

Optimistic History-Independent Sleepy Atomic Broadcast

Kwok-Yan Lam^{1*}, Zhenliang Lu^{2*✉}, Qiang Tang^{3*}, and Xinrui Zhang^{3*}

¹ Nanyang Technological University, Singapore
kwokyan.lam@ntu.edu.sg

² City University of Hong Kong, Hong Kong, China
zhenliang.lu@cityu.edu.hk

³ The University of Sydney, Sydney, Australia
{qiang.tang,xinrui.zhang1}@sydney.edu.au

Abstract. The latency of sleepy consensus protocols has been improved significantly, with recent designs achieving constant-round confirmation. However, existing protocols either rely on unrealistic additional assumptions about access to historical messages, where messages sent in round t may depend on messages received in rounds $[t - k, t - 1]$ for some $k > 1$, or do not consider optimizing performance under optimistic conditions. In this work, we study history-independent sleepy atomic broadcast protocols while simultaneously considering optimistic conditions, where messages sent in round t depend only on messages received in round $t - 1$. We first design a sleepy atomic broadcast protocol that achieves a confirmation latency of 2Δ (Δ is the upper bound on network delay) and an optimal latency of Δ under up to $1/5$ corruption. Second, we present a super-optimistic atomic broadcast protocol, achieving a best-case latency of 2Δ under fault-free executions. Building on this insight, we propose an optimistic sleepy atomic broadcast protocol that relaxes the super-optimistic condition to tolerate sub-optimal faults. As a building block, we introduce an optimistic graded agreement protocol, **OptimisticGA**, which enables a dynamic optimistic quorum with four decision grades. Built on **OptimisticGA**, our optimistic atomic broadcast protocol achieves a good-case latency of 2Δ when the leader is honest, while tolerating up to $1/6$ corruptions rather than requiring a fault-free setting. When corruption is more than $1/6$, it seamlessly adapts to $1/3$ resilience with a good-case latency of 3Δ . All protocols require no historical messages and tolerate an adversary that grows proportionally with honest nodes, with a minimal delay of Δ .

Keywords: History-independent · Atomic broadcast · Optimistic · Sleepy.

1 Introduction

Byzantine Fault Tolerant (BFT) protocols have garnered renewed substantial attention, especially with the advent of cryptocurrencies. Traditional BFT protocols [29, 38, 5, 10, 4, 6] generally assume a static set of participants, where all

* Authors are listed alphabetically. Xinrui Zhang led the effort.

honest nodes are expected to remain continuously online. Any deviation from this behavior is treated as a fault. However, this assumption becomes increasingly unrealistic in large-scale systems, where nodes may frequently go offline due to network instability, resource constraints, or voluntary actions. In contrast, longest-chain protocols exemplified by Bitcoin’s Proof-of-Work (PoW) [31] tolerate the intermittent availability of honest nodes without classifying them as faulty. However, their reliance on energy-intensive PoW significantly limits scalability and efficiency. To model the dynamic participation in modern consensus protocols and address the energy consumption limitation, Pass and Shi introduced the *Sleepy* model [32], which relaxes the traditional BFT assumption by allowing honest nodes to participate intermittently. In this model, only an *unknown* subset n_t of the total N participants is active (i.e., online) at any given time t , where $0 < n_t \leq N$. Protocol safety is maintained as long as a sufficient fraction of the n_t online nodes behave honestly during each execution step. Alongside the model, they also proposed the **PS-Sleepy** protocol [32], which does not rely on the PoW mechanism but adopts the same longest-chain approach used in protocols like Bitcoin. On the downside, those longest-chain-style protocols suffer from long confirmation latencies tied to a security parameter, resulting in a trade-off between latency and probabilistic safety guarantees.

Current major efforts [30, 26, 27, 7, 23] have focused on reducing the confirmation latency of consensus in the sleepy model. Starting with [30], which was the first to achieve constant-round latency in the sleepy setting with a confirmation delay of 16Δ , where Δ is the upper bound on network delay. [27] later improved the design and reduced the latency to 4Δ . Protocols proposed in [26] further reduced the latency to 3Δ and 2Δ , at the cost of a lower corruption threshold. Only very recently, [14] designed a Byzantine agreement that matches this upper bound when all participants have the same input, achieving the optimal good-case latency of Δ .

Limitations of existing sleepy protocols: history-independent⁴ & latency. However, existing protocols suffer from two major limitations. First, they either assume only a *non-growing* adversary (to be explained below) to achieve optimal fault tolerance, or tolerate only a limited growing adversary with improved latency at the cost of reducing the effective fault tolerance. In both cases, these protocols may still be influenced by later-awakened nodes. Moreover, they could require impractical assumptions, such as allowing a node to receive all past messages even if it awakens for the first time much later in the execution. Second, these protocols maintain the same latency even when the actual corruption level is much smaller than the maximum tolerated by the protocol, as in 1/3-MMR [26], despite significant efforts to reduce latency.

Limitation-1: non-growing adversary (history dependent). Early sleepy protocols such as [32, 30, 19] tolerate only a constant (non-growing) adversary. Although

⁴ If the messages sent at time t depend only on messages received during the most recent interval $[t - \Delta, t]$, then any adversarial behavior simulated before this interval becomes irrelevant to the current execution and can therefore be discarded. We refer to such a protocol as *history-independent*.

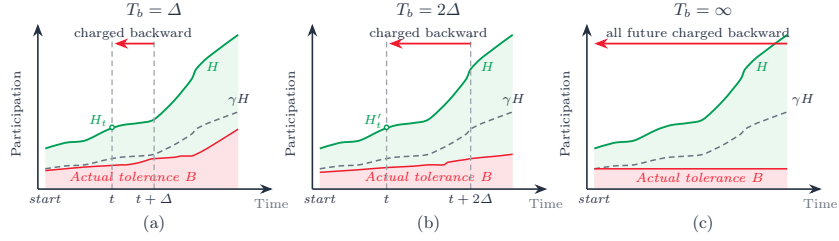


Fig. 1. T_b measures the number of rounds a corrupt party is counted as awake before it wakes up. An example of how varying T_b affects a protocol's actual Byzantine budget ($\gamma = 1/2$). When $T_b = \Delta$, the honest population H_t at time t bounds the actual corruption at time $t + \Delta$. Let $B_{t+\Delta}$ denote the corruption upper bound at time $t + \Delta$, and let γ denote the fault tolerance rate. Then $|B_{t+\Delta}| = \gamma|H_t|$. In the second scenario, when $T_b = 2\Delta$, H'_t bounds the actual corruption $B'_{t+2\Delta}$, so similarly, $|B'_{t+2\Delta}| = \gamma|H'_t|$. Suppose that $H_t = H'_t$ for both scenarios, as the adversary grows monotonically, $|B_{t+2\Delta}| \geq |B_{t+\Delta}| = |B'_{t+2\Delta}| = \gamma|H_t|$, we have $|B_{t+2\Delta}| \geq |B'_{t+2\Delta}|$. Furthermore, if $T_b = \infty$, the actual tolerance of any time is bounded by the starting population of execution, leading to a constant rather than growing adversary. Otherwise, newly corrupted nodes could be simulated as appearing in the previous phase, thereby violating the maximum fault tolerance bound of that phase. As a result, a smaller T_b permits a larger actual Byzantine participation at any given time, corresponding to a more aggressively growing adversary, closer to the ideal of instantaneous adversarial growth, i.e., $T_b = 0$.

they allow up to $1/2$ corruption, the effective corruption threshold is determined by the *minimum* participation observed during execution. Consequently, safety depends on maintaining a sufficiently large honest set at all times, which limits scalability as participation grows. This limitation arises from costless *backward simulation*. In longest-chain-style sleepy protocols, adversarial nodes that join later can fork from an earlier point in time and simulate continuous participation, potentially misleading new nodes into accepting a forged history.

We denote T_b as the maximum time window over which the adversary can perform such a simulation. Such longest-chain-style protocols give the adversary a backward simulation window throughout the entire execution, i.e., $T_b = \infty$. A protocol with $T_b = \infty$ corresponds to a non-growing adversary model, as it effectively assumes that corrupted nodes can simulate participation arbitrarily far into the past, thus all later corruptions can contribute to any previous decisions and need to be counted into the earlier corruption budgets. Such protocols rely on accumulated historical messages to make a decision for later rounds, making them inherently history-dependent. Moreover, as shown in Figure 1.(c), as time progresses, the actual corruption rate decreases in order to preserve protocol safety. Otherwise, if newly awakened corrupted nodes appear, they can collude to forge the decisions since the initial time, thereby violating safety.

Limitation-2: Limited growing adversary (partially history dependent). To reduce the history dependence window from $T_b = \infty$, several approaches [7, 23, 26] have been proposed, achieving $T_b = k\Delta$ ($k \geq 2$). That is, if a node becomes corrupted (or awakened) at time t , it is retroactively simulated as an online malicious node since time $t - k\Delta$. Such protocols still rely on multiple rounds of historical mes-

sages to make decisions, and therefore are not fully history-independent. Besides, this assumption still incurs a hidden cost, as shown in Figure 1(b): the protocol achieves a lower effective fault tolerance in each round than the claimed bound when the number of awake nodes n_t grows. This limitation also motivates the need for a stronger adversarial model that tolerates more natural and abrupt adversarial growth while still ensuring safety and progress with low latency. As shown in Table 1, the most efficient protocol so far, 1/4-MMR, still requires a two-round backward-simulation window, which remains insufficient under realistic system requirements. Moreover, although these efforts reduce the latency to a constant number of rounds, the latency is still relatively high compared with Byzantine consensus protocols in the static setting. Thus, the remaining challenge is not merely to reduce confirmation latency, but to do so while eliminating this conservative backward-simulation window, so that the adversary’s allowed growth can more closely track the current participation level.

A natural approach to reducing the backward-simulation window to Δ is to eliminate the protocol’s dependence on older history, since partial history independence remains insufficient. Eliminating history dependence also resolves the message buffering problem. In the sleepy model, newly awake nodes may need to recover past information to catch up with the latest chain state and behave correctly. Existing protocols [30, 15, 14, 7] address this issue by assuming message buffering before a node wakes up, i.e., a node activated at time t is assumed to automatically receive all messages sent before t . However, such an assumption is difficult to guarantee in practice. Even relying on several previous rounds remains problematic under Δ -synchrony, as it implicitly assumes nodes stay online long enough for messages to propagate. History-independent protocols eliminate this issue entirely, since decisions depend only on messages received within the most recent Δ -window, which is already guaranteed by the synchrony assumption. As a result, no additional buffering or recovery mechanism is required.

Limitation-3: Growing adversary (history-independent) without optimistic case. 1/3-MMR is the first protocol to achieve history independence in the sleepy setting, but it incurs higher latency than partially history-independent protocols [14], including 1/4-MMR [26]. Moreover, several sleepy BFT protocols [33, 12, 13, 8] achieve constant-round latency with optimistic outputs using optimistic quorums. However, these quorums are fixed according to the total number of participants N . In the sleepy setting, where participation changes over time, such fixed quorums may prevent optimistic confirmation when only a small number of nodes are awake. This motivates the need for dynamic optimistic quorums that adapt to the current participation level. Designing such mechanisms is challenging, especially because optimistic outputs must remain consistent with normal-case outputs under dynamically changing participation. Moreover, dynamic optimistic quorums in the history-independent setting remain largely unexplored.

Based on the above discussions, sleepy protocols face several fundamental challenges, including backward-simulation attacks and high latency even under low corruption levels. Moreover, existing sleepy protocols still exhibit a noticeable gap from near-optimal round complexity under a growing adversary. In

addition, no existing sleepy protocol supports an optimistic path under dynamic participation. This naturally raises the following question:

Can we design a history-independent sleepy atomic broadcast protocol that tolerates a growing adversary while still enabling fast confirmation with near-optimal latency of 2Δ (or even Δ) under low corruption?

Contributions. We address the aforementioned question by introducing three optimistic atomic broadcast protocols in the sleepy setting. We highlight our improvement ⁵ in Table 1. Here is a summary of our contributions:

1. Our first protocol tolerates up to $1/5$ corruption and achieves a good-case latency of 2Δ . Notably, to the best of our knowledge, this is the first sleepy atomic broadcast protocol to attain optimal Δ -latency under a growing adversary when all honest nodes start with the same input.
2. We propose two sleepy protocols that incorporate a dynamic optimistic quorum. We first consider the super-optimistic (i.e., fault-free) case, in which we achieve confirmation with a best-case latency of 2Δ and a good-case latency of 3Δ . Subsequently, we relax the super-optimistic condition and present an optimistic protocol that achieves a good-case latency of 2Δ under up to $1/6$ corruption. In both cases, when the corruption level exceeds the super-optimistic (0) or optimistic ($1/6$) threshold, the protocol immediately falls back to a regular confirmation rule that tolerates up to $1/3$ corruption, while still guaranteeing a good-case latency of 3Δ . Our protocols do not require an explicit view-change mechanism.
3. All of our protocols tolerate an adversary capable of simulating behavior up to Δ time into the past, matching the stronger adversarial model considered in existing sleepy BFT protocols. Moreover, our protocols assume that a node $\mathcal{P}$ that wakes up at time t can receive the messages sent during the interval $[t - \Delta, t]$, aligning with existing work with minimal buffer requirements.

