HOTSTUFF

Consensus in the Lens of Blockchain.

PRESENTATION BY-
Amy Vu,
Alyssa Yee,
Alex Gao,
Guransh Singh Anand

BACKGROUND

- PBFT Supports $n \geq 3f+1$, where n = total nodes, f = # of faulty nodes
- BFT(precursor to PBFT) was designed for system size n=4 or n=7, not 1000's of nodes in modern systems

PROBLEM?

- Not practical, makes many assumptions
- Provides inefficient way of solving problem (only theoretical solution)
- Does not assume delays in the network

PBFT made to be use in practical environment, BFT made for theoretical concept

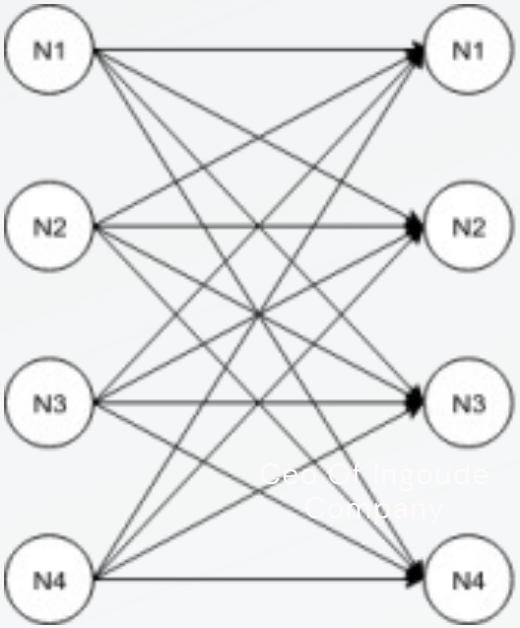
WHY HOTSTUFF?

01 **Linear View Change:** Reduces communication complexity

Protocol	Normal Operation	View-Change (Failure)	Continuous Failure
PBFT	$O(n^2)$	$O(n^3)$	$O(fn^3)$
HotStuff	$O(n)$	$O(n)$	$O(fn)$

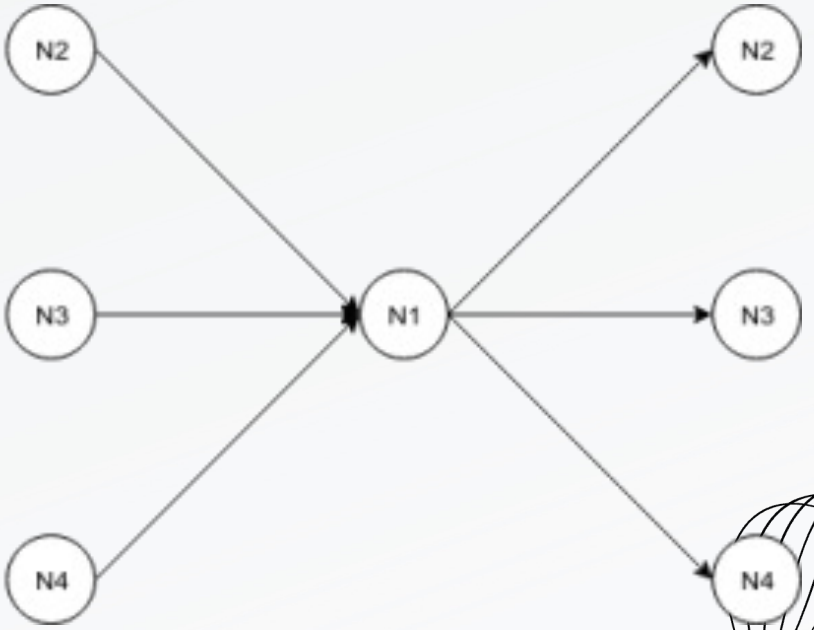
PBFT

Mesh Communication Network



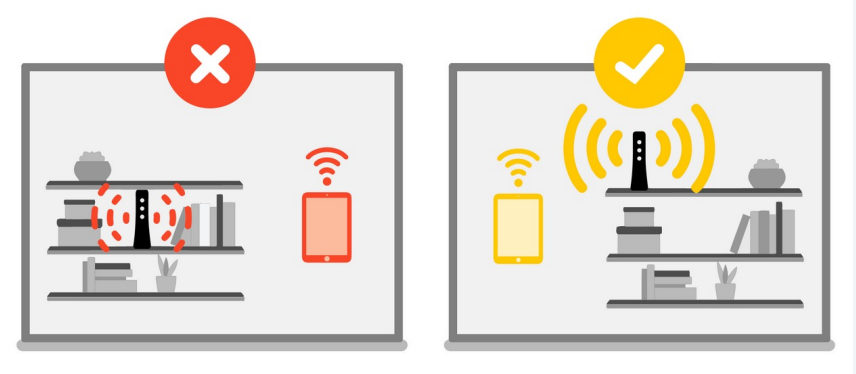
Hotstuff

Star Communication Network



02 **Optimistic Responsiveness:** performs on actual network conditions (not worse-case)

