Faster Leader-Based Consensus via Certificates of Exclusivity and Overlapping Iterations

Matthieu Rambaud

Télécom Paris

Abstract. Blockchain consensus protocols, also known as BFT state-machine replication, enable a system of n players to decide an ever-growing chain of blocks. We consider partial synchrony: after an unknown time GST, messages sent by honest players are delivered within an unknown actual delay δ, and a known bound Δ satisfies $\delta \leq \Delta$. This setting imposes $t < n/3$ corruptions. We study leader-based consensus, a class with the smallest known latency metrics when sufficiently many designated leaders behave honestly. We consider the mainstream metric of *expected latency in the view-based sense:* for a transaction known to all honest players having entered a post-GST iteration, this is the expected time until that transaction appears in a decided block, under independent random leaders. We introduce Hamster, a rotating-leader consensus protocol, which reduces this expected latency to $\Delta + 4\delta$, down from the previous $1.5\Delta + 3.5\delta$ bound achieved by Simplex (Chan-Pass, TCC'23) for the same view-based metric. Hamster also brings this latency further down to $\Delta + 3.5\delta$ for adversaries that do not get publicly caught equivocating.
The main technical novelty is a certificate of exclusivity for a block B. It is shown by the next leader as evidence that players can safely vote for a child of B. A certificate of exclusivity is an interpolation between a quorum certificate and a timeout certificate, in that it is formed from a mix of votes for a unique block B and complaint votes, proving that B is the only non-dummy block that can still obtain a decision certificate for that iteration.
The other novelty of Hamster is the use of overlapping iterations: after a process has supported a newer proposal, the protocol may still allow it to support a safe proposal of an earlier iteration. This overlap is what allows old honest proposals to be decided in time despite players advancing iterations faster.
We also analyze another metric, called the *pessimistic block proposal time*, which is the time during which a bad leader can delay the proposal of a transaction in a block which will be decided. Hamster achieves a pessimistic block proposal time of $2\Delta + 2\delta$, refined to $2\Delta + \delta$ for leaders not caught equivocating, down from $3\Delta + \delta$ for Simplex.

1 Introduction

We consider a system of $n = 3t + 1$ players $\mathcal{P} = \{P_1, \ldots, P_n\}$ connected by authenticated public channels. A static adversary $\mathcal{A}$ corrupts at most t players, and the remaining $2t + 1$ players are honest. We work in the partial synchrony model of DLS [1], in which $\mathcal{A}$ chooses a finite global stabilization time GST such that all messages sent after GST between honest players are delivered within the known bound Δ. For a fixed execution, $\delta \leq \Delta$ denotes the largest actual delay taken by such a message after GST.

A blockchain consensus protocol lets players decide an ever-growing chain of blocks. For any two decided chains, one must be a prefix of the other; every transaction input to an honest player must eventually appear in all decided chains. We focus on rotating-leader partially synchronous protocols. In each iteration, dubbed a *slot* $s = 1, 2, \ldots$, a designated leader L_s proposes a block chained to a parent block. Such protocols are PBFT [2], HotStuff [3], ICC [4], Simplex [5], DispersedSimplex [6], Moonshot [7], and TetraBFT [8], but also DAG-based protocols such as BBCA [9], Sailfish [10], and Sailfish++ [11], since they also have a single designated leader per iteration.

The best protocols in this class achieve proposal confirmation time 3δ, meaning that if an honest leader proposes a valid block after GST, the block is decided within 3δ. This 3δ value is optimal [12]. The remaining bottleneck is latency when leaders are randomly corrupt. In Simplex, a faulty or silent leader may consume up to $3\Delta + \delta$ time before the next leader can propose a block which will get decided. We formally define this time, below, as the *pessimistic block proposal time*.

As emphasized in Moonshot [7, Appendix D], compressing this delay was up to now an open challenge. Indeed if players set a timer of only 2Δ before complaining when they observe no progress, they suggest an attack enabling a corrupt leader to force players to wait anyway 3Δ before moving to the next view. The attack, which we call the "last-minute selective-proposal", consists in sending a proposal to only one honest player just before its 2Δ timer expires. As a result the player will issue a vote and wait a further Δ before complaining, preventing the players from reaching the desired quorum of $2t + 1$ complaints to advance to the next iteration. This attack is particularly annoying because the leader *need not openly cheat.*

Contributions. Hamster brings the pessimistic block proposal time down to $2\Delta + 2\delta$, and even to $2\Delta + \delta$ for all adversaries not publicly caught equivocating. As a result, Hamster achieves the best-known expected latency in the view-based sense, namely $\Delta + 4\delta$, and even $\Delta + 3.5\delta$ for adversaries that are not publicly caught equivocating, down from $1.5\Delta + 3.5\delta$ for Simplex [5], as discussed in Remark 1.

Hamster allows players to set a shorter timer of 2Δ before complaining, while avoiding the "last-minute selective proposal" attack thanks to two ingredients added to Simplex. The first ingredient is a new certificate called a ***certificate of exclusivity***: a vote/complaint certificate that is weaker than a quorum certificate but still strong enough for safety. The only way for a leader to slow down the creation of such certificates is to propose different blocks to different players, in which case it is rapidly caught because players disseminate signed leader proposals as equivocation evidence.

However, Simplex would lose its liveness if those two ingredients were naively added without further modification. Indeed the adversary could inject adversarial early complaints, triggering players to quickly obtain a certificate of exclusivity and thus advance to the next iteration before the honest block gets a chance to be certified. To counter this, we introduce another relaxation to the existing designs [2, 3, 5, 6, 7], in which voting for blocks of previous iterations was strictly forbidden. We call our relaxation ***overlapping iterations***, meaning that players are now allowed to vote for blocks of earlier slots than their current iteration, as highlighted in red in the protocol.

Historical notes. Hamster was submitted to Podc'2025, then shortly after we informed the chair of the modification necessary to allow overlapping iterations. We refer to Section A for more details and for the further editorial modifications done to Hamster following the Podc'2025 reviews. Much more recently there has been the HumBird preprint [13], which states in Theorem 8 an expected latency in the view-based sense of $\frac{28}{9}\Delta + 3\delta$ for its leader-speaks-twice partially synchronous variant. Since $\delta \leq \Delta$, this bound is larger than Hamster's $\Delta + 4\delta$ under the metric and assumptions of Theorem 4.