2 Related Work

In atomic broadcast, many works [29, 18, 25, 20, 11, 21, 36, 9, 38] have studied quorum-based settings, where forwarded messages can serve as certifications. However, the Proof-of-Work mechanism proposed by Bitcoin [31] is significantly different from this paradigm, as it introduces a dynamic model in which participation is open to nodes that can join and leave at any time, as long as the proportion of computational power controlled by faulty nodes does not exceed $1/2$ of the total computational power in the system.

The dynamic participation model was later formalized by Pass and Shi [32] as the sleepy model. They showed that under conditions of unknown and dynamic

⁵ We focus on latency and omit other metrics such as communication and message complexities. Our protocol employs the same message pattern as existing sleepy BFT protocols. Hence, it improves latency without compromising communication complexity or message complexity compared with prior sleepy BFT protocols.

Table 1. Protocol Comparison in the Sleepy Model.

	$T_b^\dagger$	Tolerance	History Independent [‡]	Confirmation Latency			
				No-fault (best-case) [¶]	1/6-fault (good-case) [♣]	1/5-fault (good-case) [♣]	Expected [♣]
Bitcoin [31]	0	1/2	✓	$O(\frac{\lambda\Delta}{\gamma})$	$O(\frac{\lambda\Delta}{\gamma})$	$O(\frac{\lambda\Delta}{\gamma})$	$O(\frac{\lambda\Delta}{\gamma})$
Pow multi-chain [1, 17, 22]	0	1/2	✓	$O(\frac{\Delta}{\gamma})$	$O(\frac{\Delta}{\gamma})$	$O(\frac{\Delta}{\gamma})$	$O(\frac{\Delta}{\gamma})$
PS-Sleepy [32]	∞	1/2	✗	$O(\frac{\lambda\Delta}{\gamma})$	$O(\frac{\lambda\Delta}{\gamma})$	$O(\frac{\lambda\Delta}{\gamma})$	$O(\frac{\lambda\Delta}{\gamma})$
Goyal et al. [19]	∞	1/2	✗	$O(\lambda\Delta)$	$O(\lambda\Delta)$	$O(\lambda\Delta)$	$O(\lambda\Delta)$
MR2022 [30]	∞	1/2	✗	16Δ	16Δ	16Δ	32Δ
Dynamic ACS [15]	∞	1/2	✗	4Δ	4Δ	4Δ	22Δ
Good-case Sleepy-1 [14]	∞	$1 - 1/\sqrt{2}$	✗	$\Delta^\S$	4Δ	4Δ	24Δ
Good-case Sleepy-2 [14]	∞	1/2	✗	2Δ	4Δ	4Δ	25Δ
Gafni-Losa [23]	$2\Delta^*$	1/2	✗	10Δ	10Δ	10Δ	20Δ
MMR2023 [27]	11Δ	1/2	✗	4Δ	4Δ	4Δ	14Δ
TOB-SVD [7]	5Δ	1/2	✗	6Δ	6Δ	6Δ	10Δ
1/4-MMR [26]	2Δ	1/4	✗	2Δ	2Δ	2Δ	8Δ
1/3-MMR [26]	Δ	1/3	✓	3Δ	3Δ	3Δ	9Δ
Our protocol 1 (Section 4)	Δ	1/5	✓	$\Delta^\S$	$\Delta^\S$	2Δ	5Δ
Our protocol 2 (Section 5)	Δ	1/3	✓	2Δ	3Δ	3Δ	9Δ
Our protocol 3 (Section 6)	Δ	1/3	✓	2Δ	2Δ	3Δ	18Δ

[†] T_b measures the number of rounds a corrupt party is counted as awake before it wakes up. We highlight protocols that achieve a $T_b \leq \Delta$.

[‡] Messages at any time t depend only on messages received during $[t - \Delta, t]$.

[¶] best-case: the confirmation latency when all nodes are honest.

[♣] good-case: the **shortest** confirmation latency with an honest leader.

[♣] Expected latency: the leader may be corrupted under corresponding fault tolerance.

^{||} λ stands for the security parameter, and γ represents participation level.

^{*} The authentication mechanism can be implemented in two rounds.

[§] This case considers that all awake honest nodes have the same input to the system at the same time. However, under this same input condition, other protocols do not achieve any improvement in round performance.

participation, prior knowledge of network delay is necessary, which implies that network synchrony is a required assumption for sleepy protocols. They also proposed a longest-chain-style protocol within this model. However, such longest-chain protocols [31, 32, 3] typically suffer from high confirmation latency. For example, Bitcoin exhibits a latency of $O(\lambda\Delta/\gamma)$, where λ is a security parameter, γ denotes the level of participation and Δ is the bound on network latency. Likewise, the latency of protocol in [32] also depends on a security parameter κ , in addition to Δ . Another line of research [30, 26, 27, 7] explores BFT-style solutions to sleepy consensus, achieving constant round confirmation latencies on the order of $O(\Delta)$. Specifically, 1/3-MMR [26] achieves a latency of 3Δ under up to 1/3 corruption, while 1/4-MMR [26] reduces the latency to 2Δ by lowering

the corruption tolerance to $1/4$ among online nodes and adopting a weaker adversary model compared to $1/3$ -MMR. Compared with the two protocols in [26], our first protocol (see Section 4) improves upon the $1/4$ -MMR adversary model. Our super-optimistic protocol (see Section 5) achieves better performance (2Δ) in the fault-free case while maintaining the same performance under $1/3$ corruption. Moreover, to relax the super-optimistic condition, we also present an optimistic protocol (see Section 6) with lower fault tolerance while retaining the same 2Δ round performance. Besides, these works [27, 7, 23] improve corruption tolerance to $1/2$, at the cost of restricting the adversary’s ability to grow over time. Among these, Gafni et al. achieve a latency of 10Δ [23], D’Amato et al. reduce it to 6Δ [7], and Malkhi et al. further improve it to 4Δ [27]. Some hybrid approaches [33, 12, 13, 8] combine an optimistic decision rule with a sleepy longest-chain protocol. However, their optimistic confirmation rules are inapplicable under low participation because their quorum sizes are fixed, that is static, and depend solely on the assumed total participation set in advance, whereas our work adopts a dynamic quorum even for optimistic cases. Beyond the sleepy model, prior work has also explored reducing the latency of atomic broadcast [37, 35]. Early-stopping Byzantine broadcast [24] provides a related approach to reducing latency in executions with fewer faults.

Apart from latency, another crucial aspect of the sleepy model is the dynamic participation of potentially corrupted nodes. Early sleepy protocols cannot tolerate a growing adversary because they require the corruption level to remain below a fixed threshold relative to the lowest participation level observed during the entire protocol execution. This assumption is often overly restrictive in practice. To address this limitation, $1/3$ -MMR and $1/4$ -MMR introduced a graded agreement mechanism that enables tolerance of a growing adversary. Classical graded agreement [16, 2] is typically defined in the non-sleepy setting with a fixed set of participating parties and at least two grades. Sleepy graded agreement follows a similar graded-agreement structure but formulates its guarantees in the sleepy setting. In $1/3$ -MMR, the adversary is allowed to grow proportionally with the number of honest nodes in each round. In contrast, $1/4$ -MMR supports only moderate adversarial growth, as its design allows for backward simulation of up to 2Δ . Both of our protocols adopt the same adversary model as $1/3$ -MMR.

3 Model, Primitive and Design Goal

3.1 System Model

Setup. We assume the existence of a public-key infrastructure (PKI) and a set of N nodes, denoted by $\{\mathcal{P}_1, \dots, \mathcal{P}_N\}$. The identities of the nodes are public and verifiable through the PKI. Each node $\mathcal{P}_i$ is associated with a unique public/private key pair (pk_i, sk_i) . It is assumed that every node knows the public keys of all nodes, i.e., $\{pk_j\}_{j=1}^N$, but has access only to its own secret key sk_i . We use $\langle m \rangle_{\mathcal{P}_i}$ to denote a message m signed by $\mathcal{P}_i$ using sk_i .

Network model. We assume a synchronous network in which any message sent by a node at time t is delivered to all awake nodes by time $t + \Delta$, where Δ

denotes the upper bound on communication delay. This synchrony assumption is essential for protocols operating under the sleepy model [32]. For simplicity, we assume the existence of a perfectly synchronized global clock, allowing all nodes to start each round simultaneously. However, using the round transformation technique proposed in [30], the protocol can be adapted to systems with loosely synchronized clocks that may differ by up to Δ . Additionally, we assume that the communication channel is unauthenticated. To ensure message integrity and authenticity, all messages are signed by their sender, and any message lacking a valid signature is discarded.

Adversary model. We consider a computationally bounded adaptive adversary that can corrupt nodes during protocol execution but cannot retract messages that have already been sent. Once corrupted, it becomes Byzantine and remains permanently under the full control of the adversary, capable of deviating arbitrarily from the protocol. Any node that is not corrupted is honest and follows the protocol faithfully. We use the terms corrupt and Byzantine interchangeably.

Sleepy model. Honest nodes may dynamically switch between awake (online) and asleep (offline) states, under the adversary’s control. Awake honest nodes faithfully execute the protocol, whereas asleep nodes neither send nor receive messages. Asleep honest nodes are not adversarially controlled and do not count toward the corruption budget. Time is divided into rounds of length Δ . If the current time $t \in [r\Delta, (r+1)\Delta]$, we refer to the current round as round r (where $r \geq 0$). Protocol execution starts from time $t = 0$. At the beginning of each round r , i.e., $t = r\Delta$, a subset of nodes is awake, denoted by O_r , where $0 < |O_r| \leq N$. Among these, H_r denotes the set of awake honest nodes in round r .

We consider a growing adversary with monotonic corruptions, i.e., $B_r \subseteq B_{r'}$ for all $r < r'$. In other words, once a node is corrupted, it remains corrupted and stays awake in all subsequent rounds, as the adversary can conduct a *forward simulation* using the corrupted node’s internal state in any subsequent rounds. Corrupted nodes may perform *backward simulation* [27], meaning that a node awakened at $r\Delta$ can retroactively inject messages as if active at $r\Delta - T_b$ with no extra cost. Since the protocol cannot effectively prevent such costless simulation, a corrupt node is conservatively counted as active for an extra T_b time backward beyond its actual active period. We therefore let B_r denote the set of corrupted nodes that may affect the result of round r , which includes Byzantine nodes awakened during the time interval $[r\Delta, r\Delta + T_b]$. Fig. 1 illustrates how varying T_b affects the effective Byzantine budget. Our protocol targets $T_b = \Delta$, where protocol messages only depend on the previous round; that is, (newly) awake parties participate in round r based on messages received in round $r - 1$.

3.2 Preliminaries

Notation. Before presenting the preliminaries used in this paper, we first define the notation for log. Throughout this paper, we assume that each node’s input is a block B containing a batch of messages. A *log* A is represented as an ordered sequence of blocks $[B_0, B_1, \dots, B_k]$. If there are two logs $A_1 = (B_0, \dots, B_i)$, and

$\Lambda_2 = (B'_0, \dots, B'_j)$, when $i \leq j$ we have $B_k = B'_k$ for all $k \leq i$, then Λ_1 is a *prefix* of Λ_2 , denoted as $\Lambda_1 \preceq \Lambda_2$. We say that a log Λ *extends* another log Λ' if Λ' is a prefix of Λ , denoted as $\Lambda \succeq \Lambda'$. Two logs Λ and Λ' are said to be *conflict* if neither $\Lambda \succeq \Lambda'$ nor $\Lambda' \succeq \Lambda$ holds. Conversely, two logs are *compatible* if at least one extends the other.

Latency. Latency is measured from the moment the designated leader (selected by VRF) inputs a value v , to the moment when value v gets decided by any honest node. We define the *best-case* latency as the latency when all nodes are honest, and *good-case* latency as the latency when at least the leader is honest under the assumed corruption model. By contrast, the *expected-case* latency captures executions in which the leader may be Byzantine. We remark that the super-optimistic case can correspond to the best case, whereas the optimistic case assumes that the leader is honest while some corruptions may still exist.

Digital Signature. We assume a secure digital signature scheme [34]. A message m signed by node $\mathcal{P}$ is denoted by $\langle m \rangle_{\mathcal{P}}$. Recipients automatically verify signatures and discard invalid messages. Unless stated otherwise, we assume all signatures are valid.

Verifiable Random Function. We assume a secure verifiable random function (VRF) [28]. Given an input μ , node $\mathcal{P}$ computes $(\rho, \pi) \leftarrow \text{VRF}_{\mathcal{P}}(\mu)$, where ρ is the pseudorandom output and π is the proof. Recipients verify VRF proofs and discard invalid messages. Unless stated otherwise, all VRF outputs are valid.

Graded Agreement. The graded agreement (GA) [26] protocol is executed among a set of n nodes, with at most $1/3$ of them being corrupted. Each node begins with an input log Λ and broadcasts it to all participants. The protocol guarantees that each node outputs a log along with a grade, denoted $(\Lambda, *)$, where $*$ $\in \{0, 1\}$. The graded agreement protocol satisfies the following properties:

1. *Graded consistency:* If an awake honest node outputs $(\Lambda, 1)$, then all awake honest nodes at least output the Λ with grade 0.
2. *Integrity:* If an awake honest node outputs $(\Lambda, *)$, then at least one awake honest node has input a log Λ' s.t. $\Lambda' \succeq \Lambda$.
3. *Validity:* If all awake honest nodes input a log that extends Λ , then all awake honest nodes output $(\Lambda, 1)$.
4. *Uniqueness:* If an awake honest node outputs $(\Lambda, 1)$, and another awake honest node outputs $(\Lambda', 1)$, then either $\Lambda' \succeq \Lambda$ or $\Lambda \succeq \Lambda'$.
5. *Bounded divergence:* Each awake honest node outputs at most two conflicting logs with grade 0.