- Receives majority votes, instantly commits

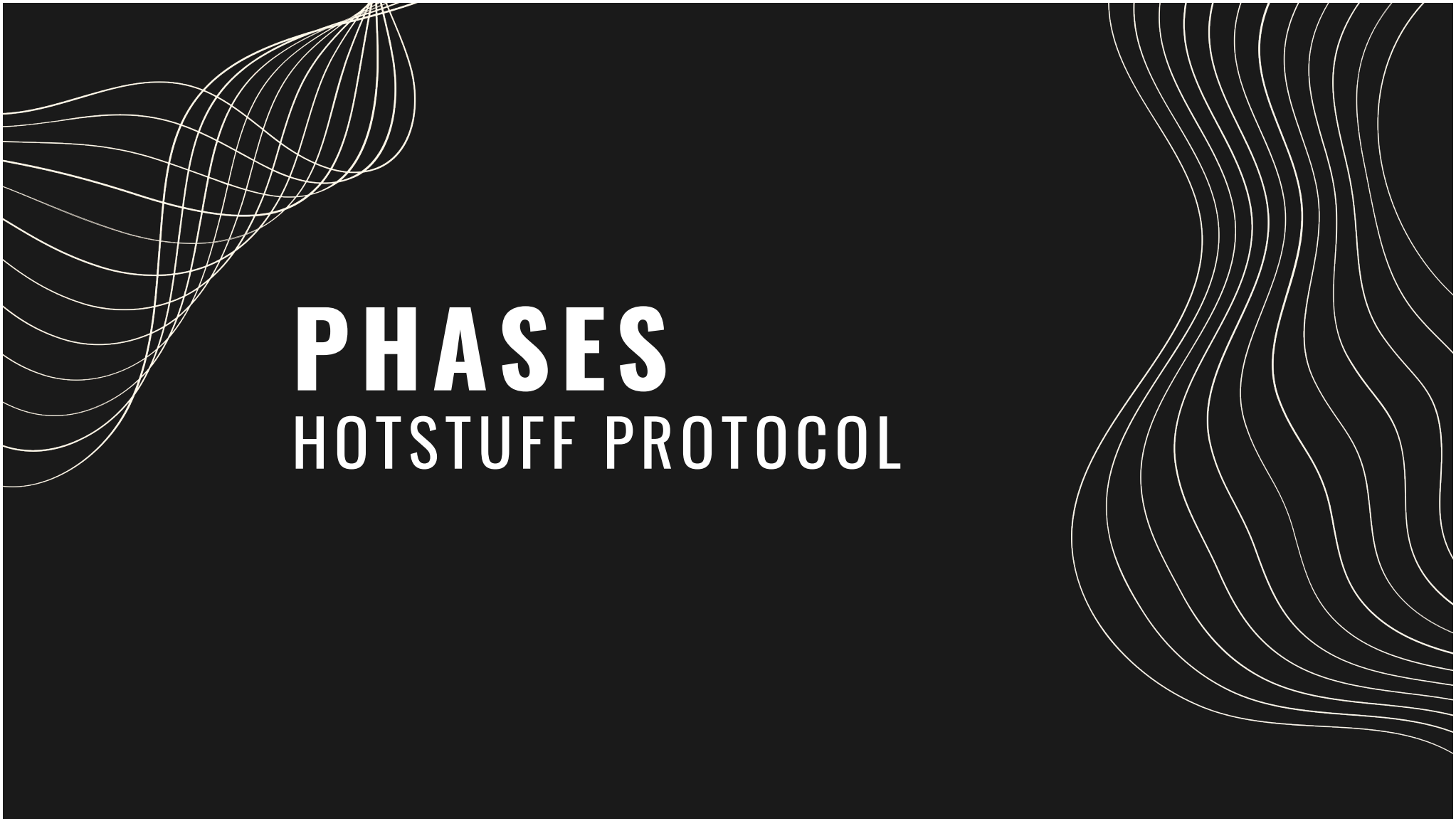


03 **Simple:** Easy to understand, implement, and improve

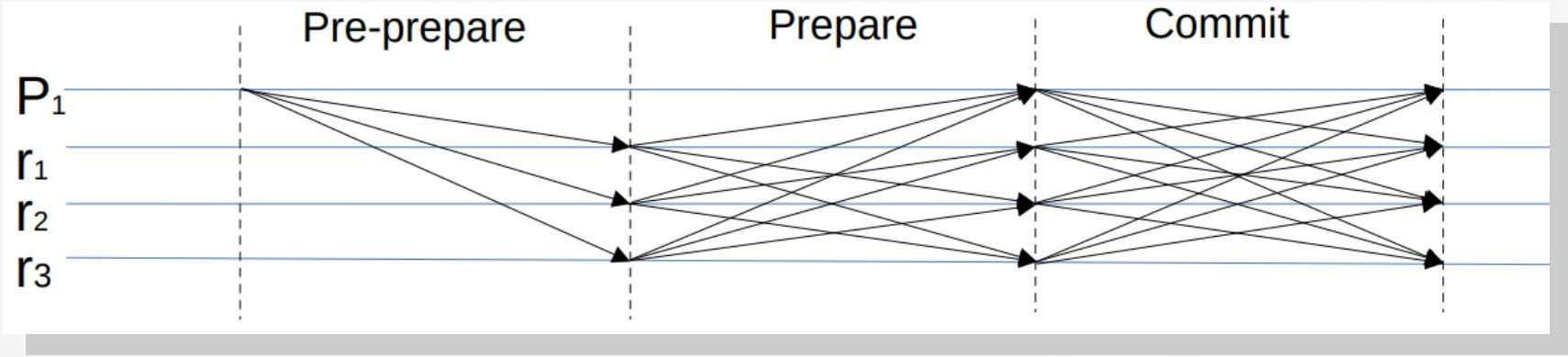
- Hotstuff-2, Swift Hotstuff, QuickBFT

04 **Practical:** Partially synchronous, used in real-world applications

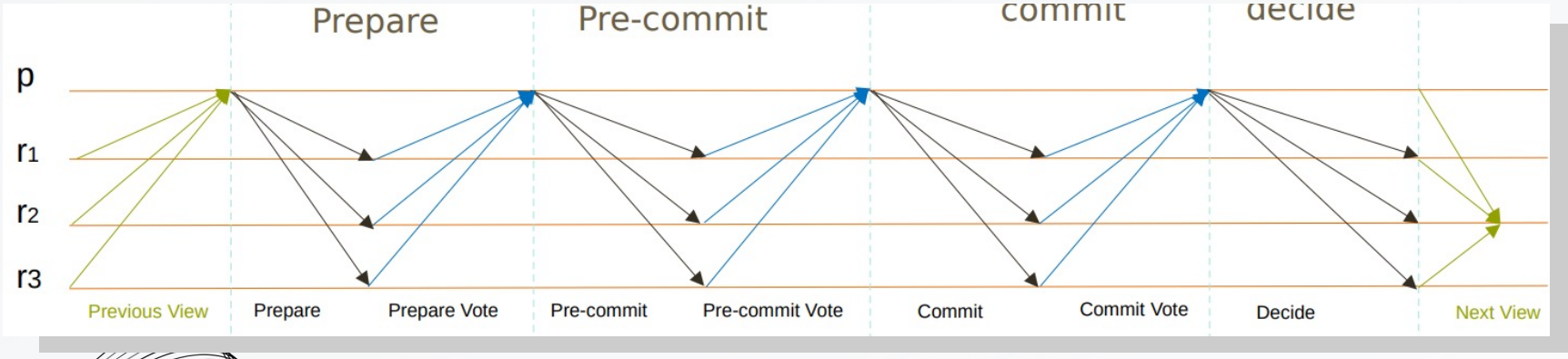
- Facebook, SafeStake


PHASES
HOTSTUFF PROTOCOL

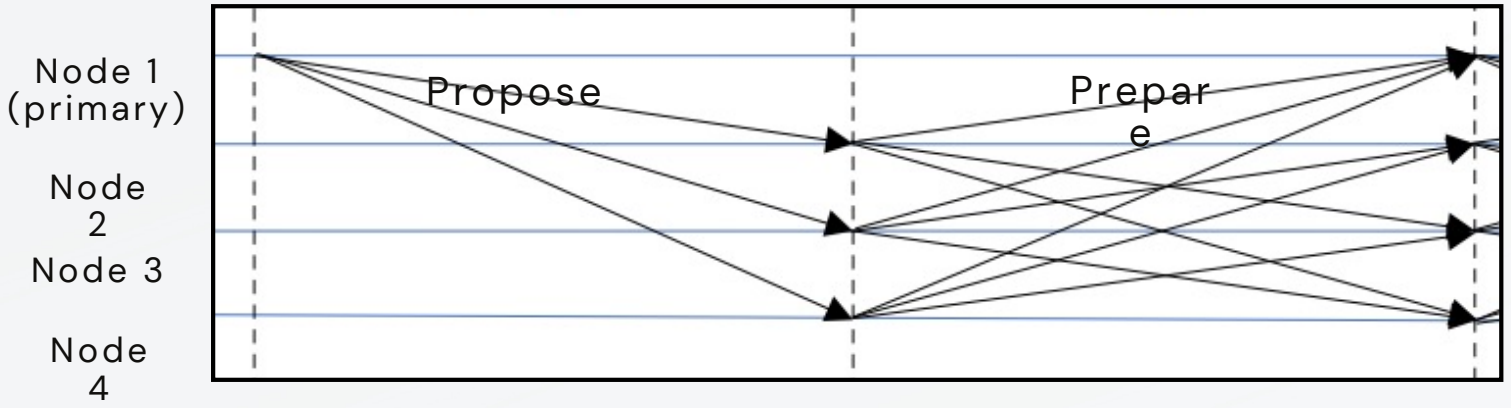
PBFT



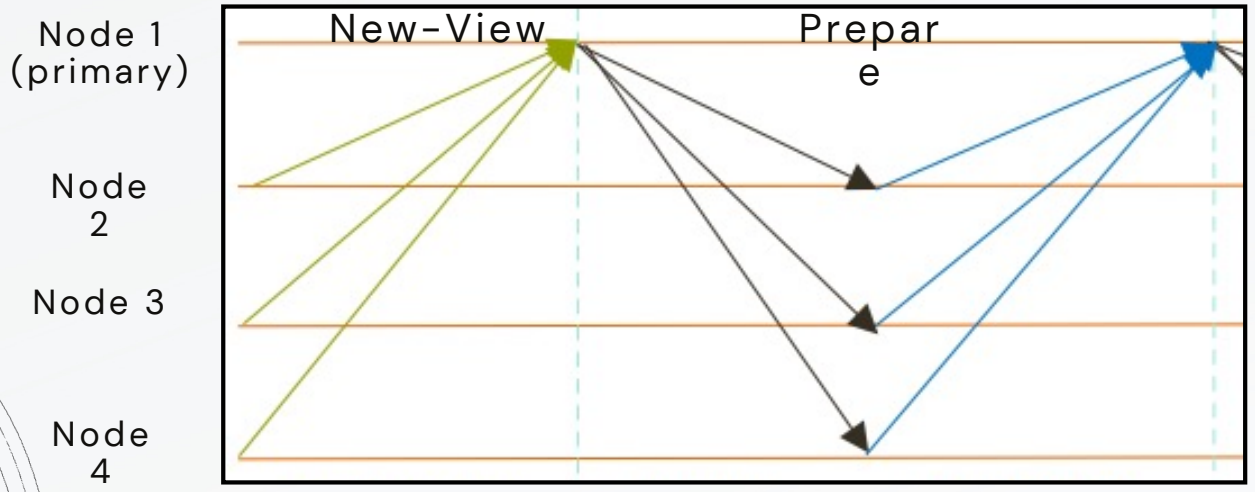
Hotstuff



PREPARE PHASE



PBF
T



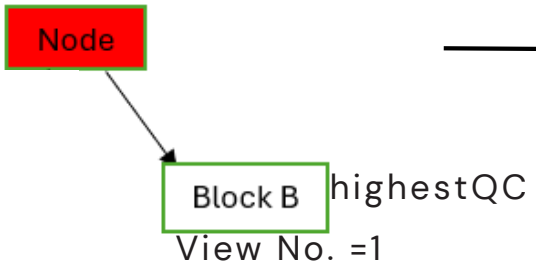
HOTSTUF
F

TREE and BRANCHES

- Commands in the protocol are wrapped in nodes that are part of a larger **tree** structure
- Each node contains parent link which is the hash digest of the parent node
- **Branches** extend from nodes and represent different possible paths

Trees-the differentiating factor

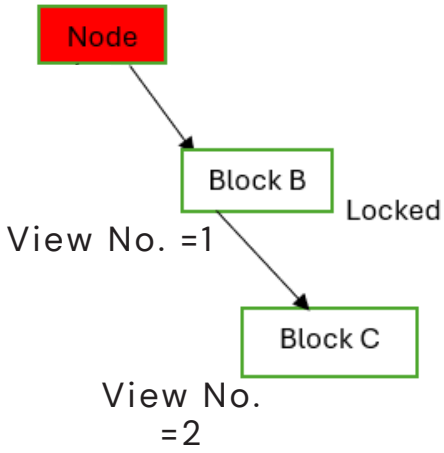
Local tree structure in all nodes after committing transaction B in last view. When B is committed it gets Block B is locked.