2 Model and Definitions

To multicast a message is to send it to all players. Multicast does not by itself provide consistency or delivery guarantees when the sender is corrupt. We write $\langle m \rangle_i$ for the message m accompanied by a digital signature of P_i on m. Standard unforgeability guarantees that, except with negligible probability, no honest player accepts $\langle m \rangle_i$ unless P_i signed m.

Definition 1 (Blockchain consensus). *In a blockchain consensus protocol, players receive input transactions and output, dubbed* decide*, chains of blocks, dubbed blockchains. It guarantees:*

Consistency. *For any two blockchains ever decided, one is a prefix of the other.*
Liveness. *Every transaction input to an honest player is eventually included in all blockchains decided by honest players.*

- ***proposal confirmation time.*** A protocol has proposal confirmation time $T(\delta)$ if an honest leader's proposal made after GST is decided by all honest players within time $T(\delta)$.

- ***expected latency in the view-based sense.*** We quote Simplex [5, p4] almost verbatim. Essentially all protocols in the literature consider a "view-based liveness" notion that is tailored for rotating leader based protocols. Following the literature, we will also consider this notion of expected "view-based" liveness. Suppose that the protocol proceeds in incrementing "views" or "iterations" $s := 1, 2, \ldots$ where each view s is assigned a single leader L_s. Fix some view number s ahead of time, and suppose that a transaction is provided to every honest player just before the time they enter view s. The expected latency in the view-based sense is the time that it takes (in expectation) from this point for the transaction to be finalized, assuming the leaders are independently honest with probability $(n-t)/n$.
- The ***pessimistic block proposal time.*** Fix some view number s ahead of time with a corrupt leader L_s, such that the next leader L_{s+1} is honest, and suppose that a transaction is provided to every honest player at the time they enter view s. The pessimistic block proposal time is the time that it takes, from this point, for the transaction to be proposed by L_{s+1} in a block which will get decided.

Remark 1 (Metric used for the comparison with Simplex). Simplex states and proves its Theorem 3.4 [5] as an upper bound of $3.5\delta + 1.5\Delta$ for the expected latency in the view-based sense metric. Hamster brings this same view-based quantity down to $\Delta + 4\delta$ and, in the non-public-equivocation case, to $\Delta + 3.5\delta$.

The Simplex introduction also uses the informal phrase "expected pessimistic confirmation time". We do not need to interpret that phrase here: all comparisons in this paper are to the formal view-based bound proved in Simplex Theorem 3.4, as well as with the pessimistic block proposal time proven in Simplex Lemma 3.6. If one wants a stronger transaction-arrival metric in which a transaction may arrive just after an honest leader has already sent its proposal, one should $2/3.3\delta$ as additional waiting term to the expected latency in the view-based sense of both Simplex and Hamster.

3 Protocol Data Objects and Local State

For simplicity we describe Hamster assuming players use the same naive dissemination layer as in Simplex, i.e., that players multicast all blocks and certificates received. We refer to Section 7 for how to divide this cost by n using the standard techniques of DispersedSimplex [6], with a tweak.

3.1 Blocks and blockchains

The genesis block is B_0. For $s \geq 1$, a slot-s block B_s is either the dummy block $\perp_s$, dubbed a *complaint*, or a tuple (s, s', h, txs) where $s' < s$, h commits to the prefix chain ending at slot s', and txs is a sequence of transactions. A dummy block contains no transactions and has no parent.

A blockchain of height s is a sequence $(B_0, B_1, \ldots, B_s)$ indexed by slots, where each non-dummy B_j points to the prefix chain preceding it. Collision resistance of the hash function ensures that a non-dummy block determines a unique prefix chain except with negligible probability.

3.2 Votes, quorum certificates, and decisions

A quorum certificate on a slot-s block B_s, denoted $\mathcal{C}_s(B_s)$, is a set of signed messages $\langle \mathsf{vote}, B_s \rangle_i$ from at least $2t+1$ distinct players. A block with a quorum certificate is called certified. Quorum intersection implies that two conflicting non-dummy slot-s blocks cannot both have quorum certificates, except with negligible probability.

A dummy block may be certified: $\mathcal{C}_s(\perp_s)$ is a quorum certificate made of "dummy votes", which we dub "complaint votes". By Lemma 1, a dummy certificate $\mathcal{C}_s(\perp_s)$ prevents any non-dummy slot-s block from obtaining a slot-s decision certificate.

A decision wildcard for slot s, denoted $\mathsf{DecisionWildcard}_s$, is a set of signed messages of the form $\langle \mathsf{decision\text{-}vote}, s\rangle_i$ from at least $2t+1$ distinct players. A decision certificate for a non-dummy slot-s block B_s is

$$\mathsf{DecCert}_s(B_s) = (\mathcal{C}_s(B_s), \mathsf{DecisionWildcard}_s).$$

Upon obtaining $\mathsf{DecCert}_s(B_s)$ for a non-dummy block B_s as well as the chain ending at B_s, then a player decides the whole chain.

3.3 Certificates of exclusivity

Definition 2 (Certificate of exclusivity). *For a non-dummy slot-s block B_s, a certificate of exclusivity $\mathsf{CertExcl}_s(B_s)$ is a set of signed messages from at least $2t+1$ distinct players, where each signed message is one of the following two forms: a vote for the unique block B_s or a complaint vote for the slot, written as a vote for $\perp_s$:*

$$\langle \mathsf{vote}, B_s\rangle_i \qquad \textit{or} \qquad \langle \mathsf{vote}, \perp_s\rangle_i.$$

A quorum certificate $\mathcal{C}_s(B_s)$ is therefore a $\mathsf{CertExcl}_s(B_s)$. For a non-dummy B_s, a certificate of exclusivity is required to contain at least one vote for B_s; if all $2t+1$ messages are complaint votes, the object is instead the certificate $\mathcal{C}_s(\perp_s)$ for the dummy slot-s block.

