Faster Asynchronous Blockchain Consensus

Xiaohai Dai
Chaozheng Ding
xhdai@hust.edu.cn
chaozhengding@hust.edu.cn
Huazhong University of Science and Technology
China

Matthieu Rambaud
matthieu.rambaud@telecom-paris.fr
Télécom Paris
France



Figure 1: s2PAC$^{\mathbf{BIG}}$, **powered by speed-vote.**

Abstract

The fastest asynchronous (block-chained) consensus protocols, a.k.a. BFT SMR, follow a design that dates back at least to AMS (Podc'19). AMS compiles any responsive consensus, Π, based on rotating proposers, into an asynchronous consensus. It is tempting to instantiate AMS with Π equal to the fastest-known proposer-based consensus with quadratic complexity, which is 2-phase Hotstuff (2HS, Podc'19). However, we demonstrate a lack of liveness of AMS(2HS). The fastest known chained consensus, called 2-chain VABA (Podc'21 and FC'22), is a tweak of AMS intended to circumvent the livelessness of AMS(2HS). We describe attacks on its consistency and liveness, leading to the conclusion that the AMS approach is hopeless with 2HS. We introduce a new design, called 2PAC (2-phase asynchronous consensus), which modifies both the AMS design and the voting rule of the compiled consensus Π.

Of independent interest, we then address the issue that, in all previous approaches (2-chain VABA, Cordial Miners (DISC'23), GradedDAG, LightDAG) a block pipelined by a leader in the middle of the view had almost twice larger latency than the non-pipelined block. We introduce a fast decision mechanism which decides the pipelined block (and all non-leader blocks referred to within) with even smaller latency than the non-pipelined block in a wide set of scenarios (even with faults and message reorderings). Instantiated over 2PAC it gives "s2PAC".

Instantiating s2PAC with the appropriate modifications of, respectively, 2HS, and a DAG-ified version of Jolteon* (in production by Aptos), we obtain the fastest asynchronous consensus in their respective complexity classes: s2PAC^{lean} (in $O(n^2)$, i.e., quadratic) and s2PAC^{BIG}-DAG ($O(n^3)$, i.e., cubic). Our large-scale experiments confirm their superiority over the state of the art, i.e., a chained version of (sMVBA, CCS'22) in $O(n^2)$; and (GradedDAG, SRDS'23) in $O(n^3)$.

CCS Concepts

• **Computing methodologies** → **Distributed computing methodologies**.

Keywords

Asynchronous BFT SMR, blockchain consensus

ACM Reference Format:
Xiaohai Dai, Chaozheng Ding, and Matthieu Rambaud. 2026. Faster Asynchronous Blockchain Consensus. In . ACM, New York, NY, USA, 20 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

A (block-chained) asynchronous consensus protocol, a.k.a. BFT SMR, involves $n = 3t + 1$ processes: $\mathcal{P}_{[n]} = \mathcal{P}_1, \ldots, \mathcal{P}_n$, of which t are maliciously corrupt. They continuously receive input transactions, and must decide an ever-growing chain of blocks of transactions. The *consistency* requirement is that for any two decided

Conference'17, Washington, DC, USA
2026. ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

chains, then one is the prefix of the other ([41, 65, 19]). All the fastest existing asynchronous chained consensus protocols [25, 70, 48, 51, 24, 21, 23] follow a design which dates back at least from [2] and which we call AMS. The AMS protocol proceeds in iterations called *views* $v = 1, 2, \ldots$, in each of them it runs n parallel iterations, $\{(v, i) : i \in [n]\}$ of a consensus based on rotating proposers. Each process P_i, $i \in [n]$ plays the role of the proposer of view (v, i). Upon receiving a proof of completion of sufficiently many views (typically $2t+1$), processes elect one of them at random: $L_v = \mathcal{P}_\ell \in \mathcal{P}_{[n]}$ called the (a posteriori elected) *leader*, then move to the next view $v+1$. All the data produced in the view (v, ℓ) is called *endorsed* (following the terminology of [42]), whereas processes retrospectively ignore messages related to the other non-endorsed $n-1$ previous parallel views $(v, i \neq \ell)$. Although AMS uses a slow proposer-based consensus in a non-black box way, we observe that their design applies a priori to any black-box proposer-based consensus protocol Π, as illustrated in Figure 2. Both consistency and liveness follow

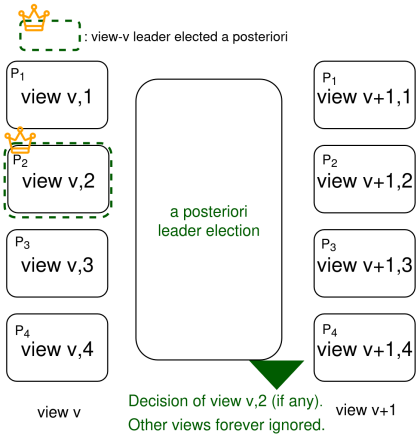

Figure 2: **the AMS design**

from the fact that, since processes ignore non-endorsed views, the analysis of an execution comes down to an execution of the baseline consensus, i.e., a sequence of views: $1, 2, \ldots, v, \ldots$ each with a single proposer L_v.

The structural limitation of AMS. It is natural to consider instantiating AMS with the fastest existing proposer-based consensus with linear communication complexity, which is Hotstuff-2 (HS-2 [59]). By contrast, Hotstuff [74] has a third phase of vote)[1]. However, in HS-2, a leader entering a view is sometimes instructed to wait for a *fixed timer*. Hence, we observe that AMS(HS-2) would not be an asynchronous protocol. Hence, it would be tempting to instantiate AMS with the modification of HS-2 in which a new proposer would *never wait* for a timer, i.e., in which the timer would be 0. We call "2-phase Hotstuff" (2HS) such a modification. Unfortunately, we observe that AMS(2HS) has no liveness. This is because the protocol 2HS lacks liveness, to the point that it was used [74, §4.4] as a textbook example (*cf.* their "livelessness" paragraph) [2].

[1]In related consensus [45, 49, 42, 28] such as SBFT or Jolteon, the authenticator complexity ([74]) is quadratic because the proposed block comes appended with a timeout certificate consisting of $2t+1$ signatures over non-equal messages, possibly aggregated with BGLS [49, 4], which takes n pairings to be verified. Applying AMS to them would thus result in cubic authenticator complexity.

[2]They use 2HS as a pedagogical explanation of the necessity of a timer in Casper [15], Tendermint [14] and HS-2.

Let us now illustrate explicitly an attack against the liveness of AMS(2HS) under asynchrony. At the top of Figure 3 we recall the structure of a view of 2HS/HS-2, with process $\mathcal{P}_i$ which stays leader during 2 phases of vote, omitting the leader-change subroutine and the step where processes report their highest-seen QC. $\mathcal{P}_i$ proposes a block, denoted $\boldsymbol{b_{v,i,1}}$, dubbed the *height-1 block*. Then it collects $2t+1$ signatures over it, called a *quorum certificate* (QC) and denoted $\boldsymbol{\mathcal{C}(b_{v,i,1})}$. From now on, we say that the block $b_{v,i,1}$ is *certified*. Then $\mathcal{P}_i$ proposes a child block of $\boldsymbol{b_{v,i,1}}$, denoted $\boldsymbol{b_{v,i,2}}$ and dubbed the *height-2* block, or also the *pipelined block*. Then it collects $2t+1$ signatures over $\boldsymbol{b_{v,i,2}}$, resulting in a $\boldsymbol{\mathcal{C}(b_{v,i,2})}$, which enforces the decision of the parent block $\boldsymbol{b_{v,i,1}}$. In 2HS (as well as in our protocols), a block b is a child of a block b' if it contains a QC over b': we denote this $\mathcal{C}(b') \leftarrow b$. Processes accept to vote for b only if their highest-seen QC, dubbed their "lock", is not higher than the $\mathcal{C}(b')$ included in b.

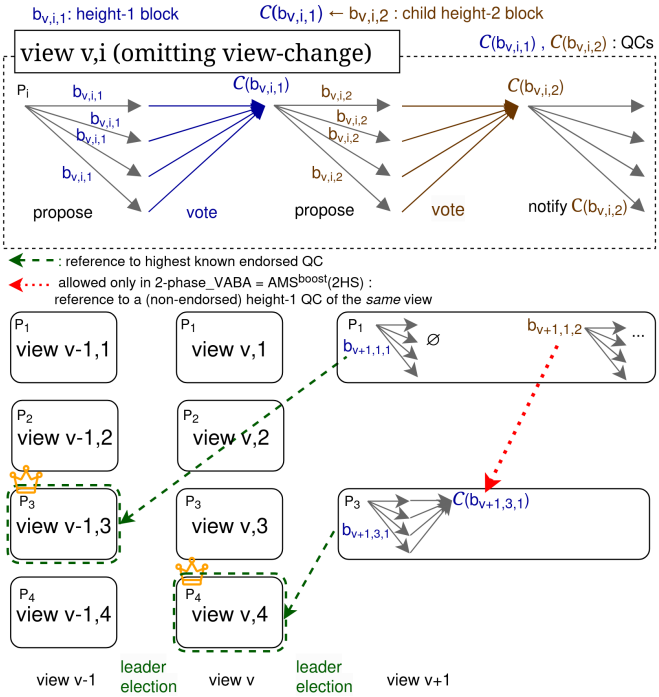

Figure 3: 2-chain_VABA**. red dotted arrow: only in** AMS$^{\textbf{boost}}$

At the bottom of Figure 3 we describe a scenario in AMS(2HS) (in which the dotted red arrow does not exist) where the highest endorsed QC known to $\mathcal{P}_1$ at the beginning of view $v+1$, is the one for a block which was proposed by $\mathcal{P}_3$ in view $v-1$: $b_{v-1,3,*}$. Whereas, the process $\mathcal{P}_3$ has since seen a higher endorsed QC, which is for a block proposed by $\mathcal{P}_4$ in view v: $\boldsymbol{b_{v,4,*}}$. Hence, $\mathcal{P}_3$ refuses to vote for the view-$(v+1)$ proposal of $\mathcal{P}_1$, $\boldsymbol{b_{v+1,1,1}}$. Without any modification to AMS, the execution would freeze here: assuming that $\mathcal{P}_2$ is crashed, then $\mathcal{P}_1$ would never collect $2t+1$ votes for its proposal, hence its view would never complete, hence only the $2 < 2t+1$ views $(v+1, i = 3, 4)$ would ever complete, hence processes would never go to leader election. Note that this scenario explains the reason for a nonzero timer in HS-2: *under the eventual*

synchrony assumption of HS-2, $\mathcal{P}_1$ would have waited long enough to learn the highest QC held by an honest process, which is the one for a $b_{v,4,*}$. The same attack would also apply under asynchrony to AMS(HS-2), i.e., even if assuming that $\mathcal{P}_1$ waits for a predetermined timer before proposing, because the adversary could always delay the notification to $\mathcal{P}_1$ of the QC over $b_{v,4,*}$ until after $\mathcal{P}_1$ proposes.

1.1 Contributions

1.1.1 Attacks showing that the limitation is unavoidable. The protocol 2-chain_VABA [42] was, until this paper, the fastest known asynchronous blockchain consensus with quadratic bit complexity. 2-chain_VABA was used as the slow path of the BFTs called "Ditto" [42] and "Abraxas*" [11, Sec. 6]. 2-chain_VABA can be conceptualized as AMS^{boost}(2HS), where AMS^{boost} is a relaxation of AMS aiming at circumventing the livelessness of AMS(2HS). The relaxation is what they call the *boosting strategy*, we depict it by the red dotted arrow in Figure 3. It allows a proposer, such as $\mathcal{P}_1$, to create its height-2 block: $b_{v+1,1,2}$ as a child of the block of *any* proposer of view v+1, not necessarily its own proposed block. Hence, this departs from AMS since, in AMS, each view $(v+1, i)$ treats the other views $(v+1, j \neq i)$ as non-existing. In Figure 3, $\mathcal{P}_1$ receives the $\mathcal{C}(b_{v+1,3,1})$ for the block of $\mathcal{P}_3$, and builds its height-2 block as a child of it: $\mathcal{C}(b_{v+1,3,1}) \leftarrow b_{v+1,1,2}$. Hence, this unlocks the previous scenario where $\mathcal{P}_1$ could never obtain a $\mathcal{C}(b_{v+1,1,1})$ hence never propose a $b_{v+1,1,2}$. However the decision rule in 2-chain_VABA stays strict: consider the view-$(v+1)$ elected leader L_{v+1}, here $\mathcal{P}_1$, suppose that its height-2 block $b_{v+1,1,2}$ is the child of the block of *another proposer*, here: $b_{v+1,3,1}$, then the height-2 QC over it: $b_{v+1,1,2}$ *does not* enforce decision of $b_{v+1,3,1}$, nor of any block. This gives rise to the following attack blocking liveness forever, which we communicated to the authors on 2023-10-13. The adversary corrupts any fixed process, say $\mathcal{P}_3$. It delivers all messages to $\mathcal{P}_3$, so that it has a lock at least as high as the highest lock of honest processes. In each view v, $\mathcal{P}_3$ quickly collects votes over its height-1 block and diffuses the obtained QC: $\mathcal{C}(b_{v,3,1})$ to the other processes. As a result, they all build their height-2 block over this QC, so by the strict decision rule above, no block is ever decided in view v.

This raises the question whether one could tweak differently AMS while keeping the lightweight voting rule of 2HS, which is to vote for a height-1 block only if the QC of the (endorsed) parent is at least as high as one's highest seen (endorsed) QC. We identified at least 5 possible tweaks to the design of 2-chain_VABA, of which: the boosting strategy, lifting various checks for endorsement (some of them lifted in the implementation [73]), electing the leader as soon as $2t + 1$ processes obtain one height-2 QC (suggested in the [42] paper), and finally a relaxation of the decision rule so that, above, a $\mathcal{C}(b_{v+1,1,2})$ would enforce the decision of its parent $b_{v+1,3,1}$ (which is done in the implementation [73]). We attacked all the 2^5 possibilities of variants resulting in activating each tweak or not. Our attacks either broke their consistencies, or, at best, slowed-down their latencies by multiplying by n the expected number of views until a decision. The least obvious attack breaks consistency when the decision rule is relaxed: it is described in Section D.D.1. All attacks were quickly confirmed by the authors of 2-chain_VABA (on 2023-10-25), who shared our conclusion that the AMS^{boost}(2HS) design inherently does not work. They immediately repaired Ditto by replacing 2-chain_VABA with a generic consensus, which they suggest ([43]) to instantiate with our new design (below). This conclusion was since adopted by the community ([55]).

1.1.2 Our new design. We introduce a new design called 2PAC (2-phase asynchronous consensus).

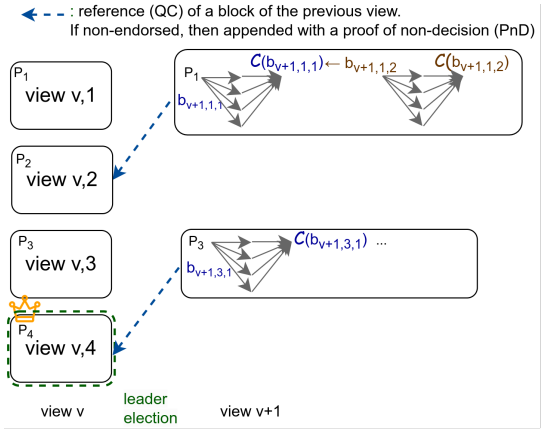

Figure 4: 2PAC^{lean}: 2PAC **instantiated with 2HS**

The novelty of 2PAC is that every proposer in a new view, v+1, *always* proposes a block which is a child of some view-v *height*-2 block. In the case where this view-v *height*-2 would *not* be endorsed, then the proposer *proves* that its proposal does not conflict with a view-$(v-1)$ decided block, by appending a lightweight proof to it. The proof is called a *proof of non-decision* (PnD) and consists of a threshold signature of $2t+1$ processes which all report that they did not see any endorsed view-$(v-1)$ height-1 QC. This proves that *no view-$(v-1)$ block could have been decided*. For instance on Figure 4, $\mathcal{P}_1$ proposes a child, $b_{v+1,1,1}$, of a non-endorsed *height*-2 block of the previous view, $b_{v-1,i,2}$. We emphasize that other view-$(v+1)$ proposers could well have observed an endorsed view-v *height*-2 block, and thus propose a child of it, as $\mathcal{P}_3$ in Figure 4. This however does not harm safety, since the PnD (exhibited by $\mathcal{P}_1$) precisely consists of $2t + 1$ signed declarations of *not* ever voting for an endorsed view-v *height*-2 block, thereby ruling out the possibility for any view-v block to be ever decided.