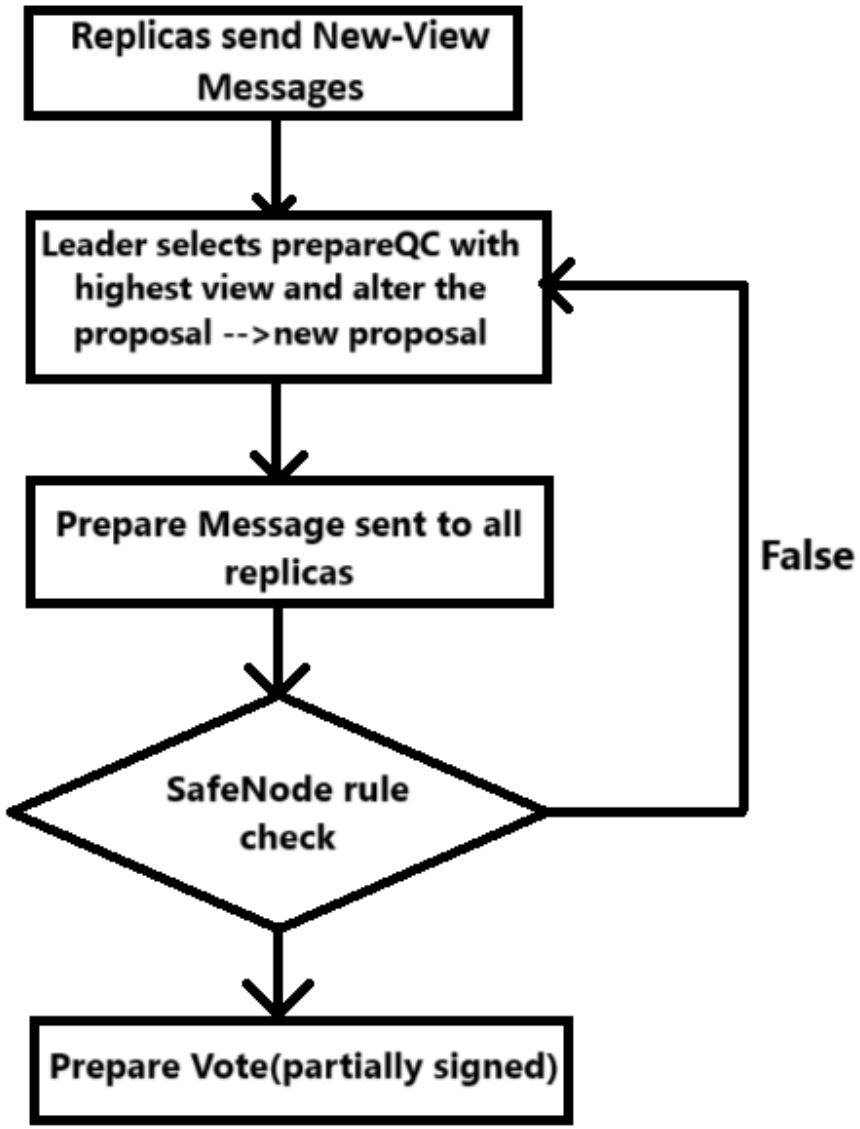


Now leader want to commit transaction C

For safety, it will build on block that had highest approval in the last view. Here its B. The leader will then add a new leaf node(block C) pointing to B.

Local tree in primary.



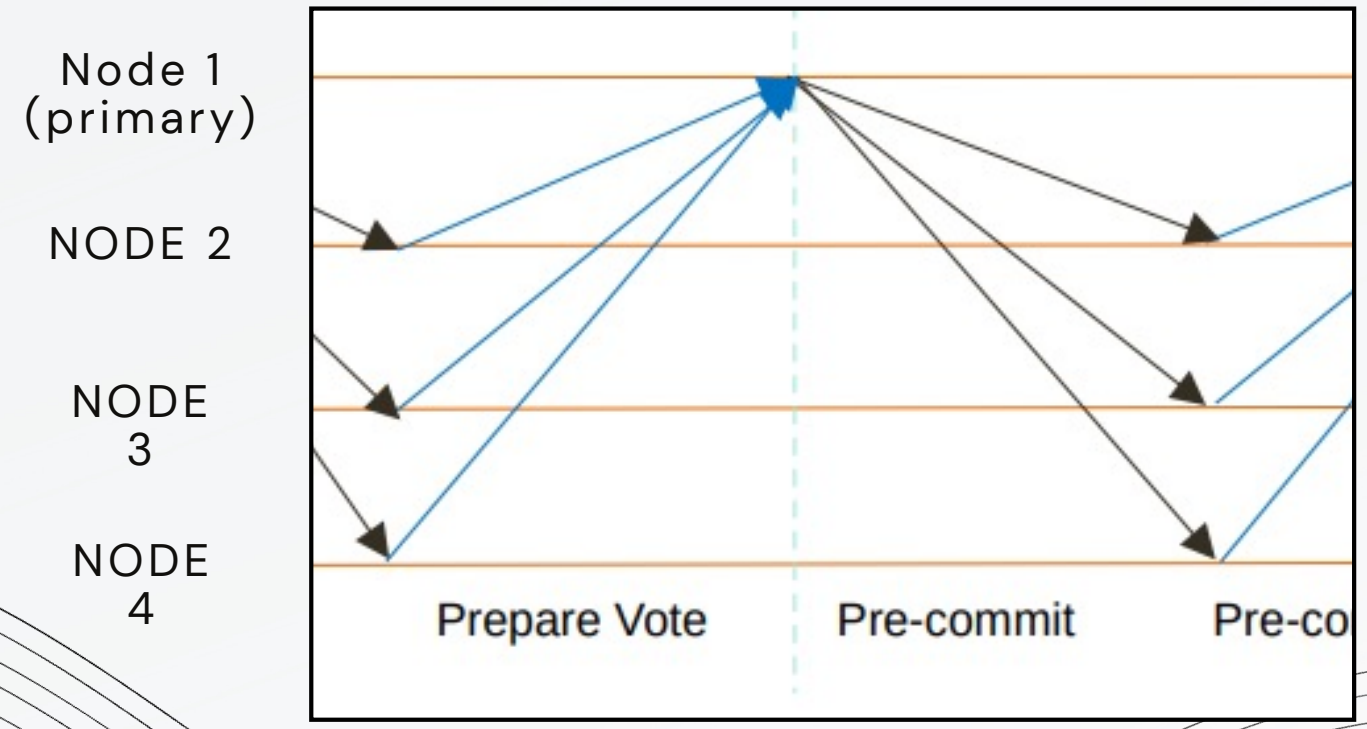

Replicas send New-View Messages
Leader selects prepareQC with highest view and alter the proposal -->new proposal
Prepare Message sent to all replicas
SafeNode rule check
False
Prepare Vote(partially signed)

SAFENODE

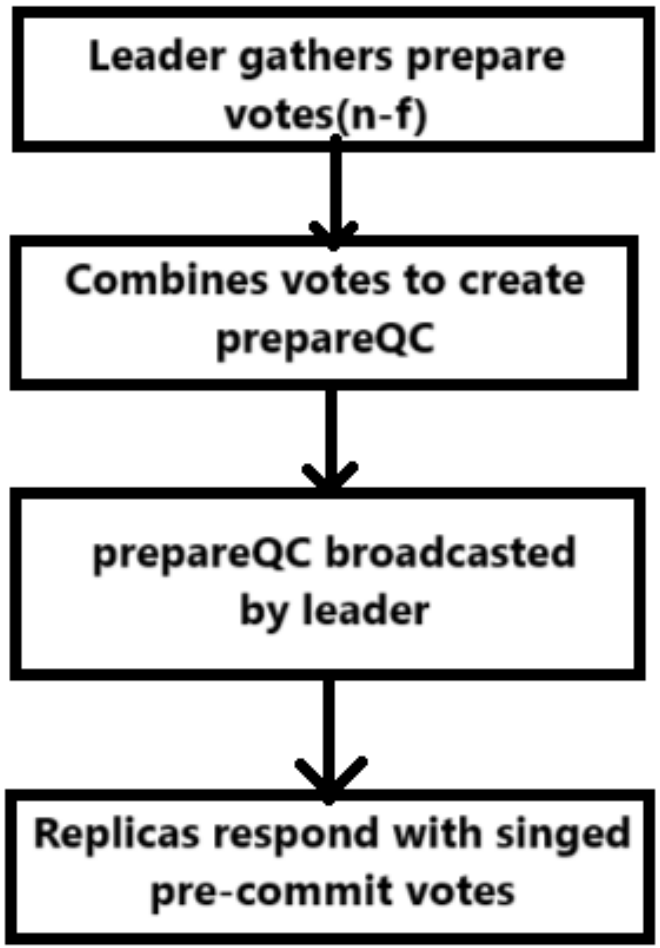
Safety: node extends from previous locked node

Liveliness: viewNumber higher than previous

PRE-COMMIT PHASE

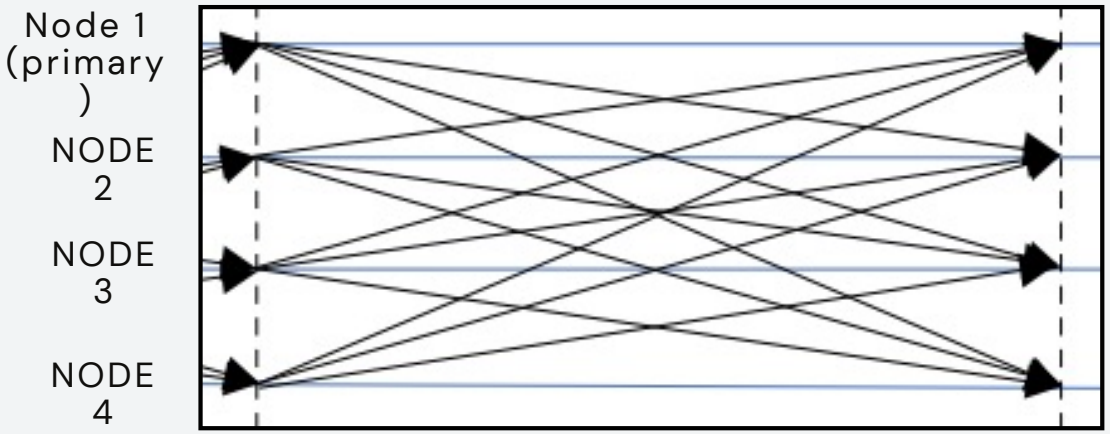


HOTSTUF
F

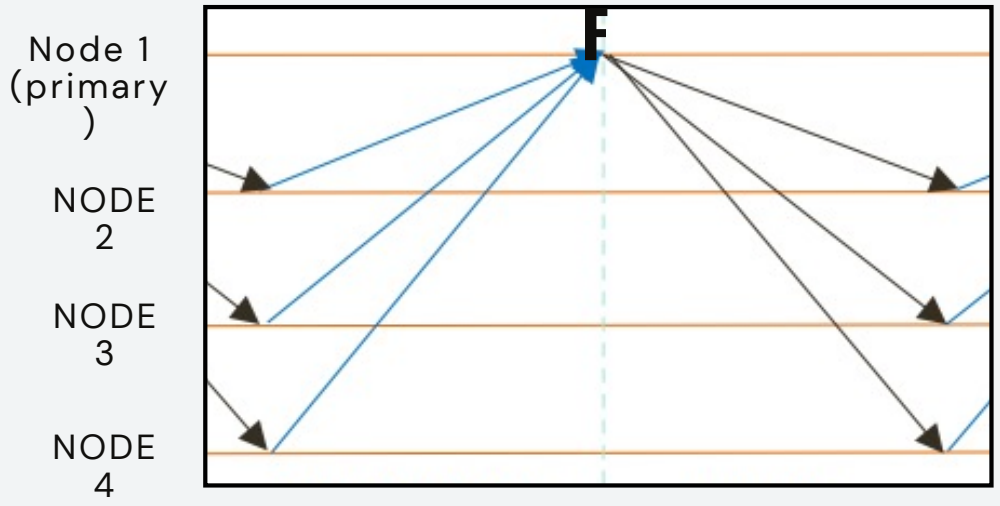

Leader gathers prepare votes(n-f)
Combines votes to create prepareQC
prepareQC broadcasted by leader
Replicas respond with singed pre-commit votes