Abspoel, Rambaud and Tonkikh [14] build related proofs of exclusivity with respect to a fast path; by contrast, Hamster's certificates of exclusivity apply to the normal path.

A safely proposable blockchain $(B_0, \ldots, B_s, S)$ consists of a blockchain $(B_0, \ldots, B_s)$ and one certificate S_j for each $j \in \{1, \ldots, s\}$, where S_j is $\mathcal{C}_j(\perp_j)$ if $B_j = \perp_j$, and is either $\mathcal{C}_j(B_j)$ or $\mathsf{CertExcl}_j(B_j)$ if B_j is non-dummy. The leader of slot $s+1$ may propose a child of the top non-dummy block of such a chain.

3.4 Local state

Each player P_i maintains a current iteration number, s_{cur}, initialized to 1. It maintains a pool of certificates and signed proposals it has seen, together with local variables

$$\begin{aligned} &s_{\mathsf{cur}}, \qquad \mathsf{voted}[s] \in \{\emptyset\} \cup \{\text{non-dummy slot-}s\text{ blocks}\},\\ &\mathsf{dummyVoted}[s], \mathsf{decisionVoted}[s] \in \{0,1\}. \end{aligned}$$

Initially, all $\mathsf{voted}[s] = \emptyset$, $\mathsf{dummyVoted}[s] = 0$, and $\mathsf{decisionVoted}[s] = 0$. The array $\mathsf{voted}[s]$ records only the player's non-dummy vote for slot s, if any; dummy votes are tracked separately by $\mathsf{dummyVoted}[s]$. This enables to keep the decision-vote condition explicit. Once either Boolean flag is set to 1, it is never reset. A vote may concern an older slot, so the slot index of a vote need not be equal to s_{cur}.

3.5 Local tree picture

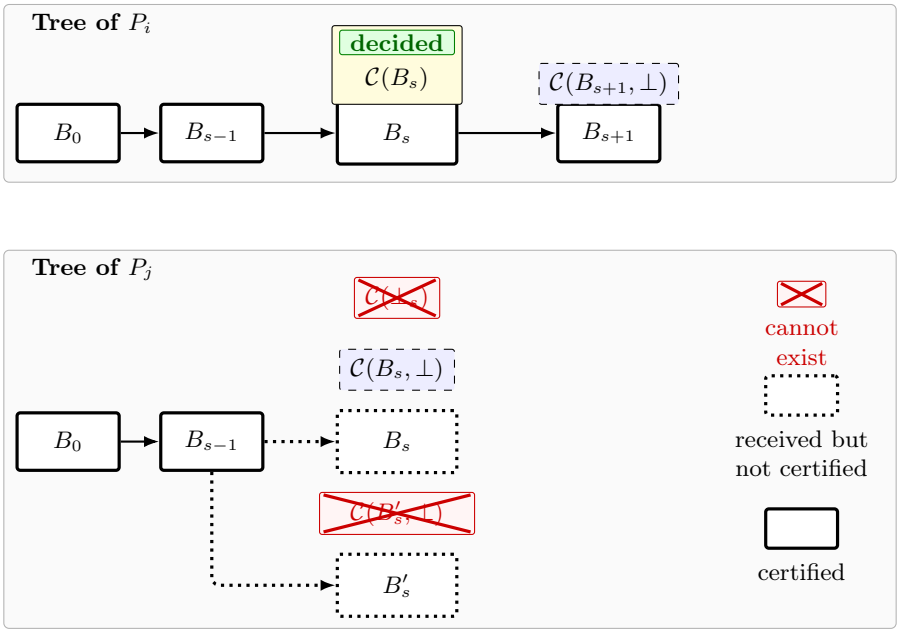


Fig. 1. Two local block trees can contain different evidence for the same slot without containing two full quorum certificates for conflicting blocks. Certificate labels are placed above the block or dummy marker to which they refer. The decided stamp is placed on the top side of the $\mathcal{C}(B_s)$ box, meaning that P_i received $2t+1$ $\mathsf{DecisionWildcard}_s$'s for slot s. The certificates shown are all compatible with the decision certificate for B_s: a full quorum certificate $\mathcal{C}(B_s)$, a certificate of exclusivity $\mathcal{C}(B_s, \perp)$, and certificate of exclusivity for a child $\mathcal{C}(B_{s+1}, \perp)$. A conflicting certificate of exclusivity $\mathcal{C}(B'_s, \perp)$, nor a full complaint quorum $\mathcal{C}(\perp_s)$, cannot possibly coexist with the decision certificate of B_s.

The figure shows two players' local block trees and the certificate information that may be known at the same time. For the same slot s, one local view may contain the full certificate $\mathcal{C}(B_s)$ while another contains the certificate of exclusivity $\mathcal{C}(B_s, \perp)$; no local view can also contain a certificate of exclusivity for a conflicting slot-s block. What guarantees the safety of a child of the decided block B_s is that *no tree can possibly contain a certificate of exclusivity for any other block, $\mathcal{C}(B'_s, \perp)$, nor a quorum certificate for the dummy block $\mathcal{C}(\perp_s)$*. A certificate of exclusivity such as $\mathcal{C}(B_{s+1}, \perp)$ in another local tree is compatible with the decision of B_{s+1}, whereas the existence of the crossed-out certificates is incompatible with the decision of B_s.

4 Hamster Protocol

The protocol instructions proceed by iterations $s = 1, 2, \ldots$. A player can advance to iteration $s_{\mathsf{cur}} + 1$ and later still cast a permitted vote for an older slot $j \leq s_{\mathsf{cur}}$, so the voting rule is stated per slot rather than per current iteration.

The protocol enforces the following voting rule. An honest player votes for at most one non-dummy block per slot. A dummy vote does not erase the record of a non-dummy vote, and a timeout does not create a dummy vote after the player has already non-dummy-voted in that slot. It sends a decision-vote for a slot only if it has not sent a complaint vote for that slot and its recorded non-dummy vote for the slot is for the certified block being decision-voted.

