Faster Asynchronous Blockchain Consensus and MVBA

Matthieu Rambaud
Télécom Paris

Abstract—**Blockchain consensus, a.k.a. BFT SMR, are protocols enabling n processes to decide on an ever-growing chain. The fastest known asynchronous one is called 2-chain VABA (PODC'21 and FC'22), and is used as fallback chain in Abraxas* (CCS'23). It has a claimed 9.5δ expected latency when used for a single shot instance, a.k.a. an MVBA. We exhibit attacks breaking it. Hence, the title of the fastest asynchronous MVBA with quadratic messages complexity goes to sMVBA (CCS'22), with 10δ expected latency. Our positive contributions are two new and complementary designs.**

• 2PAC (2-phase asynchronous consensus). It has a simpler and lighter chaining than in previous approaches. Instantiated with either quadratic or cubic phases of voting, it yields:

2PAC$^{\text{lean}}$: $+90\%$ throughput and 9.5δ expected latency, with quadratic ($O(n^2)$) messages complexity. In both 2-chain VABA and sMVBA (as if chained, with pipelining), the quorum-certified transactions which were produced in the worst-case 1/3 of views with a slow leader were dumped, so the work was lost. The simpler design of 2PAC inserts such blocks in straight-line in the chain. Thus, contrary to naive uncle-referencing, this comes with no computational overhead, yielding a net $+50\%$ throughput gain over chained sMVBA. Both the remaining throughput and latency (-0.5δ) gains, come from the lighter interactive construction of proofs of consistency appended to proposed blocks, compared to sMVBA.

2PAC$^{\text{BIG}}$: the fastest asynchronous blockchain consensus with cubic ($O(n^3)$) messages complexity. Fault-free single shot MVBA runs decide in just 4δ, as soon as no message is delivered more than twice faster than others: GradedDAG (SRDS'23) required furthermore no messages reordering.

• Super Fast Pipelined Blocks. This is an upgrade of previous approaches for pipelining: in 2-chain VABA, Cordial Miners (DISC'23) and GradedDAG, a block pipelined by a leader in the middle of the view had almost twice larger latency than the non-pipelined block. Our design provides a fast path deciding the pipelined block with even smaller latency than the non-pipelined block. The fast delay is guaranteed in all executions with a fair scheduler, but remarkably, whatever the behaviors of faulty processes. Consistency is preserved by a lightweight mechanism, of one threshold signature appended per proposal. Instantiated with the previous protocols, it yields: s2PAC$^{\text{lean}}$, with fast decision of pipelined blocks in 4δ; s2PAC$^{\text{BIG}}$, in 3δ; and sGradedDAG, in 3δ.

INTRODUCTION

A. Model and Definitions

Network and corruptions. The system consists of $n = 3f+1$ processes $\mathcal{P}_1, \dots, \mathcal{P}_n$, of which f are actively corrupt by the adversary. For simplicity we consider static corruptions, we refer to Sec. VII-2 for adaptive security. Processes are linked by pairwise asynchronous channels without authentication nor secrecy, of which the delays are controled by the adversary. They have access to a global common coin (Sec. I-3), as in all efficient related works [2, 40, 46, 65, 29, 73, 24, 22, 33, 21]. As in most works, they have access to a PKI and to signatures. We denote δ the largest (unknown) delivery delay of a message between honest processes in a given execution. To multicast a message m is the instruction to send it to all processes.

Asynchronous Blockchain Consensus (a.k.a. async. BFT SMR). 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, σ) = true then we say that σ is a decision certificate (DecCert) vouching for the block b (following the formalism of "PVABA" in [23]). Each block contains a pointer to a parent block, thus forming a chain of blocks. A DecCert for a block automatically constitutes a DecCert for all the chain of its ancestors. When a process outputs a DecCert for a block, we say that the block and all its ancestors are *decided*.

Definition 1. ([20]) A *blockchain consensus* must satisfy

- **liveness**: in an infinite execution, there is an infinitely long chain of decided blocks;
- **consistency**: if two chains of decided blocks exist, then one must be prefix of the other;

- p-**quality** ([39, 2, 44]) for some $p > 0$: the asymptotic proportion of decided blocks in which transactions were input by honest processes, is at least p.

These specifications are matched by [59, 28, 53, 65, 46, 38, 73, 49, 23, 11, 64, 24, 22, 21]. By contrast, the liveness of [30, 57, 62, 5] 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 [41] for developping asynchronous blockchain consensus.

Throughput. We use the following estimate, equal to the computation time done per decided block per process. Since our protocols and previous works proceed by iterations called *views* (a.k.a. "waves"), the estimate is itself equal to

$$\text{throughput} = \frac{\textit{blocks/view}}{\textit{computation/view}} \qquad (1)$$

where *blocks/view* is the average number of decided blocks per view and *computation/view* the average computation time per view per process.

Latency measured on a single-shot MVBA. The definition of the latency to decide a transaction in a blockchain protocol varies in the literature. Either the timer is started when one process receives the transaction (the scripts of [63] and [5]; or when all processes receive it ([20]); or, later, when furthermore all players are in the same view ([20] "view-based latency"). We thus use a more universal metric, which is the latency of a *single-shot instance*, also known as "MVBA" (or "VABA"). MVBA is one of the most important ingredients in distributed systems (Sec. C-1). In an MVBA protocol, each process i starts with an input block b_i at time $t = 0$: we start the timer here. We call *latency of the execution* the first time t at which *one* honest process outputs a DecCert for a block (b, σ). This convention, of when to stop the timer, is in line with [63]. 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. The worst-case expected latency, over all adversaries, is what we call the expected latency of the MVBA. For further emphasis, we will sometimes prefix it with "(worst-case)". We adopt our definitions everywhere, including for related works.

Good-case latency. sMVBA [46, 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 2. 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 [49, 24] use the terminology "good case" to denote instead the best case latency ([46, 22]), which is the smallest achievable one.

The number of messages complexity., resp. the *bit complexity*, is the expected number of messages, resp. of bits, sent by honest processes per decided block.

The fastest blockchain consensus with a quadratic ($O(n^2)$) number of messages complexity, is called 2-chain_VABA [40]. Its claimed worst-case expected latency on a single-shot MVBA, according to our definition, is 9.5δ. Its good-case latency is 6δ. 2-chain_VABA is used as the asynchronous blockchain in Abraxas* [11, §6].

B. Attacks, Insight and New Approach

Both this subsection and Sec. II follow from the request of a reviewer, to outline our attacks early in the paper and to explain how the insight from the attacks is utilized in the new design. 2-chain_VABA proceeds by iterations called *views*. In each view v, processes run n parallel instances: (v, i), $\forall i \in [n]$ of a view of the consensus called "2-phase Hotstuff" [71, 56] (not to be mistaken with Jolteon [41]). Process $\mathcal{P}_i$ acts as the proposer of the i-th instance (v, i), $\forall i \in [n]$. Note that this general structure, of running n views in parallel then electing one a posteriori (to be detailed), dates back at least from [2] and has been adopted in at least [65, 46, 49, 24, 22]. Each instance (v, i) proceeds by two phases of vote. First, proposer $\mathcal{P}_i$ multicasts a block $\boldsymbol{b_{v,1,i}}$, called of "*height* 1", then hopefully receives $2f + 1$ votes on $\boldsymbol{b_{v,1,i}}$ which it combines into a quorum certificate (QC): $\boldsymbol{qc_{v,1,i}}$ vouching for $\boldsymbol{b_{v,1,i}}$. Then, the proposer $\mathcal{P}_i$ initiates a second phase of vote by multicasting a child block: $\boldsymbol{qc_{v,1,i}} \leftarrow \boldsymbol{b_{v,2,i}}$ called of "*height* 2". processes reply by a *height*-2 vote, enabling $\mathcal{P}_i$ to form a *height*-2 QC: $\boldsymbol{qc_{v,2,i}}$ and multicast it. Upon obtaining enough *height*-2 QCs, processes elect a view-v leader a posteriori: $\mathcal{P}_\ell = \mathsf{lead}(v)$, $\ell \in [n]$ then go to the next view $v{+}1$ and repeat. The QCs on blocks proposed by the leader $\mathsf{lead}(v)$ are called (a posteriori) **endorsed** QCs. A DecCert on a view-v *height*-1 block $\boldsymbol{b_{v,1,\ell}}$, is by definition a "2-chain" of endorsed QCs: $\boldsymbol{qc_{v,1,\ell}} \leftarrow \boldsymbol{qc_{v,2,\ell}}$ of heights 1 and 2, on $\boldsymbol{b_{v,1,\ell}}$ and one some *height*-2 block $\boldsymbol{b_{v,1,\ell}}$. Hence, only blocks proposed by leaders can possibly be decided. Note that the *height*-2 block $\boldsymbol{b_{v,2,\ell}}$ is not decided yet: we dub it the pipelined block, it will be decided at best only in the next view (we will overcome this with our fast path). For the sake of preserving Consistency despite the parallel instances, processes act in later views $\geqslant v{+}1$ as if only the view-v instance of the leader: $(v, \mathsf{lead}(v))$ had existed. Namely, they ignore the QCs for blocks produced in other parallel instances.

2-chain_VABA introduces relaxations to this baseline, opening the way to a number of attacks. The first one (Sec. II-1) breaks the consistency of both the published and implemented versions, it leverages a bug allowing processes to accept non-endorsed QCs. All next attacks apply after a fix to the bug (suggested by the authors on 2023-10-13). The second one (Sec. II-2) breaks liveness, it leverages a relaxation called "boosting" allowing interactions between parallel instances. The third one (Sec. II-3) breaks consistency of the implemented version [69]. It leverages a relaxed definition of a leader's QC. The fourth one (Sec. II-4) breaks liveness of a variant of 2-chain VABA suggested in DumboNG [38, Footnote 5], it leverages a relaxed trigger allowing early leader election. The authors confirmed all the attacks on 2023-10-25, then withdrew 2-chain_VABA: since the 2023-12-23 update, the Ditto BFT [41] is now instantiated with a generic MVBA.

Insight: the structural livelock of 2-chain_VABA**.** In Sec. II-5 we make a last attempt at fixing 2-chain_VABA, consisting in removing all the previous relaxations, which turned out to be insecure. We then describe a liveness attack on this last attempt, which shows that the *structural livelock of* 2-chain_VABA *comes from the voting rule of 2-phase Hotstuff* [16, 71, 56]. Namely, a process cannot vote for a block descendent of an (endorsed) QC: *qc*, if the process saw an (endorsed) QC of higher view number: qc_{high} on a conflicting branch. We are faced with a dead-end: it is well-known ([70, §4.4]) that in 2-phase Hotstuff, the only way to escape the livelock is to *wait* for the maximum eventual network delay: Δ. More precisely, a proposer in a new view v which does not have a view-$(v-1)$ QC, is instructed to wait Δ (in both [71, 56]). Then it selects the QC of the highest view which it has: qc_{max}, and proposes a child block: $qc_{max} \leftarrow b$. But under asynchrony no such upper-bound Δ is known. Thus, we use instead as baseline the classical voting mechanism of PBFT/SBFT/Jolteon [43, 41]. There, processes vote for a block if it comes appended with a proof: π, that no other conflicting block could have been committed in any earlier view. But in [70, §7.2] it is stressed that the classical proofs used have linear $O(n)$ complexity (in PBFT/SBFT/Jolteon: consist of $2f+1$ signatures on *distinct* messages; in Diem [47, 27, 4]: aggregated with BGLS [13], costing $f+1$ pairings to verify). Since n proposers run in parallel, this would blow up the total complexity to $O(n^3)$. Building such a small proof π was achieved by the breakthrough sMVBA [46], used since in [74, 72, 23, 45]. However there is potential for simplification and speedup, in particular in the chained regime as we are going to see. The proof π in sMVBA: is built incrementally as the concatenation of several threshold signatures (line 30 of Algorithm 3 of [46]), takes two rounds to be built (PreVote then Vote), and the specification of its validity predicate takes 19 lines ([46, Algorithm 4]).