COMMIT PHASE

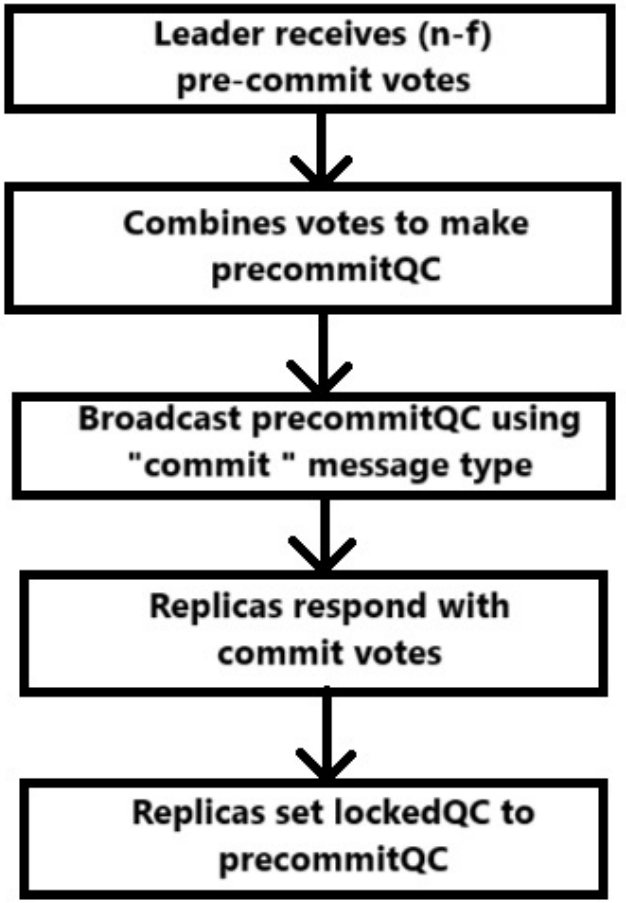
PBF



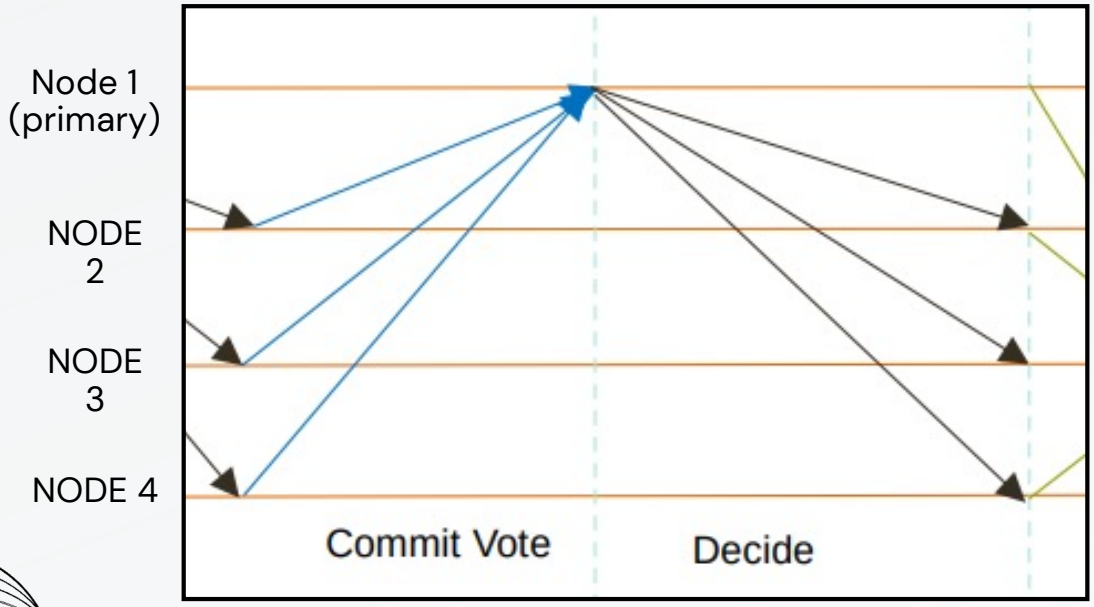
HOTSTUFF



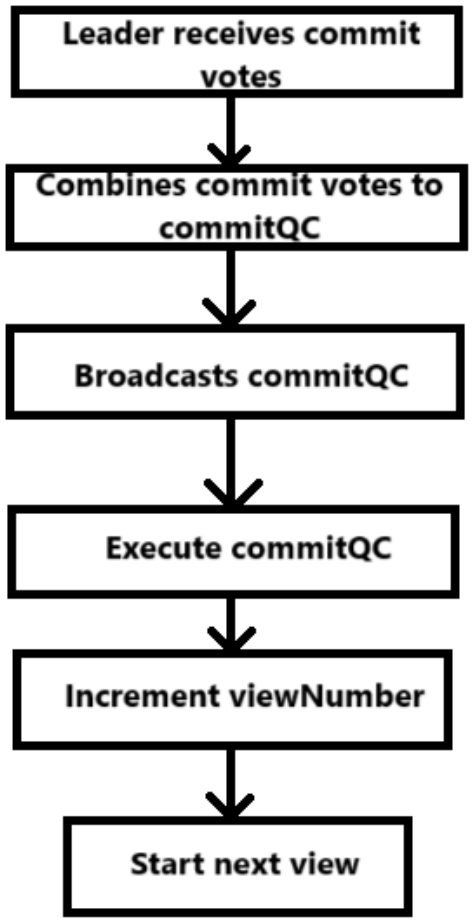
The last step provides safety incase the proposal become a consensus decision



DECIDE PHASE



HOTSTUF
F


Leader receives commit votes
Combines commit votes to commitQC
Broadcasts commitQC
Execute commitQC
Increment viewNumber
Start next view

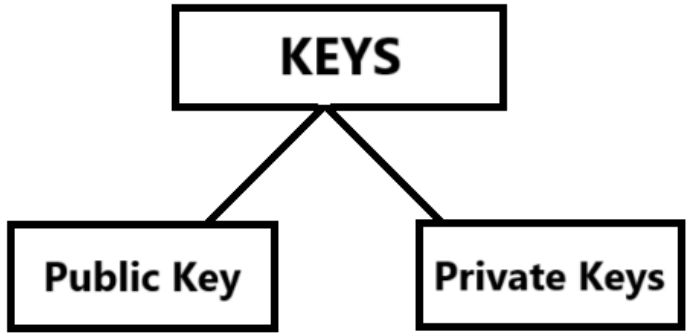
NEXTVIEW INTERRUPT

- All phases, replica waits for a message at view *viewNumber* for a timeout period.
- If nextView interrupt, replica increments *viewNumber* and starts the next view.