When P_i enters iteration s_{cur}, it starts a local timer $T_{s_{\mathsf{cur}}}$ set to fire after 2Δ time. For the newly entered slot $s = s_{\mathsf{cur}}$, the local flags $\mathsf{dummyVoted}[s]$ and $\mathsf{decisionVoted}[s]$ are initially 0. Then it executes the following handlers concurrently.

Remark 2. When a message delivery and a local timeout become enabled at the same real time, we assume that the message handler may be processed before the timeout handler. Equivalently, one may set the timer to $2\Delta + \varepsilon$ and add the corresponding ε terms to the latency bounds. This convention only affects boundary cases such as $\delta = \Delta$.

1. **Proposal for $s = s_{\mathsf{cur}}$.** If $P_i = L_s$, choose a parent chain $(B_0, \ldots, B_{s-1})$ with certificates S such that $(B_0, \ldots, B_{s-1}, S)$ is safely proposable. Create a non-dummy child B_s containing every input transaction known to P_i that is not already in the parent chain. Multicast only the signed proposal metadata

$$\langle \mathsf{propose}, s, B_s, \mathsf{parent}(B_s), S \rangle_i.$$

   The parent chain and payload fragments are made available by the DispersedSimplex-like dissemination layer of Section 7.

2. **Voting for any slot $s \leqslant s_{\mathsf{cur}}$.** Upon receiving any signed proposal

$$\langle \mathsf{propose}, s, B_s, \mathsf{parent}(B_s), S \rangle_{L_s},$$

   check that $B_s \neq \bot_s$, that the parent chain certified by S is safely proposable, and that $\mathsf{voted}[s] = \emptyset$, using the locally available blocks, certificates, and fragments obtained from the dissemination layer. If the checks pass, multicast both the $\langle \mathsf{vote}, B_s \rangle_i$ and the signature of the leader over its proposal (for equivocation detection). Then set $\mathsf{voted}[s] \leftarrow B_s$. [1]

3. **Equivocation detection for any s.** If P_i sees two signed proposals by L_s for conflicting non-dummy slot-s blocks for a slot $s \leq s_{\mathsf{cur}}$, it treats the leader as publicly equivocating. If it has not yet dummy-voted in slot s and has not decision-voted in slot s, it multicasts $\langle \mathsf{vote}, \bot_s \rangle_i$ and sets $\mathsf{dummyVoted}[s] \leftarrow 1$. This complaint may be sent even if P_i previously cast a non-dummy vote in slot s; the decision-vote guard is what prevents unsafe decision.

4. **Timeout for $s = s_{\mathsf{cur}}$.** When T_s fires, if $\mathsf{dummyVoted}[s] = 0$, $\mathsf{decisionVoted}[s] = 0$, and $\mathsf{voted}[s] = \emptyset$, multicast $\langle \mathsf{vote}, \bot_s \rangle_i$ and set $\mathsf{dummyVoted}[s] \leftarrow 1$.
   Note that if the player has already cast a non-dummy vote in slot s, the timeout does not produce a complaint; that non-dummy vote can instead be combined with complaints from other players to form $\mathsf{CertExcl}_s(B_s)$.

5. **Decision vote for any slot $s \leqslant s_{\mathsf{cur}}$.** Whenever P_i has a certified non-dummy slot-s block B_s, multicast $\langle \mathsf{decision\text{-}vote}, s \rangle_i$ only if all checks pass:
   (i) $\mathsf{dummyVoted}[s] = 0$;
   (ii) $\mathsf{decisionVoted}[s] = 0$;
   (iii) $\mathsf{voted}[s] = B_s$.
   If the checks pass, set $\mathsf{decisionVoted}[s] \leftarrow 1$. Condition (iii) checks the second part of this voting rule: the player sends a decision-vote only for the certified non-dummy block that it previously supported in that slot.
   Note that the decision-vote handler is also enabled after P_i casts a vote for a block for which it just obtained a quorum certificate.

6. **Advancing from $s = s_{\mathsf{cur}}$.** Upon seeing a safely proposable blockchain $(B_0, \ldots, B_s, S)$ of height $s = s_{\mathsf{cur}}$, multicast it then enter iteration $s + 1$ by setting $s_{\mathsf{cur}} \leftarrow s + 1$. For every slot $j \leq s$ such that B_j is non-dummy, if $\mathsf{voted}[j] = \emptyset$ and this vote is permitted by the one-non-dummy-vote-per-slot rule, it multicasts $\langle \mathsf{vote}, B_j \rangle_i$ and sets $\mathsf{voted}[j] \leftarrow B_j$.
   For each such newly recorded vote, if P_i already has $\mathcal{C}_j(B_j)$, it also runs the decision-vote handler for slot j.

[1] Note that if P_i already has a quorum certificate $\mathcal{C}_s(B_s)$, it immediately runs the decision-vote handler below for slot s. Note that invalid proposals are ignored, but we could speed-up the protocol by having players cast dummy votes upon receiving such.

7. **Deciding.** Upon seeing $\mathsf{DecCert}_s(B_s)$ for a non-dummy block and a safely proposable chain, including its certificate witnesses, ending at B_s, decide that chain.

5 Safety

We use the following standard quorum-intersection fact: any two subsets of $[n]$ of size at least $2t+1$ intersect in at least $t+1$ players, hence in at least one honest player.

Lemma 1. *For any slot s, if $\mathcal{C}_s(\perp_s)$ exists, then no $\mathsf{DecCert}_s(B_s)$ can exist for a non-dummy slot-s block B_s.*

Proof. Assume that both $\mathcal{C}_s(\perp_s)$ and $\mathsf{DecCert}_s(B_s)$ exist. The decision certificate contains a decision wildcard $\mathsf{DecisionWildcard}_s$. By quorum intersection, an honest player P_i signed both a dummy vote in $\mathcal{C}_s(\perp_s)$ and a decision-vote in $\mathsf{DecisionWildcard}_s$. If P_i signed the dummy vote first, check (i) prevents it from later signing $\langle \mathsf{decision\text{-}vote}, s\rangle_i$. If it signed the decision-vote first, then $\mathsf{decisionVoted}[s] = 1$, and the timeout/equivocation handlers never subsequently send a dummy vote for slot s. Contradiction.