Beyond its simplicity, the new 2PAC design brings two gains, which are best illustrated by comparison with a chained version of the sMVBA consensus [48] (used in [77, 75, 22, 47]), see Sec. 1.2.1. (1) First, PnD's consist of only one threshold signature. Whereas in sMVBA, of which the structure improves over AMS(Jolteon), the proposer proposes a child of an endorsed certified block *of possibly several views before*, and convinces the processes that no decision took place since this certified block. To this end, it exhibits a proof which is built incrementally as the concatenation of one threshold signature per elapsed view (line 30 of Algorithm 3 of [48]), and which takes two rounds to be built (PreVote then Vote, constituting the view-change), and the specification of its validity predicate takes 19 lines ([48, Algorithm 4]). (2) Second, while in chained sMVBA there are no blocks in the chain in views where the elected leader was too slow or crashed, in 2PAC by design there

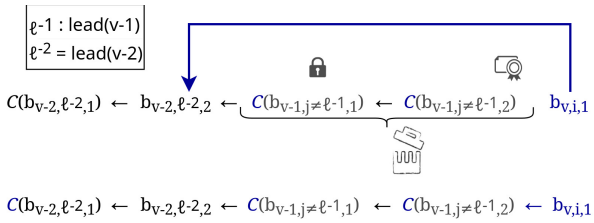

Figure 5: **New view-v proposal by $\mathcal{P}_i$: $b_{v+1,i,1}$ when no endorsed view-$(v-1)$ QC is known to $\mathcal{P}_i$. Top: in chained sMVBA (&** 2-chain_VABA**). Bottom: in** 2PAC**.**

are always two blocks per view in the chain, cf Figure 5. In addition, the work needed is not higher, since a proposed block still points to one unique parent, i.e., a block of the last view. As a result, the number of decided blocks per unit of work, i.e., the throughput, is higher.

1.2 Leveraging 2PAC as a tool

1.2.1 Faster asynchronous quadratic consensus. Instantiating 2PAC with 2HS yields the protocol called 2PAC^{lean}, which has quadratic ($O(n^2)$) bit complexity. It achieves a record 9.5worst-case expected latency, vs 10δ for a chained version of sMVBA [48] (used in [77, 75, 22, 47]). We then further bring down its average latency as follows. We observe the limitation that the *height*-2 block $b_{v,\ell,2}$ pipelined by the leader L_v is not decided in the view v. Instead, it has to wait until it becomes a direct child or ancestor of a decided block, to become *indirectly* decided. Hence, pipelined blocks have nearly twice as large a latency as *height*-1 blocks proposed by leaders. This limitation equally hits all previous asynchronous blockchain consensus: 2-chain_VABA [42], sMVBA [48] (chained) and [25, 51, 24, 21]. In Sec. 4 we upgrade the 2PAC design into s2PAC, enabling the *height*-2 block pipelined by the leader to be decided with even shorter latency than the *height*-1 block (cf Table 18). It is decided upon receiving $2t+1$ *speed-votes*. The safety is preserved by a careful mechanism, as lightweight as PnD's, enabling a proposer of a new view to *prove* that its *height*-1 proposal does not conflict with a possible fast-decided *height*-2 block of the previous view. [3]

Our large-scale experiments (Sec. 6) of s2PAC^{lean} against sMVBA demonstrate has a twice larger throughput, due to the combination of the two gains described in Sec. 1.1.2, and an improved latency, due to the one-less round than in sMVBA to create the proof, further sped-up by speed-votes.

1.2.2 Faster asynchronous cubic consensus. We introduce s2PAC^{BIG}: an instantiation of 2PAC with $O(n^3)$ complexity, which matches 4δ latency in *all* good-case executions, i.e., which are fault-free and enjoy what we call a *half-fair scheduler* (*cf.* Definition 3: such that the slowest message takes no more than twice the delay of the fastest message). s2PAC^{BIG} can be roughly seen as the instantiation of the s2PAC design, applied to the "Simple Moonshot" partially synchronous consensus [29]. Concretely, compared to s2PAC^{lean}, the votes are now multicast all-to-all (hence each of the two phases of vote takes δ instead of 2δ), and each $\mathcal{P}_i$ (acting as proposer) pipelines its *height*-2 block $b_{v,i,2}$ as soon as obtaining $2t+1$ other proposals (hence even before it obtains a QC for its parent $b_{v,i,1}$). In Figure 6 we depict the messages of s2PAC^{BIG} relatively to only one proposer in one view (all other processes acting as proposers in parallel), before the speed-votes & leader election phase.

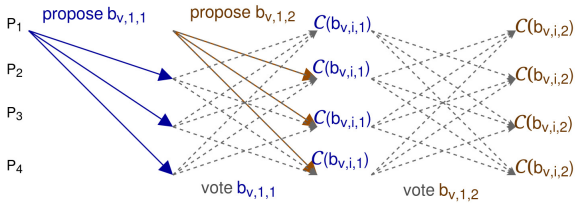

Figure 6: **View of 2PAC^{BIG} for $n=4$ before the election phase, for proposer $\mathcal{P}_1$ only ($\mathcal{P}_2$, $\mathcal{P}_3$, $\mathcal{P}_4$ also act as proposers in parallel, not displayed). $\mathcal{P}_1$ proposes $b_{v,1,1}$ then waits until receiving $2t+1$ proposals, then proposes the child block $b_{v,1,2}$. The dotted arrows are all-to-all blocks.**

In order to compare s2PAC^{BIG} with the recent cubic consensus [25, 51, 24, 21], in our benchmarks, we tuned it following [54, 3], as follows. Clients split their transactions across the n processes (instead of sending all of them to all). A process i proposing an *height*-1 block: $b_{v,i,1}$, must include inside references to distinct $2t+1$ $(v-1)$-blocks via their QCs (of which at least t+1 of *height*-2). A process i proposing a *height*-2 block: $b_{v,i,2}$, must include inside $2t+1$ references to *height*-1 blocks. A DecCert on a block automatically constitutes a DecCert on all the tree of referenced blocks, chained w.r.t. some predetermined topological order. Our experiments (Sec. 6) confirm that the latency is substantially lower and the maximal throughput higher than the state of the art GradedDAG [24].

Note that we choose to describe our protocol with fully explicit messages, instead of wrapping it as a DAG. The difference is purely syntactical, e.g., our no_endorsed_height-1_QC message mechanism has since been adapted to partially synchronous DAGs [67]. There, they are instead sent when the timeout expires. Our choice is also because, although the *height*-2 QCs in 2PAC^{BIG} are equal to outputs of reliable broadcast, hence are equal to the "vertices" in DAG protocols, we believe that 2PAC^{BIG} cannot easily be factored-out as a DAG. Indeed, processes in 2PAC^{BIG} pipeline *height*-2 blocks without waiting for the ancestor *height*-1 blocks to become vertices. In any case, QCs over blocks in 2PAC^{BIG} are proofs that all their ancestors have been seen by $t+1$ processes, hence are efficiently retrievable.

1.2.3 Evaluation by applying 2PAC *to ACS.* It is to be noted from Figure 14 that as much as 73% resp. 92% of views benefited from a speed decision of pipelined blocks, in s2PAC^{lean} resp. s2PAC^{BIG}. This motivates the following proposition to use 2PAC for agreement on a common subset (ACS [31]), as follows. The view-1 *height*-1 blocks contain the inputs, then each view-1 *height*-2 block contains $2t+1$ references to *height*-1 blocks (via their QCs, which guarantee their retrievability), then once a view-1 *height*-2 block is decided (at the end of view 1 via speed votes, or indirectly later), it enforces the output of the $2t+1$ inputs which it references.

[3] Note that speed-votes would also apply to the *first* pipelined block in a view of LightDAG2, but not to the second one (nor to the pipelined block in GradedDAG), since it is proposed too late.

2 Model and Definitions

Network and corruptions. The system consists of $n = 3t+1$ processes $\mathcal{P}_1, \ldots, \mathcal{P}_n$, of which t are actively corrupt by the adversary. For simplicity we consider static corruptions, we refer to Appendix B.7 for adaptive security. Processes are linked by peer-to-peer (P2P) asynchronous channels without authentication nor secrecy. The delays of the messages sent by honest processes are arbitrarily set by the adversary. Hence, our measurement of latency is in terms of the largest (unknown, possibly infinite) message delay in the execution, which we denote δ. To multicast a message m means to send it to all the processes over the P2P channels.

Asynchronous Blockchain Consensus (a.k.a. async. BFT SMR). We consider the classic definition [62, 30, 56, 70, 48, 40, 76, 51, 22, 11, 69, 24, 21, 20] upgraded with *provability*, following [22, §2.2.2]. Processes continuously receive input transactions, and output an ever-growing chain of decided blocks. In more detail: there exists a public predicate called *valid_DecCert*. If *valid_DecCert*$(b, \sigma) = \text{true}$ then we say that σ is a decision-certificate (DecCert) vouching for the block b. Each block contains a pointer to a parent block, thus forming a chain of blocks. A DecCert for a block constitutes as well an (indirect) DecCert for all its ancestors. When a process outputs a DecCert for a block, we say that the block and all its ancestors are *decided*.

Definition 1. ([19]) A *blockchain consensus* must satisfy
liveness: every input transaction eventually gets included in a decided block;
consistency: consider two chains of decided blocks, then one must be a prefix of the other;
***p*-quality** ([41, 2, 46]) for some $p > 0$: the asymptotic proportion of decided blocks in which transactions were input by honest processes, is at least p.

2.1 Notation: View, Height, QC, Endorsement

The notation introduced in this section holds throughout the paper, including for 2-chain_VABA and sMVBA (modulo a swap of *proposer* and *height*, in Appendix D). It simplifies the notation used in [42], since we drop their notion of "rounds", and of "fallback" vs "regular" blocks and QCs. These notions were used only in their higher-level protocol, called Ditto [42], which switches back and forth from a partially synchronous protocol. Instead, in both 2-chain_VABA and 2PAC, the tuple of (*view*, *height*) is enough to uniquely rank blocks.

Threshold signatures. We denote $\langle m \rangle_i$ a standalone signature of $\mathcal{P}_i$ on the message m, and say that $\mathcal{P}_i$ is the signer of the "signed message $\langle m \rangle_i$". A $2t+1$-threshold signature scheme [12, 7] takes as input $2t+1$ "individual" (or "partial") signatures from *any* distinct signers $\mathcal{P}_i$'s on the same message m: $\{m\}_i$ and combines them into a short multi- (or threshold) signature on m: $\{m\}$. *In practice, threshold signatures may simply be implemented as concatenations of signatures [48]* or as multi-signatures [66, 52, 38]. In those implementations, the signing public key(s) are also appended because required for verification. All previous notations imply that the message m is appended to the signature.

Local View number. Each process $\mathcal{P}_i$ has a local counter denoted v and called *the current view number of* $\mathcal{P}_i$.

▷ denoted instead v_{curr} in [42], it could also have been denoted v_i.

We say that $\mathcal{P}_i$ *is in view* v. When v advances to a higher number v', we say that $\mathcal{P}_i$ *enters view* v'.

Common coin. ([17, §4.3][36]) For any view number v, each process can generate a coin share for v. Then there is a public algorithm which takes any $2t+1$ (valid) coin shares from distinct processes for the same v, and combines them into a certificate called coin-certificate: $\mathcal{C}_{\text{coin}}$, vouching for unique value $i \in [n]$. We then say that process $\mathcal{P}_i$ is the elected leader of view v, denoted $L_v = \mathcal{P}_i$. The probability of the adversary to predict the outcome i of the election, before it sees any view-v coin share from an honest process, is at most $1/n + \text{negl}(\kappa)$. It is worth noting that in 2-chain_VABA, the threshold number of coin shares was only $t+1$. Our choice of $2t+1$ is important to achieve our claimed 6δ good-case latency of 2PAC, in particular, even if the scheduler is only half-fair. The intuition is that with only $t+1$ shares, an isolated honest process could transition too fast to a higher view, preventing it from helping other players to finish their two-chains: cf Appendix E.4. *Only for simplicity of the exposition*, we further assume that coin-QCs are constant-sized. This smallness assumption holds when coin-QCs are implemented as non-interactive unique threshold signatures [12, 72, 7] (the latter is not pairing-based, which positively answers [76, 31]). Note that this smallness assumption of coin-QCs can be lifted: as explained in Appendix B.1, using a Bracha-like termination gadget for election (see Appendix B.1) removes the need for processes to forward coin-QCs to each other. This alternative preserves all theoretical latencies of the 2PAC protocols. Thus, post-quantum lattice-based coins are usable, such as [13, §5 & §7.1], improved by [34], or also [61]. Note that a common coin is used in all the most efficient consensus algorithms: [63, 64, 2, 42, 48, 70, 31, 76, 24, 21, 35, 20], including the signature-free ones.

Blocks. A block is $b = \left\{v, i, h, parent, txn, id\right\}$, where
- v is called its view number;
- $i \in [n]$ is the index of its *proposer* $\mathcal{P}_i$;
- $h \in \{1, 2\}$ is called its *height*;
- *parent* is a short reference to its parent block (see below). We denote $\widetilde{b} \leftarrow b$ a block b with parent $\widetilde{b}$, and call b a *child* of $\widetilde{b}$. Blocks are chained by parent-child relations.
- *txn* is a batch of transactions;
- $id = \text{H}(parent, v, height, proposer, txn)$ is the unique hash digest of $(parent, v, height, proposer, txn)$;

We denote $b.id$, $b.v$ etc. the *id*, view number etc. of a block b. We will use $b_{v,i,h}$ to denote a view-v, *proposer*-i and *height*-h block. We sometimes omit from the description *txn*, *id* and i, leaving $b_{v,h}$.

Quorum certificate. A QC for a block $b = \left\{v, i, h, parent, txn, id\right\}$, denoted $\mathcal{C}(b)$, is a $(2t+1)$-threshold signature on the message $(b.v, i, h, b.id)$. We then say that b is *certified*. For $v := b.v$ and $h := b.height$, we say that $\mathcal{C}$ is a view-v *height*-h QC. We denote $\mathcal{C}_{v,i,1}$ resp. $\mathcal{C}_{v,i,2}$ a QC for some (unspecified) view-v *proposer*-i *height*-1 resp. 2 block.

Ranking. Blocks and QCs are ranked by their view numbers, then by their heights, i.e., by $2v + height$. Hence, we abuse notation and shorten as $\mathcal{C} \geqslant \mathcal{C}'$ the relation $\mathcal{C}.(2v + height) \geqslant \mathcal{C}'.(2v +$

height). The rank must strictly increase along a chain of blocks, e.g. $b_{v-2,1} \leftarrow b_{v,1} \leftarrow b_{v,2}$. In 2PAC, blocks in a decided chain will always have *consecutive* ranks, e.g. $b_{v-1,1} \leftarrow b_{v-1,2} \leftarrow b_{v,1} \leftarrow b_{v,2}$. This absence of holes in the ranks of the blocks along a chain, is partly what will explain the superior throughput of the 2PAC design over sMVBA.

Endorsed QC. Once a process has a coin-QC: $\mathcal{C}_{\text{coin}}$ for view v that elects process $\mathcal{P}_\ell$ as the leader L_v, we say that any $\boldsymbol{\mathcal{C}_{v,\ell,1}}$ resp. $\boldsymbol{\mathcal{C}_{v,\ell,2}}$ is *endorsed* (by $\mathcal{C}_{\text{coin}}$).

Parent-referencing with QCs. The notation $\widetilde{\mathcal{C}} \leftarrow b$ in the protocols, means that the reference to the *parent* $\widetilde{b} \leftarrow b$ of the block b *must* be encoded as a QC: $\widetilde{\mathcal{C}}$ on $\widetilde{b}$. In this case, we say that $\widetilde{\mathcal{C}}$ is "wrapped inside b", and may abuse notation by calling directly b a "child" of $\widetilde{\mathcal{C}}$.

Parent-referencing with hashes. The notation $\widetilde{b} \leftarrow b$ means that: if the proposer of b has a QC: $\widetilde{\mathcal{C}}$ on the parent $\widetilde{b}$, then it encodes the reference to $\widetilde{b}$ as $\widetilde{\mathcal{C}}$, else, it encodes the reference as a mere hash: $\text{H}(\widetilde{b})$. For instance in Figure 8, the notation $\mathcal{C}_{v-1,\ell,1} \leftarrow b_{v-1,\ell,2} \leftarrow \boldsymbol{b_{v,i,1}} = \{v, i, 1, \text{H}(b_{v-1,\ell,2}), txn, id\}$ means that the parent of $\boldsymbol{b_{v,i,1}}$ is referred to via its hash, and that the proposer: $\mathcal{P}_i$, diffuses in parallel its preimage $b_{v-1,\ell,2}$. A process ignores $\widetilde{b} \leftarrow b$ until it has checked the availability of its parent: $\widetilde{b}$ and of all its ancestors.

Genesis blocks. Processes are initialized with the two view-0 *genesis blocks*: $b_{0,1} \leftarrow b_{0,2}$ with empty transactions. By convention, the empty string: $\perp$ is a QC for both of them.

(ordinary) Decision-certificate (DecCert**).** In 2PAC, a DecCert over any view-w *proposer*-$\mathcal{P}_\ell$ *height*-1 block, $\boldsymbol{b_{w,\ell,1}}$, is a *2-chain of endorsed QCs*. Namely, the data of $\mathcal{C}(\boldsymbol{b_{w,\ell,1}}) \leftarrow \mathcal{C}(\boldsymbol{b_{w,\ell,2}})$ where $\boldsymbol{b_{w,\ell,2}}$ is any view-w *height*-2 *proposer* $\mathcal{P}_\ell$ child block of $\boldsymbol{b_{w,\ell,1}}$, and where $w = L_w$. We dub *ordinary* these DecCerts, because in s2PAC (Sec. 4) we will add a second type of DecCerts, called speed-DecCerts.