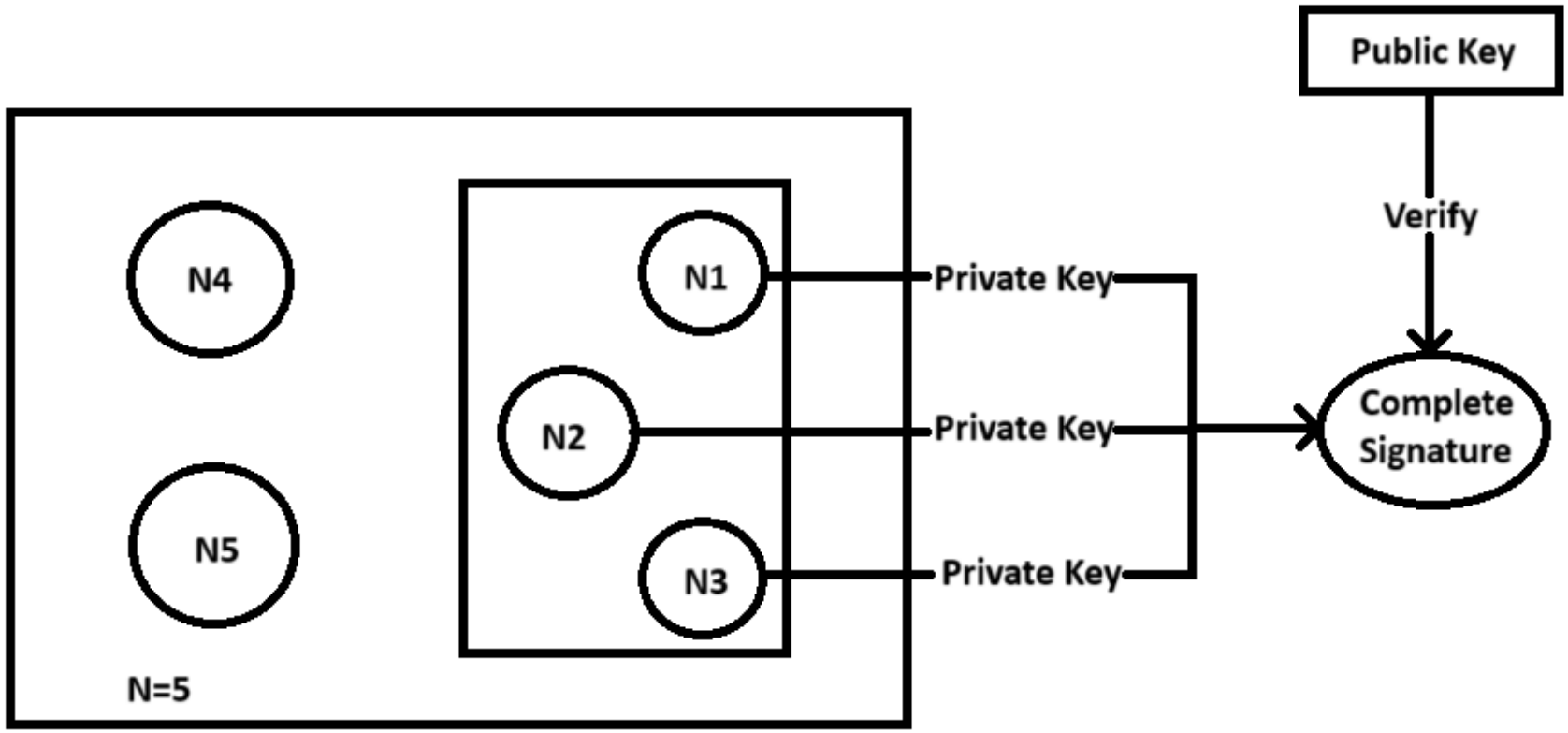
CRYPTOGRAPHIC PRIMITIVES

Used for security against Byzantine faults caused by faulty nodes.

(k,n)-Threshold Signature Scheme

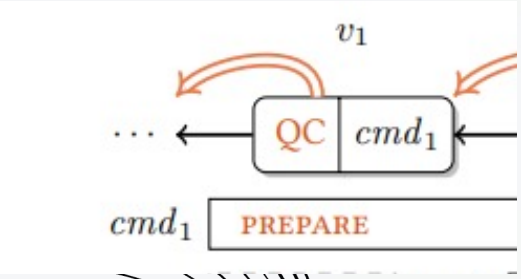


- **Public Key:** Single public key is shared by all replicas.
- **Private Keys:** Each replica has a private key which it will use to generate a partial signature.


Public Key
Verify
N4
N5
N=5
N1
N2
N3
Private Key
Private Key
Private Key
Complete
Signature

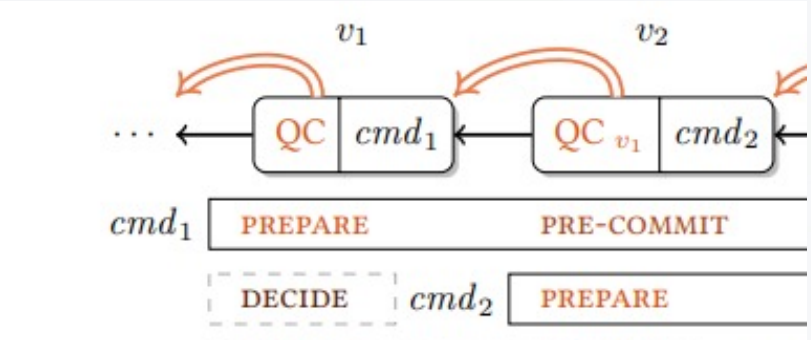
CHAINED HOTSTUFF

- Pipelines process, reduces number of messages
- genericQC: Send to next leader to delegate responsibility



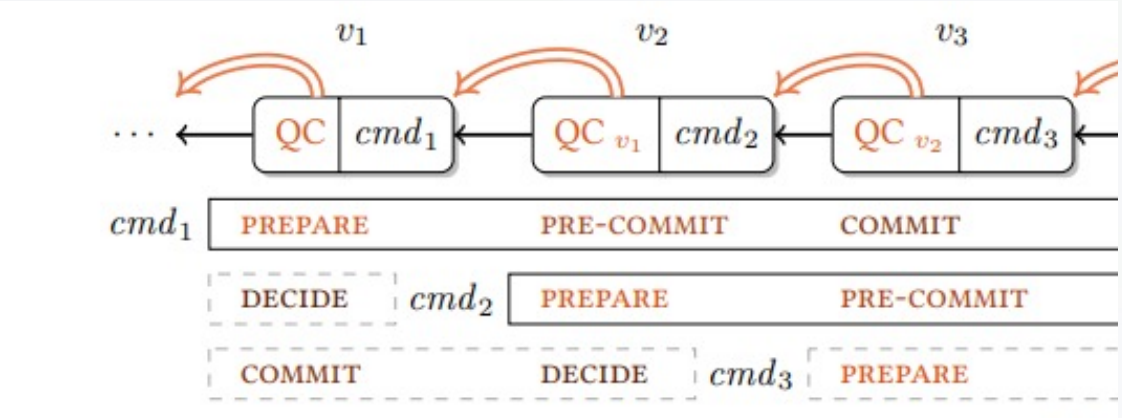
CHAINED HOTSTUFF

- Pipelines process, reduces number of messages
- genericQC: Send to next leader to delegate responsibility



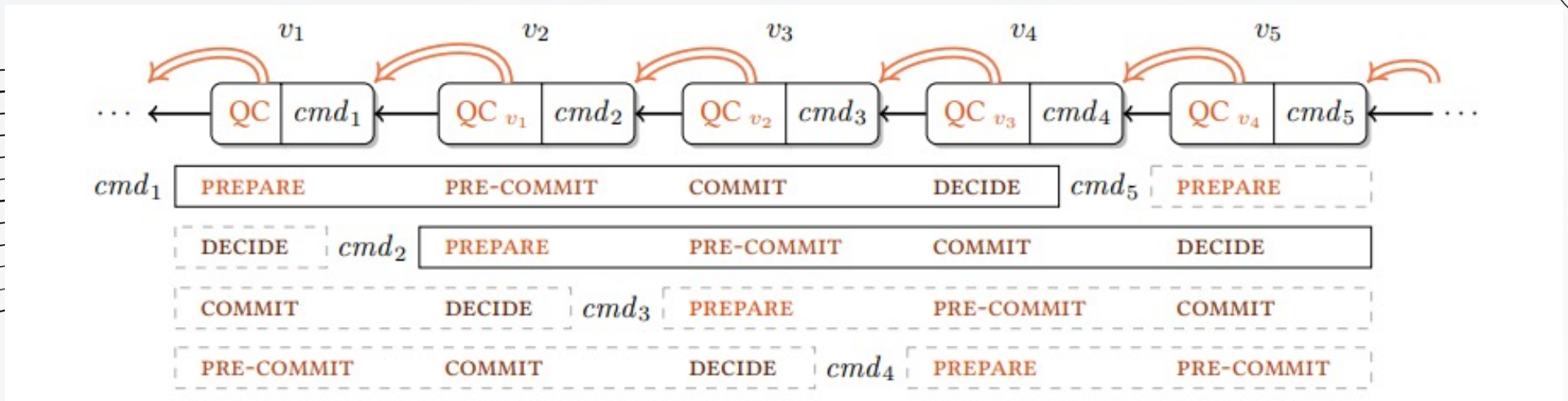
CHAINED HOTSTUFF

- Pipelines process, reduces number of messages
- genericQC: Send to next leader to delegate responsibility



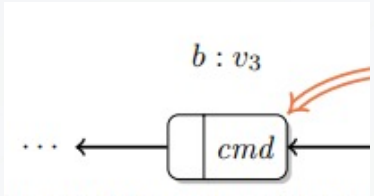
CHAINED HOTSTUFF

- Pipelines process, reduces number of messages
- genericQC: Send to next leader to delegate responsibility



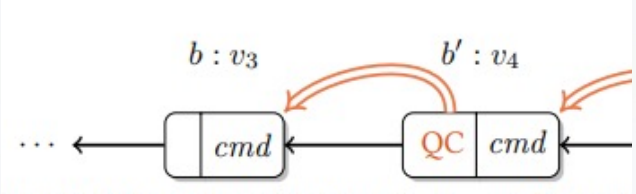
NUMBERED CHAINS

- One-Chain: PREPARE b succeed



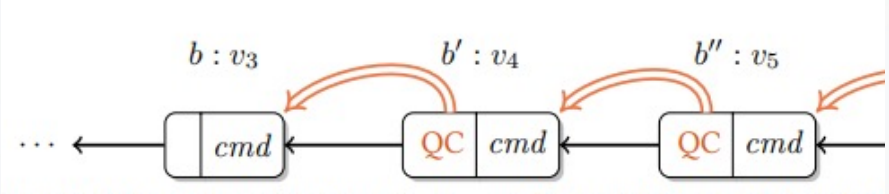
NUMBERED CHAINS

- One-Chain: PREPARE b’ succeed
- Two-Chain: PRE-COMMIT b succeed



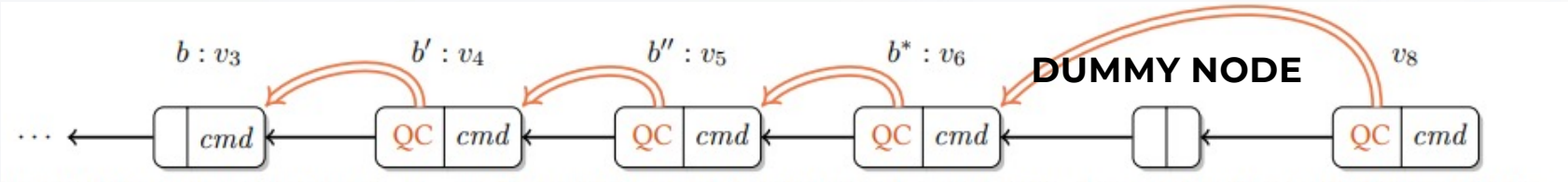
NUMBERED CHAINS

- One-Chain: PREPARE b'' succeed
- Two-Chain: PRE-COMMIT b' succeed
- Three-Chain: COMMIT b succeed