Lemma 2 (Exclusivity). *For any slot s and any non-dummy block B_s, if $\mathsf{CertExcl}_s(B_s)$ exists, then no $\mathsf{DecCert}_s(B'_s)$ can exist for any conflicting non-dummy slot-s block $B'_s \neq B_s$.*

Proof. Let S_s be the $\mathsf{CertExcl}_s(B_s)$ and suppose, toward contradiction, that $\mathsf{DecCert}_s(B'_s)$ exists for a conflicting non-dummy block $B'_s \neq B_s$. The decision certificate contains a quorum certificate $\mathcal{C}_s(B'_s)$ and a decision wildcard $\mathsf{DecisionWildcard}_s$. By quorum intersection applied to the signers of S_s and of $\mathsf{DecisionWildcard}_s$, there is an honest player P_i that signs a message in S_s and also signs $\langle \mathsf{decision\text{-}vote}, s\rangle_i$.

If P_i's message in S_s is a dummy vote, then $\mathsf{dummyVoted}[s] = 1$ before the decision-vote or the dummy vote is forbidden after $\mathsf{decisionVoted}[s] = 1$; in either order this contradicts the decision-vote and dummy-vote guards. Hence the message of P_i in S_s must be a vote for B_s.

Thus P_i recorded the non-dummy vote $\mathsf{voted}[s] = B_s$. An honest player sends $\langle \mathsf{decision\text{-}vote}, s\rangle_i$ only after seeing a certified non-dummy block equal to its recorded non-dummy vote. If this certified block were B'_s, check (iii) would fail because $B'_s \neq B_s$. If it were B_s, then $\mathcal{C}_s(B_s)$ and $\mathcal{C}_s(B'_s)$ would be two conflicting non-dummy quorum certificates, impossible by quorum intersection and the one-non-dummy-vote rule. This contradiction proves the lemma.

Theorem 1 (Consistency). *Hamster is consistent.*

Proof. Consider two decided chains

$$C = (B_0, B_1, \ldots, B_s), \qquad C' = (B_0, B'_1, \ldots, B'_s, \ldots, B'_{s'}),$$

with $s' \geq s$. As by definition there exists no decision certificate for a dummy block, both the top decided blocks B_s and $B'_{s'}$ are non-dummy. By the decision rule, the longer decision is accompanied by a safely proposable certificate vector for its prefix. In particular, the slot-s witness for B'_s is either $\mathcal{C}_s(B'_s)$ or $\mathsf{CertExcl}_s(B'_s)$ if B'_s is non-dummy, or $\mathcal{C}_s(\perp_s)$ if B'_s is dummy. The dummy case is ruled out by Lemma 1, because the shorter decided chain contains $\mathsf{DecCert}_s(B_s)$. A quorum certificate is also a certificate of exclusivity. By Lemma 2, no decision certificate can exist for any conflicting non-dummy slot-s block. Since B_s is decided, $B_s = B'_s$. Collision resistance then implies equality of the prefix chains through slot s.

6 Latency Analysis

The latency proofs below are for the abstract full-dissemination protocol described in Sections 3–4. The same proofs apply to a communication-efficient implementation only under the explicit dissemination invariant stated in Section 7: once an honest player obtains progress-enabling data after GST, all honest players obtain the corresponding verifiable data within one actual delay δ, unless they have already advanced further.

We first define the iteration counter used for the expected-latency analysis. A player initializes its counter to 1 and advances it from s to $s+1$ when it receives a safely proposable blockchain of height s.

Lemma 3 (Synchronization). *If an honest player enters iteration s by time $\tau > \mathsf{GST}$, then every honest player enters iteration s by time $\tau + \delta$.*

Proof. When the first honest player enters iteration s, it has obtained the blocks and certificates needed for a safely proposable chain of height $s-1$. Hence every honest player receives the required data by time $\tau + \delta$ and enters iteration s unless it has already entered a later iteration.

Lemma 4. *If an honest leader L_s enters iteration s and proposes a block B_s at time $\tau > \mathsf{GST}$, no honest player casts a timeout dummy vote for slot s before processing the proposal, and no honest player that votes for B_s later dummy-votes.*

Proof. Since an honest leader proposes immediately upon entering its iteration, it must be that τ is also the leader's entry time. By Lemma 3, no honest player entered iteration s before time $\tau - \delta$. Since the timer is set to 2Δ, it cannot fire before $\tau - \delta + 2\Delta \geq \tau + \delta$. The proposal reaches every honest player by time $\tau + \delta$; by Remark 2, a player that receives the proposal exactly when its timer becomes enabled may process the proposal first. It then records a non-dummy vote for B_s, and the timeout rule no longer sends a dummy vote for slot s. Because the leader is honest, no equivocation evidence exists for slot s.

Lemma 5. *If an honest leader L_s enters iteration s and proposes a block B_s at time $\tau > \mathsf{GST}$, every honest player enters iteration $s+1$ by time $\tau + 2\delta$.*

Proof. By time $\tau + \delta$, every honest player has entered iteration s and seen the proposal, unless it has already entered a later iteration. Any player still in iteration s votes for B_s by that time. If a player has already entered iteration $s+1$ or later on a safely proposable chain containing the non-dummy block B_s, the advancing rule has already caused it to vote for B_s unless it had previously done so. A safely proposable chain containing $\perp_s$ cannot exist in this honest-leader case, because Lemma 4 rules out honest dummy votes and corrupt players alone cannot form $\mathcal{C}_s(\perp_s)$. Hence all honest players vote for B_s by time $\tau + \delta$, all honest players see $\mathcal{C}_s(B_s)$ by time $\tau + 2\delta$, and all enter $s+1$ by then.