New Approach for 2-Phase Asynchronous Consensus: 2PAC**.** Instead of trying to further reduce the proof for the same predicate, we instead allow a *relaxed predicate*, thanks to a new *recycling-friendly* chaining. Our observation is that *in our context of* n *parallel instances*, a proposer $\mathcal{P}$ starting a new view v is *at least guaranteed* to timely obtain a QC: qc_{v-1} (actually $\geqslant f+1$ QCs) produced in an instance of the previous view $v-1$. With probability up to $1/3$, $\mathcal{P}$ may not obtain any *view*-$(v-1)$ *endorsed* QC, since the previous leader lead$(v-1)$ could possibly be corrupt or slow. Our key observation is that $\mathcal{P}$ can still safely propose a child block of a *non-endorsed* view-$(v-1)$ QC: $qc_{v-1} \leftarrow b$. Namely, our mechanism is that processes are allowed to vote for such b as soon as they could verify the *relaxed predicate that no decision conflicting with* qc_{v-1} *was taken in the previous view* $v-1$ *only*. Safety of our mechanism intuitively follows from a «chain of consistency». Namely, let us assume the *Statement*: no view-$(v-1)$ QCs conflicts with any prior view-$(<v-1)$-DecCert. We now prove that the statement also holds for v: it will then follow, by induction, that the statement holds for all v. *Sketch proof*: consider a *view*-v QC: qc_v. By our mechanism it is always descendent of a *view*-$(v-1)$ QC: qc_{v-1}. By the assumption, qc_{v-1} does not conflict with any *view*-$(<v-1)$ DecCert. Thus is remains to show the *Claim* that it does not conflict with any *view*-$(v-1)$ DecCert. *Either* qc_{v-1} is endorsed, then the claim follows from quorum intersection. *Or it is not*, then the DocG proves that no *view*-$(v-1)$ DecCert can possibly exist, which implies a fortiori the claim. We slightly more detail this sketch proof of consistency in Appendix A-A1 (this time the actual notation of 2PAC), and fully formalize the proof in Sec. III-1.

Our mechanism brings many advantages for free. A proof π of the relaxed predicate is small, and consists only of a threshold signature on $2f+1$ declarations «I did not vote for the leader's QC in view-v». We call this a "declaration of a certificate-less group" (DocG). It can be formed in one round instead of two in sMVBA: this is how we gain 0.5δ expected latency. Last but not least, ability to always propose a child of a QC of the last view, yields $+50\%$ more decided blocks than 2-chain_VABA (and sMVBA, as if chained with pipelining). Indeed in those prior approaches, proposers propose child blocks of the *endorsed* QC: qc_{max} of the highest view which they saw. As a result, quorum-certified blocks produced in the $1/3$ of views with a slow or corrupt leader were dumped. Our mechanism *is not uncle referencing* (as in "DAG-2PAC" below), since such non-endorsed QCs are inserted in straight line in the chain. Hence it brings **no computation overhead**: this alone explains a net $+50\%$ throughput gain. This is illustrated in Figure 3.

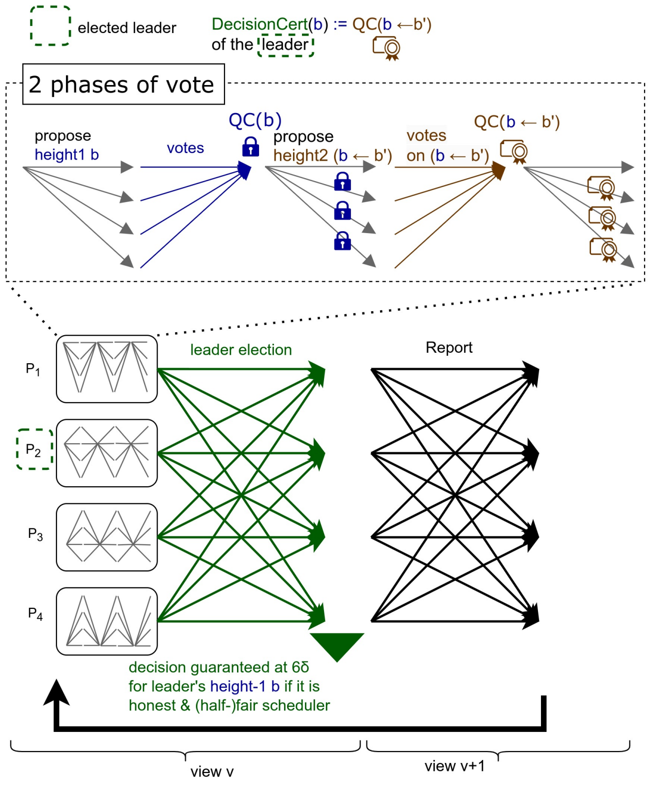


Figure 1: Bird's eye view of 2PAC$^{\text{lean}}$, omitting the new ingredients: propose-QC-of-previous-view, and DocG.

Overview of 2PAC **(without fast pipelining).** We now put the previous ideas in order. We start by the first iteration, called view $v=1$, which is simpler. Each process $\mathcal{P}_i$ acts as a so-called *proposer* of its instance $(v=1, i)$ of two-phases of vote (depicted in the dotted rectangle of Figure 1, for proposer $\mathcal{P}_i = \mathcal{P}_1$ and $n = 4$ processes). We describe an instantiation, called 2PAC$^{\text{lean}}$, in which votes are centralized by proposers, and thus which has $O(n^2)$ number of messages complexity. $\mathcal{P}_i$ initiates the first phase of vote by multicasting a so-called *height*-1 block $\boldsymbol{b_{v=1,1,i}}$ (b in Figure 1). Processes reply by signed *height*-1 votes on $\boldsymbol{b_{1,1,i}}$, which $\mathcal{P}_i$ combines into a *view*-1 *height*-1 QC: $\boldsymbol{qc_{1,1,i}}$. Then $\mathcal{P}_i$ initiates the second phase of vote by multicasting a child block of so-called *height*-2: $\boldsymbol{qc_{1,1,i}} \leftarrow \boldsymbol{b_{1,2,i}}$ (b' in Figure 1). Processes reply by a signed *height*-2 vote on $\boldsymbol{b_{1,1,i}} \leftarrow \boldsymbol{b_{1,2,i}}$. $\mathcal{P}_i$ combines them into a *height*-2 QC: $\boldsymbol{qc_{1,2,i}}$ which it multicasts, which finishes its instance. Upon receiving $2f+1$ *view*-1 *height*-2 QCs from distinct proposers, a process j multicasts a *coin share*. Any $2f+1$ coin shares can be combined into a *coin-QC* which reveals the identity of the so-called *view*-1 leader: $\mathcal{P}_\ell := \mathsf{lead}(v = 1)$ ($\ell = 2$ in Figure 1). A DecCert is by definition a 2-chain of QCs on $\mathcal{P}_\ell$'s blocks: $\boldsymbol{qc_{1,1,\ell}} \leftarrow \boldsymbol{qc_{1,2,\ell}}$, it enforces decision on $\mathcal{P}_\ell$'s *height*-1 block $\boldsymbol{b_{1,1,\ell}}$. Upon learning the leader, a process enters the next view $v = 2$. There is a probability $2/3$, not only $1/3$, that such a DecCert is formed for the leader of view $v = 1$. The reason is well-known ([2, 46]): for the leader $\mathcal{P}_\ell$ to have been revealed to the adversary, it must be that at least one honest process $\mathcal{P}$ received *beforehand* $2f+1$ *height*-2 QCs. Hence, by unpredictability of the coin, the indices of these $2f+1$ proposers are *independent* of ℓ. In particular, there is $2/3$ probability that one of them has proposer $\mathcal{P}_\ell$, hence, is a DecCert.

Upon entering such a new view $v \geqslant 2$, processes report to each other what they have seen or not. This step is called "Report". Denoting $\mathcal{P}_{\ell^{-1}} = \mathsf{lead}(v-1)$ the leader of the previous view $v-1$, each $\mathcal{P}_i$ multicasts:

- *Either* a *height*-1 endorsed QC: $qc_{v-1,1,\ell^{-1}} \leftarrow b_{v-1,2,\ell^{-1}}$, if $\mathcal{P}_i$ has such one (wrapped in a *height*-2 block of $\mathcal{P}_{\ell^{-1}}$);
- *Or* a signed declaration: $\{\text{no\_endorsed\_height-1\_QC}, v\}_i$ that $\mathcal{P}_i$ did not see any *view*-$(v-1)$ *height*-1 endorsed QC, and thus did not cast a vote for any $\mathcal{P}_{\ell^{-1}}$'s *height*-2 block.

Upon receiving (up to) $2f+1$ such reports of a new view v, a process $\mathcal{P}_i$ starts acting as proposer of the new view instance (v, i), as depicted by the back arrow at the bottom of Figure 1. It builds its new proposed block $\boldsymbol{b_{v,1,i}}$ depending on the following two cases:

- *Either* $\mathcal{P}_i$ received (or already has) an endorsed view-$(v-1)$ QC, then it builds its proposal on the top of it: $qc_{v-1,1,\ell^{-1}} \leftarrow b_{v-1,2,\ell^{-1}} \leftarrow \boldsymbol{b_{v+1,1,i}}$ (the QC is wrapped in a *height*-2 block of proposer $\mathcal{P}_{\ell^{-1}}$). Processes accept to vote for it, enabling $\mathcal{P}_i$'s instance to go through.
- *Or* it must be that $\mathcal{P}_i$ received signed declarations: $\{\text{no\_endorsed\_height-1\_QC}, v\}_j$ from $2f+1$ $\mathcal{P}_j$'s. It combines them into a threshold signature (which we called a DocG). In that case, $\mathcal{P}_i$ *freely chooses* any *view*-$(v-1)$ *height*-2 QC and builds a child of it: $qc_{v,2,j} \leftarrow \boldsymbol{b_{v,1,i}}$. Processes accept to vote for $\boldsymbol{b_{v,1,i}}$ upon checking the DocG, enabling again $\mathcal{P}_i$'s instance to go through.

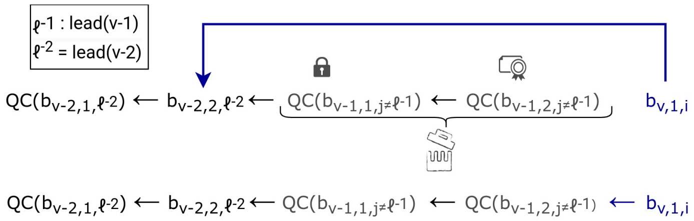
C. Quantitative Results

We now state the properties of the protocol that we have just outlined: 2PAC$^{\text{lean}}$, as well as a faster variation with cubic complexity, and a doubling of the speed of pipelined blocks. 2PAC$^{\text{lean}}$ is the new fastest blockchain consensus with quadratic ($O(n^2)$) number of messages complexity. It has 9.5δ worst-case expected latency, and the same bit complexity as previous works [2, 40, 46], i.e., $O\big((n\kappa)^2 + n^3\big)$ or $O\big((n\kappa)^2\big)$, depending if multi-signatures or constant-sized threshold signatures [12, 68] are used. In Table 2 we compare the latencies of a single-shot 2PAC$^{\text{lean}}$ MVBA, with the previously fastest one. 2PAC$^{\text{lean}}$ is formalized and proven in Sec. III.

latencies, in δ	Worst-case expected	Good-case (Def. 2) := until all processes decide, if no faults & half-fair scheduler
sMVBA [46]	10 (†)	6
2PAC$^{\text{lean}}$	**9.5**	**6**

Table 2: MVBAs with a quadratic number of messages. (†) 12δ latency for sMVBA is stated in [46, 23]. Our estimate (Sec. A) is instead 10δ, following our definition of latency. This is consistent with the updated estimate of 11δ notified on 2024-05-26 by the authors of sMVBA [46], since they measure latency instead until *all* processes output a DecCert.

$+80\%$ **to** $+100\%$ **throughput over Chained sMVBA, thanks to the new design.** Despite 2PAC$^{\text{lean}}$ being the first correct quadratic asynchronous blockchain consensus, we can compare its throughput to what would have given previous approaches. Consider a chained version of sMVBA [46], in which proposers would pipeline a child block in each of their provable broadcast (PB) instance (two per view).


ℓ-1 : lead(v-1)
ℓ-2 = lead(v-2)
QC(b_{v-2,1,ℓ-2}) ← b_{v-2,2,ℓ-2} ← QC(b_{v-1,1,j≠ℓ-1}) ← QC(b_{v-1,2,j≠ℓ-1})
b_{v,1,i}
QC(b_{v-2,1,ℓ-2}) ← b_{v-2,2,ℓ-2} ← QC(b_{v-1,1,j≠ℓ-1}) ← QC(b_{v-1,2,j≠ℓ-1}) ← b_{v,1,i}
Figure 3: New view v proposal by $\mathcal{P}_i$: $\boldsymbol{b_{v+1,1,i}}$ when no endorsed *view*-$(v-1)$ QC is known to $\mathcal{P}_i$. Top: in chained sMVBA (& 2-chain_VABA). Bottom: in 2PAC.

