Brief Announcement: It's not easy to relax: liveness in chained BFT protocols

Ittai Abraham
VMware Research, Herzliya, Israel

Natacha Crooks
UC Berkeley, CA, USA

Neil Giridharan[1]
UC Berkeley, CA, USA

Heidi Howard
Microsoft Research Cambridge, Cambridge, UK

Florian Suri-Payer
Cornell University, Ithaca, NY, USA

Abstract

Modern *chained* BFT SMR protocols have poor liveness under failures as they require multiple consecutive honest leaders to commit a single block. Siesta, our proposed new BFT SMR protocol, is instead able to commit a block that spans multiple non-consecutive honest leaders. Siesta reduces the expected commit latency of HotStuff by a factor of three under failures, and the worst-case latency by a factor of eight.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Consensus, blockchain, BFT

Digital Object Identifier 10.4230/LIPIcs.DISC.2022.39

Related Version *Full Version*: `https://arxiv.org/abs/2205.11652` [1]

1 Introduction

Blockchain systems add two constraints over traditional BFT SMR protocols: 1) only valid operations should be committed and 2) operation ordering should be fair. To address these concerns, recent blockchain protocols such as Casper FFG [2], HotStuff [5], DiemBFTv4 [4], and Fast-Hotstuff [3] are structured around two key building blocks: Chaining and Leader-Speaks-Once (LSO).

Chaining. To ensure validity, existing protocols evaluate proposed blocks against the full sequential history of the chain and check application-level preconditions are satisfied. Most BFT protocols require a (worst-case) minimum of two voting rounds (henceforth *phases*). Each voting phase aims to establish a *quorum certificate (QC)* by collecting a set of signed votes from a majority of honest replicas. Blockchain systems *pipeline* the voting phases of consecutive proposals to avoid redundant coordination and cryptography: the system can use the quorum certificate of the second voting phase of block i to certify the first phase of block $i + 1$. Each block requires (on average) generating and verifying the signatures of a single QC. This is especially important for large participant sets as QC sizes grow linearly with the number of replicas, increasing cryptographic costs.

[1] Lead author.

© Ittai Abraham, Natacha Crooks, Neil Giridharan, Heidi Howard, and Florian Suri-Payer;
licensed under Creative Commons License CC-BY 4.0

Leader-Speaks-Once (LSO). To minimize fairness concerns associated with leader-based solutions and to decrease the influence of adaptive adversaries (adversaries who control the network), BFT protocols targeted at blockchains adopt a *leader-speak-once* (LSO) model. In LSO, each leader proposes and certifies a single block after which the leader is immediately rotated out as part of a new *view*. Electing a different leader per block limits the leader's influence; it can manipulate transactions in the proposed block only. Traditional BFT protocols, in contrast, adopt a *stable-leader* paradigm in which leaders are only replaced if they fail to make progress through *view change*. View changes are intentionally costly procedures and assumed to be infrequent. Complex view changes allow for a simpler and more efficient failure-free steady state. LSO protocols instead perform view changes proactively in each round, and thus are required to place the view change on the critical path of the system.

While a joint chained leader-speak-once (CLSO) approach is desirable for blockchains, the combination of these two properties introduces a new liveness concern. We show in our full paper [1] that faulty leaders can greatly reduce the throughput of *any* CLSO protocol. Worse, this attack does not require any explicit equivocation; it suffices for a faulty leader to delay responding, making it harder to detect. The root cause of the problem is simple: CLSO protocols require the formation of k QCs in *consecutive* views in order to commit a block (where $k \in \{2, 3\}$ depending on protocol). In the remainder of the paper, we refer to this constraint as the *consecutive honest leader condition (CHLC)*.

To the best of our knowledge, the aforementioned liveness concern is present in *all* CLSO protocols today, and thus represents a significant exploit opportunity for a Byzantine attacker. Even without malicious participants, expected network asynchrony can cause spurious view-changes, precluding the system from committing blocks. Consequently, this paper asks: is CHLC fundamental, or can it be relaxed?