Revised GA. Throughout this paper, we slightly revise the original graded consistency property of the graded agreement (GA) protocol in [26] to strengthen its guarantee: if any awake honest node outputs a log with grade 1, then all awake honest nodes must output a log with grade 0, and moreover, the grade 0 log must extend the grade 1 log. For completeness, we provide the revised GA procedure (see Algorithm 1), although the modifications are minimal. In the original GA protocol [26], an awake honest node outputs a log with grade 0 if it

Algorithm 1 Graded Agreement (GA) [26] for awake node $\mathcal{P}$ at time t :

(i). If Λ' extends Λ , then $\langle \text{VOTE}, \Lambda' \rangle$ is also counted as a vote for Λ

```

1: if  $t = 0$  then
2:   multicast  $\langle \text{VOTE}, \Lambda \rangle_{\mathcal{P}}$ , where  $\Lambda$  is the input log
3: if  $t = \Delta$  then
4:   Let  $m$  be the number of received VOTE messages
5:   if the longest log  $\Lambda$  voted by  $> 2m/3$  nodes then
6:     output  $(\Lambda, 1)$ 
7:   if the longest log  $\Lambda$  voted by  $> m/3$  nodes then
8:     output  $(\Lambda, 0)$ 

```

received votes from more than $m/3$ but fewer than $2m/3$ nodes, where m is the number of votes received by the awake honest node. In our revision, we remove the upper bound condition, i.e., a grade 0 output is produced as long as the log is supported by more than $m/3$ votes. The output condition for grade 1 remains unchanged: the log must be supported by more than $2m/3$ votes.

3.3 Design Goal

Atomic broadcast. An atomic broadcast protocol allows all honest nodes to input messages and ensures that they eventually agree on the same ever-growing log. Specifically, an atomic broadcast protocol satisfies the following properties:

1. *Safety*: If an awake honest node $\mathcal{P}_i$ outputs a log Λ_i at time t , and another awake honest node $\mathcal{P}_j$ outputs a log Λ_j at time t' , then Λ_i and Λ_j must be compatible.
2. *Liveness*: If an awake honest node inputs a message m at time t , then there must exist a time $t' \geq t$, such that all awake honest nodes output a log that includes m by time t' .

Throughout this paper, we assume input dissemination as in [27], namely, that all values broadcast by honest nodes before round r are eventually included in blocks proposed in round r .

4 Atomic Broadcast with 1/5 Fault Tolerance

In this section, we present our first atomic broadcast protocol, which tolerates up to $1/5$ faults, achieves 2Δ good-case latency, and completes in expectation within 5Δ in the worst case. When all awake honest nodes start with the same input, the protocol further achieves the optimal good-case latency of Δ .

Compared with the state-of-the-art protocol 1/4-MMR [26], which tolerates $1/4$ faults, our protocol achieves the same good-case round performance and reduces the normal-case latency by one round, despite a slightly lower fault tolerance threshold ($1/5$ vs. $1/4$). More importantly, our protocol operates under a strictly stronger adversary model: the 1/4-MMR protocol cannot tolerate

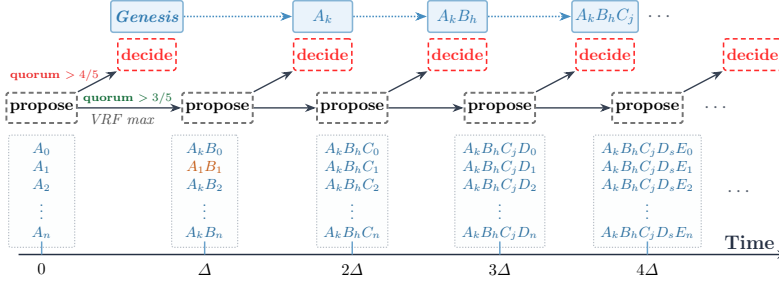


Fig. 2. Overview of Our Atomic Broadcast Protocol with 1/5 Fault Tolerance.

backward simulations shorter than 2Δ , and requires $|B_r \cup B_{r+1}| \leq |O_r|/4$. In contrast, our protocol tolerates backward simulations as short as Δ and only requires $|B_r| \leq |O_r|/5$. This allows our protocol to handle a more aggressively growing adversary, which in practice, may not lead to a worse fault tolerance within each round, even 1/5 is smaller than 1/4 (cf. Fig. 1).

At a high level, the protocol proceeds in rounds. At the beginning of each round r , every participant proposes a log. At the end of round r , each node determines two logs based on the messages received during round r : a finalized log and a candidate log, where the latter is supported by at least 3/5 of the received messages. Each node then extends the longest candidate log (ties are broken by maximum VRF) with a new block and proposes the extended log at the beginning of round $r+1$. Since the network is synchronous, the extended log reaches all honest nodes by the end of round $r+1$. If the selected candidate was proposed by an honest node at the beginning of round r , it is finalized by all honest nodes by the end of round $r+1$, achieving a latency of 2Δ . Furthermore, when all awake honest nodes propose the same value in a round, all honest nodes decide the value immediately at the end of that round, yielding a latency of Δ .

4.1 Details of the 1/5 Fault Atomic Broadcast

Our atomic broadcast protocol with 1/5 fault tolerance is presented in Algorithm 2. All awake honest nodes perform input or output operations only at the times $t = k\Delta$ (where $k \in \mathbb{N}$); at all other times, they remain in a listening state. The detailed description of the protocol's execution process is as follows:

Initial round begins at time $t = 0$. The protocol proceeds in a round-by-round manner, starting from round 0. Each round lasts for a duration of Δ , and round k spans the time interval $[k\Delta, (k+1)\Delta]$. At the beginning of each round k (i.e., at time $t = k\Delta$), every awake honest node sends a PROPOSE message, where $k \in \mathbb{N}$. In our protocol, if a log A_k extends another log A'_k , then the message $\langle \text{propose}, A_k, \text{VRF}_{\mathcal{P}}(k) \rangle_{\mathcal{P}}$ is also considered a proposal for A'_k . Additionally, it is worth noting that at the beginning of round k (i.e., at time $t = k\Delta$), any messages received before round $k-1$ (i.e., before time $t = (k-1)\Delta$) are considered expired and are excluded from the execution of round k . Furthermore, each node accepts at most one message from any given sender per round. If two distinct messages from the same sender are received within the same round,

Algorithm 2 Atomic broadcast with $1/5$ fault tolerance for awake node $\mathcal{P}$ at time t :

- (i). In time $t = (k+1)\Delta$ ($k \geq 0$), the received message $\langle \text{PROPOSE}, \Lambda_k, \text{VRF}_{\mathcal{P}}(k) \rangle_{\mathcal{P}}$ is considered invalid if any of the following conditions are met:
 - (a). Λ_k conflicts with the output log $\mathcal{F}$
 - (b). $|\Lambda_k| \neq k+1$
 - (c). $\text{VRF}_{\mathcal{P}}(k)$ is an invalid VRF output on k
- (ii). If a node $\mathcal{P}$ sends two different messages of the form $\langle \text{PROPOSE}, \Lambda_k, \text{VRF}_{\mathcal{P}}(k) \rangle_{\mathcal{P}}$ and $\langle \text{PROPOSE}, \Lambda'_k, \text{VRF}'_{\mathcal{P}}(k) \rangle_{\mathcal{P}}$ in round k with $\Lambda_k \neq \Lambda'_k$, both messages are ignored
- (iii). If Λ_k extends Λ , then $\langle \text{PROPOSE}, \Lambda_k, \text{VRF}_{\mathcal{P}}(k) \rangle_{\mathcal{P}}$ is also viewed as a propose for Λ

Initialize: $\log \mathcal{F} \leftarrow []$

1: **if** $t = 0$ **then** ▷ propose: round 0

2: **multicast** $\langle \text{PROPOSE}, \Lambda_0, \text{VRF}_{\mathcal{P}}(0) \rangle_{\mathcal{P}}$, where $\Lambda_0 := B_0$

Time t in round k , i.e., $t \in [k\Delta, (k+1)\Delta]$

3: **if** $t = k\Delta$ and $k > 0$ **then**

4: Let m be the number of received PROPOSE message ▷ sent in round $k-1$

5: Set $\mathcal{F}^{k-1}$ to the longest log that is proposed by $> 4m/5$ nodes

6: Set $\mathcal{C}^{k-1}$ to the longest log that is proposed by $> 3m/5$ nodes

7: $\mathcal{F} \leftarrow \mathcal{F}^{k-1}$

8: **output** $\mathcal{F}$ ▷ output

9: Let Λ be the longest log in the received valid PROPOSE messages with the maximum valid VRF on $k-1$ that extends $\mathcal{C}^{k-1}$

10: Let $\Lambda_k := \Lambda || B_k$

11: **multicast** $\langle \text{PROPOSE}, \Lambda_k, \text{VRF}_{\mathcal{P}}(k) \rangle_{\mathcal{P}}$ ▷ propose: round k

both messages are discarded. At time $t = 0$, every awake honest node multicasts a message $\langle \text{propose}, \Lambda_0, \text{VRF}_{\mathcal{P}}(0) \rangle_{\mathcal{P}}$ to all other nodes, where each node uses its local block B_0 as Λ_0 .

Output. At any time $t = k\Delta$ (for $k \geq 1$), each node first enters the output moment of round k . It tallies all valid $\langle \text{propose}, *, \text{VRF}(k-1) \rangle$ messages received from round $k-1$, and sets $\mathcal{F}^{k-1}$ to be the longest log that is extended by more than $4/5$ of the logs contained in the received messages. Additionally, each node sets its candidate log $\mathcal{C}^{k-1}$ to be the longest log that is extended by more than $3/5$ of the logs in the same set of received messages. Finally, $\mathcal{F}^{k-1}$ is assigned to the node's finalized output $\mathcal{F}$.

Propose at time $t \geq \Delta$. At any time $t = k\Delta$ (for $k \geq 1$), each node then enters the propose moment of round k . Every awake honest node selects all valid $\langle \text{propose}, *, \text{VRF}_{k-1} \rangle$ messages from round $k-1$ whose logs extend $\mathcal{C}^{k-1}$. Among the selected proposals, the node identifies Λ as the longest log associated with the maximum valid VRF in round $k-1$, and constructs a new log Λ_k by appending a new block B_k to Λ . This new log is then multicast to the system (cf. Algorithm 2, lines 9-11).

4.2 Security Analysis of the $1/5$ Fault Atomic Broadcast

Proof Sketch. We establish security by showing that Algorithm 2 satisfies both safety and liveness under a $1/5$ -fraction Byzantine adversary in the sleepy model.

Safety. It follows from a monotonicity property of log proposals. Specifically, once an awake honest node outputs a log Λ at the end of a round, there must be at least $3/5$ awake honest nodes that proposed Λ at the beginning of this round (since outputs require $4/5$ support and at most $1/5$ may be Byzantine), and all honest nodes must extend Λ in the subsequent round. By induction over rounds, this implies that any two logs output by honest nodes are compatible.

Liveness. It follows from the fact that any proposal sent by an honest node is always considered valid by other honest nodes. Once all honest nodes propose logs extending a common prefix, the protocol guarantees that this prefix is output in the next round. Moreover, the VRF-based leader selection ensures that, with probability at least $2/5$ (since Propose requires $3/5$ support and at most $1/5$ may be Byzantine; see Lemma 3 for details), all honest nodes extend the same log in a given round. Repeating this process over rounds implies that every value proposed by an honest node is eventually included in an output log.

Lemma 1. *If an awake honest node p_i outputs a log Λ at time $t = k\Delta$ (for $k \geq 1$), then all awake honest nodes must propose a log Λ_k that extends Λ at the same time $t = k\Delta$.*

Proof. If $\mathcal{P}_i$ outputs a log Λ at the beginning of round k , that is, at time $t = k\Delta$ with $k \geq 1$, then according to Algorithm 2, it must have received at least $4|\mathcal{M}_i|/5$ PROPOSE messages that extend Λ , where $\mathcal{M}_i$ denotes the set of all valid PROPOSE messages received from round $k-1$. We assume there is another awake honest node at time t , denoted by $\mathcal{P}_j$. Similarly, let $\mathcal{M}_j$ denote the set of all valid PROPOSE messages received by $\mathcal{P}_j$ at time t . Let $B_{r,i}$ and $B_{r,j}$ denote the sets of Byzantine messages received by $\mathcal{P}_i$ and $\mathcal{P}_j$, respectively.

The number of awake nodes at the beginning of round $k-1$ is $|O_{k-1}|$, where $|O_{k-1}| = |H_{k-1}| + |B_{k-1}|$. According to the algorithm, if a node sends two different messages, both messages will be discarded. Since Byzantine nodes may not multicast their messages, as a result, we have $|\mathcal{M}_i| \leq |O_{k-1}|$ and $|\mathcal{M}_j| \leq |O_{k-1}|$. Since the network is synchronous and all honest nodes multicast their messages to all nodes in the system, both $\mathcal{M}_i$ and $\mathcal{M}_j$ must contain, at a minimum, the messages sent by all awake honest nodes in H_{k-1} .

