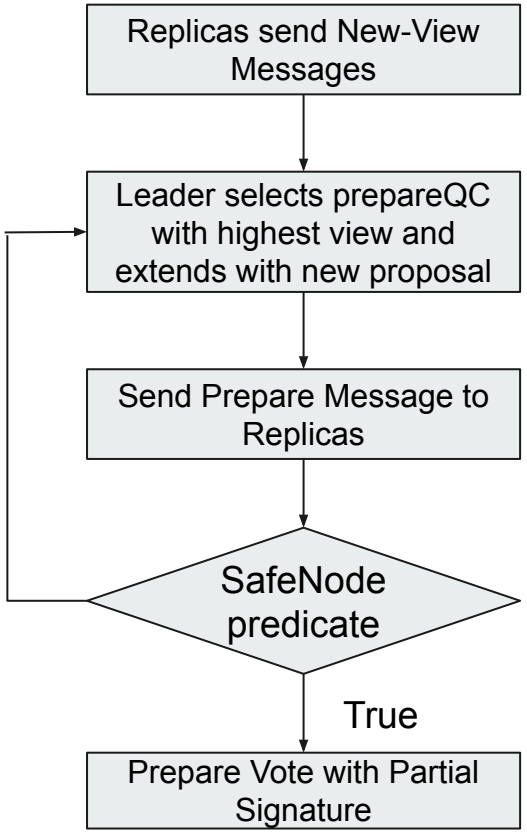
Problem:
The hidden lock problem arises when a leader fails to wait for the expiration time (Δ) of a round.

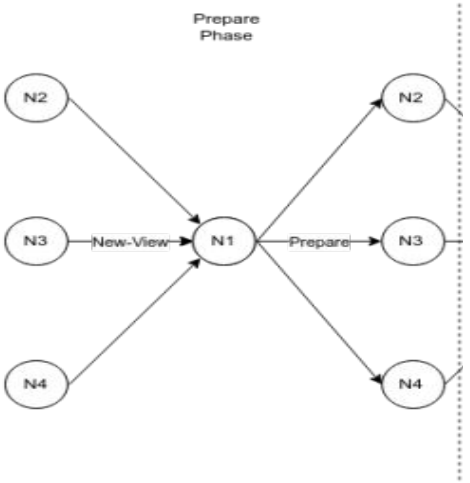
Solution:

- HotStuff introduces a precursor-lock round before the actual lock round to mitigate the hidden lock problem and prevent liveness violations caused by impatient leaders.
- In this precursor-lock round, the leader ensures hearing from 2F+1 participants, enabling it to ascertain the highest lock value proposed (though not necessarily accepted). This mechanism eliminates the necessity of waiting for the maximum Δ expiration time, enhancing system efficiency and responsiveness.