As explained in Sub-section -B, and illustrated in Figure 3, in all views with a too slow leader (up to $1/3$ of views), all quorum-certified blocks are dumped forever. Whereas 2PAC saves the work of such views, yielding a $+50\%$ to the numerator of Eq. (1). As for the denominator of Eq. (1), we empirically estimate (Sec. VI) a computation time gain of -20% to -25%, mainly brought by the removal of one round and an extra optimization (checking DocG only once).

2PAC$^{\text{BIG}}$: the fastest blockchain consensus with cubic message complexity. It is a surprisingly simple variant of 2PAC$^{\text{lean}}$.

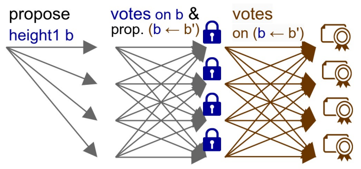


Figure 4: Compressed two phases of vote in 2PAC$^{\text{BIG}}$

As depicted in Figure 4, it simply consists in compressing each 2δ-long phase of vote, into 1δ of all-to-all sending of all votes. We compare in Table 5 its latencies on a single-shot MVBA, with those of other cubic protocols.

latencies, in δ	Worst-case expected	Good-case (Def. 2) := until all processes decide, if no faults & half-fair scheduler	Best-case
Bullshark [65]	30	–	6
Cordial Miners [49]	7.5	–	5 (*)
GradedDag [24] (†)	6.5	∞	4
LightDAG2 [22] (‡)	$12(f'+1)$	?	4
2PAC$^{\text{BIG}}$	**6.5**	**4**	**4**

Table 5: MVBAs with $O(n^3)$ messages. (*) Cordial Miners has best-case latency 5δ, since leader election is in the 5-th round. In [49, 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 [65] under a half-fair scheduler). (†) [24] see below. (‡) The "best-case" figures for [65, 22] are taken from [22, TABLE 1]. f' denotes the actual number of cheaters. We could not analyze LightDAG2 under a half-fair scheduler, as we do not have access to a pseudocode or code.

GradedDAG [24] is the closest to 2PAC$^{\text{BIG}}$ in terms of structure and performance. In [24, TABLE I] the expected latency is stated as 7.5δ. According to our definition, we estimate it instead to 6.5δ for a single-shot MVBA. This is because all their views have 5 rounds, in which voting is in the 4-th round (cf also Sec. C-2 about indexation conventions). In Sec. C-3 we describe (with a picture) a fault-free scenario in the first view with a half-fair scheduler, in which $\frac{3}{2}\delta_{\text{fast}} < \delta < 2\delta_{\text{fast}}$, such that no decision is reached in the first view. Since this scenario can repeat in all higher views with very bad luck, this explains the stated ∞ good-case latency in Table 5. The scenario is that the process $\mathcal{P}_4$ to-be-elected leader is slower, so that others refuse to vote to form a *height*-2 QC (in our terminology) for $\mathcal{P}_4$. The subtle difference enabling 2PAC$^{\text{BIG}}$ to escape such scenarios and reach 4δ latency, is that processes accept to vote for *height*-2 blocks *even after* they received $2f+1$ *height*-2 QCs.

s2PAC$^{\text{lean}}$ & s2PAC$^{\text{BIG}}$: Super Fast Pipelined Blocks. As in the previous asynchronous blockchain consensus 2-chain_VABA [40] & Cordial Miners [49], 2PAC$^{\text{lean}}$ also pays a latency price for pipelining. Consider a block pipelined by some view-v leader $\ell \in [n]$, i.e., a *height*-2 proposed block $\boldsymbol{b_{v,1,\ell}} \leftarrow \boldsymbol{b_{v,2,\ell}}$. While a *view-$v$* DecCert enforces decision only on the *height*-1 block $\boldsymbol{b_{v,1,\ell}}$, decision of the pipelined block $\boldsymbol{b_{v,2,\ell}}$ is delayed at least until the end of the next $v+1$. In Sec. IV we introduce a fast path deciding the pipelined *height*-2 block of an honest leader in just 4δ, as soon as the scheduler is fair, whatever the arbitrary behaviors of corrupt processes. This is even less than the 6δ taken by the *height*-1 block! Our fast path technique applies to both 2PAC$^{\text{lean}}$ & 2PAC$^{\text{BIG}}$ (and to GradedDAG), yielding s2PAC$^{\text{lean}}$ & s2PAC$^{\text{BIG}}$ (and sGradedDAG, in paragraph V-0a). In Table 6 we compare their latencies under a fair scheduler to those of previous works.

The technique is simply that processes in view v cast a SPEED_VOTE for a *view-v height*-2 block upon seeing a (*height*-2) QC for it (Figure 7). $2f+1$ 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 a new proposer which forked from an endorsed *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 $2f+1$ SPEED_VOTEs for a *height*-2 block can exist, hence, that no *height*-2 block of the previous view could be speed-DecCert'ed.

unit: δ	Fair scheduler	
	Arbitrary corruptions	Fault-free
sMVBA [46] (as if chained)	14	10
2PAC$^{\text{lean}}$	13.5	10
s2PAC$^{\text{lean}}$ (Thm 5)	**4**	**4**
$O(n^3)$ messages:		
Cordial Miners [49]	11.5	9
GradedDag [24]	9.5	6
2PAC$^{\text{BIG}}$	9.5	7
sGradedDag	**3**	**3**
s2PAC$^{\text{BIG}}$ (Thm 6)	**3**	**3**

Table 6: Delay between the pipelining of a *height*-2 block by an honest leader, and decision of it by *all* processes. Top: protocols with a quadratic number of messages complexity; bottom: with cubic complexity.

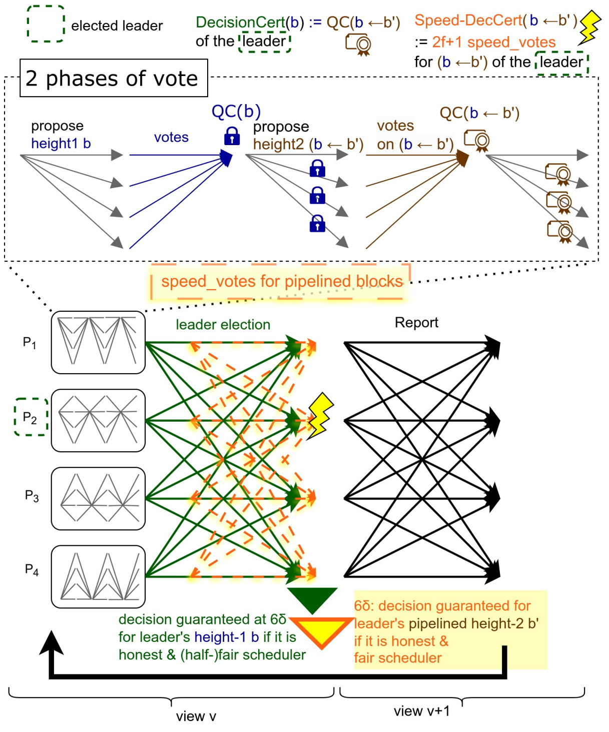

elected leader
DecisionCert(b) := QC(b ←b') of the leader
Speed-DecCert(b ←b') := 2f+1 speed_votes for (b ←b') of the leader
2 phases of vote
propose height1 b
votes
QC(b)
propose height2 (b ← b')
votes on (b ← b')
QC(b ← b')
speed_votes for pipelined blocks
leader election
Report
P1
P2
P3
P4
decision guaranteed at 6δ for leader's height-1 b if it is honest & (half-)fair scheduler
6δ: decision guaranteed for leader's pipelined height-2 b' if it is honest & fair scheduler
view v
view v+1

Figure 7: Bird's eye view of s2PAC$^{\text{lean}}$, omitting the added consistency mechanism.

DAG-2PAC: DAG-ification, with the classical trade-off. The generic compiler of Conflux [52], formalized in [3], straightforwardly applies to any of the 2PAC protocols. This compiler is, roughly, a combination of uncle-referencing and of parallelization ([65, 66]). It would give "DAG-2PAC", as follows: each process $\mathcal{P}_i$ uses a predetermined $1/n$-th fraction of the memory pool to make the blocks which it proposes. In each proposed block $b_{v,1,i}$, in addition to the reference to the parent block, $\mathcal{P}_i$ also includes QCs of $2f$ other blocks, dubbed the "uncles". More precisely, the "uncles" must be of *view* $v-1$, of which at least $f+1$ QCs of *height* 2. Then, decision of a block enforces decision on all the tree of ancestors, in a deterministic topological order as discussed in [52, 3]. This parallelization gain comes with two prices. First, uncle-referencing adds computational load. Recall that s2PAC$^{\text{lean}}$ and 2PAC$^{\text{BIG}}$ are free of uncle-referencing, which is why they achieve a net throughput gain over (a chaining of) sMVBA. Second, transactions input by non-leaders must wait $+6\delta$ longer to be decided (in the case of 2PAC$^{\text{lean}}$), compared with if processes input in their blocks every transaction which they see. Hence, we believe that the choice to DAG-ify or not a blockchain consensus should not be all-or-nothing, but instead left as a flexible implementation cursor.

I. Notation: View, Height, QC, Endorsement

In this section we set a consistent notation used throughout the paper, including for 2-chain_VABA and sMVBA. It is obtained as a simplification of the one of 2-chain_VABA [40]. In particular we drop their notion of "rounds", and of "fallback" vs "regular" blocks and QCs, since we are not concerned with switching back-and-forth from a partially synchronous protocol (called steady-state in [40]). Instead, the tuple of (*view*, *height*) is enough to uniquely rank blocks in both 2-chain_VABA and 2PAC.

1) Threshold- or multi- 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 fully non-interactive (fNI) $2f+1$-multi- (or threshold-) signature scheme [61] takes as input $2f+1$ "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: $\{m\}$. The signing public key(s) are also appended, excepted in threshold signatures. All previous notations imply that the signature is appended with the plaintext m. There exists fNI threshold signatures without pairings [7], and post-quantum fNI multi-signatures [50, 35].

2) 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 [40], 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'.

3) Common coin: ([18, §4.3]) For any view number v, each process can generate a coin share for v. Then there is a public algorithm which takes a threshold number of any valid coin shares from distinct processes for the same v, and combines them into a certificate called coin-QC: qc_{coin}, vouching for unique value $i \in [n]$. We then say that process $\mathcal{P}_i$ is the elected leader of *view* v, and denoted $\mathsf{lead}(v) = \mathcal{P}_i$. 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 $1/n + \text{negl}(\kappa)$.

- In 2-chain_VABA the threshold number is $f+1$.
- In 2PAC the threshold number is instead $\mathbf{2f+1}$.

The $2f+1$ will be important to achieve our claimed 6δ good-case latency of 2PAC, in particular even if the scheduler is only half-fair: cf Sec. C-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, 68, 7] (the latter is not pairing-based, which positively answers [73, 29]). This smallness assumption of coin-QCs can be lifted: as explained in Sec. B-1, using a Bracha-like termination gadget for election (see Sec. 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 [14, §5 & §7.1], improved by [32], or also [58].