Next, we prove that when $\mathcal{P}_i$ outputs a log Λ at time t , then at least $3|\mathcal{M}_j|/5$ of the awake honest nodes must have multicast a PROPOSE message that extends the log Λ at time t . We assume that $\mathcal{M}_i$ consists of messages sent by nodes in $H_{k-1} \cup B_{k-1,i}$, where $B_{k-1,i} \subseteq B_{k-1}$, and $\mathcal{M}_j$ consists of messages from $H_{k-1} \cup B_{k-1,j}$, where $B_{k-1,j} \subseteq B_{k-1}$. Since $|H_{k-1}| > 4|B_{k-1}|$ and $|B_{k-1,i}|, |B_{k-1,j}| \leq |B_{k-1}|$, we have $|H_{k-1}| > 3|B_{k-1,j}| + |B_{k-1,i}|$. Therefore, if $\mathcal{P}_i$ outputs a log Λ at time $t = k\Delta$, then at time $t = (k-1)\Delta$ at least $\frac{4}{5}|\mathcal{M}_i| - |B_{k-1,i}|$ awake honest nodes must have multicast a PROPOSE message extending Λ . It follows that:

$$\begin{aligned} \frac{4}{5}|\mathcal{M}_i| - |B_{k-1,i}| &= \frac{4}{5}(|H_{k-1}| + |B_{k-1,i}|) - |B_{k-1,i}| \\ &= \frac{4}{5}|H_{k-1}| - \frac{1}{5}|B_{k-1,i}| = \frac{3}{5}|H_{k-1}| + \frac{1}{5}(|H_{k-1}| - |B_{k-1,i}|) \\ &> \frac{3}{5}|H_{k-1}| + \frac{3}{5}|B_{k-1,j}| = \frac{3}{5}|\mathcal{M}_j|. \end{aligned}$$

As honest parties multicast their messages, every honest node receives these proposals. Hence, $\mathcal{P}_j$ must observe more than $3|\mathcal{M}_j|/5$ PROPOSE messages extending Λ . By Algorithm 2, $\mathcal{P}_j$ therefore proposes a log extending Λ at time $t = k\Delta$.

Lemma 2 (Safety). *If an awake honest node $\mathcal{P}_i$ outputs a log Λ_i at time t , and another awake honest node $\mathcal{P}_j$ outputs a log Λ_j at time t' , then Λ_i and Λ_j must be compatible.*

Proof. We first consider the case where $t' = t$. Without loss of generality, assume $t = t' = k\Delta$ for $k \geq 1$. Let $\mathcal{M}_i$ and $\mathcal{M}_j$ denote the sets of all valid PROPOSE messages received by nodes $\mathcal{P}_i$ and $\mathcal{P}_j$, respectively. Similar to Lemma 1, we define subsets $B_{k-1,i}$ and $B_{k-1,j}$ such that $\mathcal{M}_i$ consists of messages from $H_{k-1} \cup B_{k-1,i}$, and $\mathcal{M}_j$ consists of messages from $H_{k-1} \cup B_{k-1,j}$. Under the maximum fault tolerance assumption, we have $|H_{k-1}| > |B_{k-1}|$, and it holds that $|B_{k-1}| \geq |B_{k-1,i}|$ and $|B_{k-1}| \geq |B_{k-1,j}|$.

If Λ_i conflicts with Λ_j , then there must be at least $4/5|\mathcal{M}_i|$ nodes proposing Λ_i and at least $4/5|\mathcal{M}_j|$ nodes proposing Λ_j . Therefore, the total number of honest nodes awake at time $t = (k-1)\Delta$ satisfies:

$$\begin{aligned} |H_{k-1}| &\geq \left(\frac{4}{5}|\mathcal{M}_i| - |B_{k-1,i}|\right) + \left(\frac{4}{5}|\mathcal{M}_j| - |B_{k-1,j}|\right) \\ &= \left(\frac{4}{5}(|H_{k-1}| + |B_{k-1,i}|) - |B_{k-1,i}|\right) + \left(\frac{4}{5}(|H_{k-1}| + |B_{k-1,j}|) - |B_{k-1,j}|\right) \\ &= \frac{8}{5}|H_{k-1}| - \frac{1}{5}(|B_{k-1,i}| + |B_{k-1,j}|) > \frac{6}{5}|H_{k-1}|, \end{aligned}$$

which leads to a contradiction since it implies $|H_{k-1}| > 6|H_{k-1}|/5$. Hence, Λ_i and Λ_j must be compatible.

Now we consider the case where $t' \geq t + \Delta$. Suppose $\mathcal{P}_i$ outputs log Λ at time t . According to Lemma 1, all logs proposed by honest nodes at time t must extend Λ . Consequently, at time $t + \Delta$, each honest node must output a log Λ' such that $\Lambda' \succeq \Lambda$. Subsequently, at time $t + \Delta$, all honest nodes will propose a new log that extends Λ' , which itself extends Λ , and thus the new proposals also extend Λ . Applying this argument inductively, it follows that at any time $t' \geq t + \Delta$, all honest nodes will propose logs that extend Λ . Therefore, the output log Λ_j of any honest node $\mathcal{P}_j$ at time t' must be compatible with Λ .

Lemma 3 (Liveness). *If an awake honest node inputs a message m at time t , then there must exist a time $t' \geq t$, such that all awake honest nodes output a log that includes m by time t' .*

Proof. To prove liveness, we first show that any PROPOSE message sent by an honest node is valid to every other awake honest node. According to Algorithm 2, a PROPOSE message sent by an honest node $\mathcal{P}_i$ never violates rule (i).(b) or rule (i).(c). Regarding rule (i).(a), by Lemma 2, the proposed log Λ_i is compatible with the output of other honest nodes. Therefore, any PROPOSE message sent by an honest node is always valid in the view of other honest nodes.

Secondly, we prove that if all honest nodes propose a log that extends the same log Λ_k at time $(k+1)\Delta$, where Λ_k was multicast by an honest node at time $k\Delta$, then Λ_k will be output by all honest nodes at time $(k+2)\Delta$. Suppose that at time $t = k\Delta$, an awake honest node $\mathcal{P}_i$ inputs a value m into the system by broadcasting a $\langle \text{PROPOSE}, \Lambda_k, \text{VRF}_{\mathcal{P}}(k) \rangle_{\mathcal{P}_i}$ to all, where $m \in \Lambda_k$. By the synchronous network assumption, all awake honest nodes will receive this message by time $t = (k+1)\Delta$. If $\text{VRF}_{\mathcal{P}}(k)$ is the maximum among all valid PROPOSE messages received from round k , then according to the protocol, all awake honest nodes will propose a new log that extends Λ_k at time $(k+1)\Delta$. Since the number of awake honest nodes at round $k+1$ satisfies $|H_{k+1}| > 4|O_{k+1}|/5$, and each awake honest node $\mathcal{P}_i$ at time $(k+2)\Delta$ receives at most $|\mathcal{M}_i| \leq |O_{k+1}|$ valid messages from round $k+1$, it follows that the number of messages from honest nodes satisfies $|H_{k+1}| > 4|\mathcal{M}_i|/5$. Therefore, at least $4|\mathcal{M}_i|/5$ of the received PROPOSE messages will contain logs that extend Λ_k . By the protocol's output rule, a log extended by more than $4/5$ of the received proposals is eligible to be output. Hence, all awake honest nodes at time $(k+2)\Delta$ will output a log that extends Λ_k , which includes the value m .

Lastly, we prove that with probability at least $2/5$, every honest node multicasts a PROPOSE message at time $t = (k+1)\Delta$ extending the log Λ , previously multicast by an honest node at time $t = k\Delta$. According to Algorithm 2, all honest nodes multicast a new log that extends $\mathcal{C}^k$, where $\mathcal{C}^k$ is the longest log supported by more than $3/5$ of the received proposals in round k . Given that $|H_r| > 4|B_r|$, two conflicting $\mathcal{C}^k$ logs cannot simultaneously exist, ensuring compatibility among honest nodes. At time $t = (k+1)\Delta$, each honest node selects from the valid PROPOSE messages the longest log that extends $\mathcal{C}^{k-1}$ and has the maximum valid VRF output for round k . In the worst case, for any honest node, the number of received PROPOSE messages in round r is $m = |H_r| + |B_r|$. Some honest nodes may only consider a subset of VRF outputs generated by $2m/5$ honest nodes (denoted as the set H'), while others consider VRF outputs from all honest nodes. Since VRF outputs are pseudorandom, the probability that all honest nodes select the same proposal is at least $2/5$, namely, the global maximum VRF value is generated by a node in H' .

In sum, at time $t = (k+1)\Delta$, if all honest nodes multicast a PROPOSE message that extends the same log Λ proposed by an honest node at time $t = k\Delta$, then it requires only one more round for Λ to be output. That is, for any message $m \in \Lambda$, if m was proposed at time $t = k\Delta$, then it will be output at time $t = (k+2)\Delta$. Since the probability that all honest nodes extend the same log Λ (originating from the same honest node) is at least $2/5$, it follows that all values input by honest nodes into the atomic broadcast protocol are eventually included in an output log, thereby satisfying the liveness property.

Lemma 4 (Optimal Round). *If all awake honest nodes propose the same log $\Lambda_k = \Lambda \| B_k$ at time $t = k\Delta$, then all awake honest nodes will output Λ_k at time $(k+1)\Delta$.*

Proof. The lemma follows immediately from the second part of the proof in Lemma 3.

Theorem 1. *Algorithm 2 implements an atomic broadcast protocol with $1/5$ fault tolerance in the sleepy model.*

Proof. Lemma 2 and Lemma 3 complete the proof.

5 Super Optimistic Atomic Broadcast

In this section, we first study a sleepy consensus protocol that achieves a latency of 2Δ in the super-optimistic case, where every awake node is honest (fault-free). When Byzantine nodes are present but constitute less than one-third of the awake nodes, the protocol continues to guarantee a good-case latency of 3Δ .

5.1 Preparation: GA with a Super-optimistic Case

To achieve a latency of 2Δ , we first consider an additional case with grade 2 for the GA instance and define the modified GA protocol as **SuperOGA**.

Super-optimistic Graded Agreement (SuperOGA). Each participating node begins with an input log Λ and broadcasts a signed message $\langle \text{VOTE}, \Lambda \rangle$ to all participants. The protocol guarantees that each node outputs a log along with a grade, denoted by $(\Lambda, *)$, where $*$ $\in \{0, 1, 2\}$. Formally, **SuperOGA** satisfies the following properties:

1. *Bounded divergence, Integrity:* The same as **GA**.
2. *Graded consistency:* If an awake honest node $\mathcal{P}_i$ outputs (Λ_i, g) for $g = 2$, then any awake honest node $\mathcal{P}_j$ must output a Λ'_j with grade 1 such that $\Lambda'_j \succeq \Lambda_i$, and a Λ''_j with grade 0 such that $\Lambda''_j \succeq \Lambda'_j$. Additionally, if an awake honest node $\mathcal{P}_i$ outputs (Λ_i, g) for $g = 1$, then all awake honest nodes must output a Λ' with grade 0 that extends Λ_i .
3. *Validity:* If all awake honest nodes input a log that extends Λ , then all awake honest nodes output (Λ, g) for $g \geq 1$.
4. *Uniqueness:* If any awake honest node $\mathcal{P}$ output (Λ, g) , then no awake honest nodes output a conflicting (Λ', g) for $g \geq 1$.

Details and security analysis of the SuperOGA. Our super-optimistic protocol **SuperOGA** is presented in Algorithm 3. At time $t = 0$, each awake node multicasts a **VOTE** message to support its own input log Λ . By time $t = \Delta$, awake nodes collect the **VOTE** messages they have received and determine their outputs. Importantly, with the presence of dynamic participation and Byzantine nodes, nodes lack global knowledge of participation and may receive votes from different subsets of nodes. Consequently, each node makes decisions based on its own local view of participation. In our protocol, if Λ' extends Λ , then $\langle \text{VOTE}, \Lambda' \rangle$ is also counted as a vote for Λ . For any honest awake node, if a log Λ is supported by all of the received **VOTE** messages, the node outputs Λ with grade 2. If a log Λ is supported by more than $2/3$, or $1/3$ of the received **VOTE** messages, the node outputs Λ with grade 1, or 0, respectively. We have the following theorem:

Algorithm 3 Super-optimistic Graded Agreement (SuperOGA) for awake node $\mathcal{P}$ at time t :

- (i). If Λ' extends Λ , then $\langle \text{VOTE}, \Lambda' \rangle$ is also counted as a vote for Λ
- (ii). If a node sends different votes, all of its votes are ignored. Votes from the same node are counted at most once.

```

1: if  $t = 0$  then
2:   multicast  $\langle \text{VOTE}, \Lambda \rangle_{\mathcal{P}}$ , where  $\Lambda$  is the input log
3: if  $t = \Delta$  then
4:   Let  $m$  be the number of received VOTE messages
5:   if the longest log  $\Lambda$  voted by  $m$  nodes then
6:     output  $(\Lambda, 2)$  ▷ Super optimistic output
7:   if the longest log  $\Lambda$  voted by  $> 2m/3$  nodes then
8:     output  $(\Lambda, 1)$ 
9:   if the longest log  $\Lambda$  voted by  $> m/3$  nodes then
10:    output  $(\Lambda, 0)$ 

```

Theorem 2. *Algorithm 3 implements a super-optimistic graded agreement protocol.*

Proof. Assume honest nodes $\mathcal{P}_i$ and $\mathcal{P}_j$ are awake at time $t = \Delta$ of round r , and let $\mathcal{M}_i$ and $\mathcal{M}_j$ denote the sets of messages they have received since time $t = 0$ of round r . According to the model assumption, at time $t = 0$, the number of awake honest nodes satisfies $|H_r| > 2|O_r|/3$, where $|O_r| = |H_r| + |B_r|$. This implies that $|H_r| > 2|B_r|$. Let $B_{r,i}$ and $B_{r,j}$ denote the sets of Byzantine messages received by $\mathcal{P}_i$ and $\mathcal{P}_j$, respectively. We now prove each property separately.