Phases of HotStuff

1) **PREPARE PHASE**



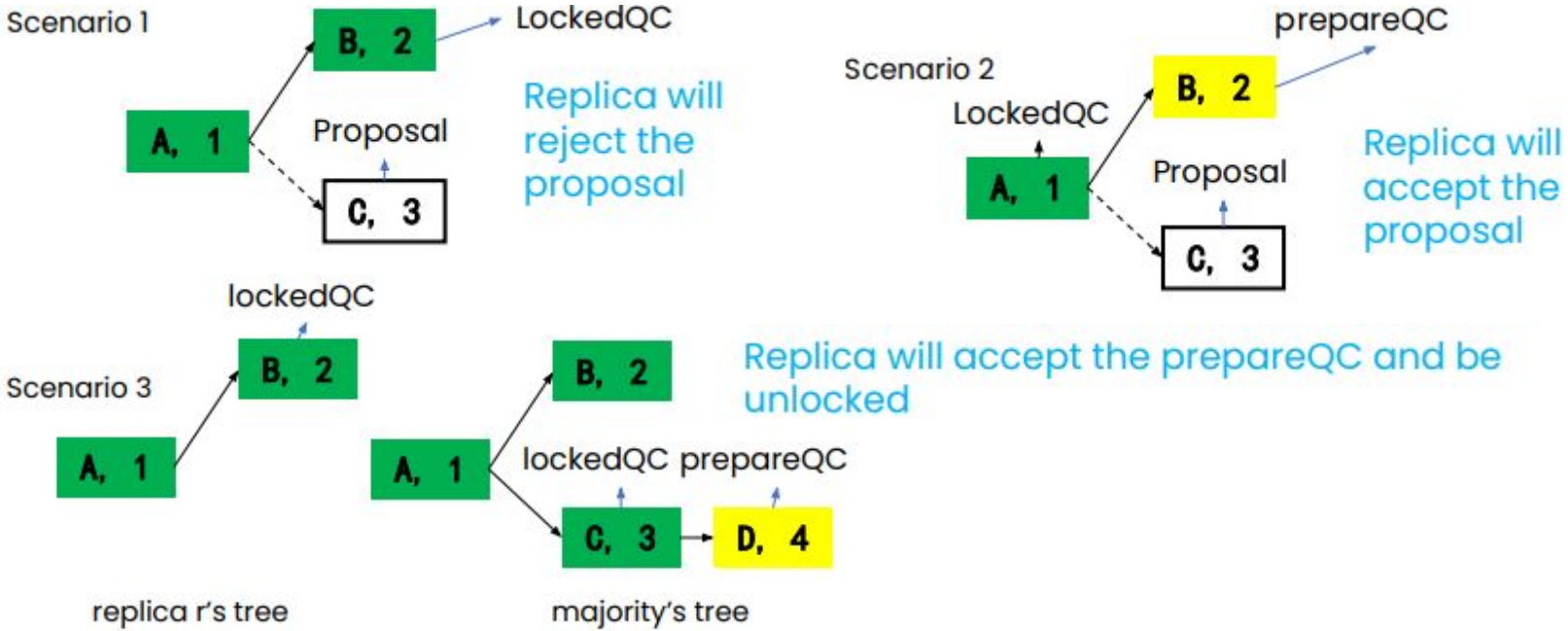


Proposal	View Number	Quorum Certificate

PrepareQC

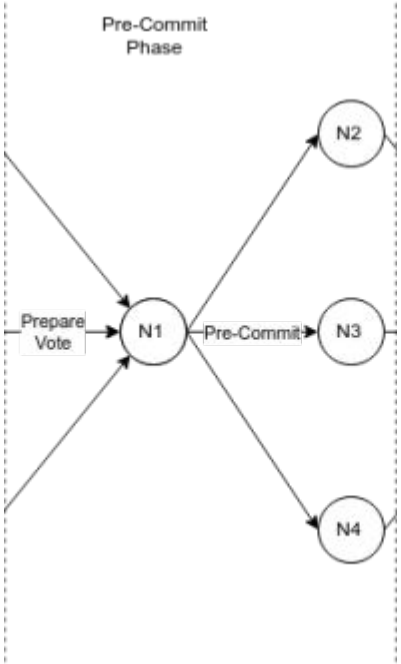
Safety: node extends from lockedQC .node
Liveness: QC.viewNumber > lockedQC.viewNumber

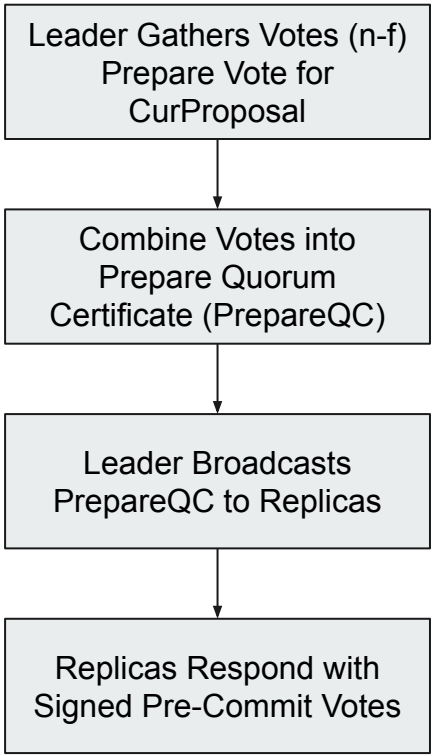
SafeNode Check



Phases of HotStuff

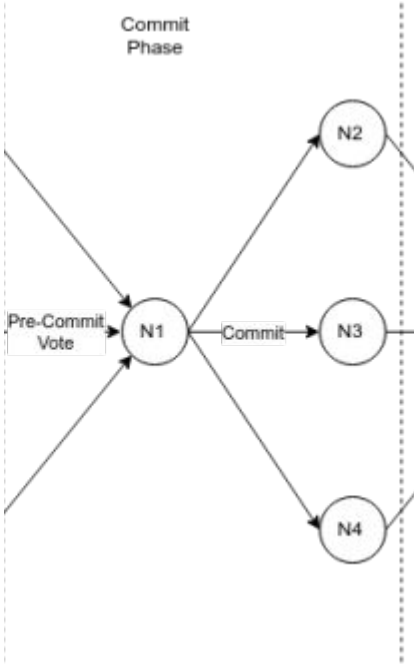
2) Pre-commit Phase:





Phases of HotStuff

3) Commit Phase:



Receive (n-f) pre-commit votes

Combine votes into PreCommitQC

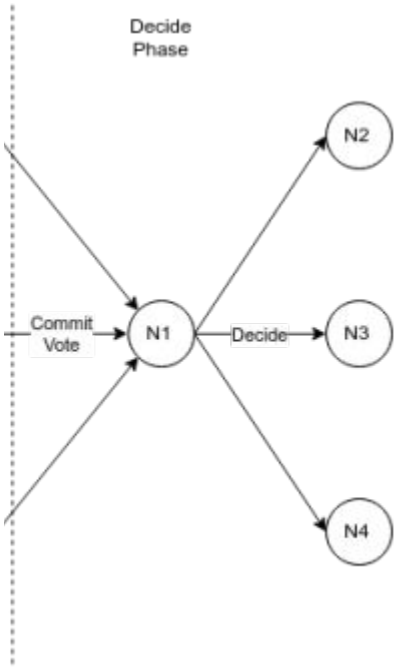
Broadcast PreCommitQC using commit messages