Recall that a DecCert on a block b enforces the decision of b and of all its ancestors, hence, constitutes (indirectly) a DecCert over each block in the chain of ancestors of b.

All blocks are retrievable off the critical path. Processes can form QCs and DecCerts out of votes even if they did not receive the *height*-1 block: $\boldsymbol{b_{w,\ell,1}}$ itself. Note that the QC guarantees that $t+1$ honest processes have received $\widetilde{b}$, hence that it is efficiently retrievable ([27, 39]) off the critical path. Moreover, in both s2PAC^{lean} and s2PAC^{BIG}, in the case of a block b pointing to its parent $\widetilde{b}$ by its mere hash, we nevertheless enforce: (i) the proposer of b must also diffuse the hash preimage preimage $\widetilde{b}$; (ii) by design we always have that $\widetilde{b}$ points to its parent via a QC. Thus (i) and (ii) imply that a node receiving a proposal b from an honest proposer is guaranteed (i) to receive soon its parent $\widetilde{b}$ (ii), which, itself, contains a proof (QC) that the whole chain of ancestors is efficiently retrievable.

This is the reason why we formalize latency in Appendix A.1 as the delay until the output of a DecCert, which could happen before learning the actual content of the decided block.

Notation	Meaning
L_v	elected leader of view v
$\boldsymbol{b_{v,i,1}}$ (resp. $\boldsymbol{b_{v,i,2}}$)	a view-v proposer-$\mathcal{P}_i$ and height-1 (resp. 2) block
$\mathcal{C}(b)$	QC for a block b
$\mathcal{C}(b') \leftarrow b$	block b containing a pointer to its *parent* b' in the form of a QC over b'.
$\boldsymbol{\mathcal{C}_{v,i,1}}$ (resp. $\boldsymbol{\mathcal{C}_{v,i,2}}$)	QC for some unspecified $\boldsymbol{b_{v,i,1}}$ (resp. $\boldsymbol{b_{v,i,2}}$)
endorsed $\boldsymbol{b_{v,i,1}}$ or $\boldsymbol{b_{v,i,2}}$	s.t. $\mathcal{P}_i = L_v$
endorsed $\mathcal{C}(b)$	s.t. b is endorsed
(ordinary) DecCert($\boldsymbol{b_{w,\ell,1}}$)	a $\mathcal{C}(\boldsymbol{b_{w,\ell,1}}) \leftarrow \mathcal{C}(\boldsymbol{b_{w,\ell,2}})$ s.t. $\ell = L_w$.

Table 7: **Summary of notation**

3 2PAC

We present the 2PAC design. More precisely, we examplify 2PAC when applied to 2HS, which yields the consensus "2PAC^{lean}" which we now outline. In Figure 8 we fully formalize 2PAC^{lean}. In Appendix A.1 we formalize the two theoretical latency metrics which we consider (good-case, and expected worst-case), then in Theorem 4 we state both the latencies of 2PAC, and its consistency. In Appendix A.2 we prove the former and in Appendix A.3 the latter. In Appendix A.3.4 we analyze its quality. In Sec. B we sketch some variants and optimizations. For conciseness we make the choice to present 2PAC^{lean} without the SPEED_VOTE mechanism, we will add it in Sec. 4, yielding s2PAC^{lean}.

A view v of 2PAC^{lean} goes as follows. Each process $\mathcal{P}_i$ acts as a proposer and does the following:

- processes first send to all signed *reports* of the endorsed QCs which they saw or not in the previous view (to be detailed)
- $\mathcal{P}_i$ forms a *height*-1 block $\boldsymbol{b_{v,i,1}}$ which is a child of a well-chosen view-$(v-1)$ *height*-2 block $\boldsymbol{b_{v-1,j,2}}$, appended with data to convince processes to vote for it (to be detailed). It proposes the block by multicasting it.
- upon gathering $2t+1$ votes for $\boldsymbol{b_{v,i,1}}$, it gathers them into a QC: $\mathcal{C}(\boldsymbol{b_{v,i,1}})$. Then it creates a child *height*-2 block, $\mathcal{C}(\boldsymbol{b_{v,i,1}}) \leftarrow \boldsymbol{b_{v,i,2}}$ and proposes it by multicasting it.
- upon gathering $2t+1$ votes for $\boldsymbol{b_{v,i,2}}$, it gathers them into a QC, $\mathcal{C}(\boldsymbol{b_{v,i,2}})$, and multicasts it.
- upon receiving $2t+1$ $\boldsymbol{\mathcal{C}_{v,k,2}}$'s, i.e., QCs for *height*-2 blocks of $2t+1$ proposers $\mathcal{P}_k$'s, a process multicasts it coin-share (and we have the Bracha amplification mechanism that a process multicasts it as well upon receiving $t+1$ coin shares)
- upon receiving $2t+1$ coin shares, a process forms the coin-certificate $\mathcal{C}_{\text{coin}}$ which reveals the identity of the leader $\mathcal{P}_\ell$ of v, denoted $L_v \leftarrow \mathcal{P}_\ell$. It then moves to the next view $v+1$. The $\mathcal{C}_{\text{coin}}$ *endorses* the blocks proposed by $\mathcal{P}_\ell$. In particular, upon seeing a 2-chain of endorsed QCs $\mathcal{C}(\boldsymbol{b_{v,\ell,1}}) \leftarrow \mathcal{C}(\boldsymbol{b_{v,\ell,2}})$ then any process decides $\boldsymbol{b_{v,\ell,1}}$.

We now elaborate on the mechanism to safely propose and vote.

(1) *If* $\mathcal{P}_i$ knows an *endorsed* and certified block of the previous view, $\boldsymbol{b_{v-1,L_{v-1},1}}$, then the *height*-2 endorsed block

$b_{v-1,L_{v-1},2}$ is $\mathcal{P}_i$'s preferred choice of the parent $b_{v,i,1}$. This case is thus similar to 2-chain_VABA[4].

(2) *Else*, $\mathcal{P}_i$ picks *any* view-$(v-1)$ block as the parent of its proposed block $b_{v,i,1}$ and proves its safety by appending to it a *proof of non-decision* (PnD). A PnD is a declaration of $2t+1$ processes reporting that they did not see any endorsed view-$(v-1)$ height-1 QC, hence that *no view-$(v-1)$ block could have been decided*. Note that such steps, where processes report what they saw or not in previous views, are often called "view-change" in the literature. Processes then vote for the proposed block $b_{v,i,1}$ upon checking the PnD. In conclusion, the design of 2PAC departs from AMS in that the previous non-endorsed views are no longer ignored; on the contrary, these non-endorsed blocks are the key to unlock the scenario Figure 3 where AMS(2HS) was stuck.

3.1 Intuition of the proof of consistency of 2PAC^{lean} . The proof of consistency intuitively follows from a ***chain of consistency*** and is fully formalized in Appendix A.2. Namely, let us make the induction assumption that no view-$(v-1)$ QC conflicts with any decision in a prior view, i.e., with any view-$(<v-1)$ DecCert. Let us prove that the assumption holds for view-v QCs. It will follow, by induction, that the statement holds for all views.

Consider a view-v QC, $\mathcal{C}_{v,1}$. By the 2PAC design, it is a descendant of a view-$(v-1)$ QC: $\mathcal{C}_{v-1}$. By the assumption, $\mathcal{C}_{v-1}$ itself does not conflict with any view-$(<v-1)$ DecCert. As a result we have *Claim (i):* no view-$(<(v-1))$ DecCert conflicting with $\mathcal{C}_{v,1}$ can ever exist.

In order to prove the desired assumption for v, it remains to finally prove the *Claim (ii):* no view-$(v-1)$ DecCert conflicting with $\mathcal{C}_{v,1}$ can exist. We have two cases. *Either* $\mathcal{C}_{v-1}$ is endorsed, then the Claim (ii) follows from quorum intersection (no two endorsed QCs can exist in the same view). *Or it is not*, thus at least one honest process (actually $t+1$ of them) checked existence of a threshold signature {no_endorsed_height-1_QC, v}, i.e, a PnD. The PnD proves that no view-$(v-1)$ certified block could possibly be decided, i.e., no view-$(v-1)$ DecCert could be made, which implies a fortiori the Claim (ii).

4 s2PAC: Speeding-up Pipelined Blocks

The idea of s2PAC is simply that processes in a view v cast a SPEED_VOTE for a view-v *height*-2 block upon seeing a (*height*-2) QC for it. $2t+1$ SPEED_VOTEs for a *height*-2 block: b' pipelined by the leader, constitute a "speed-DecCert" for b', enforcing its decision. To preserve consistency, we simply require that, in case the proposer in a new view picks the parent of its proposal equal to an endorsed *height*-2 block *without a QC*, hence such that only the *height*-1 QC of the previous view, must exhibit a combined quorum of declarations *not* to have seen an endorsed *height*-2 QC of the previous view. This proves that no quorum of $2t+1$ SPEED_VOTEs for a *height*-2 block can exist, hence, that no *height*-2 block of the previous view could be speed-DecCert'ed. We now formalize the idea on the example of s2PAC^{lean}. It is described in Figure 9.

[4]there is still an added value of 2PAC there, which is that we further allow to build over a height-2 block: $b_{v-1,L_{v-1},1}$, even if $\mathcal{P}_1$ saw an endorsed QC only over the height-1 parent $b_{v-1,L_{v-1},1}$, where 2-chain_VABA would have instructed to build over the parent $b_{v-1,L_{v-1},1}$.

In s2PAC^{lean} there are two ways to decide a block and all its ancestors, i.e., two types of DecCerts. The first type is the same as in 2PAC^{lean}, i.e., a 2-chain of endorsed QCs of the same view is a DecCert for the *height*-1 block:

- **ordinary** DecCert For any view number w, a DecCert for a view-w *height*-1 block with *proposer* $\mathcal{P}_\ell = L_w$, $b_{w,\ell,1}$, is: a 2-chain of endorsed QCs $\mathcal{C}_{w,\ell,1} \leftarrow \mathcal{C}_{w,\ell,2}$, i.e., QCs on the blocks $b_{w,\ell,1} \leftarrow b_{w,\ell,2}$ where $b_{w,\ell,2}$ is a view-w *height*-2 *proposer* $\mathcal{P}_\ell$ block.

The new type of DecCerts in s2PAC^{lean} is over *height*-2 blocks:

- **A speed**-DecCert **for a *height*-2 ("pipelined") block** $\mathcal{C}_{v,\ell,1} \leftarrow b_{v,\ell,2} = \{v, \ell, 2, \mathcal{C}_{v,\ell,1}, txn, id\}$ proposed by the leader $\mathcal{P}_\ell$ of view v, is a $2t+1$ multi- (or threshold) signature on the message $(\text{SPEED_VOTE}; v, \ell, 2, id)$.

5 2PAC$^{\mathbf{BIG}}$ and s2PAC$^{\mathbf{BIG}}$

The modifications to go from 2PAC^{lean} to 2PAC^{BIG}, and likewise to go from s2PAC^{lean} to s2PAC^{BIG} are simply as follows: players now multicast their votes; they collect the votes for all proposers $j \in [n]$ and form their QCs themselves; finally a proposer i multicasts a *height*-2 block: $b_{v,i,2}$ without waiting to receive a QC on its *height*-1 parent block: $b_{v,i,1}$. So the child $b_{v,i,2}$ does not refer anymore to its parent $b_{v,i,1}$ via a QC, but instead simply via a hash (the preimage having been diffused previously). We emphasize that despite this no-waiting, QCs over blocks in 2PAC^{BIG} are still proofs that all their ancestors have been seen by $t+1$ processes, hence are efficiently retrievable.

Since 2PAC^{BIG}/s2PAC^{BIG} reduces by 2δ the latency per view compared to 2PAC^{lean}/s2PAC^{lean}, and since there are 1.5 views in expectation for a single-shot MVBA, it follows that 2PAC^{BIG}/s2PAC^{BIG} have $9.5\delta - 1.5 \times 2\delta = 6.5\delta$ (worst-case) expected latency. We formalize its properties as the following theorem. The proofs of consistency are identical to the ones of 2PAC^{lean} and s2PAC^{lean} (the key point is that processes wait to receive a *height*-1 QC on the parent before voting for a *height*-2 block, so we are brought back to the situation of 2PAC where the *height*-1 QC was wrapped in the *height*-2 block).

s2PAC^{BIG} is further formalized in Figure 10, the footnote symbols are pointers to the following comments. $(*)$: this could be sped-up by: **upon** receiving $t+1$ proposals (at the cost of potentially including less transactions in $b_{v,2,i}$). (†): processes could possibly multicast the QCs for further speedup, but this is not needed to match any of our latency bounds. Multicasting the SPEED_VOTE is necessary to make *all* processes decide the pipelined block within 3δ.

The proofs of the latencies (expected, good-case, and pipelined) are also identical to the ones of 2PAC^{lean} and s2PAC^{lean}, so we also skip them.

THEOREM 2. *2PAC^{BIG} & s2PAC^{BIG} are blockchain consensus. When used for a single-shot MVBA, they both have 6.5δ (worst-case) expected latency; and 4δ good-case latency (Def. 3).*

- In s2PAC^{BIG}, if the scheduler is fair, then a block pipelined by an honest leader at a time τ gets decided by all *players by $\tau + 3\delta$.*

2PAC^{lean}

Instructions for each process $\mathcal{P}_i$, with local view number denoted v. At the beginning of the protocol it enters $v = 1$, i.e., sets $v \leftarrow 1$. Upon entering a view v, it initializes the flags *proposedHeight1* and *proposedHeight2* to 0.

New view and Report. **upon** entering a view $v \geqslant 2$

if it has a view-$(v-1)$ *height*-1 endorsed QC: $\mathcal{C}_{v-1,\ell,1} \leftarrow b_{v-1,\ell,2}$ (wrapped in a *height*-2 block of $\ell = L_{v-1}$)
it multicasts it;

else it freely chooses any view-$(v-1)$ *height*-2 QC: $\mathcal{C}_{v-1,\ell,2}$
and multicasts it appended with a signed declaration $\{\text{no_endorsed_height-1_QC}, v\}_i$.

Propose *height*-1. While *proposedHeight1* $= 0$
upon obtaining an endorsed view-$(v-1)$ *height*-1 QC wrapped in a *height*-2 block: $\mathcal{C}_{v-1,\ell,1} \leftarrow b_{v-1,\ell,2}$ $(\ell = L_{v-1})$
▷ It might already have one upon entering v.

creates a child view-v block: $\mathcal{C}_{v-1,\ell,1} \leftarrow b_{v-1,\ell,2} \leftarrow \boldsymbol{b_{v,i,1}} = \{v, i, 1, \mathrm{H}(b_{v-1,\ell,2}), txn, id\}$
▷ Optimization (will become standard in s2PAC^{lean}): in case it has a QC on $b_{v-1,\ell,2}$, then shortens the reference as: $\mathcal{C}_{v-1,\ell,2} \leftarrow b_{v,i,1}$.

multicasts $\langle \mathcal{C}_{v-1,\ell,1} \leftarrow b_{v-1,\ell,2} \leftarrow \boldsymbol{b_{v,i,1}} \rangle_i$ and sets $1 \leftarrow$ *proposedHeight1*.

upon receiving from $2t+1$ processes $\mathcal{P}_j$'s: $\{\text{no_endorsed_height-1_QC}, v\}_j$
freely chooses a view-$(v-1)$ *height*-2 QC, forms a child block: $\mathcal{C}_{v-1,2} \leftarrow \boldsymbol{b_{v,i,1}} = \{v, i, 1, \mathcal{C}_{v-1,2}, txn, id\}$
combines the signed declarations into $\{\text{no_endorsed_height-1_QC}, v\}$
▷ PnD (proof of non-decision): proof that no view-$(v-1)$-DecCert can ever exist, thus $\mathcal{C}_{v-1,2}$ can safely be extended.

multicasts $\left(\langle \mathcal{C}_{v-1,2} \leftarrow \boldsymbol{b_{v,i,1}} \rangle_i, \{\text{no_endorsed_height-1_QC}, v\}\right)$ and sets $1 \leftarrow$ *proposedHeight1*.

Vote *height*-1. $\forall j \in [n]$, **upon** receiving the first proposal for a view-v *height*-1 *proposer-j* block $\boldsymbol{b_{v,1,j}}$, i.e., of the form:

- either the raw signed block $\langle \mathcal{C}_{v-1,1} \leftarrow b_{v-1,2} \leftarrow \boldsymbol{b_{v,j,1}} = \{v, j, 1, b_{v-1,2}, txn, id\} \rangle_j$ where $\mathcal{C}_{v-1,\ell=L_{v-1},1}$ is endorsed;
- or, appended with a PnD, i.e., $\left(\langle \mathcal{C}_{v-1,2} \leftarrow \boldsymbol{b_{v,j,1}} = \{v, j, 1, \mathcal{C}_{v-1,2}, txn, id\} \rangle_j, \{\text{no_endorsed_height-1_QC}, v\}\right)$;

then votes for the block by replying $\{v, j, 1, id\}_i$ to $\mathcal{P}_j$.

