HotStuff: BFT Consensus with Linearity and Responsiveness

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

Danai Balla

National Technical University of Athens

June 3rd, 2023

Contents

Hotstuff overview

Hotstuff [4] is a BFT consensus protocol that solves the State Machine Replication problem.

- n (fixed) participants, f are faulty, $n \geq 3f + 1$
- *Partially syncronous setting*: at some time after an unknown Global Stabiliziation Time, all messages are delivered after Δ time (unknown to the protocol)
- Protocol works in a succession of *views* ($viewNumber = 1, 2, 3, \dots$), each view has a leader
- Leader communicates with replicas, replicas do not communicate with each other

Hotstuff Performance

- Linear view change:
  - Correct leader: $O(n)$ communication per decision
  - Worst case $O(n^2)$
- Optimistic responsiveness: If no faults, then $O(\delta)$ time per decision, where δ is **actual** network delay
- The costs for a new leader to drive the protocol to consensus is no greater than that for the current leader.

Cryptographic primitives & complexity measure

Cryptographic primitives

- Threshold signatures
  - Common public key & individual private keys
  - A replica can sign a *partial signature*
  - Leader can combine k signatures into one signature
  - Any replica can verify that k distinct valid signatures were used
- Collision resistant hash function

Complexity measure

- Total # of aggregated signatures
- single signature = threshold signature = 1 message

Contents

Protocol outline

- Each replica stores a tree of pending commands as its local data structure.
- During the protocol, a monotonically growing branch becomes committed.
- 4 steps: Prepare, Pre-commit, Commit, Decide
- *Quorum Certificate (QC)*: collection of $n - f$ signatures (for some phase)
- In all phases, a replica waits for a message for a timeout period. If it does not receive any message, it increments *viewNumber* and starts the next view.

The Hotstuff protocol I

- PREPARE — propose commands to run
  - Leader selects a node to extend and creates a new node
  - Replicas check if it is consistent for them (SAFENODE predicate), if it is, they sign and send the signature
- PRE-COMMIT — inform that for $n - f$ replicas commands are consistent
  - If $n - f$ replicas can accept, leader forms a QC and sends it to replicas
  - Replicas hash the node, sign, and send signature to leader

The Hotstuff protocol II

- Commit — inform that $n - f$ replicas are informed
  - If leader has received $n - f$ signatures, makes QC, sends it to replicas
  - Replicas sign, send the signature and become *locked* on this QC
- Decide — tell replicas to move to next view
  - If leader has received $n - f$ signatures, makes QC and sends it to replicas
  - Replicas execute the commands and proceed to next view

Hotstuff communication diagram

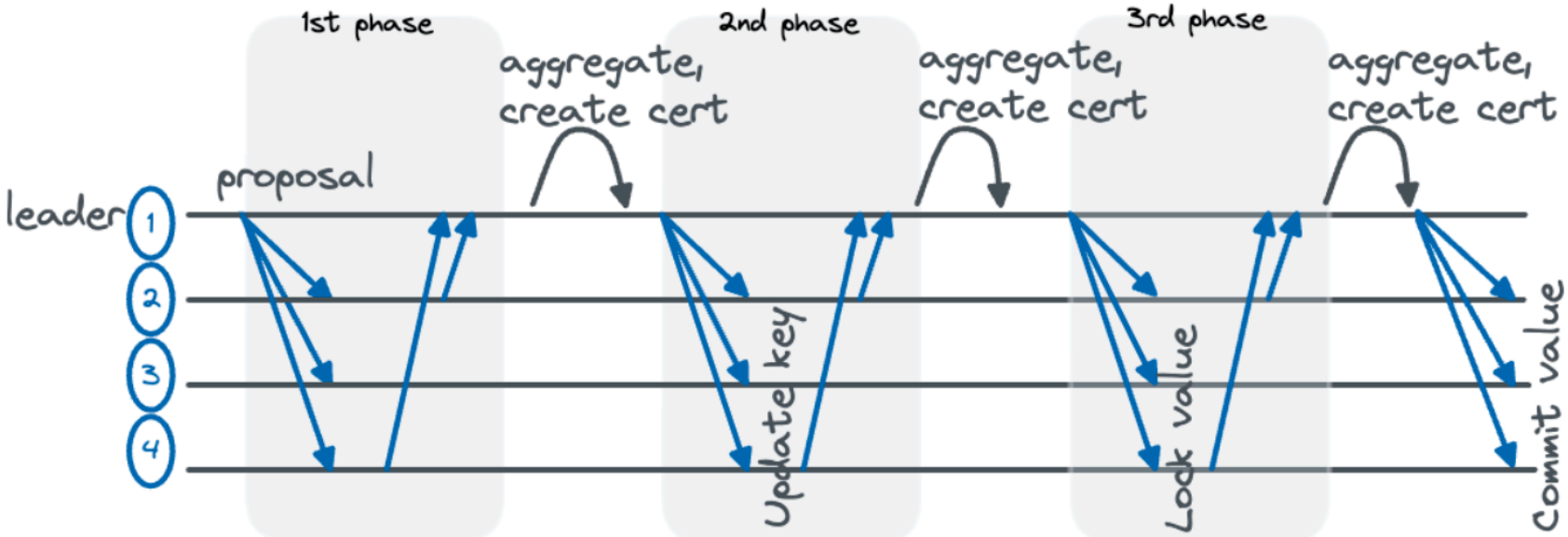

Figure 1: Hotstuff 3-step protocol. Figure taken from [6].