4) Blocks, QCs, Endorsement:

- **Block format**. A block is formatted as $b = [v, h, i, \mathit{parent}, \mathit{txn}, \mathit{id}]$ where:
  - v is the *view* number of b;
  - $h \in \{1, 2\}$ is called the *height*;
  - $i \in [n]$ is the index of the *proposer* $\mathcal{P}_i$ of the block;
  - *parent* is the parent block. 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;
  - $\mathit{id} = \mathrm{H}(\mathit{parent}, v, \mathit{height}, \mathit{proposer}, \mathit{txn})$ is the unique hash digest of $(\mathit{parent}, v, \mathit{height}, \mathit{proposer}, \mathit{txn})$;

  For simplicity we may omit *txn* and *id* from the description. We denote $b.\mathit{id}$, $b.v$ etc. the *id*, view number etc. of a block b. We will use $b_{v,h,j}$ to denote a *view-v height-h* block with *proposer* j, and omit j when not necessary.
- **Quorum certificate.** A QC: *qc* vouching for a block b is a $(2f+1)$-multi (or threshold) signature on the message $(b.v, b.\mathit{height}, b.\mathit{proposer}, b.\mathit{id})$. For $v := b.v$ and $h := b.\mathit{height}$, we say that *qc* is a *view-v height-h* QC.
- **Ranking.** Blocks and QCs are ranked by their view numbers, then by their heights, i.e., by $2v + \mathit{height}$. Hence, we abuse notation and shorten as $qc \geqslant qc'$ the relation $qc.(2v + \mathit{height}) \geqslant qc'.(2v + \mathit{height})$. 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 will yield a $+50\%$ increase in the numerator of the throughput formula (Eq. (1)), compared to a chaining of sMVBA.
- **Endorsed QC.** Once a process has a coin-QC: qc_{coin} for *view* v that elects process $\mathsf{lead}(v)$ as the leader, we say that any *view-v proposer-*$\mathsf{lead}(v)$ QC is *endorsed* (by qc_{coin}).
- **Parent-referencing with QCs.** In 2-chain_VABA (and an as-if chained version of sMVBA), the *parent* $\widetilde{b} \leftarrow b$ of a block b is always encoded as a QC: $\widetilde{qc}$ on $\widetilde{b}$. We then say that $\widetilde{qc}$ is "wrapped inside b", and denote $\widetilde{qc} \leftarrow b$. We then abuse notation by calling directly b a "child" of $\widetilde{qc}$.
- **Parent-referencing without QCs.** There are cases where the process *proposer* of a block b has no QC on its parent. In this case, for simplicity of the exposition we will consider that the proposer encodes the *parent* $\widetilde{b}$, signed by $\widetilde{b}$'s proposer, as a full copy inside b. The optimized way would be of course instead to reference the *parent* $\widetilde{b}$ via its hash *id*, and diffuse the signed $\widetilde{b}$ in case it had been badly diffused. Then, processes would vote for b only if they received a preimage $\widetilde{b}$ of the hash (and all its ancestors). Hence, a QC on b would

still imply that all its ancestors are efficiently retrievable, since $f+1$ honest processes would have received them.

- **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.

II. Attacks

2-chain_VABA is defined page 8 & Appendix C of [40] as a particular instantiation of the protocol called Ditto, published in [42, 40]. 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 [40, Figure 5]. In particular, the steady-state path of Ditto described in their [40, 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 [40], we make it explicit in Figure 8. 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. The 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 [69]. The authors confirmed (on 2023-10-13) that this was an oversight and that they should be added.

1) Without the checks of endorsement, an attack on consistency: Absence of the two highlighted checks allowed the following obvious attack. The isolated view-1 to-be-elected leader ℓ forms a 2-chain of QCs: $qc_{1,1,\ell} \leftarrow qc_{1,2,\ell}$ but delivery of $qc_{1,2,\ell}$ to other processes is delayed very long. Nevertheless, since ℓ is leader then $qc_{1,1,\ell} \leftarrow qc_{1,2,\ell}$ is a DecCert on $b_{1,1,\ell}$. In view 2, a corrupt to-be-elected leader $\mathcal{P}_i$ ignores the wrapped *height*-1 QC: $qc_{1,1,\ell} \leftarrow b_{1,2,\ell}$ received from ℓ, and instead proposes a block child of its own non-endorsed view-1 *height*-2 QC: $qc_{1,2,i} \leftarrow b_{2,1,i}$. Processes accept to vote for it, ultimately leading to a DecCert for $b_{2,1,i}$, which enforces decision on all its ancestors, in particular on $b_{1,1,i}$ conflicting with $b_{1,1,\ell}$.

2) Leveraging the boosting strategy, an attack blocking liveness: The following attack applies to the published version [40], after adding the highlighted endorsement checks. The rough idea is that a fixed Process $\mathcal{P}_k$ is corrupt, and in each view v it is made "fast-then-oblivious" of height-2 votes. First, $\mathcal{P}_k$ proposes an *height*-1 block $b_{1,1,k}$, then collects very quickly votes on it, forming an height-1 QC $qc_{1,1,k}$, which it multicasts (wrapped in a *height*-2 block, say $b_{1,2,k}$). All processes receive it before any other *height*-1 QC, so they follow the [Propose *height*-1] step and build their *height*-2 block as childs of $qc_{1,1,k}$. As a result, all their 2-chains of QCs: $qc_{1,1,k} \leftarrow qc_{1,1,i}$ are with different proposers $i \neq k$ so none of them is a DecCert. Whereas $\mathcal{P}_k$ ignores all *height*-2 votes so never forms either a DecCert. Thus if a DecCert is ever formed, it can be only in view 2 or after. The attack can be mounted again from the same single corrupt $\mathcal{P}_k$ in higher views, preventing any DecCert to be ever formed.

3) Leveraging the relaxed endorsement definition in the implemented version: an attack on consistency: In the implemented version [69], processes consider any *height*-1 QC: $qc_{v,1,j}$ as endorsed as soon as it is wrapped inside a *height*-2 block of which the proposer is the leader, i.e., $qc_{v,1,j} \leftarrow b_{v,2,\ell}$ with $\ell = \mathsf{lead}(v)$, whereas possibly $\ell \neq j$. [Technically: lines 627-628 of [69] $\mathcal{P}_\ell$ sets itself as the "acceptor" of $qc_{v,1,j}$, 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: $qc_{v,1,\ell} \leftarrow qc_{v,2,\ell}$ to an isolated process. In parallel, it generously endorses another QC: $qc_{v,1,j} \leftarrow b'_{v,2,\ell}$ which it diffuses to other processes. As a result, in the next view all processes propose child blocks of this concurrent $qc_{v,1,j}$, hence it ultimately gets decided.

4) Leveraging the trigger for early leader election: a liveness attack on the Dumbo-NG variant: In Dumbo-NG [38] they observe that 2-chain_VABA does not have quality, because of the boosting strategy. Hence, they suggest a variant of Ditto, consisting in removing the boosting strategy[1]. It could be formalized as the following modification:

Propose *height*-2. Upon the ~~first~~ height-2 block $b_{h,i}$ ~~(by *any* process j)~~ by $\mathcal{P}_i$ itself is certified by some qc, …

We observe that it allows the following attack which slows-down latency by $n/3\times$. Surprisingly, a strenghtening of the attack actually prevents quality. The attack leverages the trigger for early leader election highlighted in footnote (iv) of Figure 8. The idea is simply that in each view, a very fast proposer $\mathcal{P}$ delivers to all processes an endorsed *height*-2 QC: qc before they receive any other. Following [trigger leader election], they all sign-then-multicast qc. Reception of $2f+1$ such signed copies of qc by an honest process triggers it to multicast a coin share: helped by the remaining f coin shares from the adversary, processes learn the leader then move to the next view. Since $\mathcal{P}$ is elected with probability only $1/n$, the expected number of views before output is n. Moreover if $\mathcal{P}$ is fast enough, then other proposers will not even have the time to form their height-1 QCs. So the only value possibly ever decided is the one of qc: if $\mathcal{P}$ is corrupt, then the quality is thus equal to 0.

5) Last Failed Attempt: It is interesting to study a last (failed) attempt at fixing 2-chain_VABA, consisting in removing both the boosting strategy and the trigger for early leader election. Namely, we revert to the classical leader election of AMS/sMVBA [2, 46] & 2PAC, i.e., that a process must wait for *height*-2 QCs from $2f+1$ *distinct* proposers. Such a variant of 2-chain_VABA would thus be equal to n parallel 2-phase Hotstuff [71, 56] (not to be mistaken with Jolteon [41], which is analogous to SBFT [43]). It is well-known that in 2-phase Hotstuff, a new leader must wait Δ (the upper bound on message delivery) before it can propose, otherwise the protocol is not live: cf [70, §4.4]. This inspires the following attack. In view 1, a malicious elected leader delivers its endorsed *height*-2 QC only to an isolated honest process $\mathcal{P}$. Then in view 2 the

[1][38, page 7] footnote 5: «the honest nodes would propose its height-2 f-block chained to any earliest height-1 f-block that it receives, and unfortunately the adversary can always propose the fastest height-1 f-block to manipulate the output.»

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* qc_{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 $r_{vote}[j] \leftarrow 0$ and $h_{vote}[j] \leftarrow 0$ for all $j \in [n]$;
- *then* ***(Propose height-1)*** multicast a *height*-1 block: $b_{v,1,i} = [v, 1, i, qc_{high}, txn, id]$.

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

- if $h = 1$, and $b_{v,1,j} = [v, 1, j, qc, txn, id]$ such that $qc \geqslant qc_{high}$ and such that qc is endorsed; or
- if $h = 2$, and $b_{v,2,j} = [v, 2, j, qc, txn, id]$ such that qc 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 qc

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

Leader Election. Upon receiving $2f+1$ 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 qc_{coin} of view $\geqslant v$ or $f+1$ valid coin shares to form a qc_{coin} of view $\geqslant v$, multicast qc_{coin}. Update $v \leftarrow qc_{\text{coin}}.v + 1$. Execute Lock and Decision.

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

Decision certificate. For any view number w, a DecCert for a view-w *height*-1 block with *proposer* $\mathcal{P}_\ell = \mathsf{lead}(w)$: $b_{w,1,\ell}$, is a 2-chain of endorsed QCs $qc_{w,1,\ell} \leftarrow qc_{w,2,\ell}$, 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 [40, 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 $qc_{v,1,j} \leftarrow b_{v,2,i}$. In the code [69] 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 [69] there is another predicate called "valid_qc(qc)", 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 [40, Figure 5], this predicate is written under the equivalent form: "$h = qc.height + 1$". The simpler equivalent form under which we wrote it, i.e., $qc.height = 1$, turns out to be also the way the public implementation does it (line 630 of [69]).

(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 ([69] lines 821-822). This will be leveraged by our attack of Sec. II-4. Note that the proofs of latency in [40] instead assumes them from distinct proposers, our last attack Sec. II-5 will break this alternative version. Both our attacks on consistency are compatible with both versions (either no condition on these QCs, or, requiring from distinct proposers).

(v) The specifications [40, Fig. 4 & 5] instruct explicitely to do [Lock] at two specific places, i.e., in [Timeout] and [Exit Fallback]. So said like this, one would deduce that qc_{high} is *not* updated elsewhere. But in the implemented version [69], processes update their qc_{high} at anytime upon receiving an endorsed QC. All our attacks are compatible with both versions.

Figure 8

height-1 proposal of $\mathcal{P}$ (child of the endorsed *height*-2 QC) is delivered too slowly: all processes have already proposed their block before receiving it. So they all propose blocks childs of the *view*-0 QC, thus $\mathcal{P}$ refuses to vote for these blocks. Provided all f corrupt processes stay silent, no quorum of $2f+1$ votes is reached for the *height*-1 proposals of the $2f$ honest proposers other than $\mathcal{P}$. Thus those $2f$ proposers are unable to form *height*-2 QCs, thus leader election is never triggered.

III. 2PAC$^{\text{LEAN}}$

We present 2PAC$^{\text{lean}}$ in Figure 9, in Theorem 3 we state then prove both its expected and good-case latencies, and its consistency. In Sec. III-4 we analyze its quality. In Sec. VII we sketch some variants and optimizations, further ones are deferred to Appendix B. The definition of a DecCert in 2PAC$^{\text{lean}}$ is the same as in 2-chain_VABA (Figure 8), let us recall it.

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

Recall that a DecCert on a block b is by definition also a DecCert for each block in the chain of its ancestors. Processes can form QCs and DecCerts out of votes even if they did not receive the *height*-1 block: $\boldsymbol{b_{w,1,\ell}}$ itself. Note that existence of the $\boldsymbol{qc_{w,1,\ell}}$ guarantees that the block and all its ancestors were received by $f+1$ honest processes, and thus that they are efficiently retrievable ([26, 36]). This is the reason why we measure latency only until output of a DecCert, which could happen before learning the block.