QC *height*-1 & Propose *height*-2. if *proposedHeight2* $= 0$, **upon** receiving $2t+1$ votes for its *height*-1 block $\boldsymbol{b_{v,i,1}}$:
combines the $2t+1$ signatures into a view-v *height*-1 QC: $\boldsymbol{\mathcal{C}_{v,i,1}} = \{v, i, 1, id\}$ on $\boldsymbol{b_{v,i,1}}$
creates a child *height*-2 block: $\boldsymbol{\mathcal{C}_{v,i,1}} \leftarrow \boldsymbol{b_{v,i,2}} = \{v, i, 2, \boldsymbol{b_{v,i,1}}, txn, id\}$, signs then multicasts it, then sets $1 \leftarrow$ *proposedHeight2*.

Vote *height*-2. $\forall j \in [n]$, **upon** receiving the first *height*-2 proposal: $\langle \boldsymbol{\mathcal{C}_{v,j,1}} \leftarrow \boldsymbol{b_{v,j,2}} = \{v, j, 2, \boldsymbol{\mathcal{C}_{v,j,1}}, txn, id\} \rangle_j$:
▷ i.e., a view-v *height*-1 *proposer-j* QC, wrapped in a *height*-2 block $\boldsymbol{b_{v,j,2}}$ of same proposer j
votes for the *height*-2 block by replying $\{v, j, 2, id\}_i$ to $\mathcal{P}_j$.

QC *height*-2. For a view number $w \in \{v-1, v\}$, **upon** receiving $2t+1$ votes for its view-w proposal $\boldsymbol{b_{w,i,2}}$,
▷ i.e. $\{w, i, 2, id\}_j$'s signed by distinct processes $\mathcal{P}_j$'s
combines the $2t+1$ signatures into a view-w *height*-2 QC, $\boldsymbol{\mathcal{C}_{w,i,2}}$.
if $w = v$, it multicasts it.
▷ $w = v-1$ allows a slow L_{v-1} to form a *height*-2 QC while in view v: this will be used in the proof of 6δ good-case latency (Appendix A.3.3).

Leader election. **upon** receiving $2t+1$ view-v *height*-2 QCs of distinct proposers,
multicasts its view-v coin share.

Coin QC and Advance view. **upon** receiving or forming a view-v coin-QC, then it enters view $v+1$
and multicasts the coin-QC ▷ can be replaced by Bracha-amplification, cf Appendix B.1

Decision. **upon** receiving or forming a DecCert for any view number w, i.e., a 2-chain $\boldsymbol{\mathcal{C}_{w,j,1}} \leftarrow \boldsymbol{\mathcal{C}_{w,j,2}}$ s.t. $j = L_w$
it multicasts it and outputs it.

Figure 8

s2PAC^{lean}: with Fast Decision of Pipelined Blocks

Instructions for each process $\mathcal{P}_i$, with local view number denoted v. At the beginning of the protocol, it enters $v = 1$, i.e., sets $v \leftarrow 1$. Upon entering a view v, it initializes the flags *proposedHeight1* and *proposedHeight2* to 0.

New view and Report. Upon entering a view $v \geqslant 2$

if it has a view-$(v-1)$ *height*-2 endorsed QC: $\mathcal{C}_{v-1,2}$, it multicasts it.

elseif it has a view-$(v-1)$ *height*-1 endorsed QC: $\mathcal{C}_{v-1,\ell,1} \leftarrow b_{v-1,\ell,2}$ (wrapped in a *height*-2 block of $\ell = L_{v-1}$),
it multicasts it, appended with a signed declaration $\{\text{no_endorsed_height-2_QC}, v\}_i$

else multicasts $(\text{report}, \{\text{no_endorsed_height-1_QC}, v\}_i)$.

Propose *height*-1. While *proposedHeight1* = 0 (throughout we denote $\mathcal{P}_\ell = L_{v-1}$)

upon obtaining a view-$(v-1)$ *height*-2 endorsed QC: $\mathcal{C}_{v-1,\ell,2}$
creates a child view-v block: $\mathcal{C}_{v-1,\ell,2} \leftarrow \boldsymbol{b_{v,i,1}} = \{v, i, 1, \mathcal{C}_{v-1,\ell,2}, txn, id\}$, multicasts it and sets $1 \leftarrow$ *proposedHeight1*.

upon receiving from $2t+1$ processes $\mathcal{P}_j$'s: $\{\text{no_endorsed_height-2_QC}, v\}_j$

chooses any view-$(v-1)$ *height*-1 endorsed QC: $\mathcal{C}_{v-1,\ell,1} \leftarrow b_{v-1,\ell,2}$ (wrapped in a *height*-2 block of $\mathcal{P}_\ell$)
creates a child view-v block: $\mathcal{C}_{v-1,\ell,1} \leftarrow b_{v-1,\ell,2} \leftarrow \boldsymbol{b_{v,i,1}} = \{v, i, 1, b_{v-1,\ell,2}, txn, id\}$
combines the signed declarations into $\{\text{no_endorsed_height-2_QC}, v\}$ ▷ proof that no view-$(v-1)$-speed-DecCert can ever exist.
multicasts $(\langle \mathcal{C}_{v-1,1} \leftarrow b_{v-1,2} \leftarrow b_{v,i,1} \rangle_i, \{\text{no_endorsed_height-2_QC}, v\})$ and sets $1 \leftarrow$ *proposedHeight1*.

upon receiving from $2t+1$ processes: $(\text{report}, \{\text{no_endorsed_height-1_QC}, v\}_j)$

chooses any view-$(v-1)$ *height*-2 QC, forms a child block: $\mathcal{C}_{v-1,2} \leftarrow b_{v,i,1} = \{v, i, 1, \mathcal{C}_{v-1,2}, txn, id\}$
combines the signed declarations into $\{\text{no_endorsed_height-1_QC}, v\}$
▷ proof that no $(v-1)$ DecCert can ever exist, thus $\mathcal{C}_{v-1,2}$ can safely be extended.
multicasts $(\langle \mathcal{C}_{v-1,2} \leftarrow b_{v,i,1} \rangle_i \ \{\text{no_endorsed_height-1_QC}, v\})$ and sets $1 \leftarrow$ *proposedHeight1*.

Vote *height*-1. For any $j \in [n]$, **upon** receiving for the first time a well-formed proposal for a view-v *height*-1 *proposer-j* block $\boldsymbol{b_{v,j,1}} = \{v, j, 1, parent, txn, id\}$, i.e., of one of the following forms:

either the raw signed block $\langle \mathcal{C}_{v-1,\ell,2} \leftarrow \boldsymbol{b_{v,j,1}} \rangle_j$ with $\mathcal{C}_{v-1,\ell,2}$ endorsed,

or, of the form $(\langle \mathcal{C}_{v-1,\ell,1} \leftarrow b_{v-1,2} \leftarrow \boldsymbol{b_{v,j,1}} \rangle_j, \{\text{no_endorsed_height-2_QC}, v\})$ with $\mathcal{C}_{v-1,\ell,1}$ endorsed,

or, of the form $(\langle \mathcal{C}_{v-1,2} \leftarrow \boldsymbol{b_{v,j,1}} \rangle_j, \{\text{no_endorsed_height-1_QC}, v\})$;

then it votes for the *height*-1 block by replying $\{v, j, 1, id\}_i$ to $\mathcal{P}_j$.

QC *height*-1, Vote *height*-2. and QC *height*-2. same as in 2PAC^{lean} (and 2HS/HS-2), cf Figure 8, as well as in s2PAC^{BIG}, cf Figure 10.

Speed-Decision vote (for *height*-2 blocks).

For any $j \in [n]$, **upon** receiving for the first time a view-v *height*-2 *proposer-j* QC: $\mathcal{C}_{v,j,2} = \{w, j, 2, id\}$
it replies $\{\text{SPEED_VOTE}, v, j, 2, id\}_i$ to $\mathcal{P}_j$. ▷ optimization: if furthermore $\mathcal{C}_{v,j,2}$ is endorsed, then it multicasts the SPEED_VOTE.

Leader election. same as in 2PAC^{lean} (and 2HS/HS-2), cf Figure 8, as well as in s2PAC^{BIG}, cf Figure 10.

Coin QC and Advance view. **upon** receiving or forming a view-v coin-QC: $\mathcal{C}_{\text{coin}}$, then it multicasts it (optional) and enters view $v+1$.

Decision. **upon** receiving or forming an ordinary DecCert for any view number w, i.e., *either* a 2-chain $\boldsymbol{\mathcal{C}_{w,j,1}} \leftarrow \boldsymbol{\mathcal{C}_{w,j,2}}$ s.t. $j = L_w$, *or* a speed-DecCert$\{\text{SPEED_VOTE}, \boldsymbol{\mathcal{C}_{w,j,2}}\}$; then it multicasts it and outputs it.

Figure 9: **Differences with 2PAC^{lean} are highlighted.**

s2PAC^{BIG} (differences with s2PAC^{lean} in red)

New view and Report. **upon** entering a view $v \geqslant 2$

if it has a view-$(v-1)$ *height*-2 endorsed QC: $\mathcal{C}_{v-1,2}$, it multicasts it.

else if it has a view-$(v-1)$ *height*-1 endorsed QC: $\mathcal{C}_{v-1,\ell,1} \leftarrow b_{v-1,\ell,2}$ (wrapped in a *height*-2 block of $\ell = L_{v-1}$), it multicasts it, appended with a signed declaration $\{\text{no_endorsed_height-2_QC}, v\}_i$

else multicasts $(\text{report}, \{\text{no_endorsed_height-1_QC}, v\}_i)$.

Propose *height*-1. While $proposedHeight1 = 0$ (throughout we denote $\mathcal{P}_\ell = L_{v-1}$)

upon obtaining a view-$(v-1)$ *height*-2 endorsed QC: $\mathcal{C}_{v-1,\ell,2}$
creates a child view-v block: $\mathcal{C}_{v-1,\ell,2} \leftarrow \boldsymbol{b_{v,i,1}} = \{v, i, 1, \mathcal{C}_{v-1,\ell,2}, txn, id\}$, multicasts it and sets $1 \leftarrow proposedHeight1$.

upon receiving from $2t+1$ processes $\mathcal{P}_j$'s: $\{\text{no_endorsed_height-2_QC}, v\}_j$, then

chooses any view-$(v-1)$ *height*-1 endorsed QC: $\mathcal{C}_{v-1,\ell,1} \leftarrow b_{v-1,\ell,2}$ (wrapped in a *height*-2 block of $\mathcal{P}_\ell$)
creates a child view-v block: $\mathcal{C}_{v-1,\ell,1} \leftarrow b_{v-1,\ell,2} \leftarrow \boldsymbol{b_{v,i,1}} = \{v, i, 1, b_{v-1,\ell,2}, txn, id\}$
combines the signed declarations into $\{\text{no_endorsed_height-2_QC}, v\}$ ▷ proof that no view-$(v-1)$-speed-DecCert can ever exist.
multicasts $(\langle \mathcal{C}_{v-1,1} \leftarrow b_{v-1,2} \leftarrow b_{v,i,1} \rangle_i, \{\text{no_endorsed_height-2_QC}, v\})$ and sets $1 \leftarrow proposedHeight1$.

upon receiving from $2t+1$ processes: $(\text{report}, \{\text{no_endorsed_height-1_QC}, v\}_j)$, then

chooses any view-$(v-1)$ *height*-2 QC, form a child block: $\mathcal{C}_{v-1,2} \leftarrow b_{v,i,1} = \{v, i, 1, \mathcal{C}_{v-1,2}, txn, id\}$
combines the signed declarations into $\{\text{no_endorsed_height-1_QC}, v\}$
▷ proof that no $(v-1)$ DecCert can ever exist, thus $\mathcal{C}_{v-1,2}$ can safely be extended.
multicasts $(\langle \mathcal{C}_{v-1,2} \leftarrow b_{v,i,1} \rangle_i \{\text{no_endorsed_height-1_QC}, v\})$ and sets $1 \leftarrow proposedHeight1$.

Vote *height*-1. For any $j \in [n]$, **upon** receiving for the first time a well-formed proposal for a view-v *height*-1 *proposer*-j block $\boldsymbol{b_{v,j,1}} = \{v, j, 1, parent, txn, id\}$, i.e., of one of the following forms:

either the raw signed block $\langle \mathcal{C}_{v-1,\ell,2} \leftarrow \boldsymbol{b_{v,j,1}} \rangle_j$ with $\mathcal{C}_{v-1,\ell,2}$ endorsed,

or, of the form $(\langle \mathcal{C}_{v-1,\ell,1} \leftarrow b_{v-1,2} \leftarrow \boldsymbol{b_{v,j,1}} \rangle_j, \{\text{no_endorsed_height-2_QC}, v\})$ with $\mathcal{C}_{v-1,\ell,1}$ endorsed,

or, of the form $(\langle \mathcal{C}_{v-1,2} \leftarrow \boldsymbol{b_{v,j,1}} \rangle_j, \{\text{no_endorsed_height-1_QC}, v\})$;

then votes for the *height*-1 block by multicasting $\{v, 1, j, id\}_i$.

Propose *height*-2. if $proposedHeight2 = 0$ **upon** receiving *height*-1 blocks from $2t+1$ proposers $^{(*)}$
creates a child *height*-2 block: $\boldsymbol{b_{v,1,i}} \leftarrow b_{v,2,i}$ and multicasts it, then set $1 \leftarrow proposedHeight2$.

QC *height*-1 $\forall j$: upon receiving $2t+1$ votes for $\mathcal{P}_j$'s *height*-1 block: $\{v, 1, j, id\}_k$'s
combines the $2t+1$ signatures into a view-v *height*-1 QC: $\mathcal{C}_{v,1,j} = \{v, 1, j, id\}$. $^{(\dagger)}$

Vote *height*-2. For any $j \in [n]$, **upon** receiving for the first time a *height*-2 proposal: $\langle \boldsymbol{b_{v,1,j}} \leftarrow \boldsymbol{b_{v,2,j}} = \{v, 2, j, \boldsymbol{b_{v,1,j}}, txn, id\} \rangle_j$
waits until obtaining a $\boldsymbol{\mathcal{C}_{v,1,j}}$ on $\boldsymbol{b_{v,1,j}}$ then votes by multicasting $\{v, 2, j, id\}_i$.

QC *height*-2. $\forall j, \forall w \in \{v-1, v\}$,**upon** receiving for the first time $2t+1$ votes for $\mathcal{P}_j$'s *height*-2 block, $\{w, 2, j, id\}_k$, then
combines the $2t+1$ signatures into a view-w *height*-2 QC: $\mathcal{C}_{w,2,j}$ $^{(\dagger)}$.
If $w = v$, it multicasts it.

Fast Decision vote (for *height*-2 blocks).

For any $j \in [n]$, **upon** receiving for the first time a view-v *height*-2 *proposer*-j QC: $\mathcal{C}_{v,2,\boldsymbol{j}} = \{v, 2, \boldsymbol{j}, id\}$
it multicasts $\{\text{SPEED_VOTE}, v, 2, j, id\}_i$.

Leader election. **upon** receiving $2t+1$ view-v *height*-2 QCs of distinct proposers, *or*, $t+1$ view-v coin shares, then it multicasts its view-v coin share.

Coin QC and Advance view. **upon** receiving or forming a view-v coin-QC, then it enters view $v+1$ and multicasts the coin-QC ▷ can be replaced by Bracha-amplification, cf Appendix B.1

Decision. **upon** receiving or forming an ordinary DecCert for any view number w, i.e., *either* a 2-chain $\boldsymbol{\mathcal{C}_{w,j,1}} \leftarrow \boldsymbol{\mathcal{C}_{w,j,2}}$ s.t. $j = L_w$, *or* a speed-DecCert$\{\text{SPEED_VOTE}, \boldsymbol{\mathcal{C}_{w,j,2}}\}$; then it multicasts it and outputs it.

Figure 10

6 Computational Evaluation

We implemented[5] s2PAC^{lean} and s2PAC^{BIG}-DAG in Go, as well as a chained version of sMVBA [48] and of GradedDAG [24]. We did benchmarks on AWS with the following parameters. Transaction size: 512B payload size: 1000 × 512B queue capacity: 100000 maximum number of payloads contained in a block: 32 Ubuntu 20.04 LTS Instance type: m5d.xlarge processes were located in the four AWS regions: us-east-1, eu-north-1, ap-northeast-1, ap-southeast-2. Each point is averaged over a 30 secondes-long experiment, as in [24, 21].