NUMBERED CHAINS

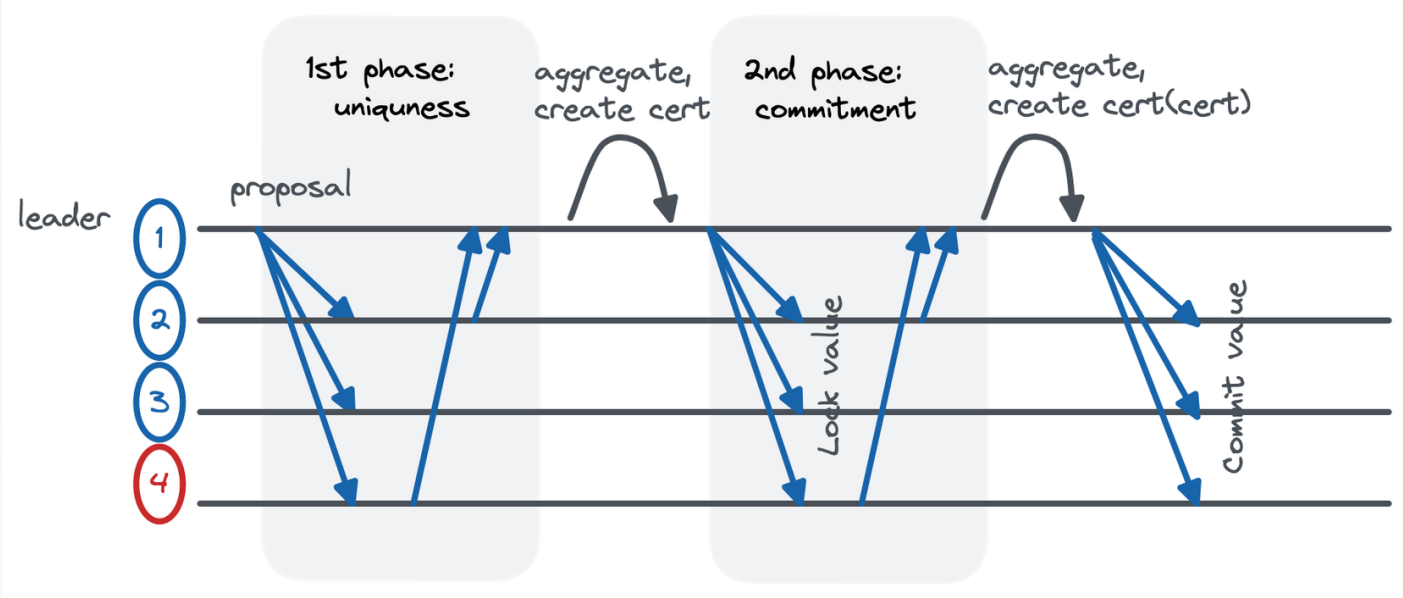
- One/Two/Three - Chain
- If at any point fail due to fault, chain breaks and calls nextView
  - Create dummy node in tree to maintain height



HOTSTUFF-2

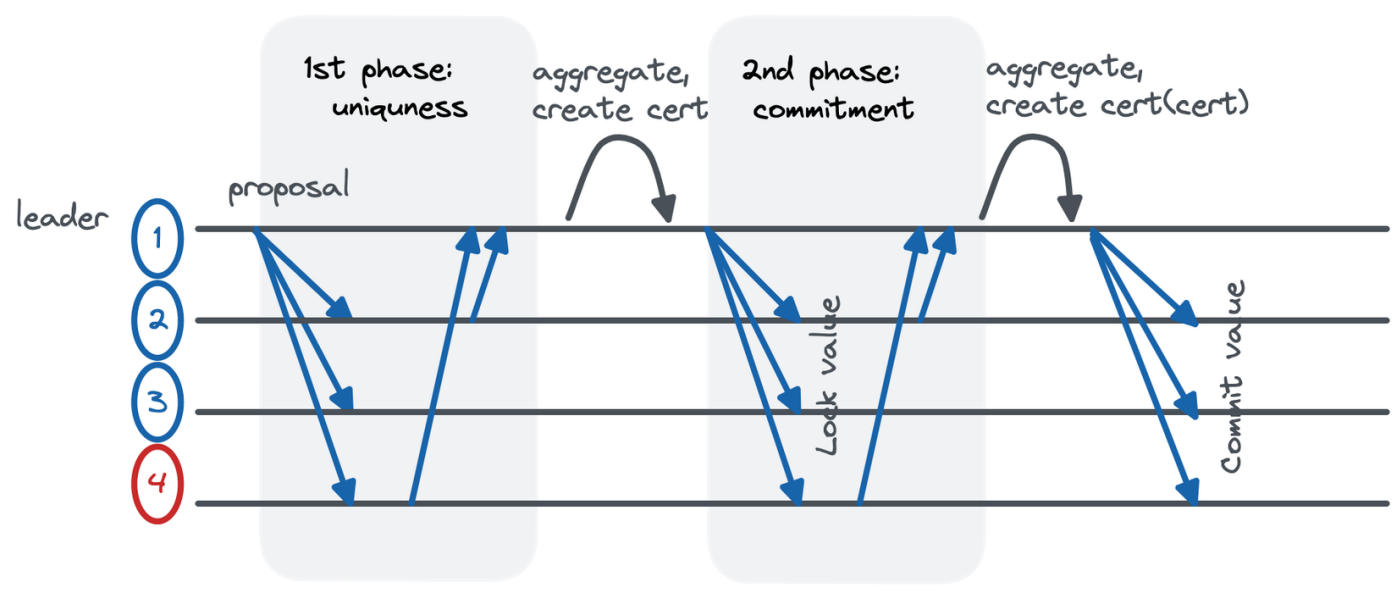
Two phases while maintaining all necessary properties

- No PRE-COMMIT phase, locked on prepareQC



HOTSTUFF-2

Locked nodes will not vote on a conflicting value unless proof is shown that it is safe



PACEMAKER

- New leader obtains QC from previous leader
  - Advance to next view immediately

- New leader doesn’t get QC after time delay
  - Waits delay for nodes to enter next view, then proposes

EVALUATION

HotStuff (2 and 3-phase)
versus BFT-SMaRT

LATENCY/THROUGHPUT

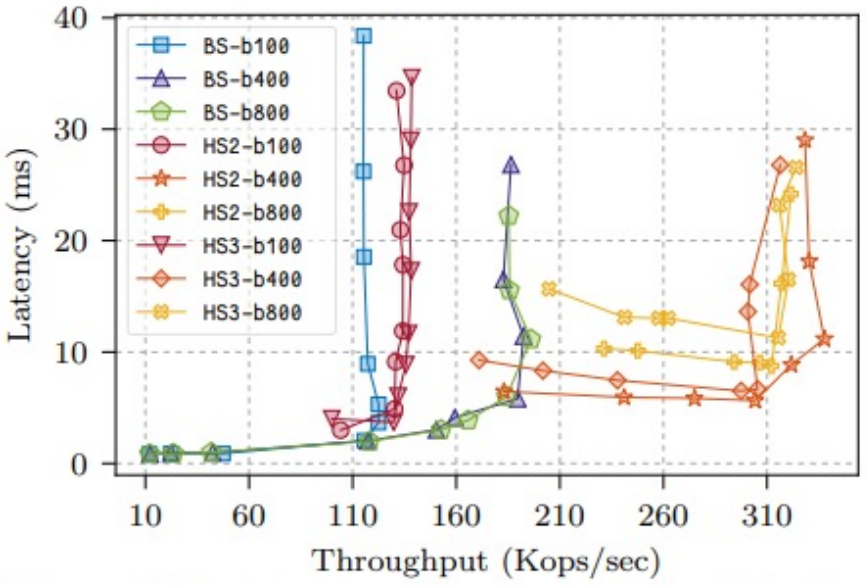


Figure 4: Throughput vs. latency with different choices of batch size, 4 replicas, 0/0 payload.

LATENCY/THROUGHPUT

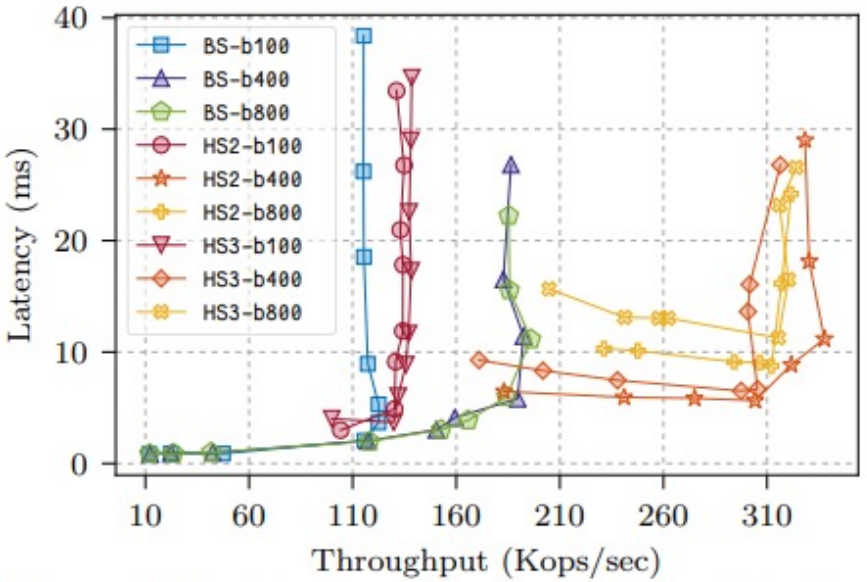


Figure 4: Throughput vs. latency with different choices of batch size, 4 replicas, 0/0 payload.