Theorem 3. *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. 2) in every fault free execution*

$2\text{PAC}^{\text{lean}}$

```
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: $qc_{v-1,1,\ell} \leftarrow b_{v-1,2,\ell}$ (wrapped in a height-2 block of $\ell = \mathsf{lead}(v-1)$)
        it multicasts it;
    else it freely chooses any view-$(v-1)$ height-2 QC $qc_{v-1,2,\ell}$
        and multicasts it appended with a signed eclaration $\{\text{no\_endorsed\_height-1\_QC}, v\}_i$.
Propose height-1. While proposedHeight1 $= 0$
    upon obtaining an endorsed view-$(v-1)$ height-1 QC wrapped in a height-2 block: $qc_{v-1,1,\ell} \leftarrow b_{v-1,2,\ell}$ ($\ell = \mathsf{lead}(v-1)$)
                                                            ▷ It might already have one upon entering $v$.
        creates a child view-$v$ block: $qc_{v-1,1,\ell} \leftarrow b_{v-1,2,\ell} \leftarrow \boldsymbol{b_{v,1,i}} = [v, 1, i, b_{v-1,2,\ell}, txn, id]$
            ▷ Optimization (will become standard in s2PAC^lean): in case it has a QC on $b_{v-1,2,\ell}$, then shortens the reference as: $qc_{v-1,2,\ell} \leftarrow b_{v,1,i}$.
        multicasts $\langle qc_{v-1,1,\ell} \leftarrow b_{v-1,2,\ell} \leftarrow \boldsymbol{b_{v,1,i}} \rangle_i$ and sets $1 \leftarrow$ proposedHeight1.
    upon receiving from $2f+1$ 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: $qc_{v-1,2} \leftarrow \boldsymbol{b_{v,1,i}} = [v, 1, i, qc_{v-1,2}, txn, id]$
        combines the signed declarations into $\{\text{no\_endorsed\_height-1\_QC}, v\}$
            ▷ DocG (declaration of a certificate-less group): proof that no $(v-1)$-DecCert can ever exist, thus $qc_{v-1,2}$ can safely be extended.
        multicasts $\big(\langle qc_{v-1,2} \leftarrow \boldsymbol{b_{v,1,i}} \rangle_i, \{\text{no\_endorsed\_height-1\_QC}, v\}\big)$ 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 $\boldsymbol{b_{v,1,j}}$,
    i.e., of one of the following forms:
    - either the raw signed block $\langle qc_{v-1,1} \leftarrow b_{v-1,2} \leftarrow \boldsymbol{b_{v,1,j}} = [v, 1, j, b_{v-1,2}, txn, id] \rangle_j$ such that $qc_{v-1,1}$ is endorsed;
    - or, appended with a DocG, i.e., of the form $\big(\langle qc_{v-1,2} \leftarrow \boldsymbol{b_{v,1,j}} = [v, 1, j, qc_{v-1,2}, txn, id] \rangle_j, \{\text{no\_endorsed\_height-1\_QC}, v\}\big)$;
                                                            ▷ Optimization: check a DocG only once per view, cf. Appendix B-2
        then votes for the block by replying $\{v, 1, j, id\}_i$ to $\mathcal{P}_j$.
QC height-1 & Propose height-2. if proposedHeight2 $= 0$, upon receiving $2f+1$ votes for its height-1 block $\boldsymbol{b_{v,1,i}}$
    combines the $2f+1$ signatures into a view-$v$ height-1 QC: $\boldsymbol{qc_{v,1,i}} = \{v, 1, i, id\}$ on $\boldsymbol{b_{v,1,i}}$
    creates a child height-2 block: $\boldsymbol{qc_{v,1,i}} \leftarrow \boldsymbol{b_{v,2,i}} = [v, 2, i, \boldsymbol{b_{v,1,i}}, txn, id]$, signs then multicasts it, then sets $1 \leftarrow$ proposedHeight2.
Vote height-2. For any $j \in [n]$, upon receiving for the first time a height-2 proposal: $\langle \boldsymbol{qc_{v,1,j}} \leftarrow \boldsymbol{b_{v,2,j}} = [v, 2, j, \boldsymbol{qc_{v,1,j}}, txn, id] \rangle_j$
                        ▷ i.e., a view-$v$ height-1 proposer-$j$ QC, wrapped in a height-2 block $\boldsymbol{b_{v,2,j}}$ of same proposer $j$
    votes for the height-2 block by replying $\{v, 2, j, id\}_i$ to $\mathcal{P}_j$.
QC height-2. For a view number $w \in \{v-1, v\}$, upon receiving $2f+1$ votes for its view-$w$ proposal $\boldsymbol{b_{w,2,i}}$: $\{w, 2, i, id\}_j$'s signed
    by distinct processes $\mathcal{P}_j$'s
    combines the $2f+1$ signatures into a view-$w$ height-2 QC: $\boldsymbol{qc_{w,2,i}}$. If $w = v$, it multicasts it.
        ▷ $w = v-1$ allows a slow $\mathsf{lead}(v-1)$ to form a height-2 QC while in view $v$: this will be used in the proof of $6\delta$ good-case latency (Sec. III-3).

Leader election. upon receiving $2f+1$ view-$v$ height-2 QCs of distinct proposers, then multicasts its view-$v$ coin share.
Coin QC and Advance view. upon receiving or forming a view-$v$ coin-QC: $qc_{\text{coin}}$, then it multicasts it and enters view $v+1$.
Decision. upon receiving or forming a DecCert for any view number $w$, i.e., a 2-chain $\boldsymbol{qc_{w,1,j}} \leftarrow \boldsymbol{qc_{w,2,j}}$ s.t. $j = \mathsf{lead}(w)$
    multicasts it and outputs it.
```

Figure 9

with a half-fair scheduler, all players decide by 6δ.

1) 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 4 below, $b'_{v',1}$ must be of the same view: $v' = v$. Thus both blocks have same proposer $\ell := \mathsf{lead}(v)$. Hence, we would have two QCs for conflicting *view*-v *height*-1 *proposer*-ℓ blocks: $b_{v,1,\ell}$ and $b'_{v,1,\ell}$, which is impossible by quorum intersection.

Lemma 4. *For all $v \geqslant 1$, consider a DecCert: $qc_{v,1} \leftarrow qc_{v,2}$ for a block $b_{v,1}$. Then no higher view-$(v' > v)$ height-1 QC: $qc'_{v',1}$ (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 $qc'_{v',1}$ is of minimal view $v' > v$ among those conflicting with $b_{v,1}$. By construction, $qc'_{v',1}$ is descendent of a *view*-$(v'-1)$ *height*-1 QC: $qc_{v'-1,1}$. By the classical *Claim* below, $v'-1 = v$. Existence of a DecCert for $b_{v,1}$ implies that at least $f+1$ processes had a *view*-v endorsed QC for $b_{v,1}$: $qc_{v,1}$ upon entering view $v' = v+1$. Thus no quorum of declarations:

$\{\text{no\_endorsed\_height-1\_QC}, v{+}1\}$ (DocG) 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 $qc'_{v',1}$.

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

2) Proof of (worst-case) expected latency $\leqslant 9.5\delta$: Let us fix any given adversary and denote E the expected latency of 2PAC$^{\text{lean}}$ 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 Sec. A-A. 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 $f{+}1$): $\mathcal{P}(v)$, received $2f+1$ *height*-2 QCs before the leader $\mathsf{lead}(v)$ was revealed. By unpredictability of the coin, the identity of $\mathsf{lead}(v)$ is independent of the proposers of these *height*-2 QCs. Hence, the probability that one of them is equal to $\mathsf{lead}(v)$ is $(2f{+}1)/n > 2/3$. 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 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 $E \leqslant 6\delta + 0.5 \times 7\delta = 9.5\delta$.

3) Proof of Good-case latency 6δ: Let us consider a one-shot MVBA good-case execution. Recall (Def 2) 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: t when a process enters view 2.

Since the process received a coin-QC, it must be that $6\delta_{\text{fast}} \leqslant t$ (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, $3\delta \leqslant t$. 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.

4) Quality of 2PAC$^{\mathit{lean}}$:

a) On a single-shot MVBA, if the scheduler is half-fair: If the first leader $\mathcal{P}_\ell = \mathsf{lead}(1)$ is honest, which happens with probability $> 2/3$, then all processes will receive its proposed *height*-2 block: $\boldsymbol{qc_{1,1,\ell}} \leftarrow \boldsymbol{b_{1,2,\ell}}$ 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.

b) If the scheduler is fair: If the leader $\mathcal{P}_\ell = \mathsf{lead}(v)$ of some view v is honest, which happens with probability $> 2/3$, then all processes will receive its proposed *height*-2 block: $\boldsymbol{qc_{v,1,\ell}} \leftarrow \boldsymbol{b_{1,2,\ell}}$ 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.

c) Worst-case: Let us fix any v and consider a 2-chain $b_{v,1,i} \leftarrow b_{v,2,i}$ 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 $2f+1$ *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 $f{+}1$ processes received its *height*-2 proposed block: $\boldsymbol{qc_{v,1,\ell}} \leftarrow \boldsymbol{b_{v,2,\ell}}$, thus for the same reason as above we have that the 2-chain $\boldsymbol{b_{v,1,\ell}} \leftarrow \boldsymbol{b_{v,2,\ell}}$ will end up in the decided chain (thus $i = \ell$). Since $\mathbb{P}(X) > 2/3$, and that $\mathbb{P}(\ell \text{ is honest}|X) > 1/2$, there is $> 1/3$-quality. We can obtain a higher estimate. Consider the bad event, included in $\neg X$, where no $\boldsymbol{b_{v,1,\ell}} \leftarrow \boldsymbol{b_{v,2,\ell}}$ 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 $\mathsf{lead}(w)$. Recall that when $\mathsf{lead}(w)$ built its *height*-1 proposal, it made a *free choice* between $2f+1$ certified-but-undecided chains $b_{v,1,i} \leftarrow b_{v,2,i} \leftarrow \ldots b_{w-1,2}$. 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 $\mathsf{lead}(w)$. With this definition, since $\mathbb{P}\Big(\mathsf{lead}(w) \text{ is honest}|\neg X\Big) > 1/2$, then $\mathbb{P}\Big(b_{v,1,i} \leftarrow b_{v,2,i} \text{ is corrupt input}|\neg X\Big) < 1/2$. 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 $\mathsf{lead}(w)$ (else, labeled "corrupt input" if $\mathsf{lead}(w)$ 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 $b_{v,1,j} \leftarrow b_{v,2,j}$ s.t. $j \neq \mathsf{lead}(v)$. The throughput would then drop down by 1/3.

IV. s2PAC$^{\text{LEAN}}$: SUPER FAST PIPELINING

s2PAC$^{\text{lean}}$ is described in Figure 10, in Theorem 5 we state its properties, then prove them. In addition to the DecCerts on *height*-1 blocks as in 2PAC$^{\text{lean}}$, dubbed *ordinary DecCerts*, s2PAC$^{\text{lean}}$ allows a new type of DecCerts:

- **A speed-DecCert for a *height*-2 ("pipelined") block** $\boldsymbol{qc_{v,1,\ell}} \leftarrow \boldsymbol{b_{v,2,\ell}} = \big[v, 2, \ell, qc_{v,1,\ell}, \mathit{txn}, \mathit{id}\big]$ proposed by the leader $\mathcal{P}_\ell$ of *view* v, is a $2f+1$ multi- (or threshold) signature on the message $\big(\text{SPEED\_VOTE}; v, 2, \ell, \mathit{id}\big)$.

A speed-DecCert equally enforces decision of $b_{v,2,i}$ and all its ancestors.

s2PAC$^{\text{lean}}$: with Optimistically 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: $qc_{v-1,2}$, it multicasts it.
- **elseif** it has a *view*-$(v-1)$ *height*-1 endorsed QC: $qc_{v-1,1,\ell} \leftarrow b_{v-1,2,\ell}$ (wrapped in a *height*-2 block of $\ell = \mathsf{lead}(v-1)$),
  it multicasts it, appended with a signed declaration $\{\text{no\_endorsed\_height-2\_QC}, v\}_i$
- **else** multicasts $\big(\text{report}, \{\text{no\_endorsed\_height-1\_QC}, v\}_i\big)$.