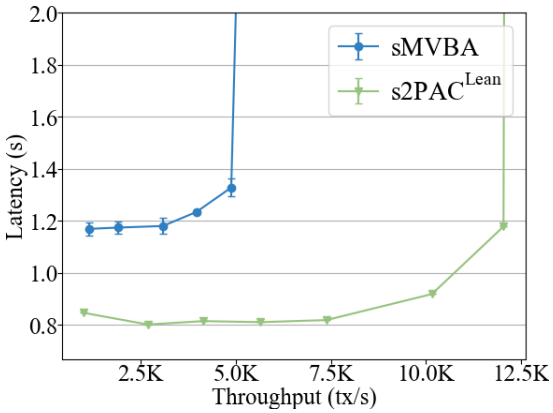

Figure 11: **Latency-throughput comparison between** s2PAC^{lean} **and chained sMVBA, fault-free**

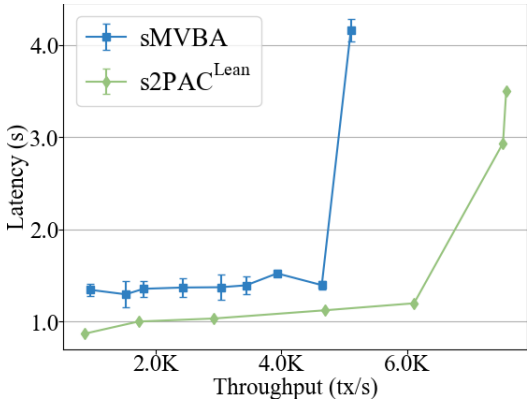

Figure 12: **Latency-throughput comparison between** s2PAC^{lean} **and chained sMVBA, with** 3 **crashes**

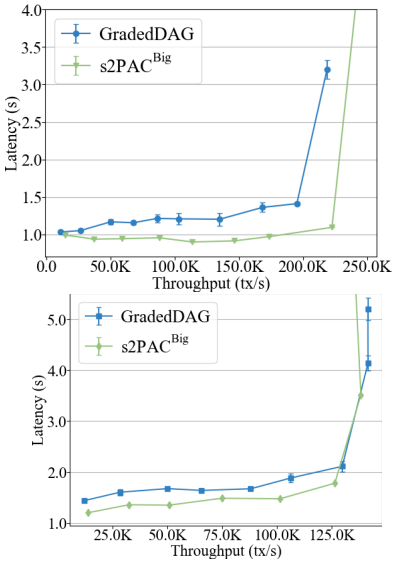

Figure 13: **Latency-throughput comparison between** s2PAC^{BIG}**-DAG and GradedDAG: fault-free (left), and with** 3 **crashes (right).**

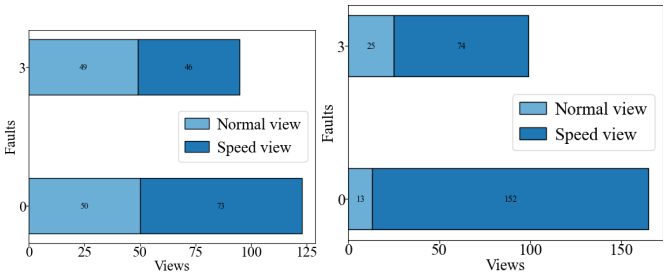

Figure 14: **Percentage of the views in which a speed-**DecCert **was formed: in** s2PAC^{lean} **(left) and** s2PAC^{BIG}**-DAG (right).**

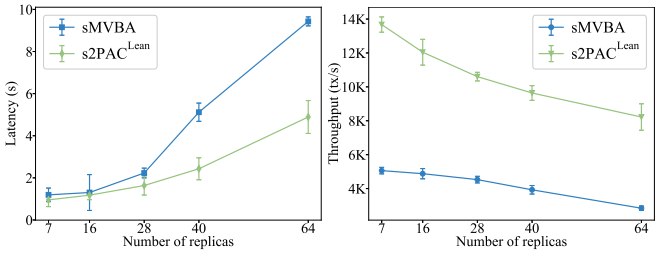

Figure 15: **Scale vs latency (left) and scale vs throughput (right) comparison between** s2PAC^{lean} **and chained sMVBA.**

[5]https://github.com/ac-dcz/s2PAC

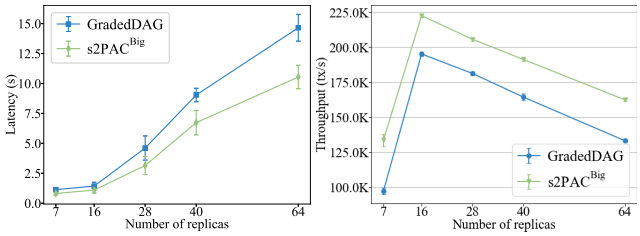

Figure 16: **Scale vs latency (left) and scale vs throughput (right) comparison between s2PAC^{BIG}-DAG and GradedDAG.**

Each point on the scalability graph was roughly estimated as the inflection point of the latency-throughput graph of the slowest of the two protocols compared, i.e., sMVBA or GradedDAG. Due to budget constraints, we were limited to 32 instances, so for the higher values: $n = 40$ and $n = 64$, we ran two nodes on the same instance.

7 Further related work

7.1 GradedDAG. In [24, TABLE I] of GradedDAG, the expected latency is stated as 7.5δ. We estimate it instead to 6.5δ for a single-shot consensus, a.k.a. an MVBA. This is because their views have 5 rounds, in which voting is in the 4-th round, then the 5th round, which serves only to prepare the next view. We identified fault-free scenarios in GradedDAG, with a half-fair scheduler such that $\frac{3}{2}\delta_{\text{fast}} < \delta < 2\delta_{\text{fast}}$, where δ_{fast} is the fast message delay, such that no decision is reached in the first view. The scenario is that the process to-be-elected leader, say $\mathcal{P}_4$, is slower so that others refuse to vote to form a *height*-2 QC (in our terminology) for $\mathcal{P}_4$. Since this scenario can repeat in all higher views with very bad luck, this explains the stated ∞ good-case latency in Table 17. The scenario is formalized and illustrated in Appendix E.3. The subtle difference enabling 2PAC^{BIG} to achieve 4δ latency in all good-case executions, hence not falling in the previous ∞ scenarios of GradedDAG, is that processes accept to vote for *height*-2 blocks *even after* they received $2t+1$ *height*-2 QCs.

Another reason for the practical gains of s2PAC^{BIG}-DAG over GradedDAG (Sec. 6) is the simplicity of its design: processes in 2PAC^{BIG}-DAG *need not wait* to retrieve the referenced blocks before proceeding, since QCs guarantee their (efficient) recoverability [27]. This enables 2PAC^{BIG}-DAG to follow the "lazy ordering" paradigm [71], i.e., of ordering without checking validity, offloading to clients the task of pruning invalid transactions. Hence, 2PAC^{BIG}-DAG does not require the complex rules of lightDAG2 [21], preventing equivocation in the tree of referenced blocks. A last gain of 2PAC^{BIG}-DAG over GradedDAG is the fast decision of pipelined blocks, while GradedDAG cannot, due to its pipelined block being proposed too late (3 messages after the *height*-1 block, vs 1 message in 2PAC^{BIG}).

We compare in Table 17 the latencies of s2PAC^{BIG} a single-shot consensus, with those of other cubic protocols.

7.2 Comparison with the follow-up work TockOwl. In Appendix B of the follow-up work [55], the authors present a chained consensus with quadratic complexity, called TockCat, which has a claimed expected latency of 9.5, thus matching the one of s2PAC^{lean}. TockCat can be described as a chained version of sMVBA, but without pipelined blocks, and in which the two rounds to create a proof of safe proposal, are shrunk into one round. Hence, it can be seen from their experiments that TockCat has a latency and throughput comparable to a chained-but-non-pipelined sMVBA. While their threshold signature over the message no-leader is equal to the PnD in 2PAC^{lean}, their proof of safe proposal is more complex than just the PnD. Indeed, following sMVBA, when a process $\mathcal{P}_i$ in TockCat proposes a child of a non-endorsed block of the previous view: $b_{v-1,j,1} \leftarrow b_{v,i,1}$, then the parent must be the block proposed by $\mathcal{P}_i$ in view $v-1$, i.e., $j = i$, and must *itself* be appended with a further proof of safe proposal of the $b_{v-1,j,1}$ in view $v-1$. Comparably, in 2PAC^{lean} no such further proof is required, because the parent block: $b_{v-1,j,*}$ is pointed to via its QC, which itself is an (indirect) proof of safe proposal of $b_{v-1,j,*}$. This is the advantage (1) explained in Sec. 1.1.2. Furthermore, in 2PAC^{lean}, the parent is a *height*-2 block: $b_{v-1,j,*=2}$, allowing the chain to be denser, as explained in (2) of Sec. 1.1.2. Hence, our experiments show that s2PAC^{lean} has twice larger throughput than a *pipelined* version of sMVBA, while pushing the latency even lower, and thus the gap

latencies, in δ	Worst-case expected	Good-case (Def. 3)	Best-case
Bullshark [70]	30	_	6
Cordial Miners [51]	7.5	_	5 $^{(*)}$
GradedDag [24] $^{(\dagger)}$	6.5	∞	4
LightDAG2 [21] $^{(\ddagger)}$	$12(t'+1)$	∞	4
2PAC^{BIG}	**6.5**	**4**	**4**

Table 17: **MVBAs with $O(n^3)$ messages.** $(*)$ Cordial Miners has best-case latency 5δ, since leader election is in the 5-th round. In [51, Table 1, Async.] it is stated 5δ in a "good case", we did not analyze it under a half-fair scheduler (nor did we analyze [70] under a half-fair scheduler). (†) [24] see below. (‡) The "best-case" figures for [70, 21] are taken from [21, TABLE 1]. t' denotes the actual number of cheaters. The ∞ in LightDAG2 was confirmed by the authors as following from the same scenario as the ∞ of GradedDAG.

unit: δ	Fair scheduler	
	Arbitrary corruptions	Fault-free
sMVBA [48] (chained)	14	10
2PAC^{lean}	13.5	10
s2PAC^{lean} (Thm 8)	**4**	**4**
$O(n^3)$ messages:		
Cordial Miners [51]	11.5	9
GradedDag [24]	9.5	6
2PAC^{BIG}	9.5	7
s2PAC^{BIG} (Thm 2)	**3**	**3**

Table 18: **Decision delay of a ("pipelined") *height*-2 block by an honest leader. Fast decisions in s2PAC^{BIG} actually happen in many more scenarios than a fair scheduler, as confirmed by our experiments Figure 14.**

should be even wider against the non-pipelined version of sMVBA/TockCat.

Note that the throughput of TockCat could not simply be doubled without changing the latency, by naively pipelining a block in the second phase of the broadcast in TockCat ([55, Alg. 5] line 6), for at least two reasons. *First*, in the scenario where $\mathcal{P}_i$ would not have obtained a *height*-2 QC of the previous view neither over the leader's *height*-block nor over its own *height*-2 block, then such a naive pipelined TockCat would be blocked (since TockCat imposes $\mathcal{P}_i$ to propose a child of a block proposed by itself in the previous view $v-1$). *Second*, because the pipelined blocks would be decided at best in the next view, thus hitting the average latency. Instead, the speedup of the decision of pipelined blocks in s2PAC (Sec 1.2.1 and 4) enables to preserve the average latency, since it hits nearly all pipelined blocks: this is illustrated by Table 18 and confirmed experimentally by Figure 14.

More generally, we believe that the s2PAC design is of general interest because it also applies to protocols with a cubic complexity and DAG structure, as demonstrated by s2PAC^{BIG}.

Acknowledgements

Matthieu Rambaud thanks Julian Loss for notifying of 2-chain_VABA, the authors of 2-chain_VABA especially Zhuolun Xiang for their responsiveness, Zhenliang Lu for exchanges on sMVBA, Nathan Rouillé for useful comments on the paper, and is supported by ANR-21-CE39-0009-BARRACUDA.

References

[1] Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. 2023. Communication complexity of byzantine agreement, revisited. *Distributed Comput.*

[2] Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. 2019. Asymptotically optimal validated asynchronous byzantine agreement. In *PODC*.

[3] Ignacio Amores-Sesar and Christian Cachin. 2023. We will DAG you. In *Esorics Workshops*.

[4] Aptos. 2024. Implementation of aptos consensus, following diem. https://github.com/aptos-labs/aptos-core/blob/main/consensus/consensus-types/src/timeout_2chain.rs Retrieved on June 23, 2024. (2024).

[5] Balaji Arun, Zekun Li, Florian Suri-Payer, Sourav Das, and Alexander Spiegelman. 2024. Shoal++: high throughput dag bft can be fast! (2024).

[6] Renas Bacho and Julian Loss. 2022. On the adaptive security of the threshold bls signature scheme. In *CCS*.

[7] Renas Bacho, Julian Loss, Gilad Stern, and Benedikt Wagner. 2024. Harts: high-threshold, adaptively secure, and robust threshold schnorr signatures. In.

[8] Michael Ben-Or, Ran Canetti, and Oded Goldreich. 1993. Asynchronous secure computation. In *STOC*.

[9] Michael Ben-Or and Ran El-Yaniv. 2003. Resilient-optimal interactive consistency in constant time. *Distributed Comput.*

[10] Michael Ben-Or, Boaz Kelmer, and Tal Rabin. 1994. Asynchronous secure computations with optimal resilience (extended abstract). In *PODC*.

[11] Erica Blum, Jonathan Katz, Julian Loss, Kartik Nayak, and Simon Ochsenreither. 2023. Abraxas: throughput-efficient hybrid asynchronous consensus. In *CCS*.

[12] Alexandra Boldyreva. 2003. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC*. Latest long version at https://faculty.cc.gatech.edu/~aboldyre/papers/b.pdf.

[13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. 2013. Key homomorphic prfs and their applications. In *CRYPTO*.

[14] Ethan Buchman. 2016. *Tendermint: Byzantine Fault Tolerance in the Age of Blockchains*. Ph.D. Dissertation. University of Guelph.

[15] Vitalik Buterin and Virgil Griffith. 2017. Casper the friendly finality gadget. *arxiv 1710.09437*.

[16] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. 2001. Secure and efficient asynchronous broadcast protocols. In *CRYPTO*.

[17] Christian Cachin, Klaus Kursawe, and Victor Shoup. 2005. Random oracles in constantinople: practical asynchronous byzantine agreement using cryptography. *J. Cryptol.*

[18] Miguel Castro and Barbara Liskov. 1999. Practical byzantine fault tolerance. In *OSDI*.

[19] Benjamin Y Chan and Rafael Pass. 2023. Simplex consensus: a simple and fast consensus protocol. In *TCC*.

[20] Hao Cheng, Yuan Lu, Zhenliang Lu, Qiang Tang, Yuxuan Zhang, and Zhenfeng Zhang. 2024. Jumbo: fully asynchronous bft consensus made truly scalable. (2024).

[21] Xiaohai Dai, Guanxiong Wang, Jiang Xiao, Zhengxuan Guo, Rui Hao, Xia Xie, and Hai Jin. 2024. Lightdag: a low-latency dag-based bft consensus through lightweight broadcast. In *IPDPS*.

[22] Xiaohai Dai, Bolin Zhang, Hai Jin, and Ling Ren. 2023. Parbft: faster asynchronous bft consensus with a parallel optimistic path. ePrint 2023/679. (2023).

[23] Xiaohai Dai, Zhaonan Zhang, Zhengxuan Guo, Chaozheng Ding, Jiang Xiao, Xia Xie, Rui Hao, and Hai Jin. 2024. Wahoo: A dag-based BFT consensus with low latency and low communication overhead. *IEEE Trans. Inf. Forensics Secur.*

[24] Xiaohai Dai, Zhaonan Zhang, Jiang Xiao, Jingtao Yue, Xia Xie, and Hai Jin. 2023. Gradeddag: an asynchronous dag-based bft consensus with lower latency. In *SRDS*.

[25] George Danezis, Eleftherios Kokoris Kogias, Alberto Sonnino, and Alexander Spiegelman. 2022. Narwhal and tusk: a dag-based mempool and efficient bft consensus. In *EuroSys*.

[26] Sourav Das and Ling Ren. 2023. Adaptively secure bls threshold signatures from ddh and co-cdh. ePrint 2023/1553. (2023).

[27] Sourav Das, Zhuolun Xiang, and Ling Ren. 2021. Asynchronous data dissemination and its applications. In *CCS*.

[28] Diem. 2021. *DiemBFT v4: State Machine Replication in the Diem Blockchain*. https://developers.diem.com/papers/diem-consensus-state-machine-replication-in-the-diem-blockchain/2021-08-17.pdf.

[29] Isaac Doidge, Raghavendra Ramesh, Nibesh Shrestha, and Joshua Tobkin. 2024. Moonshot: optimizing block period and commit latency in chain-based rotating leader bft. In *DSN*.

[30] Sisi Duan, Michael K. Reiter, and Haibin Zhang. 2018. Beat: asynchronous bft made practical. In *CCS*.

[31] Sisi Duan, Xin Wang, and Haibin Zhang. 2023. Fin: practical signature-free asynchronous common subset in constant time. In *CCS*.

[32] Sisi Duan, Haibin Zhang, Xiao Sui, Baohan Huang, Changchun Mu, Gang Di, and Xiaoyun Wang. 2024. Dashing and star: byzantine fault tolerance with weak certificates. In.

[33] Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. 1988. Consensus in the presence of partial synchrony. *J. ACM*.

[34] Muhammed F. Esgin, Ron Steinfeld, Dongxi Liu, and Sushmita Ruj. 2023. Efficient hybrid exact/relaxed lattice proofs and applications to rounding and vrfs. In *Crypto*.

[35] Hanwen Feng, Zhenliang Lu, Tiancheng Mai, and Qiang Tang. 2025. Faster hash-based multi-valued validated asynchronous byzantine agreement. (2025).

[36] Hanwen Feng and Qiang Tang. 2025. Asymptotically optimal adaptive asynchronous common coin and DKG with silent setup. In *CRYPTO*.