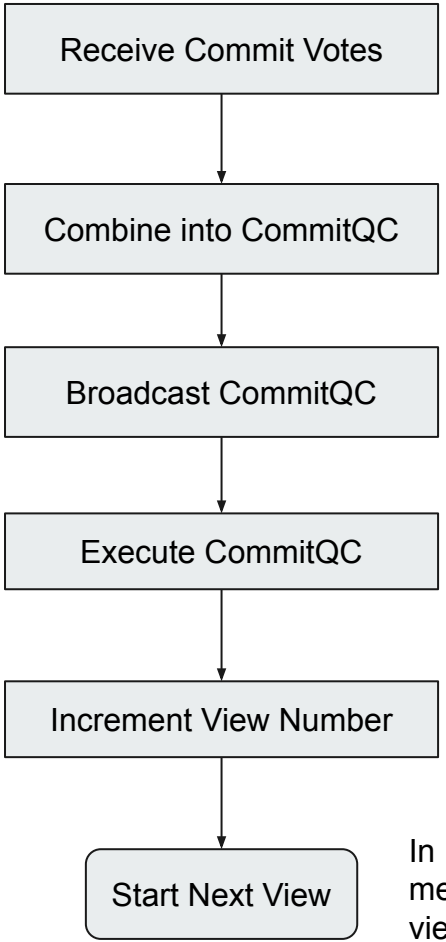
Replica assigns m.justify to local lockedQC

Replicas respond with commit votes

Phases of HotStuff

4) Decide Phase:

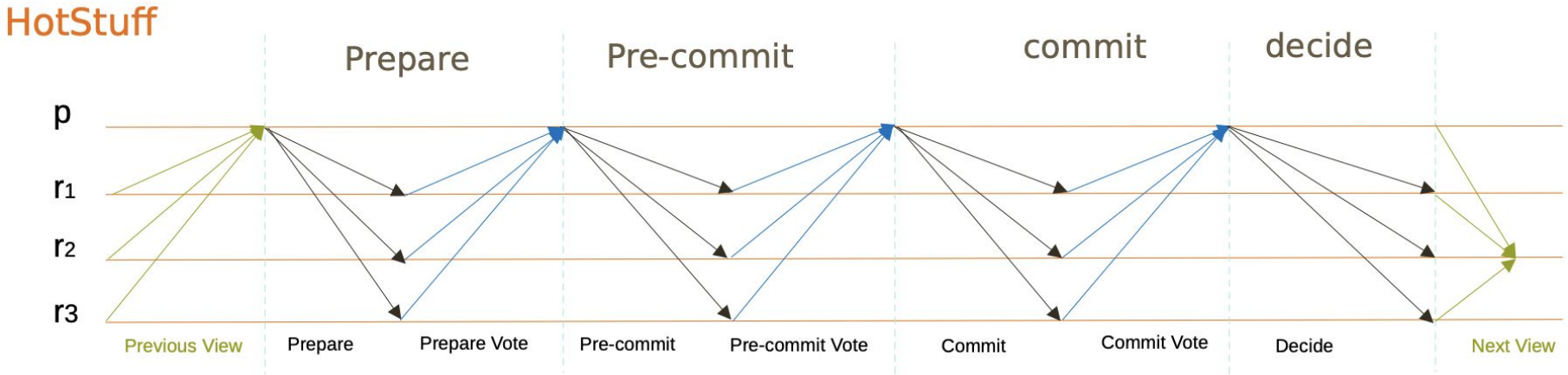




In HotStuff, if a replica waits too long for a message, it automatically moves to the next view to maintain progress.

NextView Interrupt

- Waiting for Messages
- Utility Function: nextView(ViewNumber)



Safety

1. No 2 conflicting QC will have the same view number. Voting only happens once.

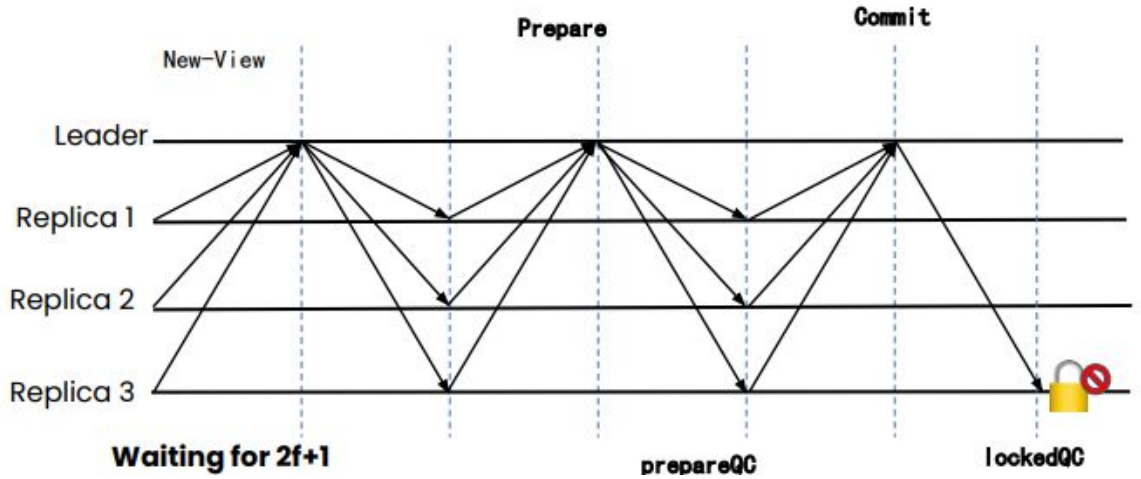
2. No 2 conflicting nodes can be committed by correct replicas at any time.