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

- **upon** obtaining a *view*-$(v-1)$ *height*-2 endorsed QC: $qc_{v-1,2,\ell}$
  creates a child *view*-v block: $qc_{v-1,2,\ell} \leftarrow \boldsymbol{b_{v,1,i}} = \big[v, 1, i, qc_{v-1,2,\ell}, \mathit{txn}, \mathit{id}\big]$, multicasts it and sets $1 \leftarrow$ *proposedHeight1*.
- **upon** receiving from $2f+1$ processes $\mathcal{P}_j$'s: $\{\text{no\_endorsed\_height-2\_QC}, v\}_j$
  chooses any *view*-$(v-1)$ *height*-1 endorsed QC: $qc_{v-1,1,\ell} \leftarrow b_{v-1,2,\ell}$ (wrapped in a *height*-2 block of $\mathcal{P}_\ell$)
  creates a child *view*-v block: $qc_{v-1,1,\ell} \leftarrow b_{v-1,2,\ell} \leftarrow \boldsymbol{b_{v,1,i}} = \big[v, 1, i, b_{v-1,2,\ell}, \mathit{txn}, \mathit{id}\big]$
  combines the signed declarations into $\{\text{no\_endorsed\_height-2\_QC}, v\}$ ▷ proof that no *view*-$(v-1)$-speed-DecCert can ever exist.
  multicasts $\Big(\langle qc_{v-1,1} \leftarrow b_{v-1,2} \leftarrow b_{v,1,i}\rangle_i, \{\text{no\_endorsed\_height-2\_QC}, v\}\Big)$ and set $1 \leftarrow$ *proposedHeight1*.
- **upon** receiving from $2f+1$ processes: $\big(\text{report}, \{\text{no\_endorsed\_height-1\_QC}, v\}_j\big)$
  chooses any *view*-$(v-1)$ *height*-2 QC, form a child block: $qc_{v-1,2} \leftarrow b_{v,1,i} = \big[v, 1, i, qc_{v-1,2}, \mathit{txn}, \mathit{id}\big]$
  combines the signed declarations into $\{\text{no\_endorsed\_height-1\_QC}, v\}$
  ▷ proof that no $v-1$ DecCerts can ever exist, thus $qc_{v-1,2}$ can safely be extended.
  multicasts $\Big(\langle qc_{v-1,2} \leftarrow b_{v,1,i}\rangle_i \ \{\text{no\_endorsed\_height-1\_QC}, v\}\Big)$ 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 $\boldsymbol{b_{v,1,j}} = \big[v, 1, j, \mathit{parent}, \mathit{txn}, \mathit{id}\big]$, i.e., of one of the following forms:

- either the raw signed block $\langle qc_{v-1,2,\ell} \leftarrow \boldsymbol{b_{v,1,j}}\rangle_j$ with $qc_{v-1,2,\ell}$ endorsed,
- or, of the form $\Big(\langle qc_{v-1,1,\ell} \leftarrow b_{v-1,2} \leftarrow \boldsymbol{b_{v,1,j}}\rangle_j, \{\text{no\_endorsed\_height-2\_QC}, v\}\Big)$ with $qc_{v-1,1,\ell}$ endorsed,
- or, of the form $\Big(\langle qc_{v-1,2} \leftarrow \boldsymbol{b_{v,1,j}}\rangle_j, \{\text{no\_endorsed\_height-1\_QC}, v\}\Big)$

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

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

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

QC *height*-2. $\forall w \in \{v-1, v\}$, **upon** receiving $2f+1$ votes for its view-w *height*-2 block $\boldsymbol{b_{w,2,i}}$: $\{w, 2, i, \mathit{id}\}_j$'s signed by distinct processes $\mathcal{P}_j$'s
combines the $2f+1$ signatures into a view-w *height*-2 QC: $\boldsymbol{qc_{w,2,i}}$. If $w = v$, it multicasts it.

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: $qc_{v,2,j} = \big\{w, 2, i, \mathit{id}\big\}$
it replies $\big\{\text{SPEED\_VOTE}, v, 2, j, \mathit{id}\big\}_i$ to $\mathcal{P}_j$. ▷ optimization: if furthermore $qc_{v,2,i}$ is endorsed, then multicast the SPEED_VOTE.

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

Coin QC and Advance view. **upon** receiving or forming a *view*-v coin-QC: qc_{coin}, then it multicasts it and enters view $v+1$.

Decision. **upon** receiving or forming an ordinary DecCert for any view number w, i.e., a 2-chain $\boldsymbol{qc_{w,1,j}} \leftarrow \boldsymbol{qc_{w,2,j}}$ s.t. $j = \mathsf{lead}(w)$ or a speed-DecCert: $\{\text{SPEED\_VOTE}, qc_{w,2,j}\}$, then
multicasts it and outputs it.

Figure 10: Differences with 2PAC$^{\text{lean}}$ are highlighted, they are due to the optimistically fast decision of pipelined blocks.

Theorem 5. s2PAC^{lean} *is a blockchain consensus. When used for a single-shot MVBA, then it has* 9.5δ *(worst-case) expected latency; and* 6δ *good-case latency (Def. 2).*

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

Both the proof of expected latency and of good case latency are identical to the ones for 2PAC$^{\text{lean}}$ (Sec. III-2 then Sec. A-A, and Sec. III-3), so we skip them.

1) Proof of 4δ-decision of pipelined blocks if fair scheduler: Consider a view v, by definition at time t the (to-be-elected) leader $\mathcal{P}_\ell = \mathsf{lead}(v)$ proposes its *height*-2 block $\boldsymbol{b_{v,2,\ell}}$. By the Claim below, all $2f+1$ honest processes cast a SPEED_VOTE for $\boldsymbol{b_{v,2,\ell}}$, thus by $t+3\delta$. In conclusion, a DecCert is formed (at least by $\mathcal{P}_\ell$) by $t+4\delta$. We now state then prove the Claim.

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

2) Proof of consistency of s2PAC^{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: $qc_{v',h'}$ (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 same proposer $\mathsf{lead}(v)$. By quorum intersection it cannot be of same height h, so we must have $h = 1$ and $h' = 2$. But by construction we must have $qc_{v,h=1} \leftarrow qc'_{v,h'=2}$ (otherwise processes would not have voted to form $qc'_{v,h'=2}$), a contradiction.

[*Proof of the Claim.* W.l.o.g. we can assume that $qc' = qc_{v',h}$ 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 $b_{v',1,j} \leftarrow b'_{v',h'=2,j}$, 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 $qc' = qc_{v',1,j}$ for a block $b'_{v',1,j}$, 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 qc':

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

ii) or, upon receiving a proposal $\langle qc_{v'-1,1} \leftarrow b_{v'-1,2} \leftarrow b_{v',1,j}\rangle_j$ appended with $\{\text{no\_endorsed\_height-2\_QC}, v'\}$ with $qc_{v'-1,1}$ endorsed. By minimality, $qc_{v'-1,1}$ 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 $qc_{v'-1,1}$, 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 $\langle qc_{v'-1,2} \leftarrow b'_{v',1,j}\rangle_j$ appended with $\{\text{no\_endorsed\_height-1\_QC}, v'\}$. By minimality, $qc_{v'-1,2}$ must be on the same chain as $b_{v,h}$, thus also $b'_{v',1,j}$, a contradiction.

V. 2PAC$^{\text{BIG}}$ AND s2PAC$^{\text{BIG}}$

The modifications to go from 2PAC$^{\text{lean}}$ to 2PAC$^{\text{BIG}}$, and from s2PAC$^{\text{lean}}$ to s2PAC$^{\text{BIG}}$, are identical. Those modifications are only that: 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: $b_{v,2,i}$ without waiting to receive a QC on its *height*-1 parent block: $b_{v,1,i}$. So the child $b_{v,2,i}$ does not refer anymore to its parent $b_{v,1,i}$ via a QC, but instead simply via a hash (the preimage having been diffused previously).

Since 2PAC$^{\text{BIG}}$/s2PAC$^{\text{BIG}}$ reduce by 2δ the latency per view compared to 2PAC$^{\text{lean}}$/s2PAC$^{\text{lean}}$, and since there are 1.5 views in expectation for a single-shot MVBA, it follows that 2PAC$^{\text{BIG}}$/s2PAC$^{\text{BIG}}$ 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$^{\text{lean}}$ and s2PAC$^{\text{lean}}$ (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). The proofs of the latencies (expected, good-case and pipelined) are also identical to the ones of 2PAC$^{\text{lean}}$ and s2PAC$^{\text{lean}}$, so we also skip them.

Theorem 6. 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. 2).*

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

a) sGradedDAG: the same kind of technique provides a fast track for GradedDag [24] (reminders are given in Sec. C-3). In GradedDAG, the two phases of vote per view are for only one block, and are called a GBC. Then, as in 2PAC$^{\text{BIG}}$ there is a leader election (called the first round of a CBC), and furthermore processes pipeline a block in their coin-share messages. We now sketch how to add a fast-track, yielding sGradedDAG. In its GBC proposal of a new view $v+1$, a process: (a) either it has an endorsed QC: $qc_{v,2}$ for a CBC block of the previous view v, then it casts a SPEED_VOTE for the corresponding block $b_{v,2}$, *and* proposes a child of it: $qc_{v,2} \leftarrow b'$. (b) Or it has none, then it proposes a child b' of the highest endorsed QC: $qc_{w,h}$ which it has. It appends to its proposal a quorum of $2f+1$ declarations, testifying not to have seen a more recent endorsed QC than $qc_{w,h}$. In particular, such declarations rule-out the existence of $2f+1$ SPEED_VOTEs on any *view*-$v' \geqslant w$ endorsed CBC block $b_{v',2}$, hence of a conflicting view-v' speed-DecCert for $b_{v',2}$.

VI. COMPUTATIONAL EVALUATION

In the spreadsheet 2PAC_timings.xls (https://perso.telecom-paristech.fr/rambaud/articles) we measure the computation time per process per view, i.e., the denominator of the throughput formula Eq. (1), for both 2PAC$^{\text{lean}}$ and sMVBA (as if chained with pipelining), for some values of n from

2PAC$^{\text{BIG}}$ and s2PAC$^{\text{BIG}}$

```
Vote height-1. ...  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 $2f+1$ proposers $^{(*)}$
    create a child height-2 block: $b_{v,1,i} \leftarrow b_{v,2,i}$ and multicast it, then set $1 \leftarrow proposedHeight2$.
QC height-1 $\forall j$: upon receiving $2f+1$ votes for $\mathcal{P}_j$'s height-1 block: $\{v, 1, j, id\}_k$'s
    combines the $2f+1$ signatures into a view-$v$ height-1 QC: $qc_{v,1,j} = \{v, 1, j, id\}$. $^{(\dagger)}$
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 $\boldsymbol{qc_{v,1,j}}$ 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 $2f+1$ votes for $\mathcal{P}_j$'s height-2 block: $\{w, 2, j, id\}_k$'s
    combines the $2f+1$ signatures into a view-$w$ height-2 QC: $qc_{w,2,j}$ $^{(\dagger)}$
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: $qc_{v,2,j} = \{v, 2, j, id\}$
      it multicasts $\{\text{SPEED\_VOTE}, v, 2, j, id\}_i$.
```

Figure 11: Only changes from 2PAC$^{\text{lean}}$ (for 2PAC$^{\text{BIG}}$) and from s2PAC$^{\text{lean}}$ (for s2PAC$^{\text{BIG}}$) are displayed (in black, vs gray if unchanged). $(*)$: thus it waits no longer than δ before *height*-2 proposing. This could be sped-up by: **upon** receiving $f+1$ proposals (at the cost of potentially including less transactions in $b_{v,2,i}$). $(\dagger)$: it could possibly multicast the QCs for further speed, 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.

22 to 121. We use the benchmarked computation times of https://zka.lc/ for elementary operations on the curve BLS-381 implemented by gnark-crypto [15], running on an EC2 m5.2x large instance. From these timings, we deduce the time taken by the three main basic operations in both sMVBA and 2PAC, which are: verification of $2f+1$ signatures, combination into a BLS-based multisignature [61], and verification of it. We obtain that the total computation time of 2PAC$^{\text{lean}}$ is smaller by -20% to -27%, depending if a parameter called "Optimistic" (below) is set to 0 or 1. Since the numerator of Eq. (1) is $+50\%$ larger in 2PAC$^{\text{lean}}$ than in sMVBA, its throughput is thus larger by $+80\%$ to $+104\%$. When Optimistic is set to 1, the verification of $2f+1$ signatures is done in batch (this is the methodology in [72]). Namely: by first combining them, then verifying the multisignature. When Optimistic is set to 0, the verification is done one by one (this is the methodology in [46]). Since naively verifying $2f+1$ BLS signatures would be prohibitive, we instead measure the verification time of a nice speedup suggested in [37]. Namely, we consider that a BLS signer also provides a Chaum-Pedersen proof of equality of discrete logarithms between its signature: sk.$\mathrm{H}(m)$ and its public key: sk.G_2. Then, BLS verification amounts to verification of this proof, which takes less than twice the time to verify a Schnorr signature. The reason why the computational gain of 2PAC$^{\text{lean}}$ is comparatively higher when Optimistic is set to 1, is that when so, then the optimization to verify DocGs only once saves a cost comparatively closer to optimistically combining and batch-verifying votes.