[37] Matthias Fitzi and Martin Hirt. 2006. Optimally efficient multi-valued byzantine agreement. In *Podc*.

[38] Nils Fleischhacker, Gottfried Herold, Mark Simkin, and Zhenfei Zhang. 2023. Chipmunk: better synchronized multi-signatures from lattices. In *CCS*.

[39] Fangyu Gai, Jianyu Niu, Ivan Beschastnikh, Chen Feng, and Sheng Wang. 2023. Scaling blockchain consensus via a robust shared mempool. In *ICDE*.

[40] Yingzi Gao, Yuan Lu, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. 2022. Dumbo-ng: fast asynchronous BFT consensus with throughput-oblivious latency. In *CCS*.

[41] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The bitcoin backbone protocol: analysis and applications. In *EUROCRYPT*.

[42] Rati Gelashvili, Lefteris Kokoris-Kogias, Alberto Sonnino, Alexander Spiegelman, and Zhuolun Xiang. 2022. Jolteon and ditto: network-adaptive efficient consensus with asynchronous fallback. In *FC*. we refer to the 18 June 2021 version on arxiv.

[43] Rati Gelashvili, Lefteris Kokoris-Kogias, Alberto Sonnino, Alexander Spiegelman, and Zhuolun Xiang. 2024. Jolteon and ditto: network-adaptive efficient consensus with asynchronous fallback. In version 2024-07-09. Note that the 2023-12 version fixes the Ditto of the FC'22 version, and the 2024-04-30 further fixed a bug in Jolteon.

[44] Rati Gelashvili, Lefteris Kokoris-Kogias, Alexander Spiegelman, and Zhuolun Xiang. 2021. Brief announcement: be prepared when network goes bad: an asynchronous view-change protocol. In *PODC*.

[45] Guy Golan-Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael K. Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. 2019. SBFT: A scalable and decentralized trust infrastructure. In *DSN*.

[46] Guy Goren, Yoram Moses, and Alexander Spiegelman. 2023. Probabilistic indistinguishability and the quality of validity in byzantine agreement. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*.

[47] Vincent Gramoli, Zhenliang Lu, Qiang Tang, and Pouriya Zarbafian. 2024. Optimal asynchronous byzantine consensus with fair separability. ePrint. (2024).

[48] Bingyong Guo, Yuan Lu, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. 2022. Speeding dumbo: pushing asynchronous bft closer to practice. In *NDSS*.

[49] Mohammad M. Jalalzai, Jianyu Niu, Chen Feng, and Fangyu Gai. 2023. Fast-hotstuff: a fast and resilient hotstuff protocol. *IEEE Transactions on Dependable and Secure Computing*.

[50] Alireza Kavousi, Zhipeng Wang, and Philipp Jovanovic. 2024. Sok: public randomness. In *EuroS&P*.

[51] Idit Keidar, Oded Naor, Ouri Poupko, and Ehud Shapiro. 2023. Cordial miners: fast and efficient consensus for every eventuality. In *DISC*.

[52] Irakliy Khaburzaniya, Konstantinos Chalkias, Kevin Lewi, and Harjasleen Malvai. 2022. Aggregating and thresholdizing hash-based signatures using starks. In *Asia CCS*.

[53] Andrew Lewis-Pye and Ittai Abraham. 2023. Fever: optifmal responsive view synchronisation. In *Opodis*.

[54] Chenxing Li, Peilun Li, Dong Zhou, Zhe Yang, Ming Wu, Guang Yang, Wei Xu, Fan Long, and Andrew Chi-Chih Yao. 2020. A decentralized blockchain with high throughput and fast confirmation. In *USENIX ATC*.

[55] Minghang Li, Qianhong Wu, Zhipeng Wang, Bo Qin, Bohang Wei, Hang Ruan, Shihong Xiong, and Zhenyang Ding. 2025. Tockowl: asynchronous consensus with fault and network adaptability. In *Usenix*.

[56] Chao Liu, Sisi Duan, and Haibin Zhang. 2020. EPIC: efficient asynchronous BFT with adaptive security. In *DSN*.

[57] Yuan Lu, Zhenliang Lu, and Qiang Tang. 2022. Bolt-dumbo transformer: asynchronous consensus as fast as the pipelined bft. In *CCS*.

[58] Yuan Lu, Zhenliang Lu, Qiang Tang, and Guiling Wang. 2020. Dumbo-mvba: optimal multi-valued validated asynchronous byzantine agreement, revisited. In *PODC*.

[59] Dahlia Malkhi and Kartik Nayak. 2023. Extended abstract: hotstuff-2: optimal two-phase responsive bft. ePrint 2023/397. (2023).

[60] Dahlia Malkhi, Chrysoula Stathakopoulou, and Maofan Yin. 2024. BBCA-CHAIN: one-message, low latency BFT consensus on a DAG. In *FC*.

[61] Easwar Vivek Mangipudi and Aniket Pundlik Kate. 2022. D-kode: distributed mechanism to manage a billion discrete-log keys. In *FC*.

[62] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. 2016. The honey badger of BFT protocols. In *CCS*.

[63] Achour Mostéfaoui, Hamouma Moumen, and Michel Raynal. 2015. Signature-free asynchronous binary byzantine consensus with t < n/3, o(n2) messages, and o(1) expected time. *J. ACM*.

[64] Achour Mostéfaoui and Michel Raynal. 2017. Signature-free asynchronous byzantine systems: from multivalued to binary consensus with t< n/3, o(n^2) messages, and constant time. *Acta Informatica*.

[65] Rafael Pass and Elaine Shi. 2018. Thunderella. In *EUROCRYPT*.

[66] Thomas Ristenpart and Scott Yilek. 2007. The power of proofs-of-possession: securing multiparty signatures against rogue-key attacks. In *EUROCRYPT*.

[67] Nibesh Shrestha, Rohan Shrothrium, Aniket Kate, and Kartik Nayak. 2025. Sailfish: towards improving latency of dag-based bft. In *SP*.

[68] Alberto Sonnino. 2021. Implementation of jolteon. https://github.com/asonnino/hotstuff/. (2021).

[69] Alexander Spiegelman, Balaji Aurn, Rati Gelashvili, and Zekun Li. 2023. Shoal: improving DAG-BFT latency and robustness. In *FC*.

[70] Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. 2022. Bullshark: DAG BFT protocols made practical. In *CCS*.

[71] Ertem Nusret Tas, Dionysis Zindros, Lei Yang, and David Tse. 2024. Light clients for lazy blockchains. In *FC*.

[72] Alin Tomescu, Robert Chen, Yiming Zheng and Ittai Abraham, Benny Pinkas, Guy Golan-Gueta, and Srinivas Devadas. 2020. Towards scalable threshold cryptosystems. In *IEEE S&P*.

[73] Zhuolun Xiang. 2021. Implementation of the fallback of ditto. https://github.com/danielxiangzl/Ditto/blob/main/consensus/src/fallback.rs Retrieved on August 17 2023. (2021).

[74] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. 2019. Hotstuff: BFT consensus with linearity and responsiveness. In *PODC*. we refer to the arxiv v6 long version.

[75] Thomas Yurek, Zhuolun Xiang, Yu Xia, and Andrew Miller. 2023. Long live the honey badger: robust asynchronous dpss and its applications. In *USENIX security*.

[76] Haibin Zhang, Sisi Duan, Boxin Zhao, and Liehuang Zhu. 2023. WaterBear: practical asynchronous BFT matching security guarantees of partially synchronous BFT. In *USENIX Security*.

[77] You Zhou, Zongyang Zhang, Haibin Zhang, Sisi Duan, Bin Hu, Licheng Wang, and Jianwei Liu. 2022. Dory: asynchronous bft with reduced communication and improved efficiency. ePrint 2022/1709. (2022).

A Deferred details and analysis for 2PAC^{lean}

A.1 Definitions and theorem statement

Latency: theoretical and experimental definitions. The definition of the latency of a blockchain protocol varies in the literature. Either the timer is started when one process receives the transaction (the scripts of [68] and [5]; or when all processes receive it ([19]; and possibly furthermore only when all processes are in the same view ([19]); or when the transaction is included in a proposed block [67]. We thus use a more universal definition, which is the latency of a *single-shot instance of consensus*, also known as "MVBA" (or "VABA"). In an MVBA protocol, each process i starts with an input block b_i at time $\tau = 0$: we start the timer here. We define as the *latency of the execution* the first time τ at which *one* honest process outputs a DecCert for a block (b, σ): the timer is stopped here [6]. Then we define the (worst-case) expected latency of the MVBA as the worst-case expected latency, over all adversaries. We adopt these definitions everywhere, including for related works (Sec. 7). Our experiments (Sec. 6) use instead the classical end-to-end measurement of latency, from a client's perspective.

Good-case latency. sMVBA [48, Table I] guarantees that all processes output a DecCert by 6δ if both: there is no corruptions, and the scheduler is fair, i.e., messages are not reordered. We observe that they achieve 6δ in a wider-than-advertised (and arguably more realistic) set of executions, which we formalize as follows.

Definition 3. We say that the scheduler is half-fair if: either it is fair, or, $\delta \leqslant 2\delta_{\text{fast}}$, where δ_{fast} is the fastest message delay between honest processes.
We call an execution good case if both: it is fault-free, i.e., with no corruptions, *and* the scheduler is half-fair.
We call good case latency the delay until *all* (not only one) processes output in a good case execution.

We will use this definition everywhere, including for related works. Note that some works [51, 24] use the terminology "good case" to denote instead the best case latency ([48, 21]), which is the smallest achievable one.

Messages complexity. This is by definition the expected number of messages sent by honest processes per decided block. The *bit complexity* is the number of bits sent. All our protocols can be compiled into ones with bit complexity in $O(nL)$, asymptotically in L the size of blocks: cf Appendix B.8. On the other hand, the message complexity is a scalability bottleneck: when n gets too large, it becomes necessary to use protocols with no more than quadratic ($O(n^2)$) message complexity.

Theorem 4. *2PAC^{lean} is a blockchain consensus. When used for a single-shot MVBA, then*
- *it has 9.5δ worst case expected latency;*
- *(Good case latency, cf Def. 3) in every fault free execution with a half-fair scheduler, all players decide by 6δ.*

[6] Note that this is in line with [68]. We believe that it is relevant from the perspective of an external client subscribing to the system. From when the first process outputs a DecCert, the subscriber is guaranteed to obtain the publicly verifiable decision: (b, σ) in one message delay.

A.2 Proof of consistency

Assume existence of a DecCert for a block $b'_{v',1}$ conflicting with a DecCert for a block $b_{v,1}$, w.l.o.g. $v \leqslant v'$. By Lemma 5 below, $b'_{v',1}$ must be of the same view: $v' = v$. Thus, both blocks have the same proposer $\ell := L_v$. Hence, we would have two QCs for conflicting view-v *height*-1 *proposer*-ℓ blocks: $b_{v,\ell,1}$ and $b'_{v,\ell,1}$, which is impossible by quorum intersection.

Lemma 5. *For all $v \geqslant 1$, consider a* DecCert: $\mathcal{C}_{v,1} \leftarrow \mathcal{C}_{v,2}$ *for a block $b_{v,1}$. Then no higher view-($v' > v$) height-1 QC: $\mathcal{C}'_{v',1}$ (non-necessarily endorsed) can vouch for a block: $b'_{v',1}$ lying on a conflicting branch.*

Proof. W.l.o.g. we can assume that $\mathcal{C}'_{v',1}$ is of minimal view $v' > v$ among those conflicting with $b_{v,1}$. By construction, $\mathcal{C}'_{v',1}$ is descendent of a view-$(v'-1)$ *height*-1 QC: $\mathcal{C}_{v'-1,1}$. By the classical *Claim* below, $v'-1 = v$. Existence of a DecCert for $b_{v,1}$ implies that at least $t+1$ processes had a view-v endorsed QC for $b_{v,1}$: $\mathcal{C}_{v,1}$ upon entering view $v' = v+1$. Thus, no quorum of declarations: {no_endorsed_height-1_QC, $v+1$} (PnD) could be made. Hence, all view-$(v+1)$ *height*-1 blocks voted for by honest processes must be descendent of $b_{v,1}$, contradicting existence of a conflicting $\mathcal{C}'_{v',1}$.

Claim: $v'-1 = v$. [*proof*: assume $v'-1 > v$. Minimality of v' implies that $\mathcal{C}_{v'-1,1}$ is on the same branch as $\mathcal{C}_{v,1}$, hence also $\mathcal{C}'_{v',1}$, a contradiction.] □

A.3 Latency analysis

A.3.1 Proof of (worst-case) expected latency $\leqslant 9.5\delta$. Let us fix any given adversary and denote E the expected latency of 2PAC^{lean} under this adversary. We are going to prove the upper-bound $E \leqslant 9.5\delta$. It will follow that the max over all adversaries is $\leqslant 9.5\delta$, which will conclude our claim. We convey the main idea, the details are deferred to Section A.A.3.2. For each view number v, *either*: an endorsed *height*-2 QC of an earlier view v' was already formed, in which case processes decide at most while in view v; *or*: all processes receive a coin-QC of v. We now consider this latter case. Existence of a view-v coin-QC implies that at least one honest process (actually at least $t+1$): $\mathcal{P}(v)$, received $2t+1$ *height*-2 QCs before the leader L_v was revealed. By unpredictability of the coin, the identity of L_v is independent of the proposers of these *height*-2 QCs. Hence, the probability that one of them is equal to L_v is $(2t+1)/n > 2/3$. If this happens, then this implies that all honest processes will receive a DecCert, i.e., an endorsed view-v *height*-2 QC, by 6δ. If it does not happen (with probability $< 1/3$), at least all processes receive a view-v coin-QC by 6δ, thus move to the next view $v+1$. There, they have again $> 2/3$ probability to decide in $v+1$ and $< 1/3$ probability to move to the next view $v+2$, etc. Thus, the expected number of elapsed views until the decision is 1.5. Since the delay to decision in the first view (if successful) is 6δ in the first view and 7δ in every higher view $v \geqslant 2$ (if successful), we thus obtain the claimed $E \leqslant 6\delta + 0.5 \times 7\delta = 9.5\delta$.

A.3.2 Further details for the proof of expected latency of 2PAC^{lean} *(and of sMVBA).* We formalize the proof given in Appendix A.3.1 for the upper-bound 9.5δ on the expected latency of 2PAC^{lean}. For each view number v, we call *lucky (roulette) roll*, and denote $\text{LR}(v)$, the following event:

$\text{LR}(v)$:= {Denote $\mathcal{P}(v)$ the first honest process which multicasts a view-v coin share. Denote $\mathcal{Q}(v)$ the set of *height*-2 QCs (issued by $2t+1$ distinct proposers) which it received beforehand. Then one of them will be endorsed, i.e., L_v is among these $2t+1$ proposers.}

Let us informally explain why our terminology is justified, i.e., why electing L_v is like rolling a wheel with n numbers. The index in $[n]$ on which the wheel stops is, by unpredictability of a common coin, independent of the view of the adversary so far (this will be leveraged in Lemma 6 below). In particular it is independent of $\mathcal{Q}(v)$. If the wheel stops on the index of one of the $2t+1$ proposers in $\mathcal{Q}(v)$, then this *height*-2 QC is promoted as endorsed, hence, constitutes a DecCert. This explains why we say that the roll was "lucky".

From Proposition 7 below, it follows that the random variable L of the latency of an execution is upper-bounded by the random variable L' defined as follows. Let $v_D \geqslant 1$ (D for "decision") be the random variable equal to the smallest view number in which the roll is lucky. Namely:

$$(1) \qquad v_D := \min\Big(v \geqslant 1 \text{ s.t. } \big[\neg\text{LR}(v'), \forall v' < v\big] \ \wedge \ \text{LR}(v)\Big) .$$

Then $L' = 6\delta + 7\delta.(v_D - 1)$. Moreover, by Lemma 6 the events $\{\neg\text{LR}(v), v \in [1, \ldots, \infty[\}$ are independent. It follows that L' is "memoryless" with respect to the "past un-lucky rolls", i.e., for all v:

$$(2) \qquad E\Big[L' \mid \big\{\neg\text{LR}(v'), \forall v' \leqslant v\big\}\Big] = 6\delta + 7\delta.(v-1) + E[L'] .$$

Since by Lemma 6, each event $\neg\text{LR}(v)$ has probability $< 1/3$, we obtain the following inductive formula on $E' = E[L']$ the expectation of L':

$$(3) \qquad E' = 2/3.6\delta + 1/3(7\delta + E')$$

from which the claimed $E \leqslant E' = 9.5\delta$ follows.

We now estimate an upper-bound on the latency of sMVBA [48] by the same method, then state and prove Lemma 6 and Proposition 7. In sMVBA, each view $v \geqslant 1$ is as follows: if $\text{LR}(v)$ then an honest process decides by 6δ after all processes have entered the view. Else, there are two extra asynchronous rounds added at the end of the view, called "pre-vote and vote". So by 8δ after the view v started, if no honest process has decided yet, then all processes have entered the next view $v+1$. In conclusion, with the same notation as above, the latency of an execution of sMVBA is upper-bounded by the random variable L'' defined as $L'' = 6\delta + 8\delta.(v_D - 1)$. Since the events $\{\neg\text{LR}(v), v \in [1, \ldots, \infty[\}$ are also independent, and each of probability $< 1/3$, we obtain the inductive formula on the expectation of L'': $E'' = 2/3.6\delta + 1/3(8\delta + E'')$, which gives our estimated upper-bound $E'' = 10$ for the expected latency of sMVBA.