- Atleast 2f + 1 replicas will be non faulty
- Quorum Certificate
- View Changes
- SafeNode Predicate

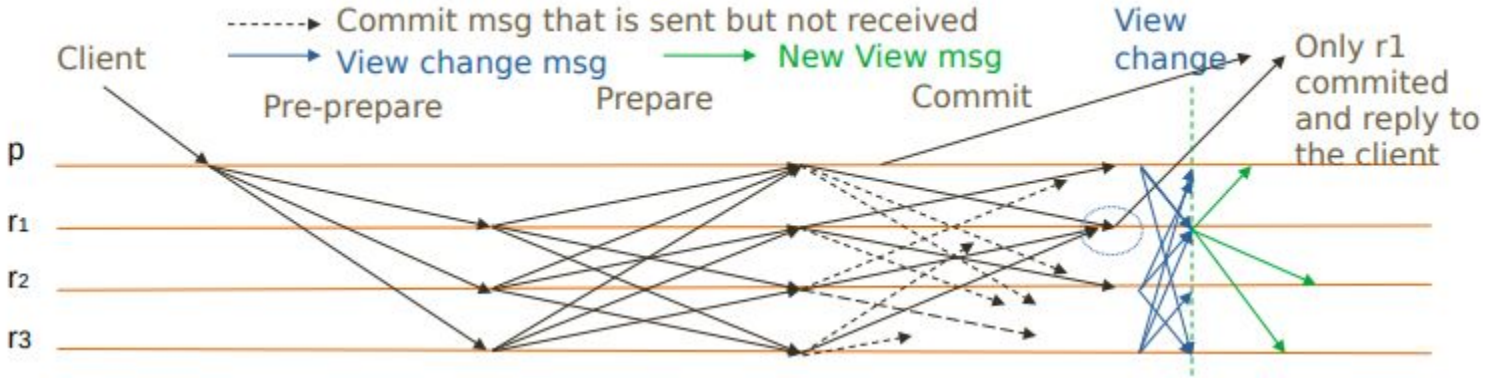
Liveness

- Prevents system from getting stalled
- Avoiding deadlocks
- Optimistically Responsive

Hotstuff:



PBFT
:



What is Chained HotStuff and why do we need it?

Decrease type of messages — Only one type of QC ever created

Simplify the protocol — Easier to code event-driven style programs

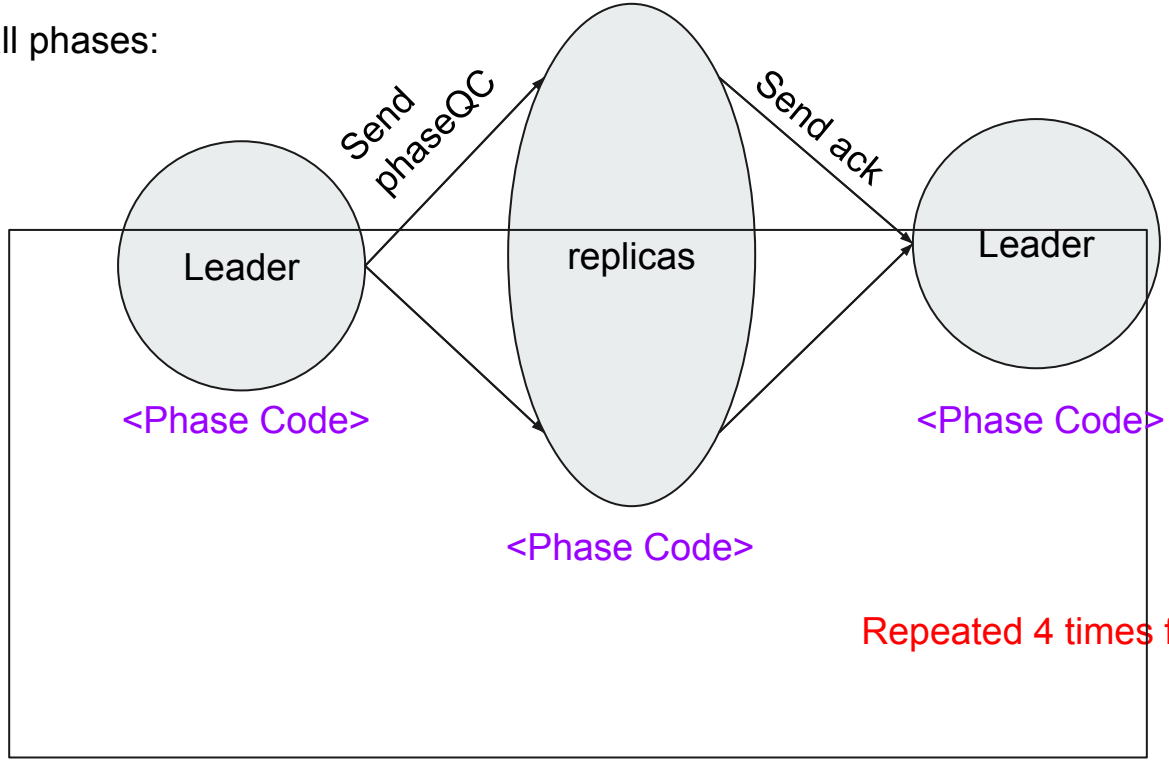
Conceptually separate safety and liveness