1. **Graded consistency.** Suppose that $\mathcal{P}_i$ outputs Λ_i with grade 2. Then, by definition, $\mathcal{P}_i$ must have received $\mathcal{M}_i$ votes for Λ_i by time $t = \Delta$. Since honest nodes multicast their votes to all participants, all honest nodes must have voted for Λ_i at time $t = 0$. By definition, $|H_r| > 2|O_r|/3 > 2|\mathcal{M}_j|/3$, thus any $\mathcal{P}_j$, must output a $(\Lambda'_j, 1)$ such that $\Lambda'_j \succeq \Lambda_i$. Furthermore, it is evident that $\mathcal{P}_j$ must also output a $(\Lambda''_j, 0)$ with $\Lambda''_j \succeq \Lambda_i$. Similarly, if $\mathcal{P}_i$ outputs Λ for grade 1, then it must have received more than $2|\mathcal{M}_i|/3$ votes for Λ . The number of honest nodes who voted for Λ is at least:

$$\begin{aligned}
\frac{2}{3}|\mathcal{M}_i| - |B_{r,i}| &= \frac{2}{3}(|H_r| + |B_{r,i}|) - |B_{r,i}| = \frac{1}{3}|H_r| + \frac{1}{3}(|H_r| - |B_{r,i}|) \\
&> \frac{1}{3}|H_r| + \frac{1}{3}|B_{r,j}| = \frac{1}{3}(|H_r| + |B_{r,j}|) = \frac{1}{3}|\mathcal{M}_j|.
\end{aligned}$$

Thus $\mathcal{P}_j$ must output a $(\Lambda', 0)$ s.t. $\Lambda' \succeq \Lambda_i$.

2. **Validity.** If all awake honest nodes input a log that extends Λ , it also implies that more than $2|O_r|/3$ nodes vote for a log extending Λ at time $t = 0$. Let $\mathcal{M}_i$ denote the set of messages received by an honest node $\mathcal{P}_i$ at time $t = \Delta$. Since $|\mathcal{M}_i| \leq |O_r|$, it follows that $\mathcal{P}_i$ receives more than $2/3|\mathcal{M}_i|$ votes for a log extending Λ . Therefore, all awake honest nodes will output Λ with at least grade 1.

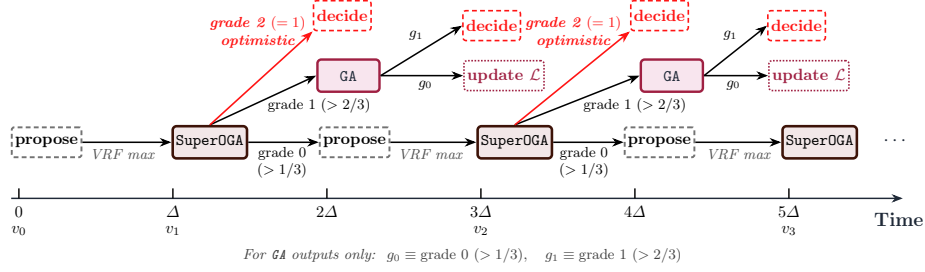


Fig. 3. Illustration of a protocol supporting the super-optimistic case.

If any awake honest node $\mathcal{P}$ output (Λ, g) , then no awake honest nodes output a conflicting (Λ', g) for $g \geq 1$.

3. **Uniqueness.** Similarly to the proof of graded consistency, if an awake honest node $\mathcal{P}_i$ outputs $(\Lambda, 1)$, then $\mathcal{P}_i$ must have received more than $2|\mathcal{M}_i|/3$ votes for Λ . If another awake honest node $\mathcal{P}_j$ outputs $(\Lambda', 1)$, then $\mathcal{P}_j$ must have received more than $2|\mathcal{M}_j|/3$ votes for Λ' . Since $M_i = H_r \cup B_{r,i}$, $M_j = H_r \cup B_{r,j}$, then the number of honest nodes that voted for Λ is at least $2|\mathcal{M}_i|/3 - |B_{r,i}| = 2|H_r|/3 - |B_{r,i}|/3$, and the number of honest nodes that voted for Λ' is at least $2|\mathcal{M}_j|/3 - |B_{r,j}| = 2|H_r|/3 - |B_{r,j}|/3$. If Λ and Λ' conflict, these honest supporters must be disjoint, as every honest node does not vote for two conflicting logs. Therefore, the sum of two honest supporters' sets should be less than $|H_r|$, thus $|H_r| \geq 4|H_r|/3 - (|B_{r,i}| + |B_{r,j}|)/3$, then we have $|H_r| \leq |B_{r,i}| + |B_{r,j}| \leq 2|B_r|$, contradicting the assumption $|H_r| > 2|B_r|$. A similar proof applies for the case $g = 2$.
4. **Bounded divergence.** Since a grade 0 output requires support from more than $1/3$ of the received votes, and each VOTE message can support only one log, at most two conflicting logs can each receive more than $1/3$ of disjoint messages from the same VOTE message set. Therefore, each node can output at most two conflicting logs.
5. **Integrity.** Suppose no awake honest node has input a log that extends Λ . Consider any awake honest node $\mathcal{P}_i$ that outputs $(\Lambda, *)$. Since $|H_r| > 2|O_r|/3 > 2|\mathcal{M}_i|/3$, and the set of VOTE messages received by $\mathcal{P}_i$ is $\mathcal{M}_i$, it follows that the number of nodes who could have voted for a log extending Λ must be less than $|\mathcal{M}_i|/3$. This contradicts the assumption that $\mathcal{P}_i$ outputs $(\Lambda, *)$, which requires at least $|\mathcal{M}_i|/3$ votes. Therefore, if an awake honest node outputs $(\Lambda, *)$, then at least one awake honest node must have input Λ' that extends Λ .

5.2 Constructing Super-optimistic Atomic Broadcast

Using **SuperOGA**, we present a super-optimistic atomic broadcast protocol that fulfills such a requirement in Figure 3 and Algorithm 4.

Details of Super-optimistic Atomic Broadcast. The atomic broadcast protocol proceeds in views, with each view lasting 2Δ , except view 0, which lasts

Algorithm 4 Atomic broadcast for awake node $\mathcal{P}$ at time t : worst-case tolerance of $1/3$ faults at 3Δ latency, and best-case tolerance of no faults at 2Δ latency

- (i). In time $t = 0$ of view v (> 0), the received message $\langle \text{PROPOSE}, \Lambda_{v-1}, \text{VRF}_{\mathcal{P}}(v-1) \rangle_{\mathcal{P}}$ is considered invalid if any of the following conditions are met:
 - (a). Λ_{v-1} conflicts with the output log $\mathcal{L}$
 - (b). $\text{VRF}_{\mathcal{P}}(v-1)$ is an invalid VRF output on $v-1$
- (ii). If a node $\mathcal{P}$ sends two different messages of the form $\langle \text{PROPOSE}, \Lambda_v, \text{VRF}_{\mathcal{P}}(v) \rangle_{\mathcal{P}}$ and $\langle \text{PROPOSE}, \Lambda'_v, \text{VRF}_{\mathcal{P}}(v) \rangle_{\mathcal{P}}$ in the view v with $\Lambda_v \neq \Lambda'_v$, both messages are ignored
- (iii). If Λ_v extends Λ , then $\langle \text{PROPOSE}, \Lambda_v, \text{VRF}_{\mathcal{P}}(v) \rangle_{\mathcal{P}}$ is also viewed as a propose for Λ
- (iv). View 0 lasts Δ time and all later views $v \geq 1$ lasts 2Δ ; for $v \geq 1$, the time $t = 2\Delta$ of view v matches time $t = 0$ of view $v+1$

```

Initialize: log  $\mathcal{F} \leftarrow []$ ;  $\mathcal{L} \leftarrow []$ 
view  $v = 0$ :
1: if time  $t = 0$  then
2:   multicast  $\langle \text{PROPOSE}, \Lambda_0, \text{VRF}_{\mathcal{P}}(0) \rangle_{\mathcal{P}}$ , where  $\Lambda_0 := B_0$  ▷ propose: view 0
view  $v > 0$ :
3: if time  $t = 0$  then
4:   Let  $m$  be the number of received PROPOSE messages ▷ sent in view  $v-1$ 
5:   Start  $\text{SuperOGA}_v$  with input  $\Lambda_v$ , where  $\Lambda_v$  is the longest log in the received valid PROPOSE messages with the largest valid VRF on  $v-1$ 
6: if time  $t = \Delta$  then
7:   if  $\text{SuperOGA}_v$  outputs  $(\Lambda_{v,2}, 2)$  then ▷ all votes
8:     if  $|\mathcal{F}| < \Lambda_{v,2}$  then
9:        $\mathcal{F} \leftarrow \Lambda_{v,2}$ ; output  $\mathcal{F}$  ▷ super-optimistic output
10:  if  $\text{SuperOGA}_v$  outputs  $(\Lambda_{v,1}, 1)$  then ▷ 2/3 votes
11:    Start  $\text{GA}_v$  with input  $\Lambda_{v,1}$ 
12:    if  $\text{OptimisticGA}_v$  outputs  $(\Lambda_{v,0}, 0)$  then ▷ 1/3 votes
13:      if there are two such  $\Lambda_{v,0}$  then
14:        if the two such  $\Lambda_{v,0}$  both extend from  $\Lambda_{v,1}$  then
15:          pick one at random
16:        else
17:          select the one that extends  $\Lambda_{v,1}$  ▷ must exist
18:      Let  $\Lambda_v := \Lambda_{v,0} || B_k$ 
19:      multicast  $\langle \text{PROPOSE}, \Lambda_v, \text{VRF}_{\mathcal{P}}(v) \rangle_{\mathcal{P}}$  ▷ propose: view  $v$ 
20: if time  $t = 2\Delta$  then
21:   if  $\text{GA}_v$  outputs  $(\Lambda, 1)$  then
22:     if  $|\mathcal{F}| < \Lambda$  then
23:        $\mathcal{F} \leftarrow \Lambda$ ; output  $\mathcal{F}$  ▷ normal output
24:   if  $\text{GA}_v$  outputs  $(\Lambda', 0)$  then
25:      $\mathcal{L} \leftarrow \Lambda'$  ▷ update lock

```

Δ . Each step depends on the messages received from the immediate last round. Detailed executions are as follows:

Propose. If view $v = 0$, each awake honest node proposes a block B_0 by multicasting the message $\langle \text{PROPOSE}, \Lambda_0, \text{VRF}_{\mathcal{P}}(0) \rangle_{\mathcal{P}}$, where $\Lambda_0 = B_0$. For any view $v \geq 1$, at time $t = \Delta$, each awake honest node $\mathcal{P}$ receives a log from SuperOGA_v

with grade 1, denoted by $(\Lambda_{v,1}, 1)$, and one (or possibly more) with grade 0, denoted by $(\Lambda_{v,0}, 0)$. Select the $\Lambda_{v,0}$ that extends $\Lambda_{v,1}$ and propose a new block B_k after $\Lambda_{v,0}$. If more than one $\Lambda_{v,0}$ extend $\Lambda_{v,1}$, pick one randomly. Multicast the message $\langle \text{PROPOSE}, \Lambda_v, \text{VRF}_{\mathcal{P}}(v) \rangle_{\mathcal{P}}$, where $\Lambda_v := \Lambda_{v,0} \parallel B_k$.

Input to SuperOGA. At time $t = 0$ of view $v \geq 1$, node starts SuperOGA_v with input Λ_v , where Λ_v is the longest log extending $\mathcal{C}_v$ among the received valid PROPOSE messages, and has the largest valid VRF value for round $v - 1$.

Optimistic output under SuperOGA. At time $t = \Delta$ in view $v \geq 1$, if SuperOGA outputs $(\Lambda_{v,2}, 2)$ and the size of $\Lambda_{v,2}$ is greater than that of output log $\mathcal{F}$, then $\Lambda_{v,2}$ is output immediately.

Normal output under GA. At time $t = \Delta$ in view $v \geq 1$, each awake honest node $\mathcal{P}$ receives an output from SuperOGA_v with grade 1 and subsequently inputs this output into GA_v . Then, at time $t = 2\Delta$ in the same view, each awake honest node $\mathcal{P}$ outputs a log $(\Lambda, 1)$ from GA_v that is longer than the output log $\mathcal{F}$. The node $\mathcal{P}$ updates its lock $\mathcal{L}$ using the grade 0 output from GA_v .

5.3 Security Analysis of Super-optimistic Atomic Broadcast

Proof Sketch. The overall structure of the proof follows the same approach as in the previous section. Here we highlight the key differences and explain why security continues to hold.