Node selection & SafeNode predicate

Node selection by leader

Out of all the nodes that have a valid *prepareQC*, leader selects to extend the one that has highest *viewNumber*.

SafeNode predicate

A replica can accept a proposal if

- proposal extends their locked node (safety), or
- proposal extends a node whose *prepareQC* has a higher *viewNumber* than *lockedQC* (liveness).

Safety & Liveness

- Safety: Replicas do not commit conflicting nodes
  (we require $2f + 1$ signatures)
- Liveness: If all replicas remain in a view for enough time and the leader is correct, a decision will be driven.

Contents

Performance Comparison

Protocol	Authenticator complexity			Optimistic responsiveness
	Correct Leader	Leader Failure	Worst case	
PBFT [1]	$O(n^2)$	$O(n^3)$	$O(fn^3)$	yes
Tendermint[2][1]	$O(n^2)$	$O(n^2)$	$O(fn^2)$	no
Tendermint[1][2]	$O(n)$	$O(n)$	$O(fn)$	no
Hotstuff	$O(n)$	$O(n)$	$O(fn)$	yes

Table 1: Performance comparison after GST, taken from [4].

[1] same performance for Casper[3]

[2] combining signatures to threshold signatures

Further Reading

- In Hotstuff paper [4]
  - Chained Hotstuff
  - *pacemaker* mechanism
- Further reading:
  - Tendermint [2]: 2-phase, linear, not responsive. Published before Hotstuff.
  - Jolteon [5]: 2 phase, optimistically linear ($O(n^2)$ for incorrect leader), responsive.

Chained Hotstuff

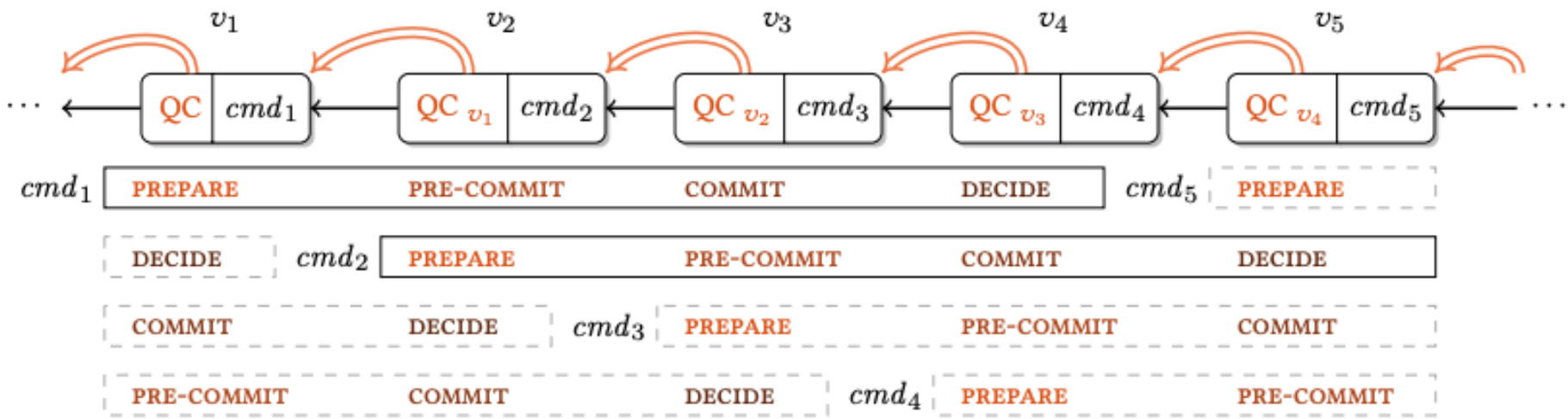

Figure 2: Chained Hotstuff. A QC can serve in different purposes simultaneously. Figure taken from [4].

Thank you

:)

Contents

Bibliography I

[1] Miguel Castro, Barbara Liskov, et al. "Practical byzantine fault tolerance". In: *OsDI*. Vol. 99. 1999. 1999, pp. 173–186.

[2] Ethan Buchman. "Tendermint: Byzantine fault tolerance in the age of blockchains". PhD thesis. University of Guelph, 2016.

[3] Vitalik Buterin and Virgil Griffith. "Casper the friendly finality gadget". In: *arXiv preprint arXiv:1710.09437* (2017).

[4] Maofan Yin et al. "HotStuff: BFT consensus with linearity and responsiveness". In: *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*. 2019, pp. 347–356.

[5] Rati Gelashvili et al. "Jolteon and ditto: Network-adaptive efficient consensus with asynchronous fallback". In: *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*. Springer. 2022, pp. 296–315.

Bibliography II

[6] Dahlia Malkhi and Maofan Yin. “Lessons from HotStuff”. In: *arXiv preprint arXiv:2305.13556* (2023).