VII. Variants, Optimizations & Comments

1) Amortization over long values, with only $+\delta$ overhead: Recall that the extension protocol Dumbo-MVBA$\star$ [55] 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 5 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 $:= (\sigma, c)$. 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 [55]. We make the observation that in 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.

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: $(\sigma, \mathrm{Hash}(c))$.

2) Adaptive corruptions: Adaptive security can be achieved in all 2PAC protocols, simply by using adaptively-secure primitives. For instance, [6, 25, 7] provide unique non-interactive threshold signatures, which can thus be hashed into adaptive common coins ([60]). We refer to the nice discussion on adaptive fairness in [46, Appendix E].

Acknowledgements

We thank Julian Loss for notifying of 2-chain_VABA, Zhuolun Xiang for exchanges on 2-chain_VABA, Zhenliang Lu for exchanges on sMVBA and Nathan Rouillé for useful comments on the paper.

References

[1] I. Abraham, T. H. Chan, D. Dolev, K. Nayak, R. Pass, L. Ren, and E. Shi. "Communication complexity of byzantine agreement, revisited". In: *Distributed Comput.* (2023).

[2] I. Abraham, D. Malkhi, and A. Spiegelman. "Asymptotically Optimal Validated Asynchronous Byzantine Agreement". In: *PODC*. 2019.

[3] I. Amores-Sesar and C. Cachin. "We will DAG you". In: *CoRR* abs/2311.03092 (2023).

[4] Aptos. *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] B. Arun, Z. Li, F. Suri-Payer, S. Das, and A. Spiegelman. *Shoal++: High Throughput DAG BFT Can Be Fast!* 2024.

[6] R. Bacho and J. Loss. "On the Adaptive Security of the Threshold BLS Signature Scheme". In: *CCS*. 2022.

[7] R. Bacho, J. Loss, G. Stern, and B. Wagner. *HARTS: High-Threshold, Adaptively Secure, and Robust Threshold Schnorr Signatures*. ePrint 2024/280. 2024.

[8] M. Ben-Or, R. Canetti, and O. Goldreich. "Asynchronous Secure Computation". In: *STOC*. 1993.

[9] M. Ben-Or and R. El-Yaniv. "Resilient-optimal interactive consistency in constant time". In: *Distributed Comput.* (2003).

[10] M. Ben-Or, B. Kelmer, and T. Rabin. "Asynchronous Secure Computations with Optimal Resilience (Extended Abstract)". In: *PODC*. 1994.

[11] E. Blum, J. Katz, J. Loss, K. Nayak, and S. Ochsenreither. "Abraxas: Throughput-Efficient Hybrid Asynchronous Consensus". In: *CCS*. 2023.

[12] A. Boldyreva. "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. 2003.

[13] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps". In: *EUROCRYPT*. 2003.

[14] D. Boneh, K. Lewi, H. W. Montgomery, and A. Raghunathan. "Key Homomorphic PRFs and Their Applications". In: *CRYPTO*. 2013.

[15] G. Botrel, T. Piellard, Y. E. Housni, A. Tabaie, and I. Kubjas. *ConsenSys/gnark-crypto: v0.6.1*. 2022.

[16] V. Buterin and V. Griffith. "Casper the Friendly Finality Gadget". In: *arxiv 1710.09437* (2017).

[17] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup. "Secure and Efficient Asynchronous Broadcast Protocols". In: *CRYPTO*. 2001.

[18] C. Cachin, K. Kursawe, and V. Shoup. "Random Oracles in Constantinople: Practical Asynchronous Byzantine Agreement Using Cryptography". In: *J. Cryptol.* (2005).

[19] M. Castro and B. Liskov. "Practical Byzantine Fault Tolerance". In: *OSDI*. 1999.

[20] B. Y. Chan and R. Pass. "Simplex Consensus: A Simple and Fast Consensus Protocol". In: *TCC*. 2023.

[21] H. Cheng, Y. Lu, Z. Lu, Q. Tang, Y. Zhang, and Z. Zhang. *JUMBO: Fully Asynchronous BFT Consensus Made Truly Scalable*. 2024.

[22] X. Dai, G. Wang, J. Xiao, Z. Guo, R. Hao, X. Xie, and H. Jin. "LightDAG: A Low-latency DAG-based BFT Consensus through Lightweight Broadcast". In: *IPDPS*. 2024.

[23] X. Dai, B. Zhang, H. Jin, and L. Ren. *ParBFT: Faster Asynchronous BFT Consensus with a Parallel Optimistic Path*. ePrint 2023/679. 2023.

[24] X. Dai, Z. Zhang, J. Xiao, J. Yue, X. Xie, and H. Jin. "GradedDAG: An Asynchronous DAG-based BFT Consensus with Lower Latency". In: *SRDS*. 2023.

[25] S. Das and L. Ren. *Adaptively Secure BLS Threshold Signatures from DDH and co-CDH*. ePrint 2023/1553. 2023.

[26] S. Das, Z. Xiang, and L. Ren. "Asynchronous Data Dissemination and its Applications". In: *CCS*. 2021.

[27] Diem. *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. 2021.

[28] S. Duan, M. K. Reiter, and H. Zhang. "BEAT: Asynchronous BFT Made Practical". In: *CCS*. 2018.

[29] S. Duan, X. Wang, and H. Zhang. "FIN: Practical Signature-Free Asynchronous Common Subset in Constant Time". In: *CCS*. 2023.

[30] S. Duan, H. Zhang, X. Sui, B. Huang, C. Mu, G. Di, and X. Wang. "Dashing and Star: Byzantine Fault Tolerance with Weak Certificates". In: 2024.

[31] C. Dwork, N. A. Lynch, and L. J. Stockmeyer. "Consensus in the presence of partial synchrony". In: *J. ACM* (1988).

[32] M. F. Esgin, R. Steinfeld, D. Liu, and S. Ruj. "Efficient Hybrid Exact/Relaxed Lattice Proofs and Applications to Rounding and VRFs". In: *Crypto*. 2023.

[33] H. Feng, Z. Lu, T. Mai, and Q. Tang. *Making Hash-based MVBA Great Again*. ePrint 2024/479. 2024.

[34] M. Fitzi and M. Hirt. "Optimally Efficient Multi-Valued Byzantine Agreement". In: *Podc*. 2006.

[35] N. Fleischhacker, G. Herold, M. Simkin, and Z. Zhang. "Chipmunk: Better Synchronized Multi-Signatures from Lattices". In: *CCS*. 2023.

[36] F. Gai, J. Niu, I. Beschastnikh, C. Feng, and S. Wang. "Scaling Blockchain Consensus via a Robust Shared Mempool". In: *ICDE*. 2023.

[37] D. Galindo and J. Liu. "Robust Subgroup Multi-signatures for Consensus". In: *CT-RSA*. 2022.

[38] Y. Gao, Y. Lu, Z. Lu, Q. Tang, J. Xu, and Z. Zhang. "Dumbo-NG: Fast Asynchronous BFT Consensus with Throughput-Oblivious Latency". In: *CCS*. 2022.

[39] J. Garay, A. Kiayias, and N. Leonardos. "The Bitcoin Backbone Protocol: Analysis and Applications". In: *EUROCRYPT*. 2015.

[40] R. Gelashvili, L. Kokoris-Kogias, A. Sonnino, A. Spiegelman, and Z. Xiang. "Jolteon and Ditto: Network-Adaptive Efficient Consensus with Asynchronous Fallback". In: *FC*. we refer to the 18 June 2021 version on arxiv. 2022.

[41] R. Gelashvili, L. Kokoris-Kogias, A. Sonnino, A. Spiegelman, and Z. Xiang. "Jolteon and Ditto: Network-Adaptive Efficient Consensus with Asynchronous Fallback". In: version 2024-04-30, fixing the FC'22 version and the 2023-12 version. 2024.

[42] R. Gelashvili, L. Kokoris-Kogias, A. Spiegelman, and Z. Xiang. “Brief Announcement: Be Prepared When Network Goes Bad: An Asynchronous View-Change Protocol”. In: *PODC*. 2021.

[43] G. Golan-Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. K. Reiter, D. Seredinschi, O. Tamir, and A. Tomescu. “SBFT: A Scalable and Decentralized Trust Infrastructure”. In: *DSN*. 2019.

[44] G. Goren, Y. Moses, and A. Spiegelman. “Probabilistic Indistinguishability and the Quality of Validity in Byzantine Agreement”. In: *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*. 2023.

[45] V. Gramoli, Z. Lu, Q. Tang, and P. Zarbafian. *Optimal Asynchronous Byzantine Consensus with Fair Separability*. ePrint. 2024.

[46] B. Guo, Y. Lu, Z. Lu, Q. Tang, J. Xu, and Z. Zhang. “Speeding Dumbo: Pushing Asynchronous BFT Closer to Practice”. In: *NDSS*. 2022.

[47] M. M. Jalalzai, J. Niu, C. Feng, and F. Gai. “Fast-HotStuff: A Fast and Resilient HotStuff Protocol”. In: *IEEE Transactions on Dependable and Secure Computing* (2023).

[48] A. Kavousi, Z. Wang, and P. Jovanovic. “SoK: Public Randomness”. In: *EuroS&P*. 2024.

[49] I. Keidar, O. Naor, O. Poupko, and E. Shapiro. “Cordial Miners: Fast and Efficient Consensus for Every Eventuality”. In: *DISC*. 2023.

[50] I. Khaburzaniya, K. Chalkias, K. Lewi, and H. Malvai. “Aggregating and Thresholdizing Hash-Based Signatures Using STARKs”. In: *Asia CCS*. 2022.

[51] A. Lewis-Pye and I. Abraham. “Fever: OptiFmal Responsive View Synchronisation”. In: *Opodis*. 2023.

[52] C. Li, P. Li, D. Zhou, Z. Yang, M. Wu, G. Yang, W. Xu, F. Long, and A. C. Yao. “A Decentralized Blockchain with High Throughput and Fast Confirmation”. In: *USENIX ATC*. 2020.

[53] C. Liu, S. Duan, and H. Zhang. “EPIC: Efficient Asynchronous BFT with Adaptive Security”. In: *DSN*. 2020.

[54] Y. Lu, Z. Lu, and Q. Tang. “Bolt-Dumbo Transformer: Asynchronous Consensus As Fast As the Pipelined BFT”. In: *CCS*. 2022.

[55] Y. Lu, Z. Lu, Q. Tang, and G. Wang. “Dumbo-MVBA: Optimal Multi-Valued Validated Asynchronous Byzantine Agreement, Revisited”. In: *PODC*. 2020.

[56] D. Malkhi and K. Nayak. *Extended Abstract: HotStuff-2: Optimal Two-Phase Responsive BFT*. ePrint 2023/397. 2023.

[57] D. Malkhi, C. Stathakopoulou, and M. Yin. “BBCA-CHAIN: One-Message, Low Latency BFT Consensus on a DAG”. In: *FC*. 2024.

[58] E. V. Mangipudi and A. P. Kate. “D-KODE: Distributed Mechanism to Manage a Billion Discrete-Log Keys”. In: *FC*. 2022.

[59] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song. “The Honey Badger of BFT Protocols”. In: *CCS*. 2016.

[60] R. Pass and E. Shi. “Thunderella”. In: *EUROCRYPT*. 2018.

[61] T. Ristenpart and S. Yilek. “The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks”. In: *EUROCRYPT*. 2007.

[62] N. Shrestha, R. Shrothrium, A. Kate, and K. Nayak. *Sailfish: Towards Improving Latency of DAG-based BFT*. ePrint 2024/472. 2024.

[63] A. Sonnino. *Implementation of Jolteon*. https://github.com/asonnino/hotstuff/. 2021.

[64] A. Spiegelman, B. Aurn, R. Gelashvili, and Z. Li. “Shoal: Improving DAG-BFT Latency And Robustness”. In: *FC*. 2023.

[65] A. Spiegelman, N. Giridharan, A. Sonnino, and L. Kokoris-Kogias. “Bullshark: DAG BFT Protocols Made Practical”. In: *CCS*. 2022.

[66] C. Stathakopoulou, M. Pavlovic, and M. Vukolic. “State machine replication scalability made simple”. In: *EuroSys’22*. 2022.