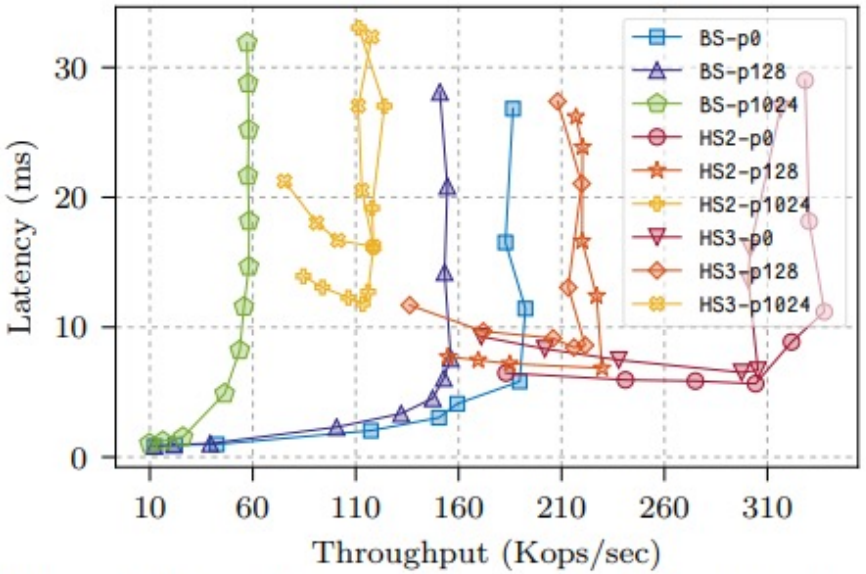
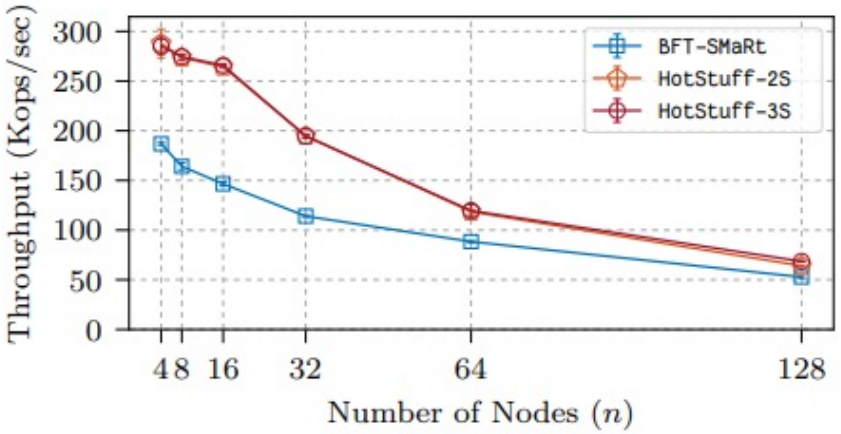
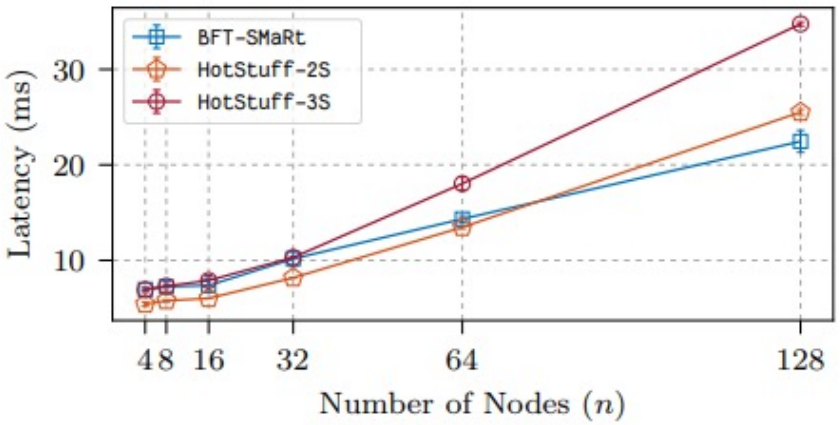


Figure 5: Throughput vs. latency with different choices of payload size, 4 replicas, batch size of 400.

SCALABILITY



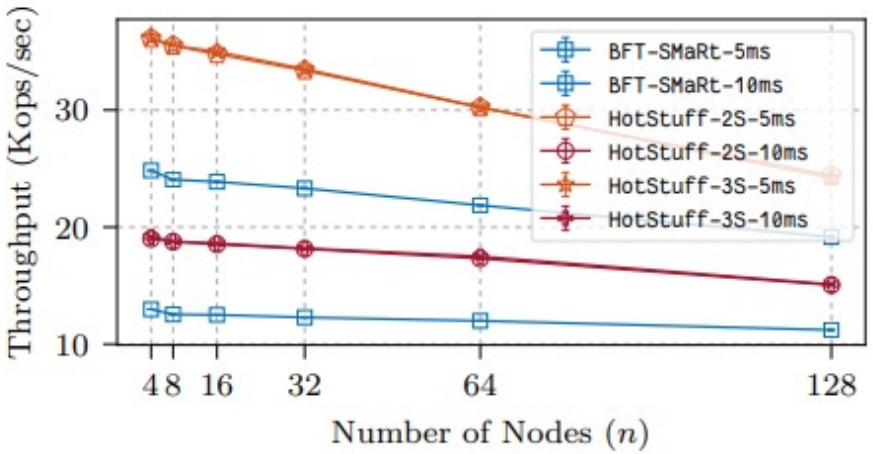
(a) Throughput



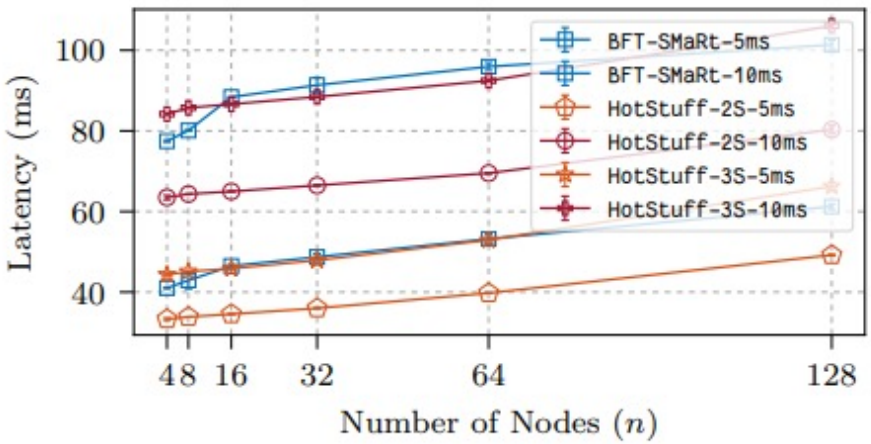
(b) Latency

Figure 6: Scalability with 0/0 payload, batch size of 400.

SCALABILITY



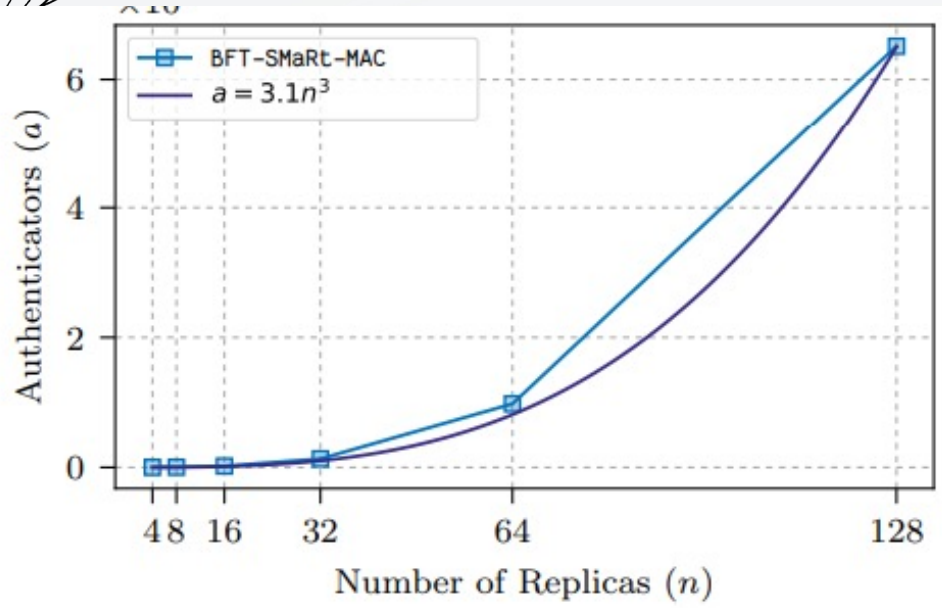
(a) Throughput



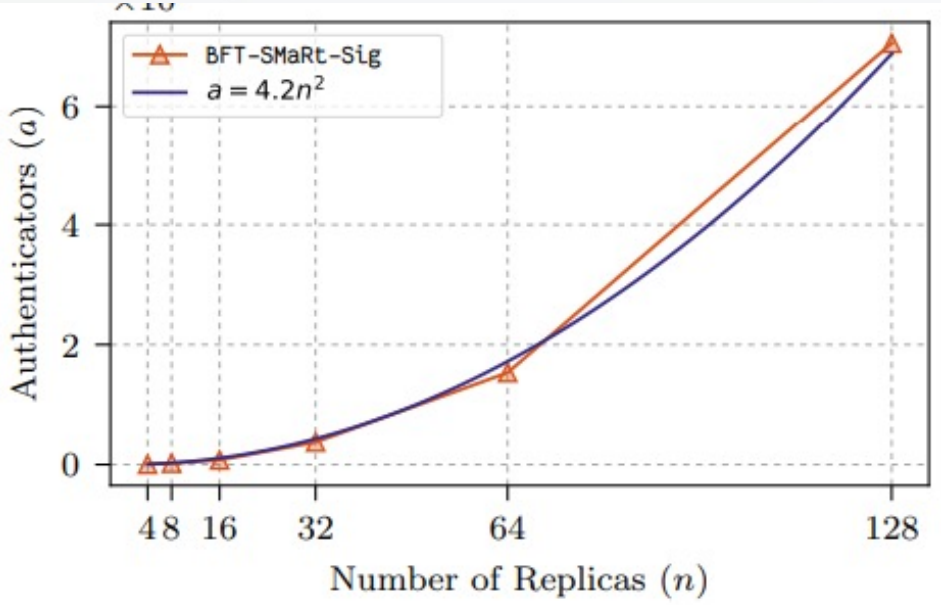
(b) Latency

Figure 8: Scalability for inter-replica latency 5ms $\pm$ 0.5ms or 10ms $\pm$ 1.0ms, with 0/0 payload, batch size of 400.