Safety. It follows from the *uniqueness* and *graded consistency* properties of SuperOGA and the *uniqueness* and *integrity* properties of GA . If two honest nodes output in the same view, then conflicting outputs are ruled out by the *uniqueness* guarantees of the underlying SuperOGA or GA . If outputs occur at different times, *graded consistency* ensures that later outputs must extend earlier ones, while the locking mechanism prevents honest nodes from accepting proposals that conflict with their current $\mathcal{L}$. As a result, all logs output by honest nodes are mutually compatible.

Liveness. It is established by showing that values proposed by honest nodes are eventually decided with probability $1/3$ in each view, as an honest node becomes leader with probability at least $2/3$, and with probability at least $1/2$ it extends the common grade-0 log (see Lemma 6). In particular, with probability $1/3$, an honest node becomes the leader and proposes a log extending the current $\mathcal{L}$ of all honest nodes. Such a proposal is accepted by all honest nodes and decided in the subsequent view. Repeating this process over views implies that every value proposed by an honest node is eventually included in an output log.

Lemma 5 (Safety). *If an awake honest node $\mathcal{P}_i$ outputs a log Λ_i at time t , and another awake honest node $\mathcal{P}_j$ outputs a log Λ_j at time t' , then Λ_i and Λ_j must be compatible.*

Proof. We first prove that if two awake honest nodes output at the same time, then their outputs must be compatible. Suppose node $\mathcal{P}_i$ outputs Λ_i at time $t = t' = \Delta$ in view $v \geq 1$, $\mathcal{P}_i$ must have received $(\Lambda_i, 2)$ from SuperOGA_v . By the

uniqueness property of **SuperOGA**, no other node $\mathcal{P}_j$ can output a conflicting log with grade $g \geq 0$. This establishes the lemma for outputs at $t = \Delta$. Similarly, if node $\mathcal{P}_i$ outputs Λ_i at time $t = t' = 2\Delta$ in the same view $v \geq 1$, it must have received $(\Lambda_i, 1)$ from $\mathbf{GA}_v$. Again, by the *uniqueness* property of **GA**, no conflicting log with grade ≥ 1 can be output by another node, and the lemma follows.

Secondly, we prove that if two awake honest nodes output at different times, their outputs must still be compatible. Let us consider the case where $t' \geq t + \Delta$.

1. **Case 1:** $t = \Delta$. Suppose node $\mathcal{P}_i$ outputs log Λ_i at time $t = \Delta$ in view $v \geq 1$ by receiving output $(\Lambda_i, 2)$ from **SuperOGA** _{v} . By the *graded consistency* property of **SuperOGA**, any awake honest node $\mathcal{P}_j$ must output a log Λ'_j that extends Λ_i with grade 1. Then $\mathcal{P}_j$ inputs Λ'_j to $\mathbf{GA}_v$ at time $t = \Delta$. By the *validity* property of **GA**, all awake honest nodes output a log that extends Λ_i with grade 1 at time $t = 2\Delta$, and decide this log, denoted by Λ_{2v} and $\Lambda_{2v} \succeq \Lambda_i$.
2. **Case 2:** $t = 2\Delta$. If an awake honest node $\mathcal{P}_i$ outputs a log Λ_i at time $t = 2\Delta$ of view $v \geq 1$ (which corresponds to time $t = 0$ of view $v + 1$) by outputting $(\Lambda_i, 1)$ from $\mathbf{GA}_v$, then, according to Algorithm 4, the input to $\mathbf{GA}_v$ is the output of **SuperOGA** _{v} with grade 1. By the *uniqueness* property of **SuperOGA**, the inputs provided by awake honest nodes to $\mathbf{GA}_v$ are non-conflicting. Moreover, by the *integrity* property of **GA**, any awake honest node $\mathcal{P}_j$ will output $(\Lambda_j, 0)$ such that $\Lambda_j \succeq \Lambda_i$, and simultaneously update its lock $\mathcal{L}$ to Λ_j . Since all PROPOSE messages that conflict with the current lock $\mathcal{L}$ are treated as invalid, all awake honest nodes will henceforth only output logs that extend Λ_j in future views. Therefore, $\mathcal{P}_j$ must eventually output a log that extends Λ_i .

Lemma 6 (liveness). *If an awake honest node inputs a message m at time t , then there must exist a time $t' \geq t$, such that all awake honest nodes output a log that includes m by time t' .*

Proof. Suppose an awake honest node $\mathcal{P}$ has the maximum VRF among all awake nodes, we refer to $\mathcal{P}$ as the honest leader in view $v \geq 1$.

Let $\mathcal{L} = \Lambda'$ denote the longest locked logs among all honest nodes at time $t = 2\Delta$ of view $v \geq 1$, then the locked log Λ' must have been output by at least one honest node from $\mathbf{GA}_v$. By the *integrity* property of **GA**, it implies that some honest node must have input a log $\Lambda'' \succeq \Lambda'$ to $\mathbf{GA}_v$ after receiving output $(\Lambda'', 1)$ from **SuperOGA** _{v} .

By the *graded consistency* property of **SuperOGA**, every honest node must have output a log $(\Lambda''', 0)$ such that $\Lambda''' \succeq \Lambda''$. Due to the *bounded divergence* property of **SuperOGA**, there can be at most two conflicting logs with grade 0 output from **SuperOGA** _{v} , one of which is Λ''' . Therefore, the probability that an honest leader proposes a new log extending Λ''' , and consequently extending the locked log $\mathcal{L}$, is at least $1/2$.

As $\mathcal{L} = \Lambda'$ is the longest locked log among all honest nodes, the honest leader's proposal extends all honest nodes' locked log, and will be accepted by all honest nodes at $t = 0$ of view $v + 1$. Since more than $2/3$ awake nodes are

honest at all times, the probability that an honest node has the maximum VRF and becomes the leader is $2/3$. Therefore, the probability of a proposal getting decided in the next view is $1/3$.

In summary, in each view, there is a $1/3$ probability that all awake honest nodes select the input Λ from the honest leader to **SuperOGA**, where Λ extends the current locks of all honest nodes. This also implies that, with probability $1/3$, the honest leader's input will be output in view v . It follows that any value proposed by an honest node to the atomic broadcast protocol will eventually appear in an output log, thereby ensuring the liveness property.

Theorem 3. *Algorithm 4 implements an atomic broadcast protocol in the sleepy model, achieving an optimistic latency of 2Δ and a latency of 3Δ under $1/3$ corruptions.*

Proof. Lemma 5 and Lemma 6 complete the proof.

6 Optimistic Atomic Broadcast

In this section, we present a sleepy consensus protocol that achieves a latency of 2Δ when fewer than $1/6$ of the online nodes are Byzantine, and 3Δ latency when the Byzantine fraction is below $1/3$. Notably, the protocol does not require view-change or view-synchronization mechanisms, even when $f > 1/6$. All operations are performed on a single chain, and we seamlessly incorporate a two-level confirmation rule to ensure both high performance and robust security.

We introduce an Optimistic Graded Agreement (**OptimisticGA**) protocol, extending the standard **GA** [26] with additional optimistic outputs. Our atomic broadcast integrates **OptimisticGA** with **GA** in the same three-phase structure as $1/3$ -MMR [26], replacing the first voting phase with **OptimisticGA** to enable immediate confirmation when optimistic thresholds are met.

While **GA** uses two grades with thresholds $1/3$ and $2/3$, **OptimisticGA** introduces four grades with thresholds $1/2$, $2/3$, and $5/6$, allowing early confirmation while preserving compatibility with subsequent **GA** decisions.

6.1 Preparation: Lifting GA Stronger

One key idea in constructing an optimistic atomic broadcast is to achieve output in the minimal number of rounds when optimistic conditions are met. To this end, we aim to extend the standard **GA** protocol into an **OptimisticGA**, which can be formalized as follows.

Optimistic Graded Agreement (OptimisticGA). The **OptimisticGA** is executed among a set of n nodes, with at most $1/3$ of them being corrupted. Each node begins with an input log Λ by broadcasting a signed message $\langle \text{VOTE}, \Lambda \rangle$ to all participants. The protocol guarantees that each node outputs a log along with a grade, denoted by $(\Lambda, *)$, where $*$ $\in \{0, 1, 2, 3\}$. The **OptimisticGA** protocol retains several key properties of the original **GA** protocol introduced in the

preliminaries section, including *Bounded divergence* and *Integrity*. However, we revise the definitions of *Graded Consistency*, *Validity* and *Uniqueness* to better suit the optimistic scenario we consider, without compromising safety in the general case. Formally, **OptimisticGA** satisfies the following properties:

1. *Bounded divergence, Integrity*: The same as **GA**.
2. *Graded consistency*: If an awake honest node $\mathcal{P}_i$ outputs (Λ_i, g) for $g = 3$, then any awake honest node $\mathcal{P}_j$ must output a Λ'_j with grade 1 such that $\Lambda'_j \succeq \Lambda_i$, and a Λ''_j with grade 0 such that $\Lambda''_j \succeq \Lambda'_j$. Additionally, if an awake honest node outputs (Λ, g) for $g = 2$, then all awake honest nodes must output a Λ' with grade 0 that extends Λ .
3. *Validity*: If all awake honest nodes input a log that extends Λ , then all awake honest nodes output (Λ, g) for $g \geq 2$.
4. *Uniqueness*: If any awake honest node $\mathcal{P}$ output $(\Lambda, 3)$, then no awake honest nodes outputs a conflicting (Λ', g) for $g \geq 1$. Additionally, if any awake honest node $\mathcal{P}$ output $(\Lambda, 2)$, then no awake honest nodes outputs a conflicting (Λ', g) for $g \geq 2$.

Details and security of the OptimisticGA. Our **OptimisticGA** is shown in Algorithm 5 which achieves consensus in a single communication round. At time $t = 0$, each awake node multicasts a VOTE message to support its own input log Λ . By time $t = \Delta$, nodes collect the VOTE messages they have received and determine their outputs. Importantly, due to the sleepy model and the presence of Byzantine nodes, nodes lack global knowledge of participation and may receive votes from different subsets of nodes. Consequently, each node makes decisions based on its own local view of participation. In our protocol, if Λ' extends Λ , then (VOTE, Λ') is also counted as a vote for Λ . For any honest awake node, if a log Λ is supported by more than $5/6$, $2/3$, $1/2$, or $1/3$ of the received VOTE messages, then the node outputs Λ with grade 3, 2, 1, or 0, respectively. We have the following theorem:

Theorem 4. *Algorithm 5 implements an optimistic graded agreement protocol.*

Proof. Assume honest nodes $\mathcal{P}_i$ and $\mathcal{P}_j$ are awake at time $t = \Delta$, and let $\mathcal{M}_i$ and $\mathcal{M}_j$ denote the sets of messages they have received since time $t = 0$. According to the model assumption, at time $t = 0$, the number of awake honest nodes satisfies $|H_r| > 2|O_r|/3$, where $|O_r| = |H_r| + |B_r|$. This implies that $|H_r| > 2|B_r|$. We now prove each property separately.

1. **Graded consistency.** Suppose that $\mathcal{P}_i$ outputs Λ_i with grade 3. Then, by definition, $\mathcal{P}_i$ must have received more than $5|\mathcal{M}_i|/6$ votes for Λ_i by time $t = \Delta$. Since honest nodes multicast their votes to all participants, then we have $\mathcal{M}_i = H_r \cup B_{r,i}$ and $\mathcal{M}_j = H_r \cup B_{r,j}$, where $B_{r,i}$ and $B_{r,j}$ are the sets of Byzantine messages received by $\mathcal{P}_i$ and $\mathcal{P}_j$, respectively. Given the model assumption that $|H_r| > 2|B_r|$, we have $|H_r| > 2|B_{r,i}|$ and $|H_r| > 2|B_{r,j}|$. It follows that: $|H_r| > |B_{r,i}| + |B_{r,j}|$. Then the number of honest nodes in $\mathcal{M}_i$

Algorithm 5 Optimistic Graded Agreement (**OptimisticGA**) for awake node $\mathcal{P}$ at time t :

- (i). If Λ' extends Λ , then $\langle \text{VOTE}, \Lambda' \rangle$ is also counted as a vote for Λ
- (ii). If a node sends different votes, all of its votes are ignored. Votes from the same node are counted at most once.

```

1: if  $t = 0$  then
2:   multicast  $\langle \text{VOTE}, \Lambda \rangle_{\mathcal{P}}$ , where  $\Lambda$  is the input log
3: if  $t = \Delta$  then
4:   Let  $m$  be the number of received VOTE messages
5:   if the longest log  $\Lambda$  voted by  $> 5m/6$  nodes then
6:     output  $(\Lambda, 3)$  ▷ optimistic output
7:   if the longest log  $\Lambda$  voted by  $> 2m/3$  nodes then
8:     output  $(\Lambda, 2)$ 
9:   if the longest log  $\Lambda$  voted by  $> m/2$  nodes then
10:    output  $(\Lambda, 1)$  ▷ optimistic candidate
11:   if the longest log  $\Lambda$  voted by  $> m/3$  nodes then
12:    output  $(\Lambda, 0)$ 

```

that voted for Λ_i is at least:

$$\begin{aligned}
\frac{5}{6}|\mathcal{M}_i| - |B_{r,i}| &= \frac{5}{6}(|H_r| + |B_{r,i}|) - |B_{r,i}| = \frac{4}{6}|H_r| + \frac{1}{6}(|H_r| - |B_{r,i}|) \\
&> \frac{2}{3}|H_r| + \frac{1}{6}|B_{r,j}| > \frac{1}{2}|H_r| + \frac{1}{6}|H_r| + \frac{1}{6}|B_{r,j}| \\
&> \frac{1}{3}|Q_r| + \frac{1}{6}|\mathcal{M}_j| > \frac{1}{3}|\mathcal{M}_j| + \frac{1}{6}|\mathcal{M}_j| = \frac{1}{2}|\mathcal{M}_j|.
\end{aligned}$$

Therefore, $\mathcal{P}_j$ must output a $(\Lambda'_j, 1)$ such that $\Lambda'_j \succeq \Lambda$. Furthermore, it is evident that $\mathcal{P}_j$ must also output a $(\Lambda''_j, 0)$ with $\Lambda''_j \succeq \Lambda$. Similarly, if $\mathcal{P}_i$ outputs Λ for grade 2, then it must have received more than $2|\mathcal{M}_i|/3$ votes for Λ . The number of honest nodes who voted for Λ is at least:

$$\begin{aligned}
\frac{2}{3}|\mathcal{M}_i| - |B_{r,i}| &= \frac{2}{3}(|H_r| + |B_{r,i}|) - |B_{r,i}| = \frac{1}{3}|H_r| + \frac{1}{3}(|H_r| - |B_{r,i}|) \\
&> \frac{1}{3}|H_r| + \frac{1}{3}|B_{r,j}| = \frac{1}{3}(|H_r| + |B_{r,j}|) = \frac{1}{3}|\mathcal{M}_j|.
\end{aligned}$$

Thus $\mathcal{P}_j$ must output a $(\Lambda', 0)$ s.t. $\Lambda' \succeq \Lambda$.