Lemma 6. *The events $\{\neg\text{LR}(v), v \in [1, \ldots, \infty[\}$, are independent. Each of them has probability $< 1/3$.*

Proof. The times $\tau_{\text{coin}}(v)$ at which the $\mathcal{P}(v)$'s send their coin shares are strictly ordered by increasing v. Thus all events $\neg\text{LR}(v')$ for $v' < v$, are independent from $\neg\text{LR}(v)$: otherwise, unpredictability of the view-v coin would be broken. This proves the first claim. The second claim follows from the fact that the indices of the proposer in the set $\mathcal{Q}(v)$ of the *height*-2 QCs received by $\mathcal{P}(v)$, are

independent from the view-v coin. Indeed, otherwise, unpredictability of the view-v coin would be broken. □

Proposition 7. *For each view number v, consider the time $\tau(v) := 6\delta + (v-1)7\delta$. Suppose that no process decided by $\tau(v)$. Then, by $\tau(v)$: (i) $\neg$LR(v') happened for all $v' \leqslant v$; (ii) and all processes have entered view $v+1$.*

Proof. We proceed by induction on v. Let us first initialize the induction by proving the proposition for $v = 1$. At 5δ:

- *if an honest process has received a coin-QC:* then all honest processes receive it by 6δ;
- *or*, since no honest process received a coin-QC yet, all honest processes are still in view 1. So each of them must have formed a *height*-2 QC and multicast it, then received $2t+1$ *height*-2 QCs, then multicast its coin share. Thus by 6δ, all processes have obtained a view-1 coin-QC.

In conclusion, all processes have received a coin-QC by 6δ. Since by assumption no honest process decided by $\tau(1) = 6\delta$, they must all have entered view $v = 2$, which proves (ii). Moreover, consider $\mathcal{P}(1)$: since by assumption it did not decide despite having received a view-1 coin-QC, it must be that none of its $2t+1$ *height*-2 QCs is endorsed, which proves (i).

Last, we assume that the proposition holds for all $v' < v$ for some $v \geqslant 2$, and deduce that it holds for v. We skip the proof of this deduction, since it is identical to the case $v = 1$, up to replacing "at 5δ" by: "at $6\delta + (v-2)7\delta + 6\delta$". □

A.3.3 Proof of Good-case latency 6δ. Let us consider a one-shot MVBA good-case execution. Recall (Def 3) that it means: fault-free, and with a half-fair scheduler, i.e., $\frac{1/2}{\delta} \leqslant \delta_{\text{fast}}$. Our goal is to show that all processes output a DecCert by 6δ. Consider the first time: τ when a process enters view 2.

Since the process received a coin-QC, it must be that $6\delta_{\text{fast}} \leqslant \tau$ (for a process to perform a step, e.g., advance view, it must be that at least t other honest processes completed the last step at least δ_{fast} before, e.g., casting their coin share). Hence, by the half-fair scheduler assumption, $3\delta \leqslant \tau$. Thus, at 3δ all n processes were still in view 1, so they had received *height*-1 proposals from all n processes, and had multicast (or were multicasting) their *height*-1 votes. Hence, at 4δ all processes will all have formed-then-multicast a *height*-2 QC for their own proposal. In conclusion, at 5δ, all *height*-2 QCs from all n processes have been received.

On the other hand, all processes will have received a coin-QC by 6δ, promoting one of their *height*-2 QCs as a DecCert, which concludes the proof.

A.3.4 Quality of 2PAC^{lean}.

On a single-shot MVBA, if the scheduler is half-fair. If the first leader $\mathcal{P}_\ell = L_1$ is honest, which happens with probability $> 2/3$, then all processes will receive its proposed *height*-2 block: $\boldsymbol{C_{1,\ell,1} \leftarrow b_{1,\ell,2}}$ by 3δ. By the half-fair assumption, they are still in view 1, thus will vote for it, which will form a DecCert. Hence, we have $> 2/3$-quality.

If the scheduler is fair. If the leader $\mathcal{P}_\ell = L_v$ of some view v is honest, which happens with probability $> 2/3$, then all processes will receive its proposed *height*-2 block: $\boldsymbol{C_{v,\ell,1} \leftarrow b_{1,\ell,2}}$ while they are still in view v (otherwise, there would have been one more round-trip between some fast honest processes, contradicting the fair scheduler assumption). Hence, for the same reason, we have $> 2/3$-quality.

Worst-case. Let us fix any v and consider a 2-chain $b_{v,i,1} \leftarrow b_{v,i,2}$ included in a decided chain, we are going to compute the quality, i.e., the probability that this 2-chain is "honest input" (to be precised). Consider the first honest process receiving $2t+1$ view-v QCs. In the event X where one of them is proposed by the to-be-elected leader ℓ, then this implies that at least $t+1$ processes received its *height*-2 proposed block: $\boldsymbol{C_{v,1,\ell} \leftarrow b_{v,2,\ell}}$, thus for the same reason as above we have that the 2-chain $\boldsymbol{b_{v,1,\ell} \leftarrow b_{v,\ell,2}}$ will end up in the decided chain (thus $i = \ell$). Since $\mathbf{P}(X) > 2/3$, and that $\mathbf{P}(\ell \text{ is honest}|X) > 1/2$, there is $> 1/3$-quality. We can obtain a higher estimate. Consider the bad event, included in $\neg X$, where no $\boldsymbol{b_{v,\ell,1} \leftarrow b_{v,\ell,2}}$ ends in the decided chain. Consider the first view $w > v$ for which there is an honest process which receives a DecCert, i.e., a *height*-2 QC of the view-w leader L_w. Recall that when L_w built its *height*-1 proposal, it made a *free choice* between $2t+1$ certified-but-undecided chains $b_{v,i,1} \leftarrow b_{v,i,2} \leftarrow \ldots b_{w-1,2}$. From there, two possible definitions of quality can be considered. *The first definition* is optimistic and always labels as "honest input" any such undecided chain freely chosen by an honest L_w. With this definition, since $\mathbf{P}\Big(L_w \text{ is honest}|\neg X\Big) > 1/2$, then $\mathbf{P}\Big(b_{v,i,1} \leftarrow b_{v,i,2} \text{ is corrupt input}|\neg X\Big) < 1/2$. Hence, the total quality is $> 2/3.1/2 + 1/3.1/2 = 1/2$.

The second definition is pessimistic, and labels as "honest input", on average, 50% of such chains freely chosen by an honest L_w (else, labeled "corrupt input" if L_w is corrupt). With this definition, the total quality would be $> 2/3.1/2 + 1/3.1/4 = 5/12$. With such a definition, there is a simple way to lift the 5/12-quality to optimal 1/2: clients simply ignore the transactions in 2-chains $b_{v,j,1} \leftarrow b_{v,j,2}$ s.t. $j \neq L_v$. The throughput would then drop down by 1/3.

B Further optimizations

B.1 Leader election without forwarding the coin-QC. Existing post-quantum common coins, recalled in Sec. 2.1, do not come with an algorithm aggregating coin shares into a constant-sized coin-QC.

Fortunately, the leader-election mechanism in the 2PAC protocols can be modified into a variant which does not require anymore to forward any coin-QC. Instead, the variant guarantees that all processes receive enough coin shares. This variant is the one of sMVBA ([48, Alg 5 line 9]), it was introduced in a related context by [28] under the name "Bracha timeout". It consists in adding the following trigger: a process multicasts its coin share *also* upon receiving $t+1$ coin shares.

First, we show that the (worst-case) expected latency 9.5δ of $\text{2PAC}^{\text{lean}}$ is unaffected. It is enough to show that Proposition 7 still holds. We prove it for the first view $v = 1$, i.e., we prove the *Claim: all honest processes entered view 2 by 6δ.* [Proof of the Claim. Assuming an execution where the Claim does not hold. Thus, at 4δ, there would exist an honest proposer $\mathcal{P}$ which did not collect $2t+1$ *height*-2 votes. For this to happen, it must be that, at 3δ, there exists a process $\mathcal{Q}$ which is already in the next view $v+1$. For this to happen, it must be that $\mathcal{Q}$ received $2t+1$ coin shares before

3δ. Since $t+1$ of them were issued by honest processes, they will reach all honest processes by 4δ, triggering them to multicast their coin shares. Thus all honest processes will have left view v by 5δ, a contradiction.]

Second (and last), we show that the good-case 6δ latency of 2PAC^{lean} is unaffected. Assume the contrary, then there would exist an execution where, at 4δ, there would exist an honest proposer $\mathcal{P}$ which did not collect $2t+1$ *height*-2 votes. For this to happen, it must be that, at 3δ, there exists a process $\mathcal{Q}$ which is already in the next view $v+1$. For this to happen, it must be that $\mathcal{Q}$ received $2t+1$ coin shares before 3δ, thus $6\delta_{\text{fast}} < 3\delta$, a contradiction.

B.2 Check only one PnD *instead of* $2t+1$. In both 2PAC^{lean} and 2PAC^{BIG}, the threshold signature {no_endorsed_height-1_QC, v} needs to be checked only once in a view v. Once checked, raize a flag DocGchecked and accept non-endorsed QCs from other proposers without checking PnDs from them. Similarly, in s2PAC^{lean} and s2PAC^{BIG}, the threshold signature {no_endorsed_height-2_QC, v} needs to be checked only once in a view v. In Appendix B.3 we describe a further optimization enabling proposers to optimistically not form nor send a PnD, without losing latency in the worst-case.

B.3 Optimistically Skipping PnD. The optimization, which applies to both 2PAC^{lean} and 2PAC^{BIG} (mutatis mutandis for s2PAC^{lean}), is that in the second case of [Propose], the proposer $\mathcal{P}_i$ optimistically sends only a bare non-endorsed view-$(v-1)$ QC in its proposal $\langle \mathcal{C}_{v-1,2} \leftarrow b_{v,1,i} \rangle_i$, without the threshold signature {no_endorsed_height-1_QC, v}. Then, a process $\mathcal{P}_j$ accepts to [Lock Vote] for such a bare proposal, only if $\mathcal{P}_j$ did not cast a view-$(v-1)$ *height*-VOTE for a conflicting proposal of L_{v-1} (notice that this is an adaptation of the unlocking mechanism of Tendermint/two-phase Hotstuff [74, p. 4.4]). It could happen that some slow processes $\mathcal{P}_j$'s refuse to [Vote - height 1] because they cast such a view-$(v-1)$ VOTE for a *height*-2 block $b_{v-1,2}$. The key observation which we make is that such slow processes must have already sent to $\mathcal{P}_i$ the endorsed $\mathcal{C}_{v-1,2} \leftarrow b_{v-1,2}$, in view-$v$ [Report]. Thus proposer $\mathcal{P}_i$ will detect such $\mathcal{P}_j$'s *straight from their [Report]-ed endorsed QCs*. To such a process $\mathcal{P}_j$, $\mathcal{P}_i$ sends the threshold signature {no_endorsed_LockCert, v}, then $\mathcal{P}_j$ accepts to Vote upon receiving it. In conclusion, no further round-trip is needed between $\mathcal{P}_i$ and $\mathcal{P}_j$.

Let us finally prove that this optimization preserves consistency. Suppose that there exists a quorum of $2t+1$ view-v *height*-1 votes on a view-$(v-1)$ non-endorsed QC. Then, it must be the case that no $t+1$ honest processes cast view-$(v-1)$ *height*-2 votes for a *height*-2 block of L_{v-1}. Thus, no view-$(v-1)$ DecCert conflicting with $\mathcal{C}_{v-1,2} \leftarrow b_{v,1,i}$ can ever exist. We then make the same kind of induction argument as in the proof of consistency, i.e., that no $(v-1)$-QC conflicts with a DecCert of a lower view, to conclude that $\mathcal{C}_{v-1,2} \leftarrow b_{v,1,i}$ does not conflict with any lower view DecCert.

B.4 Clever processing of mailbox. The proof of 6δ latency in the good case would fail in an implementation allowing a process to process incoming messages not in the order in which they were received. For instance, the process could form first a view-v coin-QC then move to view $v+1$, before processing the view-v *height*-1 QCs which arrived before, preventing it forever to *height*-2 vote for their child. To prevent this, it must be precised that processes process all received view-v messages before moving to view $v+1$.

B.5 Fast catch-up. A process advances to a higher view v' upon receiving any view-v' QC (such a QC implies existence of $2t+1$ view-$(v'-1)$ *height*-2 QCs, and of a coin-QC, so the proofs of latencies apply unchanged).

B.6 Do not re-send messages infinitely many times, while preserving liveness. In a model where messages could be lost, processes would have to re-send a message m until they obtain an ACK. Thus if the recipient is corrupt, they would re-send it infinitely, thus the complexity would blow-up linearly in v. We thus make the optimization that the only messages of prior views $v' < v$ that a process re-sends, are: the highest DecCert which it has, and the coin-QC of the previous view $v-1$. Moreover, by the previous optimization (Appendix B.5), a process can stop re-sending the latter as soon as it obtains a QC of the current view.

B.7 Adaptive corruptions. Adaptive security can be achieved in all 2PAC protocols, simply by using adaptively-secure primitives. For instance, [6, 26, 7] provide unique non-interactive threshold signatures, which can thus be hashed into adaptive common coins ([65]). We refer to the nice discussion on adaptive fairness in [48, Appendix E].

B.8 Amortization over long values, with only $+\delta$ overhead. Recall that the extension protocol Dumbo-MVBA$\star$ [58] compiles any asynchronous Mvba: C with (expected) latency into one with asymptotic bit complexity $O(Mn)$ over long M-sized input values (but same message complexity: the messages-cubic protocols of Table 17 cannot magically become linear). Its blueprint can be adapted to 2PAC as follows. Each proposer disperses its block b with error-correction encoding, then collects signatures into a *certificate of retrievability*: lock := (σ, c). Technically, c is a vector commitment to codewords of b and σ a threshold signature of declarations to have received a correct opening of a coordinate of c. After this 2δ-long round-trip, the proposer can finally propose the lock in its instance. Upon learning the elected leader, processes reconstruct its block b in one δ step, called "Re-Cast" in [58]. Since a lattice-based c can weigh hundreds of kilobytes, we propose the minor improvement to further reduce the lock in Dumbo-MVBA$\star$, as: $(\sigma, \text{Hash}(c))$.

We now introduce an optimization specific to 2PAC: the first 2δ round-trip can be spared by merging it into the proposal step. Indeed, since σ is a threshold signature on a unique digest of b: c, it plays the role of a QC on b.

C Proof of s2PAC^{lean}

THEOREM 8. s2PAC^{lean} *is a blockchain consensus. It has* 9.5δ *expected latency (worst-case adversary, for a single MVBA); and* 6δ *good-case latency (formal Definitions 3 in Appendix A.1).*

- If the scheduler is fair, *then a height-2 block pipelined by an honest leader at a time τ gets decided before $\tau + 4\delta$.*

Both the proof of expected latency and of good case latency are identical to the ones for 2PAC^{lean} (Appendix A.3.1 then Appendix A.3.2, and Appendix A.3.3), so we skip them.

Proof of 4δ-decision of pipelined blocks if fair scheduler. Consider a view v, by definition at time τ the (to-be-elected) leader

$\mathcal{P}_\ell = L_v$ proposes its *height*-2 block $\boldsymbol{b_{v,2,\ell}}$. By the Claim below, all $2t+1$ honest processes cast a SPEED_VOTE for $\boldsymbol{b_{v,\ell,2}}$, thus by $\tau + 3\delta$. In conclusion, a DecCert is formed (at least by $\mathcal{P}_\ell$) by $\tau + 4\delta$. We now state and then prove the Claim.

Claim: all $2t+1$ honest processes vote for $\boldsymbol{b_{v,\ell,2}}$, then $\mathcal{P}_\ell$ forms-then-multicasts a height-2 QC while still in view v, which is received (and thus SPEED_VOTED) by all $2t+1$ honest processes while still in view v. [Proof of the claim: assuming the contrary, then some honest process would have received.].

Proof of consistency of s2PAC$^{\textbf{lean}}$**.** First, we deduce consistency from the following claim, then we prove the claim. *Claim: Consider a (ordinary or speed-)* DecCert *for a block $b_{v,h}$, then no higher view ($v' > v$)-QC: $\mathcal{C}_{v',h'}$ (non-necessarily endorsed) can exist for a block b' on a conflicting branch.*

Consider a DecCert for a block b' conflicting with a DecCert for a block $b_{v,h}$, which we assume w.l.o.g. lower ranked, i.e., $2v + h \leqslant 2v' + h'$. By the Claim, b' must be of the same view: $v' = v$, thus both blocks have the same proposer L_v. By quorum intersection, it cannot be of the same height h, so we must have $h = 1$ and $h' = 2$. But by construction we must have $\mathcal{C}_{v,h=1} \leftarrow \mathcal{C}'_{v,h'=2}$ (otherwise processes would not have voted to form $\mathcal{C}'_{v,h'=2}$), a contradiction.