[67] E. N. Tas, D. Zindros, L. Yang, and D. Tse. “Light Clients for Lazy Blockchains”. In: *FC*. 2024.

[68] A. Tomescu, R. Chen, Y. Z. and Ittai Abraham, B. Pinkas, G. Golan-Gueta, and S. Devadas. “Towards Scalable Threshold Cryptosystems”. In: *IEEE S&P*. 2020.

[69] Z. Xiang. *Implementation of the fallback of Ditto*. https://github.com/danielxiangzl/Ditto/blob/main/consensus/src/fallback.rs Retrieved on August 17 2023. 2021.

[70] M. Yin, D. Malkhi, M. K. Reiter, G. Golan-Gueta, and I. Abraham. “HotStuff: BFT Consensus with Linearity and Responsiveness”. In: *PODC*. we refer to the arxiv v6 long version. 2019.

[71] M. Yin, D. Malkhi, M. K. Reiter, G. Golan-Gueta, and I. Abraham. ““Two-phase HotStuff” in Sections 4.4 and 6 of HotStuff: BFT Consensus with Linearity and Responsiveness”. In: *PODC*. 2019.

[72] T. Yurek, Z. Xiang, Y. Xia, and A. Miller. “Long Live The Honey Badger: Robust Asynchronous DPSS and its Applications”. In: *USENIX security*. 2023.

[73] H. Zhang, S. Duan, B. Zhao, and L. Zhu. “WaterBear: Practical Asynchronous BFT Matching Security Guarantees of Partially Synchronous BFT”. In: *USENIX Security*. 2023.

[74] Y. Zhou, Z. Zhang, H. Zhang, S. Duan, B. Hu, L. Wang, and J. Liu. *Dory: Asynchronous BFT with Reduced Communication and Improved Efficiency*. ePrint 2022/1709. 2022.

Appendix A
Deferred details of proofs

A. *Proof of expected latency of* $\mathsf{2PAC}^{lean}$ *(and of sMVBA)*

We formalize the proof given in Sec. III-2 for the upper-bound 9.5δ on the expected latency of $\mathsf{2PAC}^{\mathsf{lean}}$. For each view number v, we call *lucky (roulette) roll*, and denote $\mathrm{LR}(v)$, the following event:

$\mathrm{LR}(v)$:= $\Big\{$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 $2f+1$ distinct proposers) which it received beforehand. Then one of them will be endorsed, i.e., $\mathsf{lead}(v)$ is among these $2f+1$ proposers.$\Big\}$

Let us informally explain why our terminology is justified, i.e., why electing $\mathsf{lead}(v)$ 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 7 below). In particular it is independent of $\mathcal{Q}(v)$. If the wheel stops on the index of one of the $2f+1$ 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 8 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:

$$(2)\quad v_D := \min\Big(v \geqslant 1 \text{ s.t. } \big[\neg\mathrm{LR}(v'), \forall v' < v\big] \wedge \mathrm{LR}(v)\ \Big)\ .$$

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

$$(3)\quad E\Big[L' \mid \big\{\neg\mathrm{LR}(v'), \forall v' \leqslant v\big\}\Big] = 6\delta + 7\delta.(v-1) + E[L']\ .$$

Since by Lemma 7, each event $\neg\mathrm{LR}(v)$ has probability $< 1/3$, we obtain the following inductive formula on $E' = E[L']$ the expectation of L':

$$(4)\qquad E' = 2/3.6\delta + 1/3(7\delta + E')$$

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

We now estimate an upper-bound on the latency of sMVBA [46] by the same method, then state and prove Lemma 7 and Proposition 8. In sMVBA, each view $v \geqslant 1$ is as follows: if $\mathrm{LR}(v)$ 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 $\{\neg\mathrm{LR}(v), v \in [1, \ldots, \infty[\}$ 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 7. *The events $\{\neg\mathrm{LR}(v), v \in [1, \ldots, \infty[\}$, are independent. Each of them has probability $< 1/3$.*

Proof: The times $t_{\text{coin}}(v)$ at which the $\mathcal{P}(v)$'s send their coin shares are strictly ordered by increasing v. Thus all events $\neg\mathrm{LR}(v')$ for $v' < v$, are independent from $\neg\mathrm{LR}(v)$: 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 8. *For each view number v, consider the time $t(v) := 6\delta + (v-1)7\delta$. Suppose that no process decided by $t(v)$. Then, by $t(v)$: (i) $\neg\mathrm{LR}(v')$ 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 $2f+1$ *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 $t(1) = 6\delta$, 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 $2f+1$ *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 $6\delta + (v-2)7\delta + 6\delta$". ■

1) (One more time) intuition of the proof of consistency: Let us explain again the intuition of the proof of consistency of 2PAC$^{\text{lean}}$, formalized in Sec. III-1. Let us consider a view: $v-1$ and assume the induction statement that no view-$(v-1)$ QC conflicts with any prior view-$(< v-1)$ DecCert. Let us prove that the statement then holds for *view-v* QCs: it will follow, by induction, that the statement holds for all views.

To this end, we consider a *view-v* QC: $\boldsymbol{qc_{v,1}}$. It is descendent of a view-$(v-1)$ QC: $qc_{v-1,1}$. It always holds that (i): no view-$(<(v-1))$ DecCert conflicting with $\boldsymbol{qc_{v,1}}$ can exist (since otherwise it would conflict with $qc_{v-1,1,\mathsf{lead}(v-1)}$, contradicting the assumption). Let us now prove (ii): no view-$(v-1)$ DecCert conflicting with $\boldsymbol{qc_{v,1}}$ can exist, from which the desired statement for v will follow.

Either $qc_{v-1,1}$ is endorsed. Then (ii) holds (since otherwise a conflicting view-$(v-1)$ DecCert would conflict with the endorsed $qc_{v-1,1}$, which is impossible by quorum intersection). Hence, the desired statement holds. *Or $qc_{v-1,1}$ is not endorsed.* Thus at least one honest process (actually $f+1$ of them) checked existence of a threshold signature $\{\text{no\_endorsed\_height-1\_QC}, v\}$, i.e, a DocG. The DocG proves that no view-$(v-1)$ DecCert could be made, which in particular implies (ii)

Appendix B
Further optimizations

1) Leader election without forwarding the coin-QC: Existing post-quantum common coins, recalled in Sec. I, 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 ([46, Alg 5 line 9]), it was introduced in a related context by [27] under the name "Bracha timeout".

The sMVBA variant. It consists in adding the following trigger: a process multicasts its coin share *also* upon receiving $f+1$ coin shares.

First, we show that the (worst-case) expected latency 9.5δ of 2PAC$^{\text{lean}}$ is unaffected. It is enough to show that Proposition 8 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 $2f + 1$ *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 $2f+1$ coin shares before 3δ. Since $f+1$ 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$^{\text{lean}}$ 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 $2f + 1$ *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 $2f + 1$ coin shares before 3δ, thus $6\delta_{\text{fast}} < 3\delta$, a contradiction.

2) Check only one DocG instead of $2f+1$*:* In both 2PAC$^{\text{lean}}$ and 2PAC$^{\text{BIG}}$, the threshold signature $\{\text{no\_endorsed\_height-1\_QC}, v\}$ 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 DocGs from them. Similarly, in s2PAC$^{\text{lean}}$ and s2PAC$^{\text{BIG}}$, the threshold signature $\{\text{no\_endorsed\_height-2\_QC}, v\}$ needs to be checked only once in a view v. In Sec. B-3 we describe a further optimization enabling proposers to optimistically not form nor send a DocG, without losing latency in the worst-case.

3) Optimistically Skipping DocG*:* The optimization, which applies to both 2PAC$^{\text{lean}}$ and 2PAC$^{\text{BIG}}$ (mutatis mutandis for s2PAC$^{\text{lean}}$), 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 $\langle qc_{v-1,2} \leftarrow b_{v,1,i}\rangle_i$, without the threshold signature $\{\text{no\_endorsed\_height-1\_QC}, v\}$. 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 lead$(v-1)$ (notice that this is an adaptation of the unlocking mechanism of Tendermint/two-phase Hotstuff [70, 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 $qc_{v-1,2} \leftarrow b_{v-1,2}$, 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 $\{\text{no\_endorsed\_LockCert}, v\}$, 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 $2f+1$ *view*-v *height*-1 votes on a *view*-$(v-1)$ non-endorsed QC. Then, it must be the case that no $f+1$ honest processes cast *view*-$(v-1)$ *height*-2 votes for a *height*-2 block of lead$(v-1)$. Thus, no *view*-$(v-1)$ DecCert conflicting with $qc_{v-1,2} \leftarrow b_{v,1,i}$ 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 $qc_{v-1,2} \leftarrow b_{v,1,i}$ does not conflict with any lower view DecCert.

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 $v+1$.

5) Fast catch-up: A process advances to a higher *view* v' upon receiving any *view*-v' QC (such a QC implies existence of $2f+1$ *view*-$(v'-1)$ *height*-2 QCs, and of a coin-QC, so the proofs of latencies apply unchanged).

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 (Sec. B-5), a process can stop re-sending the latter as soon as it obtains a QC of the current view.

APPENDIX C
FURTHER COMMENTS AND RELATED WORKS

1) More on MVBA: MVBA is the costliest building block in most state-of-the-art asynchronous blockchain consensus [46, 54, 38, 23, 21, 11, 45]. In addition, MVBA is the main building block in most implementations [17, 46, 72, 29] 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 [72, 28, 53]. MVBA and/or ACS are also building blocks of proactive resharing [72], PVSS-based distributed randomness generation [48] and threshold signing [7]. A classical requirement in MVBA (also called VABA [2, 40]) is called *(external) validity* [17, 46]. 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 *Mvba_valid*$(x, \pi) =$ true. All our protocols can straightforwardly be made externally valid, simply by having processes ignore messages containing non-valid values, i.e., on which *Mvba_valid* returns false. For the use-case of blockchain consensus, having quality removes the need to impose external validity. Instead, it is advocated in [67] 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 [34] for the conflicting validity specification known as *strong unanimity* ([31]). It would be straightforward to enforce strong unanimity in 2PAC, provided a preliminary round as in [1, §6].

2) report *after starting a view:* Our convention of a view beginning *before* report is in line with [19, 70, 51]. 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, $v{+}1, \dots$) in [19]). 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").

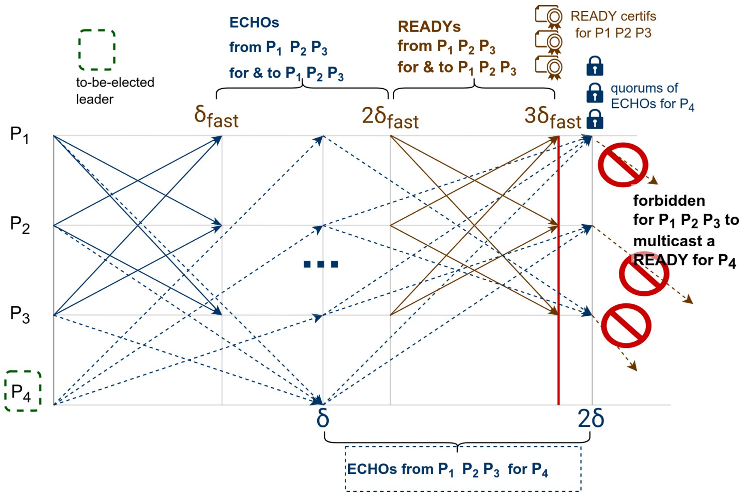

to-be-elected leader
ECHOs from P1 P2 P3 for & to P1 P2 P3
READYs from P1 P2 P3 for & to P1 P2 P3
READY certifs for P1 P2 P3
quorums of ECHOs for P4
δfast
2δfast
3δfast
P1
P2
P3
P4
forbidden for P1 P2 P3 to multicast a READY for P4
δ
2δ
ECHOs from P1 P2 P3 for P4

Figure 12: A view of GradedDAG with $\delta_{\text{fast}} < 2/3\delta$

3) GradedDAG under a half-fair scheduler: We describe a fault-free scenario where the scheduler is half-fair (Def. 2) 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 6, in 2PAC$^{\text{BIG}}$ 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 $t = 0$, all processes act as senders of a reliable broadcast (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 $2f{+}1 = 3$-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 $\mathcal{P}_4$.]

4) Why a coin with threshold only $f{+}1$ prevents 6δ good-case latency: We describe the counterexample for 2PAC as if it had been instantiated with a $(f{+}1)$-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 f 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.