2. **Validity.** If all awake honest nodes input a log that extends Λ , it also implies that more than $2|O_r|/3$ nodes vote for a log extending Λ at time $t = 0$. Let $\mathcal{M}_i$ denote the set of messages received by an honest node $\mathcal{P}_i$ at time $t = \Delta$. Since $|\mathcal{M}_i| \leq |O_r|$, it follows that $\mathcal{P}_i$ receives more than $2/3|\mathcal{M}_i|$ votes for a log extending Λ . Therefore, all awake honest nodes will output Λ with at least grade 2.
3. **Uniqueness.** Due to graded consistency, if an awake honest node $\mathcal{P}_i$ outputs $(\Lambda, 3)$, then every other awake honest node $\mathcal{P}_j$ must output a log $\Lambda' \succeq \Lambda$ with at least grade 1. This implies that $\mathcal{P}_j$ must have observed more than

$|\mathcal{M}_j|/2$ votes for Λ in its received message set $\mathcal{M}_j$, and cannot have observed more than $1/2|\mathcal{M}_j|$ votes for any log that conflicts with Λ . Therefore, if any honest node $\mathcal{P}_j$ outputs a log Λ' with grade $g \geq 1$, then Λ' must be compatible with Λ . Moreover, if two honest nodes $\mathcal{P}_i$ and $\mathcal{P}_j$ output $(\Lambda, 3)$ and $(\Lambda', 3)$, respectively, then Λ and Λ' must be compatible, as both must have received more than $5/6$ of votes for their respective logs, and such high overlap among honest votes ensures consistency. Additionally, if an honest node $\mathcal{P}_i$ outputs $(\Lambda, 2)$, then by graded consistency, every other honest node $\mathcal{P}_j$ must output a log that extends Λ with at least grade 0. That is, $\mathcal{P}_j$ must have observed more than $|\mathcal{M}_j|/3$ votes for a log extending Λ . Consequently, no conflicting log can receive more than $2|\mathcal{M}_j|/3$ votes. Therefore, no honest node can output a conflicting log with grade $g \geq 2$.

4. **Bounded divergence.** Since a grade 0 output requires support from more than $1/3$ of the received votes, and each VOTE message can support only one log, at most two conflicting logs can each receive more than $1/3$ of disjoint messages from the same VOTE message set. Therefore, each node can output at most two conflicting logs.
5. **Integrity.** Suppose no awake honest node has input a log that extends Λ . Consider any awake honest node $\mathcal{P}_i$ that outputs $(\Lambda, *)$. Since $|H_r| > 2|O_r|/3 > 2|\mathcal{M}_i|/3$, and the set of VOTE messages received by $\mathcal{P}_i$ is $\mathcal{M}_i$, it follows that the number of nodes who could have voted for a log extending Λ must be less than $|\mathcal{M}_i|/3$. This contradicts the assumption that $\mathcal{P}_i$ outputs $(\Lambda, *)$, which requires at least $|\mathcal{M}_i|/3$ votes. So, if an awake honest node outputs $(\Lambda, *)$, then at least one awake honest node must have input a log Λ' that extends Λ .

6.2 Constructing Optimistic Atomic Broadcast

In this section, we provide a detailed description of our atomic broadcast protocol, which includes an optimistic case for faster output. The protocol is presented in Algorithm 6.

As illustrated in Figure 4, our optimistic atomic broadcast follows the same three-phase structure as 1/3-MMR [26]: an initial propose phase, followed by two consecutive voting phases. After the first vote, a new propose step is triggered based on its outcome; notably, this new proposal is broadcast concurrently with the second voting phase for the original proposal.

The GA protocol in [26] employs a two-grade output rule with grade 1 at the $2/3$ threshold and grade 0 at the $1/3$ threshold. In contrast, our **OptimisticGA** defines a four-grade output rule (grades 0 through 3) by adding two new decision thresholds: $1/2$ for grade 1 and $5/6$ for grade 3. The $2/3$ threshold is assigned to grade 2 in **OptimisticGA**, corresponding to grade 1 in **GA**. These additional thresholds enable an early, fast confirmation step, combined with careful design to preserve consistency with any **GA** output in subsequent rounds.

Details of Optimistic Atomic Broadcast. The optimistic atomic broadcast protocol operates in a sequence of views, with each view comprising two rounds,

Algorithm 6 Atomic broadcast for awake node $\mathcal{P}$ at time t : worst-case tolerance of $1/3$ faults at 3Δ latency, and good-case tolerance of $1/6$ faults at 2Δ latency

- (i). In time $t = 0$ of view v (> 0), the received message $\langle \text{PROPOSE}, \Lambda_{v-1}, \text{VRF}_{\mathcal{P}}(v-1) \rangle_{\mathcal{P}}$ is considered invalid if any of the following conditions are met:
 - (a). Λ_{v-1} conflicts with the output log $\mathcal{L}$
 - (b). $\text{VRF}_{\mathcal{P}}(v-1)$ is an invalid VRF output on $v-1$
- (ii). If a node $\mathcal{P}$ sends two different messages of the form $\langle \text{PROPOSE}, \Lambda_v, \text{VRF}_{\mathcal{P}}(v) \rangle_{\mathcal{P}}$ and $\langle \text{PROPOSE}, \Lambda'_v, \text{VRF}_{\mathcal{P}}(v) \rangle_{\mathcal{P}}$ in the view v with $\Lambda_v \neq \Lambda'_v$, both messages are ignored
- (iii). If Λ_v extends Λ , then $\langle \text{PROPOSE}, \Lambda_v, \text{VRF}_{\mathcal{P}}(v) \rangle_{\mathcal{P}}$ is also viewed as a propose for Λ
- (iv). View 0 lasts Δ time and all later views $v \geq 1$ lasts 2Δ ; for $v \geq 1$, the time $t = 2\Delta$ of view v matches time $t = 0$ of view $v+1$

Initialize: $\log \mathcal{F} \leftarrow []$; $\mathcal{L} \leftarrow []$

view $v = 0$:

- 1: **if** time $t = 0$ **then**
- 2: **multicast** $\langle \text{PROPOSE}, \Lambda_0, \text{VRF}_{\mathcal{P}}(0) \rangle_{\mathcal{P}}$, where $\Lambda_0 := B_0$ ▷ propose: view 0

view $v > 0$:

- 3: **if** time $t = 0$ **then**
- 4: Let m be the number of received PROPOSE messages ▷ sent in view $v-1$
- 5: Set $\mathcal{C}_v$ to the longest log that is proposed by $> 2m/3$ nodes
- 6: Start OptimisticGA_v with input Λ_v , where Λ_v is the longest log in the received valid PROPOSE messages that extends $\mathcal{C}_v$ with the largest valid VRF on $v-1$
- 7: **if** time $t = \Delta$ **then**
- 8: **if** OptimisticGA_v outputs $(\Lambda_{v,3}, 3)$ **then** ▷ 5/6 votes
- 9: **if** $|\mathcal{F}| < \Lambda_{v,3}$ **then**
- 10: $\mathcal{F} \leftarrow \Lambda_{v,3}$; **output** $\mathcal{F}$ ▷ optimistic output
- 11: **if** OptimisticGA_v outputs $(\Lambda_{v,2}, 2)$ **then** ▷ 2/3 votes
- 12: Start GA_v with input $\Lambda_{v,2}$
- 13: **if** OptimisticGA_v outputs $(\Lambda_{v,1}, 1)$ **then** ▷ 1/2 votes
- 14: **if** OptimisticGA_v outputs $(\Lambda_{v,0}, 0)$ **then** ▷ 1/3 votes
- 15: **if** there are two such $\Lambda_{v,0}$ **then**
- 16: **if** the two such $\Lambda_{v,0}$ both extend from $\Lambda_{v,1}$ **then**
- 17: pick one at random
- 18: **else**
- 19: select the one that extends $\Lambda_{v,1}$ ▷ must exist
- 20: Let $\Lambda_v := \Lambda_{v,0} || B_k$
- 21: **multicast** $\langle \text{PROPOSE}, \Lambda_v, \text{VRF}_{\mathcal{P}}(v) \rangle_{\mathcal{P}}$ ▷ propose: view v
- 22: **if** time $t = 2\Delta$ **then**
- 23: **if** GA_v outputs $(\Lambda, 1)$ **then**
- 24: **if** $|\mathcal{F}| < \Lambda$ **then**
- 25: $\mathcal{F} \leftarrow \Lambda$; **output** $\mathcal{F}$ ▷ normal output
- 26: **if** GA_v outputs $(\Lambda', 0)$ **then**
- 27: $\mathcal{L} \leftarrow \Lambda'$ ▷ update lock

except for the initial view $v = 0$, which consists of only one round. Each round lasts for a duration of Δ time and is divided into two phases: a send phase and a

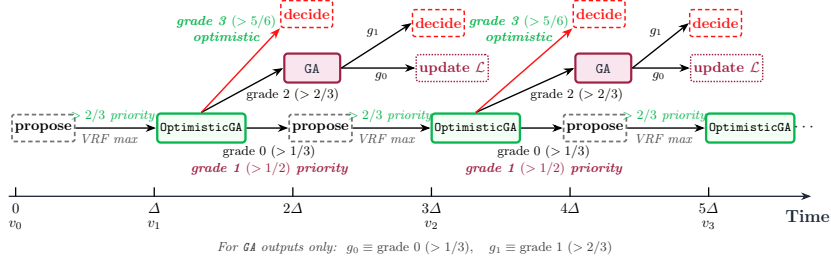


Fig. 4. Overview of Our Optimistic Atomic Broadcast Protocol.

receive phase. The send phase of a given round overlaps with the receive phase of the immediately preceding round. Likewise, the time $t = 2\Delta$ at the end of view v aligns exactly with $t = 0$ at the start of view $v + 1$. For simplicity, Algorithm 6 uses local time parameters defined within each view. Specifically, the time $t = 0$ of view $v > 0$ corresponds to $t = 2(v - 1)\Delta + \Delta$ on the overall timeline. The execution of the optimistic atomic broadcast protocol is as follows:

Propose. If view $v = 0$, each awake honest node proposes a block B_0 by multicasting the message $\langle \text{PROPOSE}, \Lambda_0, \text{VRF}_{\mathcal{P}}(0) \rangle_{\mathcal{P}}$, where $\Lambda_0 = B_0$. For any view $v \geq 1$, at time $t = \Delta$, each awake honest node $\mathcal{P}$ receives two outputs from OptimisticGA_v : one with grade 1, denoted by $(\Lambda_{v,1}, 1)$, and one (or possibly more) with grade 0, denoted by $(\Lambda_{v,0}, 0)$. Among the grade 0 outputs, the node selects a $\Lambda_{v,0}$ that extends $\Lambda_{v,1}$. If multiple such candidates exist, one is selected at random. Then, node $\mathcal{P}$ proposes a new block B_k extending the selected $\Lambda_{v,0}$, and multicasts the message $\langle \text{PROPOSE}, \Lambda_v, \text{VRF}_{\mathcal{P}}(v) \rangle_{\mathcal{P}}$, where $\Lambda_v := \Lambda_{v,0} \parallel B_k$.

Input to OptimisticGA. At time $t = 0$ of view $v \geq 1$, set $\mathcal{C}_v$ to the longest log that is proposed by $> 2/3m$ nodes. Then start OptimisticGA_v with input Λ_v , where Λ_v is the longest log in the received valid PROPOSE messages that extends $\mathcal{C}_v$ with the largest valid VRF on $v - 1$.