2 Results

▶ **Definition 1** (Gap-Tolerance). *A BFT protocol achieves gap-tolerance if it requires k QCs in non-consecutive views to commit an operation.*

We observe that CLSO protocols *can* achieve gap-tolerance in some settings: commitment with non-consecutive QCs is possible when failures comprise of omission faults only. This observation can be coupled with the active detection of commission failures to determine when to (and when not to) relax the CHLC constraint. We prove the following theorem:

▶ **Theorem 2.** *There exists a CLSO protocol that is resilient to $f < n/3$ Byzantine replicas, is safe under asynchrony, and achieves gap-tolerance when failures are omission only.*

After GST and view synchronization, if views $v < v' < v''$ have honest leaders, and block B is proposed in view v, then either block B will be committed in view v'' or a previously undetected faulty replica will be detected and slashed (excluded from any future participation).

In this paper we propose Siesta (complete description is in our full paper [1]), a new consensus protocol that satisfies Theorem 2. Siesta either (i) allows blocks to commit in non-consecutive views or (ii) immediately and reliably detects equivocation and slashes (punishes and removes) a malicious leader.

Two fundamental ideas lie at the center of Siesta:

- *No-QC Proofs.* It is safe to commit a block in non-consecutive views as long as one can prove that no QC for a conflicting block could have formed in between views. Siesta carefully designs such proofs to be efficient.

- *Equivocation proofs.* Many BFT protocols do not include phase one messages in view changes as conflicting phase one messages can exist in the presence of malicious leaders. Although these messages cannot be used to successfully commit a block, we observe that they can be used as an explicit *proof of equivocation* to hold perpetrators accountable.

2.1 Performance Results and Complexity

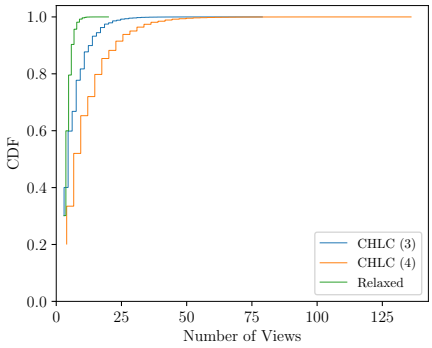

Figure 1 CDF of the number of views needed to commit an operation n = 100.

We quantify the performance gains made possible by relaxing the consecutive honest leader condition (CHLC). We compare Siesta to 1) two-phase CLSO protocols and 2) three-phase CLSO protocols. We assume a random leader election strategy; experiments with round-robin election convey similar results. We simulate each protocol and report how many rounds were necessary to satisfy each protocol's commit rule.

In CLSO protocols, the number of rounds directly influences both latency and throughput. If a round has latency x, then commit latency for an operation will be $x * rounds$ while throughput is calculated by dividing the batch size by the expected commit latency. We write CHLC(4) for protocols requiring four consecutive honest leaders, CHLC(3) for three consecutive honest leaders, and finally Relaxed for our own protocol Siesta. Figure 1 shows the resulting commit latency CDF. Siesta achieves an expected commit latency of 4.5 rounds; CHLC(3) requires ≈ 7 rounds. CHLC(4) has worst the expected performance, taking 12 rounds to commit. Worst-case commit latency is especially interesting: Siesta has relatively low worst-case latency, with 18 rounds, while CHLC(3) protocols have a worst-case commit time of 76 rounds. CHLC(4) has seven times worst latency, with a worst-case commit time of 129 rounds.

References

1 Ittai Abraham, Natacha Crooks, Neil Giridharan, Heidi Howard, and Florian Suri-Payer. It's not easy to relax: liveness in chained bft protocols, 2022. doi:10.48550/arXiv.2205.11652.
2 Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint*, 2017. doi:10.48550/arXiv.1710.09437.
3 Mohammad M. Jalalzai, Jianyu Niu, and Chen Feng. Fast-hotstuff: A fast and resilient hotstuff protocol, 2020. doi:10.48550/arXiv.2010.11454.
4 The Diem Team. DiemBFT v4: State machine replication in the diem blockchain, 2021.
5 Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, 2019.