Theorem 2 (Proposal confirmation time). *If the honest leader L_s enters iteration s and proposes a valid block B_s at time $\tau > \mathsf{GST}$, all honest players obtain $\mathsf{DecCert}_s(B_s)$ by time $\tau + 3\delta$.*

Proof. By Lemma 5, all honest players see $\mathcal{C}_s(B_s)$ by time $\tau + 2\delta$ and have recorded a non-dummy vote for B_s. By Lemma 4, no honest player has dummy-voted in slot s. Therefore the level-triggered decision-vote handler is enabled for every honest player, so each one passes the decision-vote checks and multicasts $\langle \mathsf{decision\text{-}vote}, s \rangle_i$ by time $\tau + 2\delta$. All honest players receive enough decision-votes by time $\tau + 3\delta$. This quorum of decision-votes plus the $\mathcal{C}_s(B_s)$ form a decision certificate for B_s.

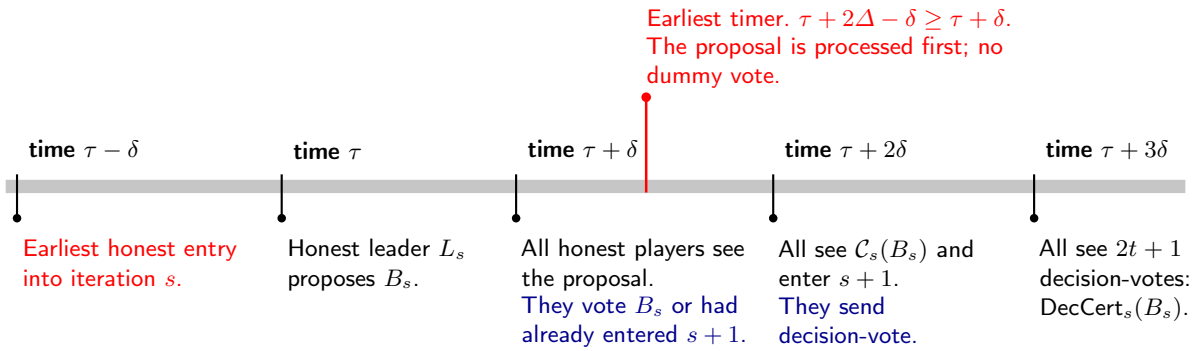


Fig. 2. Timeline for Theorem 2. The honest leader's proposal is confirmed in 3δ, while the 2Δ timeout cannot create an honest dummy vote for slot s.

Theorem 3 (Replacing a faulty leader). *Suppose all honest players are in iteration s by time $\tau >$ GST. Then all honest players enter iteration $s + 1$ by*

(a) $\tau + 2\Delta + \delta$ if L_s is not publicly caught equivocating;
(b) $\tau + 2\Delta + 2\delta$ in all cases.

In particular, the pessimistic block proposal time is upper-bounded by $2\Delta + \delta$ for adversaries not publicly equivocating, and by $2\Delta + 2\delta$ in all cases. Here "not publicly caught equivocating" means that the execution does not deliver to honest players two conflicting signed slot-s proposals from L_s; if the case then instead (b) applies.

Proof. By time $\tau + 2\Delta$, each honest player's timer has fired. For the non-equivocating case, if some honest player entered iteration $s + 1$ before time $\tau + 2\Delta$, Lemma 3 brings all honest players to $s + 1$ by $\tau + 2\Delta + \delta$. Otherwise, at time $\tau + 2\Delta$, each honest player has either cast a non-dummy vote for some proposal of L_s, or, if it has not cast any non-dummy vote, has dummy-voted by the timeout rule. Since the leader is not publicly caught equivocating, all honest non-dummy votes name the same block B_s. The honest non-dummy votes for B_s together with the honest dummy votes are either a quorum certificate $\mathcal{C}_s(\perp_s)$ when there is no non-dummy vote, or a certificate of exclusivity $\mathsf{CertExcl}_s(B_s)$ otherwise. This progress certificate reaches all honest players by $\tau + 2\Delta + \delta$, allowing entry into $s + 1$.

It remains to consider public equivocation. If some honest player enters $s+1$ by time $\tau+2\Delta+\delta$, Lemma 3 already gives entry for all honest players by $\tau+2\Delta+2\delta$. Otherwise, all honest players are still in iteration s when the conflicting signed proposals, disseminated as equivocation evidence, have reached them by $\tau + 2\Delta + \delta$. They then dummy-vote if they have not already done so and have not decision-voted. These complaint votes, together with timeout complaints already cast, reach all honest players by $\tau + 2\Delta + 2\delta$ and provide progress evidence such as $\mathcal{C}_s(\perp_s)$, $\mathcal{C}_s(B_s, \perp_s)$, or $\mathcal{C}_s(B'_s, \perp_s)$. In all cases, every honest player enters $s + 1$ by $\tau + 2\Delta + 2\delta$.

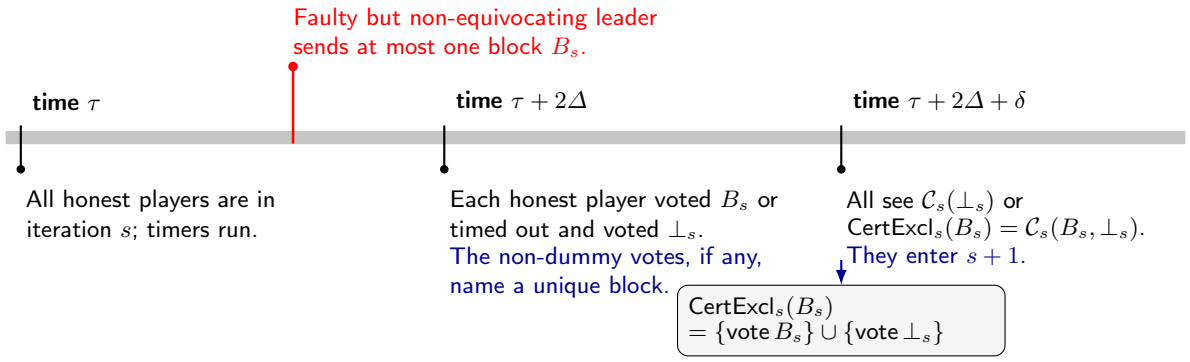


Fig. 3. Timeline for Theorem 3 in the non-equivocating faulty-leader case. A last-minute selective proposal is absorbed by a certificate of exclusivity by time $\tau + 2\Delta + \delta$.