[*Proof of the Claim.* W.l.o.g. we can assume that $\mathcal{C}' = \mathcal{C}_{v',h}$ is of minimal view $v' > v$, then height, among those conflicting with $b_{v,h}$. We first easily rule out the case $h' = 2$. Indeed, this would mean that its parent $b_{v',j,1} \leftarrow b'_{v',j,h'=2}$, of same proposer j, would be on the same chain as $b_{v,h}$, a contradiction. Thus it remains to analyze the *height*-1 case: we have $\mathcal{C}' = \mathcal{C}_{v',j,1}$ for a block $b'_{v',j,1}$, where j denotes its proposer. We consider one by one the three possibilities for the step [Vote *height*-1] by which an honest process, say $\mathcal{P}_i$, could have cast a VOTE to form $\mathcal{C}'$:

i) either upon receiving a proposal $\langle \mathcal{C}_{v'-1,2} \leftarrow b'_{v',j,1} \rangle_j$ with $\mathcal{C}_{v'-1,2}$ endorsed. By minimality, $\mathcal{C}_{v'-1,2}$ must be on the same chain as $b_{v,h}$, thus also $b'_{v',j,1}$, a contradiction;

ii) or, upon receiving a proposal $\langle \mathcal{C}_{v'-1,1} \leftarrow b_{v'-1,2} \leftarrow b_{v',j,1} \rangle_j$ appended with $\{\text{no_endorsed_height-2_QC}, v'\}$ with $\mathcal{C}_{v'-1,1}$ endorsed. By minimality, $\mathcal{C}_{v'-1,1}$ must be on the same chain as $b_{v,h}$; So the only possibility to have a fork, is that $v' = v$ and $b_{v,h=2}$ is a *child* of $\mathcal{C}_{v'-1,1}$, thus that the DecCert on $b_{v,h=2}$ is a *fast* one. But the $\{\text{no_endorsed_height-2_QC}, v' = v\}$ rules out existence of a view-v speed-DecCert, a contradiction.

iii) or, upon receiving a proposal $\langle \mathcal{C}_{v'-1,2} \leftarrow b'_{v',j,1} \rangle_j$ appended with $\{\text{no_endorsed_height-1_QC}, v'\}$. By minimality, $\mathcal{C}_{v'-1,2}$ must be on the same chain as $b_{v,h}$, thus also $b'_{v',j,1}$, a contradiction.

D Further attacks

2-chain_VABA is defined page 8 & Appendix C of [42] as a particular instantiation of the protocol called Ditto, published in [44, 42]. Thus our attacks a fortiori apply to the published version of Ditto. 2-chain_VABA is specified by setting to 0 the time-out before processes switch to the fallback path of Ditto, which is described in [42, Figure 5]. In particular, the steady-state path of Ditto described in their [42, Figure 4] boils down to the non-interactive steps called [Lock] and [Commit], before they immediately time-out and return to the fallback path. Since it is not obvious to read 2-chain_VABA from the full specifications of Ditto in [42], we make it explicit in Figure 19. Since the steady state of Ditto, i.e., Jolteon, is never triggered in 2-chain_VABA, some data structures and instructions are never used. This allows us to simplify the presentation. Departing from the rest of our paper (Sec. 2.1), in this section we revert to the convention of [42] that a block is denoted $b_{v,h,j}$, i.e., with the *height* h before the *proposer* index j.

As any blockchain consensus, the purpose of 2-chain_VABA is to produce DecCerts on blocks. The implementation of a DecCert in 2-chain_VABA is the same as in 2PAC^{lean} Sec. 3, i.e., a 2-chain of endorsed QCs of the same view. We recall it at the end of Figure 19.

Our attacks also apply when 2-chain_VABA is used to do a single-shot MVBA, since the consistency violations will be observed for the first view-1 block, and since no block is decided in the livelessness scenarios. Both the description of 2-chain_VABA and of the attacks aim at simplicity: the full details are available on demand if necessary. The two highlighted checks of endorsement were neither in the published specifications nor in the implementation [73]. Absence of these checks enabled an obvious attack on consistency, which we initially communicated to the authors. The authors quickly replied on 2023-10-13 that their intention has always been that these two checks should be there. Hence, in what follows we consider them as included. Inclusion of these two checks then allowed our further attacks. For place reasons we describe only those in Sec. 1.1.1 & SectionD.D.1, the others were privately communicated to the authors and are available on demand.

D.1 Leveraging the relaxed endorsement definition in the implemented version: an attack on consistency. In the implemented version [73], processes consider any *height*-1 QC: $\mathcal{C}_{v,1,j}$ as endorsed as soon as it is wrapped inside a *height*-2 block of which the proposer is the leader, i.e., $\mathcal{C}_{v,1,j} \leftarrow b_{v,2,\ell}$ with $\ell = L_v$, whereas possibly $\ell \neq j$. [Technically: lines 627-628 of [73] $\mathcal{P}_\ell$ sets itself as the "acceptor" of $\mathcal{C}_{v,1,j}$, then line 542 makes processes consider it as endorsed.] This allows an attack, of which the idea is simple: the leader ℓ privately delivers a DecCert: $\mathcal{C}_{v,1,\ell} \leftarrow \mathcal{C}_{v,2,\ell}$ to an isolated process. In parallel, it generously endorses another QC: $\mathcal{C}_{v,1,j} \leftarrow b'_{v,2,\ell}$ which it diffuses to other processes. As a result, in the next view all processes propose child blocks of this concurrent $\mathcal{C}_{v,1,j}$, hence it ultimately gets decided.

E Further comments and related works

E.1 More on MVBA . MVBA is the costliest building block in most state-of-the-art asynchronous blockchain consensus [48, 57, 40, 22, 20, 11, 47]. In addition, MVBA is the main building block in most implementations [16, 48, 75, 31] of the primitive called "agreement on a common/core subset" (ACS [8, 10], also known as asynchronous interactive consistency [9]). ACS is also a building block of blockchain consensus [75, 30, 56]. MVBA and/or ACS are also building blocks of proactive resharing [75], PVSS-based distributed randomness generation [50] and threshold signing [7]. A classical requirement in MVBA (also called VABA [2, 42]) is called *(external) validity* [16, 48]. Such an MVBA is parametrized by any publicly verifiable so-called validity predicate, denoted *Mvba_valid*, and it is further required that the output value x is appended with a certificate π such that $\mathit{Mvba_valid}(x, \pi) = \text{true}$. All our protocols can straightforwardly be made externally valid, simply by having

2-chain_VABA + endorsement checks

Instructions for each process $\mathcal{P}_i$. At the beginning of the protocol, enter view $v \leftarrow 1$; initialize the *highest seen quorum certificate* $\mathcal{C}_{high}$ to the default QC: $\perp$ of the genesis block of view 0.

Enter View. Upon entering a view v:

- For every process $j \in [n]$, record all the QCs of view v by process j, and keep a voted height number $h_{vote}[j]$. Initialize $r_{\text{vote}}[j] \leftarrow 0$ and $h_{vote}[j] \leftarrow 0$ for all $j \in [n]$;
- *then* ***(Propose height-1)*** multicast a *height*-1 block: $b_{v,1,i} = [v, 1, i, \mathcal{C}_{high}, txn, id]$.

Vote. Upon receiving a view-v block $b_{v,h,j}$ from process j, if $h > h_{vote}[j]$, and

- if $h = 1$, and $b_{v,1,j} = \{v, 1, j, \mathcal{C}, txn, id\}$ such that $\mathcal{C} \geqslant \mathcal{C}_{high}$ and such that $\mathcal{C}$ is endorsed; or
- if $h = 2$, and $b_{v,2,j} = \{v, 2, j, \mathcal{C}, txn, id\}$ such that $\mathcal{C}$ is $any^{(i)}$ valid$^{(ii)}$ view-v QC of *height*-1 $^{(iii)}$; $2 > h_{vote}[j]$.

set $h_{vote}[j] \leftarrow h$ and vote for the block by sending $\{v, h, j, id\}_i$ to process j.

Propose *height*-2, and trigger leader election. Upon the first height-h block $b_{v,h,j}$ (by $any^{(i)}$ process j) is certified by some $\mathcal{C}$

- if $h = 1$, create a child block $b_{v,2,i} = [v, 2, i, \mathcal{C}, txn, id]$ and multicast it;
- if $h = 2$, sign and multicast $\langle \mathcal{C} \rangle_i$;

Leader Election. Upon receiving $2t+1$ valid$^{(ii)}$ height-2 view-v QCs signed by distinct processes $^{(iv)}$, sign and multicast a coin share for view v.

Advance View *(was called: Exit Fallback).* Upon receiving a coin-QC $\mathcal{C}_{\text{coin}}$ of view $\geqslant v$ or $t+1$ valid coin shares to form a $\mathcal{C}_{\text{coin}}$ of view $\geqslant v$, multicast $\mathcal{C}_{\text{coin}}$. Update $v \leftarrow \mathcal{C}_{\text{coin}}.v + 1$. Execute Lock and Decision.

Lock. Upon seeing$^{(v)}$ a valid endorsed QC: $\mathcal{C}$ (formed by votes or contained in proposal or timeouts), update $\mathcal{C}_{high} \leftarrow \max(\mathcal{C}_{high}, \mathcal{C})$.

Decision-certificate. For any view number w, a DecCert for a view-w *height*-1 block with *proposer* $\mathcal{P}_\ell = L_w$: $b_{w,1,\ell}$, is a 2-chain of endorsed QCs $\mathcal{C}_{w,1,\ell} \leftarrow \mathcal{C}_{w,2,\ell}$, i.e., QCs on blocks $b_{w,1,\ell} \leftarrow b_{w,2,\ell}$ where $b_{w,2,\ell}$ is a view-w *height*-2 *proposer* $\mathcal{P}_\ell$ block.

(i) **Boosting strategy** (highlighted [42, page 7]). The originality of 2-chain_VABA is that a process i is allowed to propose a *height*-2 block $b_{v,2,i}$ child of *another* proposer j's *height*-1 certified block $b_{v,1,j}$, and processes are then allowed to vote for $\mathcal{C}_{v,1,j} \leftarrow b_{v,2,i}$. In the code [73] it is described line 749: "adopt others' certified fallback block", and is implemented lines 622 to 646.

(ii) "valid QC" is not defined. In the implementation [73] there is another predicate called "valid_qc($\mathcal{C}$)", which instead checks endorsement. It is not checked here (otherwise this would have blocked the protocol, no view-v QC being endorsed yet).

(iii) In [42, Figure 5], this predicate is written under the equivalent form: "$h = \mathcal{C}.height + 1$". The simpler equivalent form under which we wrote it, i.e., $\mathcal{C}.height = 1$, turns out to be also the way the public implementation does it (line 630 of [73]).

(iv) **Trigger for early leader election.** Nothing prevents these QCs from being issued by the *same* proposer, as long as they were signed by distinct processes (line $h = 2$ of Propose). This is also allowed in the implementation of the authors ([73] lines 821-822). Since AMS would have instead instructed to wait for $2t+1$ view to terminate, we believe that this instruction was unintended. We leveraged this instruction in an attack communicated to the authors (available on demand). Our other attacks do not leverage this instruction, i.e., they are also compatible with the classical trigger for leader election of AMS.

(v) The specifications [42, Fig. 4 & 5] instruct explicitly to do [Lock] at two specific places, i.e., in [Timeout] and [Exit Fallback]. So said like this, one would deduce that $\mathcal{C}_{high}$ is *not* updated elsewhere. But in the implemented version [73], processes update their $\mathcal{C}_{high}$ at anytime upon receiving an endorsed QC. All our attacks are compatible with both versions.

Figure 19

processes ignore messages containing non-valid values, i.e., on which *Mvba_valid* returns false. For the use-case of blockchain consensus, having quality actually removes the need to impose external validity. Instead, it is advocated in [71] that blockchains are faster when they outsource to clients the task of pruning invalid transactions.

Note that the terminology of *validated consensus* is used in [37] for the conflicting validity specification known as *strong unanimity* ([33]). It would be straightforward to enforce strong unanimity in 2PAC, provided a preliminary round as in [1, §6].

E.2 report *after starting a view.* Our convention of a view beginning *before* report is in line with [18, 74, 53]. There, processes switch to the next view as soon as they time-out, before they send report messages to the next leader (formatted as (NEW_VIEW, v+1, ...) in [18]). Instead in GradedDAG, the next view begins one round after leader election completed (in sMVBA: two rounds after since there is "pre-vote" then "vote").

E.3 GradedDAG under a half-fair scheduler. We describe a fault-free scenario where the scheduler is half-fair (Def. 3) but not fair, i.e., that $\delta/2 \leqslant \delta_{\text{fast}} < 2/3\delta$. For simplicity we further consider that all messages to and from $\mathcal{P}_4$ take δ, while all messages between other processes take δ_{fast}. All the discussion still holds as long as round-trips to and from $\mathcal{P}_4$ take between $3/2\times$ and $2\times$ longer than round-trips between other processes. For simplicity we consider the first view $v = 1$. Under such assumptions, by Theorem 2, in 2PAC^{BIG} it is guaranteed that a view-1 DecCert is formed (namely: all processes would receive he-2 proposals from all *proposer*s=processes while still in view 1, thus vote for all of them). We now consider what would happen in GradedDAG, following the description of [24, p. IV C]. At $\tau = 0$, all processes act as senders of a reliable broadcast

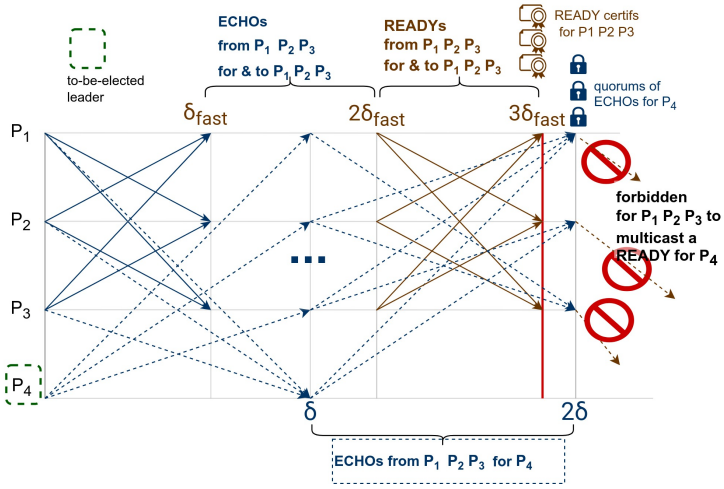

Figure 20: **A view of GradedDAG with $\delta_{\text{fast}} < 2/3\delta$**

(RBC). At $3\delta_{\text{fast}}$, all $\{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\}$ receive certificates of termination of the RBCs of $\{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\}$ (recall that certificate is a $2t+1 = 3$-quorum of READY messages). By our $\delta/2 \leqslant \delta_{\text{fast}} < 2/3\delta$ assumption, at this point, none of them received a quorum of ECHO, hence, could have sent a READY for the RBC of $\mathcal{P}_4$. We conclude by the *Claim: they will never send READY for* $\mathcal{P}_4$. The Claim implies that no grade-2 block of $\mathcal{P}_4$ will ever be formed. In conclusion, if $\mathcal{P}_4$ is elected leader, no view-1 DecCert will ever be formed. With very bad luck, such a scenario (of a slow elected leader) could repeat in all higher views.

[*Proof of the Claim.* At $3\delta_{\text{fast}}$, Processes $\{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\}$ create their CBC blocks, in which they do not reference the block sent by $\mathcal{P}_4$. Indeed, they follow the rule in bold in the last paragraph of page 4 of [24], forbidding one to reference a block for which one did not receive at least a quorum of ECHO (called a grade 1 block, in their terminology). Hence, by the rule called "to guarantee safety" on the top of page 5 in [24], all $\{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\}$ stop participating in the RBC instance of $\mathcal{P}_4$.]

E.4 Why a coin with threshold only $t+1$ prevents 6δ good-case latency. We describe the counterexample for 2PAC as if it had been instantiated with a $(t+1)$-threshold coin. The counterexample also applies to 2-chain_VABA. By $5\delta_{\text{fast}}$, there is an honest process: $\mathcal{P}(v)$ which sends its coin share. The adversary immediately delivers to $\mathcal{P}(v)$ the t corrupt coin shares from corrupt processes, as a result $\mathcal{P}(v)$ enters view $v+1$ at 5δ. On the other hand, since the scheduler is assumed only half-fair (but not fair), there could be honest processes (acting as proposers) of which the *height*-1 block was still not received by $\mathcal{P}(v)$ at $5\delta_{\text{fast}}$. Such bad event can happen, e.g., if $5/2\delta \leqslant 5\delta_{\text{fast}} < 3\delta$. Hence, if corrupt processes do not vote, then these honest processes will never obtain *height*-2 QCs on their proposals.

E.5 Partially synchronous consensus. In this line of works [32, 60, 67, 5] the liveness is conditioned to eventual synchrony and absence of DoS attacks on pre-determined leaders (a.k.a. "anchors"). Being agnostic to such DoS attacks is the motivation given in [43] for developping asynchronous blockchain consensus, such a following the 2PAC design.