View existing BFT protocols in a common framework

Idea : Pipeline the phases

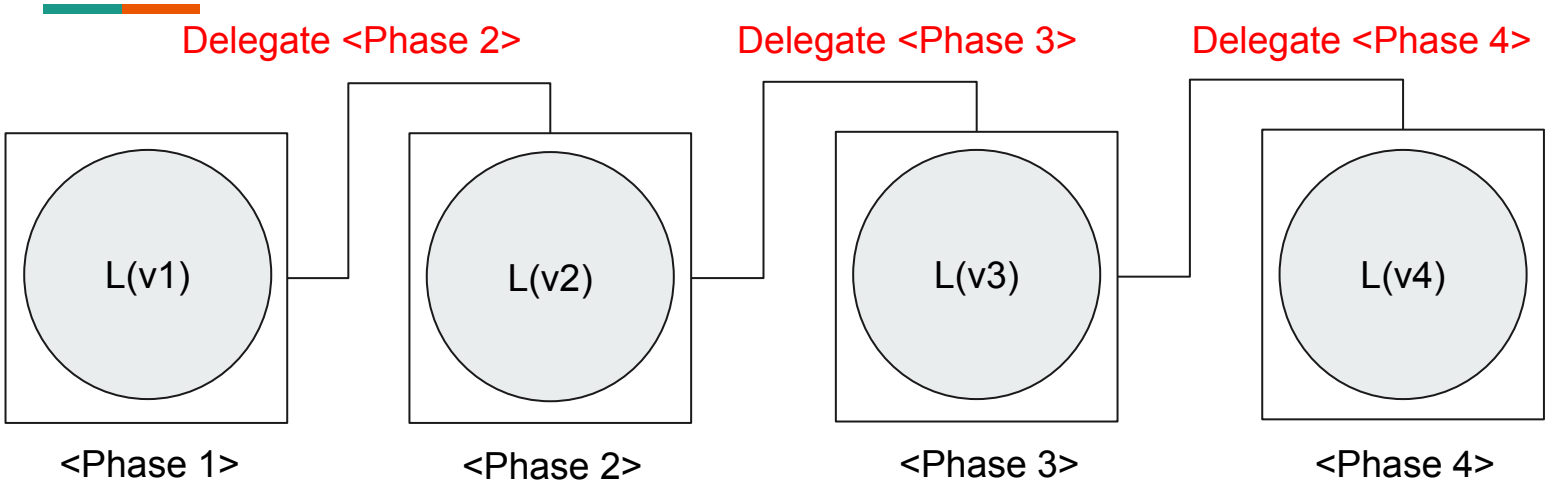
Common in all phases:

View =



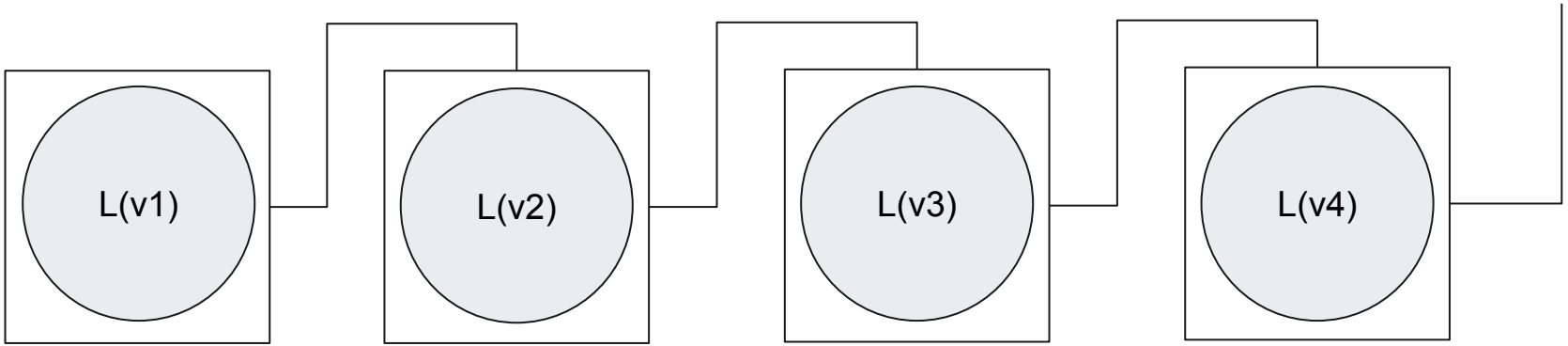
× 4

Idea: Pipeline the phases



But L(v2) needs to start its own view…

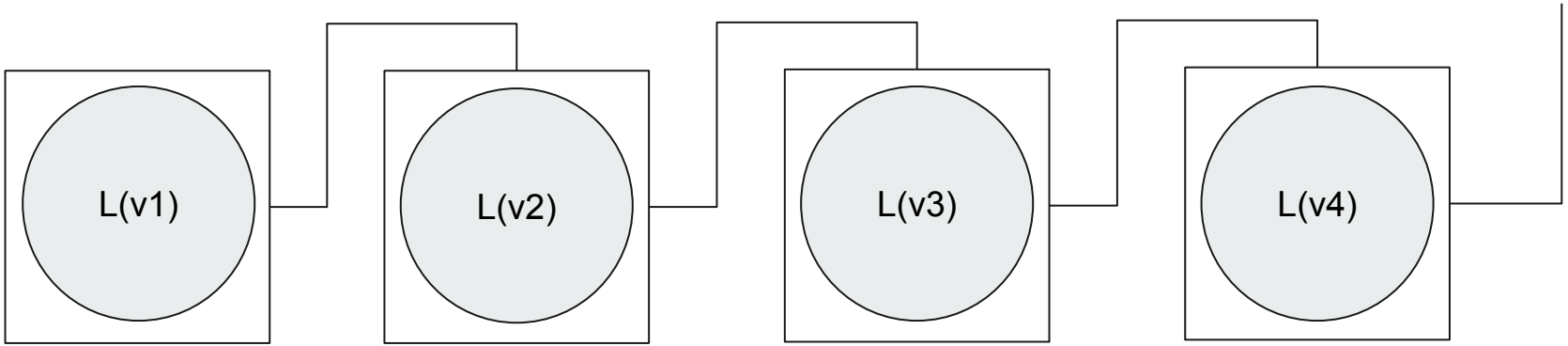
continues ….



different colours represent different views

Same for L(v3), L(v4) ….

continues ….



<Phase 1>

<Phase 2>

<Phase 1>

<Phase 3>

<Phase 2>

<Phase 1>

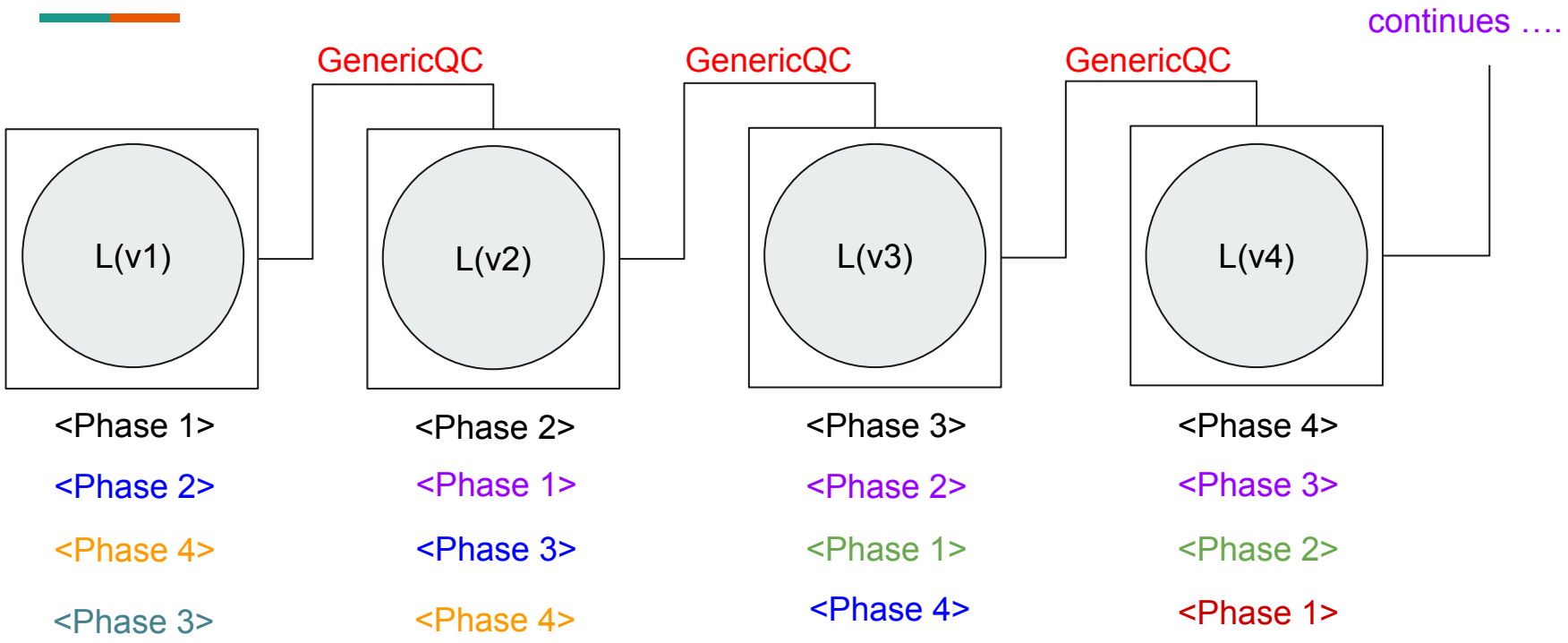
<Phase 4>

<Phase 3>

<Phase 2>

<Phase 1>

Code is identical for all views



Code for different phases

Prepare Phase:

1. Leader sends QC
2. On receiving QC, check for conflicts and vote

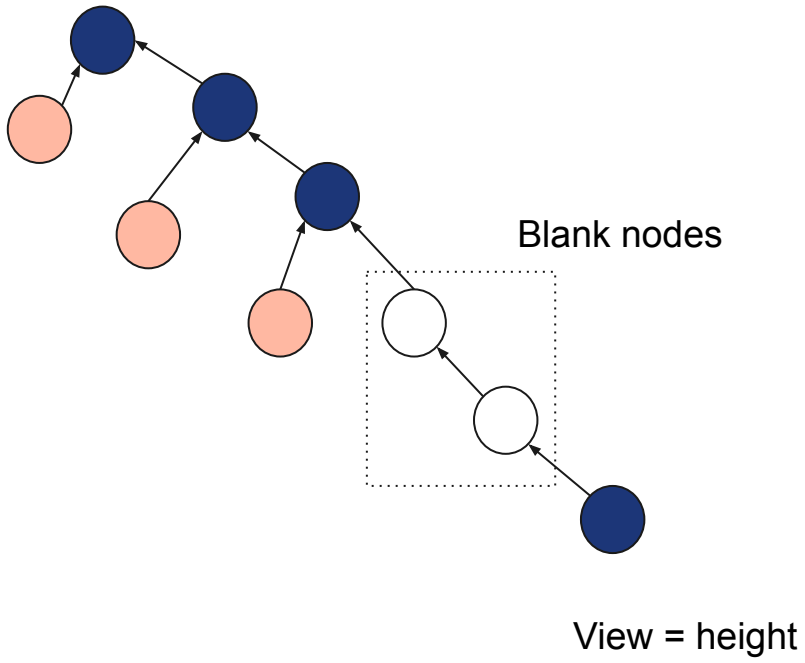
Pre-commit Phase:

1. Leader sends QC
2. On receiving QC, lock the node

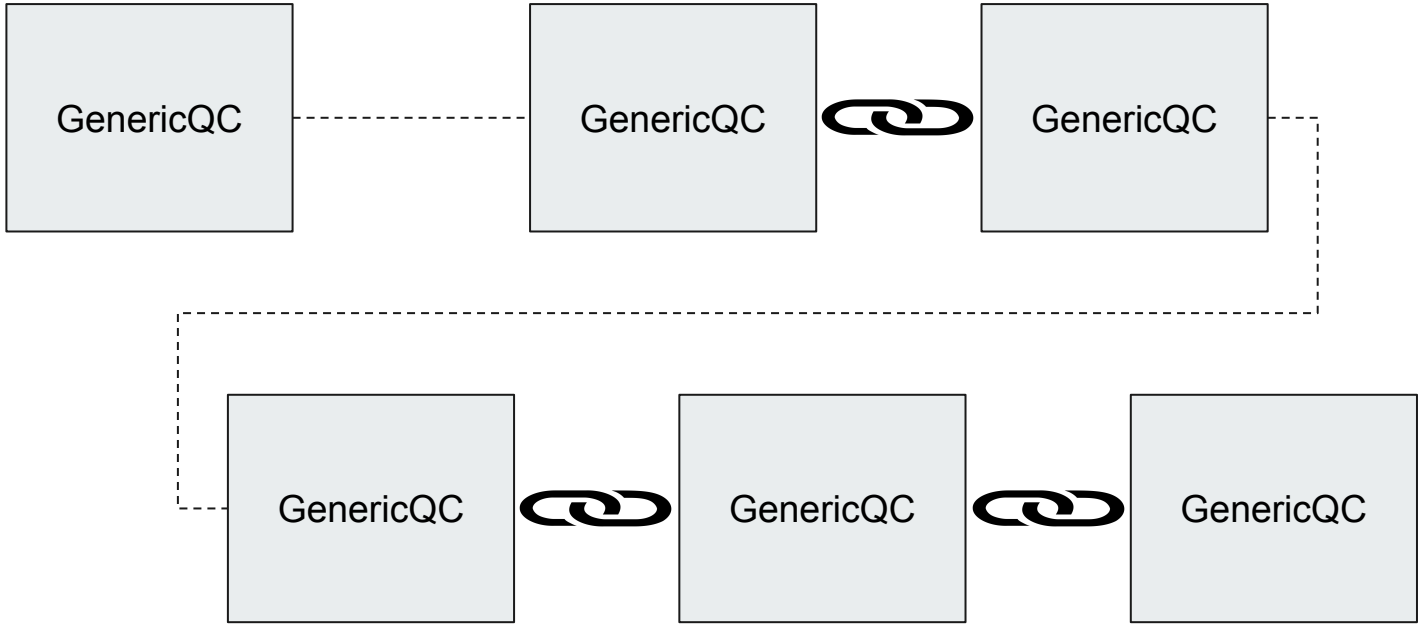
Commit Phase:

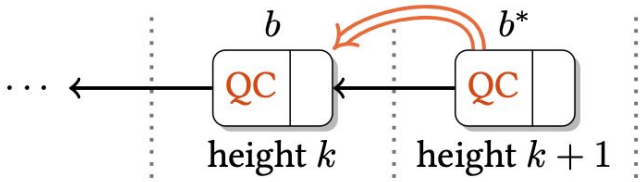
1. Leader sends QC
2. On receiving QC, check for conflicts and vote

```
viewNumber

parent

grand-parent

great-grand-parent

Quorum Certificate for message

Node digests for verification
```

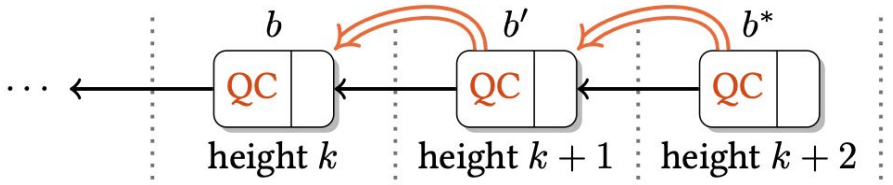


One-chain, Two-chain and Three-chains

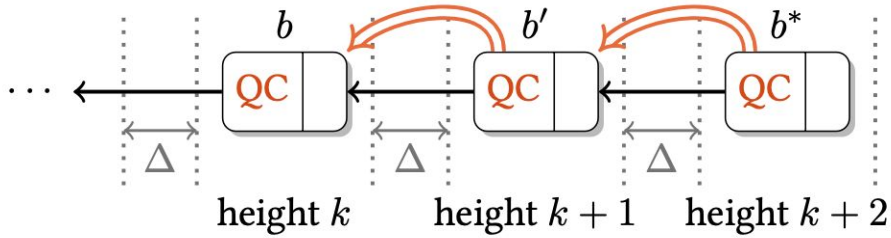




(a) One-Chain (DLS, 1988)

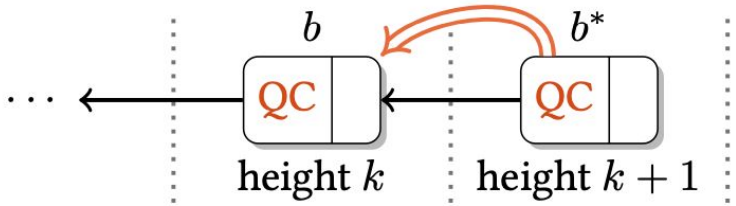


(b) Two-Chain (PBFT, 1999)



(c) Two-Chain w/ delay (Tendermint, 2016)

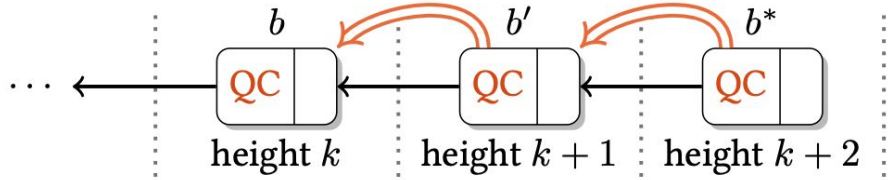
DLS - Commit after one chain



(a) One-Chain (DLS, 1988)

- Voting node = locked node
- Only leader commits
- Complicated unlock mechanism

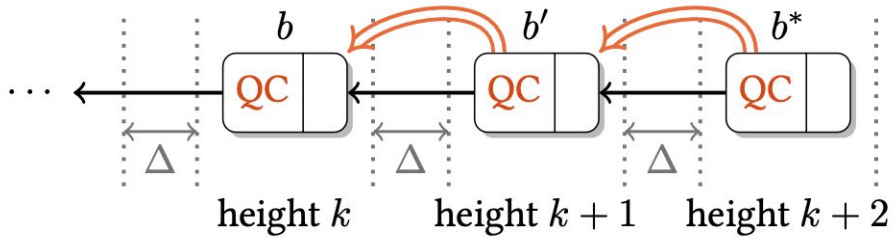
PBFT - Commit after two chains



(b) Two-Chain (PBFT, 1999)

- Lock after one phase
- Commit longest chain
- Convincing longest chain requires quadratic messages per view.

Tendermint - Commit after two chains, but wait...



(c) Two-Chain w/ delay (Tendermint, 2016)

- Wait for max - delay (after GST) each phase
- This guarantees liveness (no need for large proofs of one-chains)

Sources Used

https://arxiv.org/pdf/1803.05069.pdf

https://www.youtube.com/watch?v=GAGW-c4hADA&t=39s

https://decentralizedthoughts.github.io/2023-04-01-hotstuff-2/

https://expolab.org/ecs265-fall-2022/slides/HotStuff-presentation.pdf

https://expolab.org/ecs265-fall-2021/slides/4_HotStuff.pdf

https://medium.com/ontologynetwork/hotstuff-the-consensus-protocol-behind-facebooks-libra bft-a5503680b151

https://medium.com/@chamirachid/your-journey-to-consensus-part-4-hotstuff-c651a70b02b9

http://muratbuffalo.blogspot.com/2019/12/hotstuff-bft-consensus-in-lens-of.html

https://www.youtube.com/watch?v=ONobI3X70Rc&t=2s