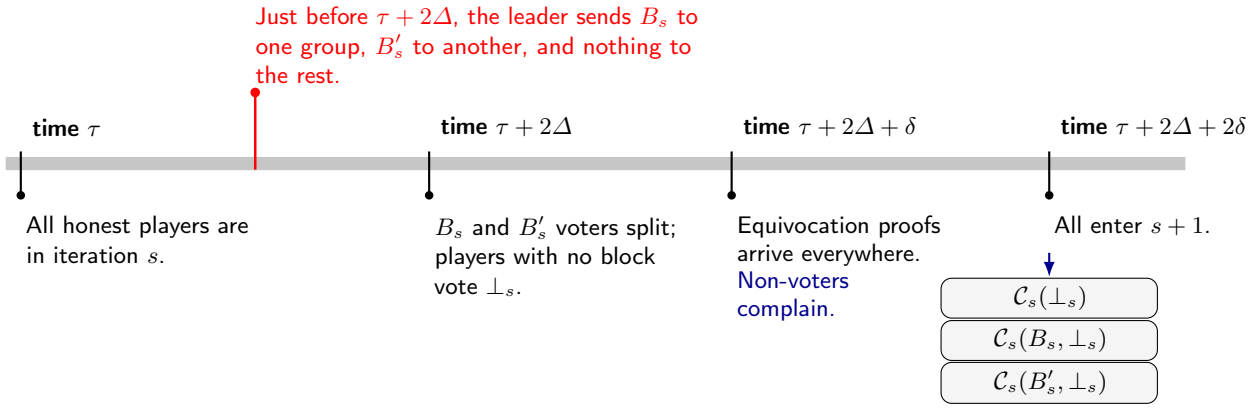


Fig. 4. Timeline for Theorem 3 in the publicly equivocating case. After one more actual delay, all players have received proofs of equivocation and those who did not vote yet cast complaint votes. Hence, after one more actual delay, each player obtains a progress evidence, either $\mathcal{C}_s(\perp_s)$, $\mathcal{C}_s(B_s, \perp_s)$, or $\mathcal{C}_s(B'_s, \perp_s)$, thus can advance iteration.

Theorem 4 (Expected latency). *With random leaders and $t/n \le 1/3$, Hamster has expected latency at most $\Delta + 4\delta$. If corrupt leaders do not get publicly caught equivocating, the expected latency is at most $\Delta + 3.5\delta$.*

Proof. Let X be the number of corrupt leaders before the first honest leader after a slot entered by all honest players. Since each leader is corrupt with probability at most $1/3$, $\mathbb{E}[X] \le 1/2$. Once an honest leader is reached, Theorem 2 decides its block within 3δ. Each corrupt leader consumes at most $2\Delta + 2\delta$ by Theorem 3, hence the expected latency is at most

$$3\delta + \mathbb{E}[X](2\Delta + 2\delta) \le 3\delta + \frac{1}{2}(2\Delta + 2\delta) = \Delta + 4\delta.$$

If no public equivocation case occurs, each corrupt leader consumes at most $2\Delta + \delta$, giving

$$3\delta + \frac{1}{2}(2\Delta + \delta) = \Delta + 3.5\delta.$$

7 Dissemination of DispersedSimplex, with a tweak

The Hamster protocol can be implemented with the same data-dissemination layer as Dispersed-Simplex [6], which we now recall, with tweaks. In the large-block regime, the leader erasure-encodes

a block into fragments, sends each player its fragment and a Merkle authentication path. When a player votes for a block, it also multicasts certified fragments of all the blocks in the chain for which it does not have a quorum certificate. A player adds a block to its complete block tree once it has a support certificate, a parent already present in its tree, and enough certified fragments to decode the payload.

We must however add tweaks to this dissemination layer to enable (1) a new leader to recover blocks B_s even when it collects few support messages for them. Indeed, without an additional mechanism it may not obtain enough fragments. We must also (2) enable players to receive all blocks in a proposed chain, despite few support messages being multicast for some of them. To solve (1), one possible rule is to have processes send the blocks B_j for which they voted to the leader of each later iteration, until they see a decided block B_i of a higher slot $i \geq j$, which implies that the prefix through B_i is recoverable by all players. To solve (2), on possible rule is that leaders piggyback all the blocks in the parent chain for which they do not have a quorum certificate, together with the corresponding certificates of exclusivity.

The latency theorems transfer from the full-multicast protocol Hamster to this dispersed implementation since it provides the following dissemination invariant: whenever an honest player obtains a safely proposable chain, a quorum certificate, a certificate of exclusivity, public equivocation evidence, or enough certified fragments to validate a proposal after GST, every honest player obtains the corresponding verifiable data within one actual delay δ, unless it has already advanced further. This is implied by the following two Claims, enforced by the two rules (1) and (2) above. (1) whenever a quorum of $2t + 1$ signed progress messages for slot s exists after all honest players have entered slot s, then within one actual delay the iteration-$(s + 1)$ leader obtains a safely proposable chain of height s; and (2) when an iteration-s leader proposes a safely proposable chain and sends the uncertified blocks, all honest players receive the whole chain within one actual delay. The proof of Claim (1) is that the blocks which players do not send to the leader of the next iteration are those for which they have a quorum certificate, and thus for which the next leader will recover the required number of $t + 1$ fragments by δ. The proof of Claim (2) is, similarly, the blocks which players do not receive from the leader are those for which the leader had a quorum certificate when it proposed, and thus for which all players will recover the required number of $t + 1$ fragments by δ after it proposed.

8 Other Comments

8.1 Optimistic responsiveness

Under the meaning introduced in HotStuff [3], optimistic responsiveness means that a new leader L_{s+1} can propose a block which will get decided, as soon as it receives any $2t + 1$ protocol messages from players looking to be honest and to wish to enter iteration $s + 1$. Simplex is also optimistically responsive, as observed in Moonshot [7]. Hamster has the analogous responsiveness property for the abstract full-dissemination protocol, and for the dispersed implementation when the needed safety witnesses and data-availability evidence are already locally verifiable by the new leader, such as enforced with the rule (1) in Section 7. In the older and weaker definition of Thunderella [15], optimistic responsiveness is instead guaranteed only if the previous leader was also honest.