EXTRA AUTHENTICATORS



(a) MACs



(b) Signatures

Figure 10: Number of extra authenticators used for each BFT-SMaRt view change.

RESOURCES

Main Paper: Yin, Maofan, et al. "HotStuff: BFT consensus in the lens of blockchain." arXiv preprint arXiv:1803.05069 (2018).

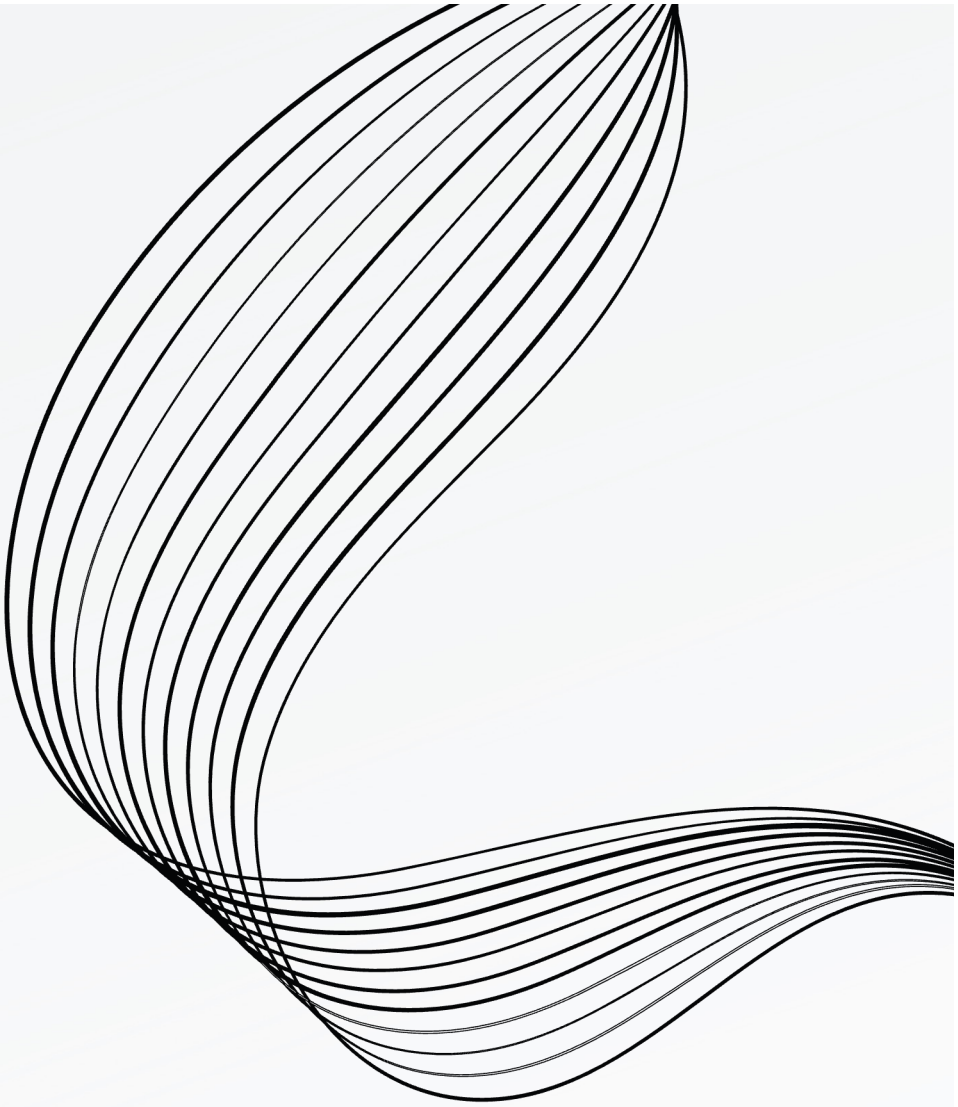
Additional Resources:

Malkhi, Dahlia and Yin, Maofan. "Lessons from HotStuff." Proceedings of the 5th workshop on Advanced tools, programming languages, and PLatforms for Implementing and Evaluating algorithms for Distributed systems. 2023.

Malkhi, Dahlia and Nayak, Kartik. “HotStuff-2: Optimal Two-Phase Responsive BFT.” Cryptology ePrint Archive, 2023.

Zhao, Siyuan, Wu, Yanqi, and Wang, Zheng. "HotStuff-2 vs. HotStuff: The Difference and Advantage." arXiv preprint arXiv:2403.18300 (2024).

QUESTIONS?



Synchronous

Known time bound on messages

- Less realistic
- Doesn’t factor in delay and reordering
- Assume messages are all received within time bound

Asynchronous

Unknown time bound on messages

- More realistic
- Factors in delay and reordering
- Messages delayed by unknown time period
- Complicates correctness and liveness

Partial Synchrony

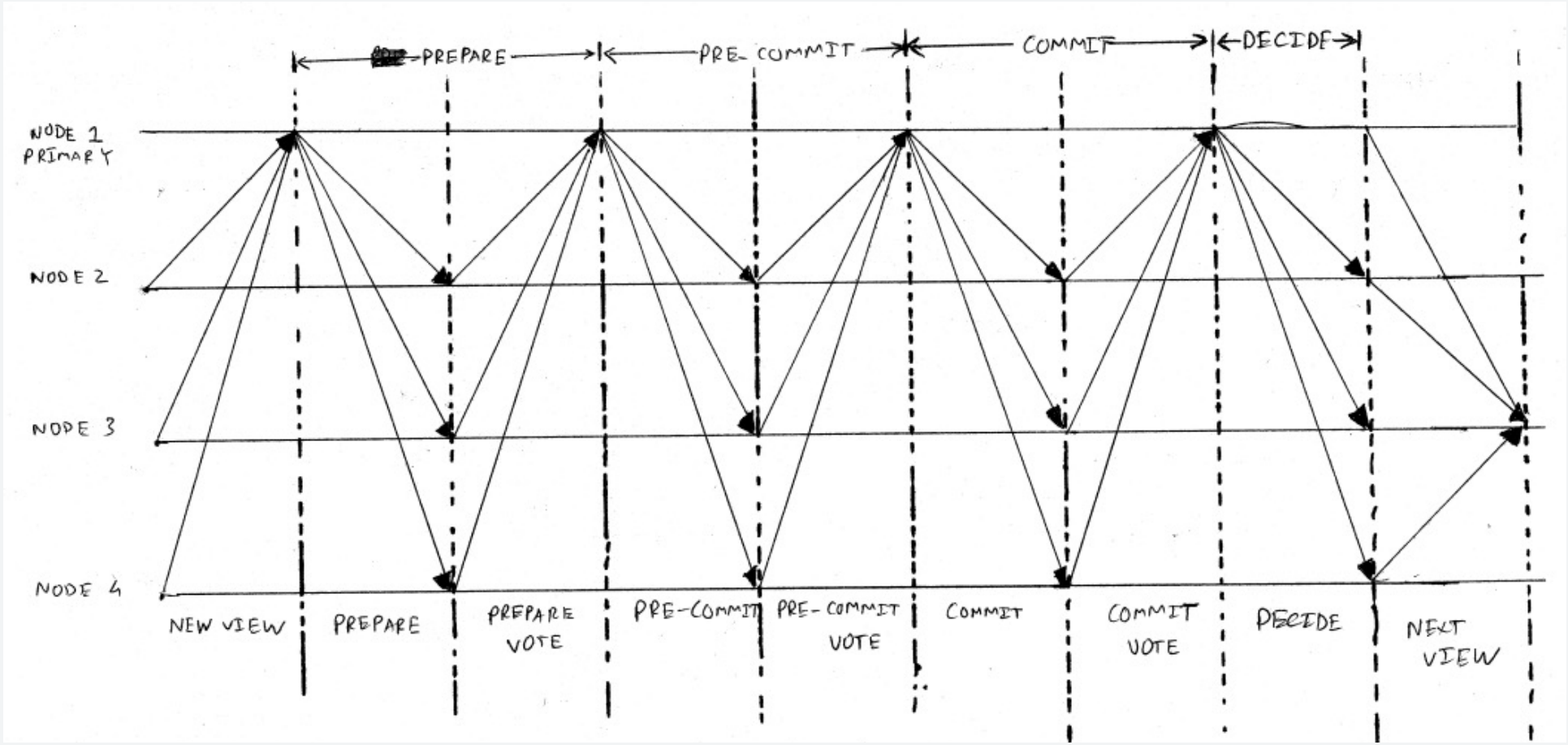
- Middle ground between asynchrony & synchrony
- Asynchronous before GST, synchronous after GST
- Unknown time bounds on communication + processing

GST (Global Stabilization Time)

- Time bounds do not hold at all times
- Holds up to limited time after GST (assume infinite time for simplicity)
- behavior can only be stable and predictable after GST

Precommit Phase

- Replica locks onto the block on separate phase before commit
- Compare own QC with leader QC
- Transition to new leader
  - Picks up from where old leader left off
  - Highest QC


PREPARE
PRE-COMMIT
COMMIT
DECIDE
NODE 1
PRIMARY
NODE 2
NODE 3
NODE 4
NEW VIEW
PREPARE
PREPARE
VOTE
PRE-COMMIT
PRE-COMMIT
VOTE
COMMIT
COMMIT
VOTE
DECIDE
NEXT
VIEW