8.2 Relation with timeout certificates

In PBFT-style view-change protocols, a new leader gathers timeout messages that report the latest decision-vote rounds of other players. A timeout certificate is the equivalent of Hamster's dummy quorum $\mathcal{C}_s(\perp_s)$, it shows that no block at the abandoned slot can obtain a slot-s decision certificate. Note that it does not by itself forbid every later safely proposable chain from carrying a non-dummy witness at slot s. A $\mathsf{CertExcl}_s(B_s)$ has a different role: it shows that if a slot-s block obtains a slot-s decision certificate, it must be B_s, so the next leader must extend B_s.

References

[1] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. "Consensus in the Presence of Partial Synchrony". In: *J. ACM* (1988).

[2] Miguel Castro and Barbara Liskov. "Practical Byzantine Fault Tolerance". In: *OSDI*. 1999.

[3] Maofan Yin et al. "HotStuff: BFT Consensus with Linearity and Responsiveness". In: *PODC*. 2019, pp. 347–356.

[4] Jan Camenisch et al. "Internet Computer Consensus". In: *PODC*. 2022.

[5] Benjamin Y. Chan and Rafael Pass. "Simplex Consensus: A Simple and Fast Consensus Protocol". In: *TCC*. 2023.

[6] Victor Shoup. "Sing a Song of Simplex". In: *DISC*. 2024.

[7] Isaac Doidge et al. "Moonshot: Optimizing Block Period and Commit Latency in Chain-Based Rotating Leader BFT". In: *DSN*. 2024.

[8] Qianyu Yu, Giuliano Losa, and Xuechao Wang. "TetraBFT: Reducing Latency of Unauthenticated, Responsive BFT Consensus". In: *PODC*. 2024.

[9] Dahlia Malkhi, Chrysoula Stathakopoulou, and Maofan Yin. "BBCA-CHAIN: One-Message, Low Latency BFT Consensus on a DAG". In: *FC*. 2024.

[10] Nibesh Shrestha et al. "Sailfish: Towards Improving Latency of DAG-based BFT". In: *S&P*. 2025.

[11] Nibesh Shrestha et al. "Optimistic, Signature-Free Reliable Broadcast and Its Applications". In: *CCS*. 2025.

[12] Ittai Abraham et al. "Good-case Latency of Byzantine Broadcast: A Complete Categorization". In: *PODC*. 2021.

[13] Chenxu Wang et al. *HumBird: Rotating Leader BFT made Simple and Fast*. Cryptology ePrint Archive, Paper 2026/894. 2026. URL: `https://eprint.iacr.org/2026/894`.

[14] Mark Abspoel, Matthieu Rambaud, and Andrei Tonkikh. "Linear View Change in Optimistically Fast BFT". In: *ACM Workshop on Developments in Consensus*. 2022.

[15] Rafael Pass and Elaine Shi. "Thunderella: Blockchains with Optimistic Instant Confirmation". In: *EUROCRYPT*. 2018.

A Podc'2025 reviews and actions taken

Hamster was submitted to Podc'2025. Shortly after, on 12 March 2025, we informed the chair that the overlapping iterations should be added in order to guarantee liveness. Namely, we informed that the following instruction should be added "Upon obtaining a safely proposable blockchain, a process votes for all the blocks with [slot] numbers in which it did not vote yet.". From this point Hamster was in its final correct form, the subsequent modifications were editorial.

Note that in this submission we also suggested applying to Hamster the technique of optimistic proposals of Moonshot [7], in order to achieve a variant with δ optimistic block time. We will fully formalize it in a future iteration.

Branding and motivation. Reviewer A: "Overall, the measure of "pessimistic confirmation against a static adversary that does not control the order of malicious leaders" seems extremely specialized. The relevance of optimizing this metric to PODC, practical blockchains, or foundational fault-tolerant distributed computing is unclear." Reviewer B: "The main metric considered is the expected pessimistic confirmation time, which assumes 1/3 of participants to behave maliciously. This is quite far removed from practice, as these leaders could immediately be punished and excluded from the network. I believe the proposed improvement also benefit other metrics, which I think would be very beneficial to add and discuss (e.g., pessimistic block proposal time)."
Actions taken. We advertized the pessimistic block proposal time metric and its improvement by Hamster. We also explained the "last-minute selective-proposal" attack in the introduction.

This attack refutes the claim of Reviewer A, because it enables a corrupt leader to achieve a pessimistic block proposal time in Simplex/Moonshot as bad as $3\Delta + \delta$ *without any player being ever detected as a cheater*.
Reviewer C: "The paper proposes a novel BFT consensus protocol in a niche category it refers to as "streamlined leader based protocols. This is very incremental as a goal, so already not very interesting for PODC." **Action taken.** We cited more protocols falling in this category, including some called "DAG-based-BFT" [9, 10, 11].

Communication complexity. Reviewers also estimated that the communication complexity, inherited from the Simplex-style multicasting of every received blockchain, was too high. The present version patches this using the standard technique used in DispersedSimplex [6].

Proofs of latencies. Reviewer B: "In my opinion, it is not trivial to understand how the authors arrived at the latencies... Instead, the reader needs to rely on Theorem 3.4 of the Simplex paper." "adding a figure or two could greatly improve understanding." **Actions taken.** We provide self-contained proofs of the latency metrics in Section 6, and added four figures.

Writing. Reviewer C: "Introduction reads as a model section. It does not really introduce anything." **Actions taken.** Although we believe the right way to introduce contributions is to first define the model and the metrics, as done in the Simplex [5] paper, we gave this suggestion a try by postponing the definition of the model and metrics after the introduction.

B AI disclosure

Generative AI was used for editorial improvements over our Podc'2025 submission. We inspected then corrected all the outputs to ensure the paper remains correct.