Optimistic output under OptimisticGA. At time $t = \Delta$ in view $v \geq 1$, if the OptimisticGA outputs $(\Lambda_{v,3}, 3)$ and the size of $\Lambda_{v,3}$ is greater than that of $\mathcal{F}$, then $\Lambda_{v,3}$ is output immediately.

Normal output under GA. At time $t = \Delta$ in view $v \geq 1$, each awake honest node $\mathcal{P}$ receives an output from OptimisticGA_v with grade 2 and subsequently inputs this output into GA_v . Then, at time $t = 2\Delta$ in the same view, each awake honest node $\mathcal{P}$ outputs a log Λ of grade 1 that is longer than $\mathcal{F}$. The node $\mathcal{P}$ updates its lock $\mathcal{L}$ using the grade 0 output from GA_v .

6.3 Security Analysis of Optimistic Atomic Broadcast

Proof Sketch. The overall structure of the proof follows the same approach as in Section 5. Here we highlight the key differences and explain why security continues to hold.

Safety. The stronger graded agreement instance OptimisticGA provides four-grade outputs and *bounded divergence*, which ensures that any log output early in a view constrains all subsequent proposals. Combined with the *uniqueness* and

integrity properties of **GA** and the locking mechanism, it is guaranteed that once an honest node outputs a log, all future outputs by honest nodes must extend it. Additionally, prioritizing Λ_v that extends $\mathcal{C}_v$ ensures later input is compatible with the optimistic output. Hence, all honest outputs remain compatible.

Liveness. The main difference with the super-optimistic protocol lies in the leader selection and the additional candidate log $\mathcal{C}_v$. In each view v , with probability $1/6$ (as the leader aligns with the common candidate log with probability at least $1/3$, and extends the priority lock with probability at least $1/2$; see Lemma 8 for details), an honest leader is selected whose proposal extends both the longest locked log $\mathcal{L}$ and the $\mathcal{C}_v$ held by honest nodes. Such a proposal is accepted by all honest nodes and is decided in the subsequent view. Repeating this process over views implies that every value proposed by an honest node is eventually included in an output log.

Lemma 7 (Safety). *If an awake honest node $\mathcal{P}_i$ outputs a log Λ_i at time t , and another awake honest node $\mathcal{P}_j$ outputs a log Λ_j at time t' , then Λ_i and Λ_j must be compatible.*

Proof. We first prove that if two awake honest nodes output at the same time, then their outputs must be compatible. Suppose the time is $t = t' = \Delta$ in some view $v \geq 1$, and node $\mathcal{P}_i$ outputs Λ_i at time t . This implies that $\mathcal{P}_i$ must have received $(\Lambda_i, 3)$ from **OptimisticGA** _{v} . By the *uniqueness* property of **OptimisticGA**, no other node $\mathcal{P}_j$ can output a conflicting log with grade $g \geq 1$. This establishes the lemma for outputs at $t = \Delta$. Similarly, if node $\mathcal{P}_i$ outputs Λ_i at time $t = t' = 2\Delta$ in the same view $v \geq 1$, it must have received $(\Lambda_i, 1)$ from **GA** _{v} . Again, by the *uniqueness* property of **GA**, no conflicting log with grade ≥ 1 can be output by another node, and the lemma follows.

Secondly, we prove that if two awake honest nodes output at different times, their outputs must still be compatible. Let us consider the case where $t' \geq t + \Delta$.

Case 1: $t = \Delta$. Suppose node $\mathcal{P}_i$ outputs log Λ_i at time $t = \Delta$ in view $v \geq 1$ by receiving output $(\Lambda_i, 3)$ from **OptimisticGA** _{v} . By the *graded consistency* property of **OptimisticGA**, any node $\mathcal{P}_j$ must output a log Λ'_j that extends Λ_i with grade 1, and further, a log extending Λ'_j with grade 0, both at time $t = \Delta$ in view v . According to Algorithm 6, every awake honest node $\mathcal{P}_j$ must multicast a PROPOSE message carrying a log Λ_v that extends Λ'_j at time $t = \Delta$ of view v . As a result, by the *uniqueness* property of **OptimisticGA**, no awake and honest node $\mathcal{P}_j$ can output a conflicting log with grade 2 from **OptimisticGA** _{v} and input it to **GA** _{v} . Additionally, if any awake and honest node outputs $(\Lambda', *)$ from **GA** _{v} , then by the *integrity* property, at least one awake and honest node must have input a log Λ'' that extends Λ' . Consequently, no honest node can output a conflicting log from **GA** _{v} at time $t = 2\Delta$ in view $v \geq 1$. This also means that the updated lock $\mathcal{L}$ is compatible with the PROPOSE message multicast at time $t = \Delta$, which further implies that any PROPOSE message sent by an awake honest node is valid. Furthermore, by time $t = 0$ of view $v + 1$, any awake and honest node $\mathcal{P}_j$ must have received PROPOSE messages from at least $2/3$ of the nodes, each carrying a log that extends Λ'_j . Therefore, $\mathcal{P}_j$ must input to **OptimisticGA** _{$v+1$} a

log that extends Λ'_j . By the *validity* property of **OptimisticGA**, all honest nodes will output (Λ'_j, g) with $g \geq 2$ from **OptimisticGA** $_{v+1}$, and subsequently input Λ'_j to **GA** $_{v+1}$. Due to the *validity* of **GA**, all honest nodes will then output $(\Lambda'_j, 1)$ from **GA** $_{v+1}$. Therefore, any awake and honest node $\mathcal{P}_j$ must eventually output a log that extends Λ_i .

Case 2: $t = 2\Delta$. If an awake honest node $\mathcal{P}_i$ outputs log Λ_i at time $t = 2\Delta$ of view $v \geq 1$ (which corresponds to time $t = 0$ of view $v + 1$) by outputting $(\Lambda_i, 1)$ from **GA** $_v$, then by the *graded consistency* property of **GA**, any awake and honest node $\mathcal{P}_j$ will output $(\Lambda_j, 0)$ such that $\Lambda_j \succeq \Lambda_i$, and at the same time update its lock $\mathcal{L}$ to Λ_j . Since all PROPOSE messages that conflict with the current lock $\mathcal{L}$ are treated as invalid, all awake and honest nodes will henceforth only output logs that extend Λ_j in future views. Therefore, $\mathcal{P}_j$ must eventually output a log that extends Λ_i .

Lemma 8 (liveness). *If an awake honest node inputs a message m at time t , then there must exist a time $t' \geq t$, such that all awake honest nodes output a log that includes m by time t' .*

Proof. Suppose an awake honest node $\mathcal{P}$ has the maximum VRF among all awake nodes, we refer to $\mathcal{P}$ as the honest leader in view $v \geq 1$. To prove the lemma, we establish the following two claims:

Claim 1: Let $\mathcal{L} = \Lambda'$ denote the longest locked log among all honest nodes at time $t = 2\Delta$ of view $v \geq 1$, then the locked log Λ' must have been output by at least one honest node from **GA** $_v$. By the *integrity* property of **GA**, it implies that some honest node must have input a log $\Lambda'' \succeq \Lambda'$ to **GA** $_v$ after receiving output $(\Lambda'', 2)$ from **OptimisticGA** $_v$.

By the *graded consistency* property of **OptimisticGA**, every honest node must have output a log $(\Lambda''', 0)$ such that $\Lambda''' \succeq \Lambda''$. Due to the *bounded divergence* property of **OptimisticGA**, there can be at most two conflicting logs with grade 0 output from **OptimisticGA** $_v$, one of which is Λ''' . Therefore, the probability that an honest leader proposes a new log extending Λ''' , and consequently extending the locked log $\mathcal{L}$, is at least $1/2$.

Claim 2: According to Algorithm 6, each honest node's $\mathcal{C}_v$ corresponds to the longest log that has been proposed by more than $1/3$ of the awake honest nodes. Since the adversary can control at most $1/3$ of the nodes, and each awake honest node always multicasts the same PROPOSE message to all other nodes. Hence, at most two such logs can exist among the awake honest nodes, denoted $\mathcal{C}_v$ and $\mathcal{C}'_v$, and these two logs must be compatible with each other. Without loss of generality, suppose $|\mathcal{C}_v| \geq |\mathcal{C}'_v|$.

If $|\mathcal{C}'_v| < |\mathcal{C}_v|$, then $\mathcal{C}'_v$ must have been extended by all PROPOSE messages from the awake honest nodes, since the intersection of any two subsets of more than $2/3$ of the nodes must contain at least one same awake honest node. In the worst case, one log $\mathcal{C}_v$ could be extended by exactly $2/3$ of the nodes (including at least $1/3$ of the awake honest nodes, denoted as AH_1), while the other log $\mathcal{C}'_v$ could be extended by all awake honest nodes (denoted as AH_2). If the node with the maximum VRF among all awake nodes belongs to AH_1 , then all honest nodes

will input the same log Λ to **OptimisticGA**, carried in a **PROPOSE** message sent by that node. Since Λ extends $\mathcal{C}_v$, it will be used as the input to **OptimisticGA**, and in the worst case, will be output by $\mathbf{GA}_v$. The probability that the node with the maximum VRF lies in AH_1 is $1/3$. On the other hand, if $\mathcal{C}_v = \mathcal{C}'_v$, then all awake honest nodes have the same input to **OptimisticGA**, carried by a **PROPOSE** message from one awake honest node. In this case, the probability that the node with the maximum VRF is honest is $2/3$. Therefore, in either case, the probability that all awake honest nodes receive the same input originating from an awake honest node and used as input to **OptimisticGA** is at least $1/3$.

In summary, in each view, there is a $1/6$ probability that all awake honest nodes select the input Λ from the honest leader to **OptimisticGA**, where Λ extends both the current locks of all honest nodes and their respective $\mathcal{C}_v$ logs. This also implies that, with probability $1/6$, the honest leader's input will be output in view v . It follows that any value proposed by an honest node to the atomic broadcast protocol will eventually appear in an output log, thereby ensuring the liveness property.

Theorem 5. *Algorithm 6 implements an atomic broadcast protocol in the sleepy model, achieving an optimistic latency of 2Δ and a latency of 3Δ under $1/3$ corruptions.*

Proof. Lemma 7 and Lemma 8 complete the proof.

7 Conclusion

We study atomic broadcast in the sleepy model without assuming static corruption or large backward windows. Our protocols tolerate monotonic adversarial growth with only the minimal Δ -bounded backward window required by synchrony. Moreover, we present the first sleepy atomic broadcast protocol with a faster output path, enabling improved latency under favorable conditions while preserving safety and liveness. Together, these results clarify the trade-offs among round complexity, dynamic participation, and adversarial growth in sleepy consensus.

Acknowledgments. We thank our anonymous reviewers for their constructive comments. This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority. The work of Qiang and Xinrui was supported in part by ARC Discovery Project (DP250101739), Research Awards from the Stellar Development Foundation, and Input Output (iog.io) as part of the Cardano Vision 2026 research initiative. The work of Zhenliang was supported in part by a grant from City University of Hong Kong (Project No. 9610761).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bagaria, V., Kannan, S., Tse, D., Fanti, G., Viswanath, P.: Prism: Deconstructing the blockchain to approach physical limits. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security. pp. 585–602 (2019)
2. Ben-Or, M., Dolev, D., Hoch, E.N.: Brief announcement: Simple gradecast based algorithms. In: International Symposium on Distributed Computing. pp. 194–197. Springer (2010)
3. Bentov, I., Pass, R., Shi, E.: Snow white: Provably secure proofs of stake. IACR Cryptol. ePrint Arch. **2016**(919) (2016)
4. Blum, E., Katz, J., Loss, J.: Synchronous consensus with optimal asynchronous fallback guarantees. In: Theory of Cryptography Conference. pp. 131–150. Springer (2019)
5. Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation. vol. 99, pp. 173–186. The USENIX Association (1999)
6. Cristian, F., Aghili, H., Strong, R., Dolev, D.: Atomic broadcast: From simple message diffusion to byzantine agreement. *Information and Computation* **118**(1), 158–179 (1995)
7. D’Amato, F., Saltini, R., Tran, T.H., Zanolini, L.: Tob-svd: Total-order broadcast with single-vote decisions in the sleepy model. arXiv preprint arXiv:2310.11331 (2023)
8. D’Amato, F., Saltini, R., Tran, T.H., Zanolini, L.: 3-slot-finality protocol for ethereum. arXiv preprint arXiv:2411.00558 (2024)
9. Danezis, G., Kokoris-Kogias, L., Sonnino, A., Spiegelman, A.: Narwhal and tusk: a dag-based mempool and efficient bft consensus. In: Proceedings of the Seventeenth European Conference on Computer Systems. pp. 34–50 (2022)
10. Dolev, D., Fischer, M.J., Fowler, R., Lynch, N.A., Strong, H.R.: An efficient algorithm for byzantine agreement without authentication. *Information and Control* **52**(3), 257–274 (1982)
11. Duan, S., Reiter, M.K., Zhang, H.: Beat: Asynchronous bft made practical. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 2028–2041 (2018)
12. D’Amato, F., Neu, J., Tas, E.N., Tse, D.: No more attacks on proof-of-stake ethereum. arXiv preprint arXiv:2209.03255 (2022)
13. D’Amato, F., Zanolini, L.: A simple single slot finality protocol for ethereum. In: European Symposium on Research in Computer Security. pp. 376–393. Springer (2023)
14. Efron, Y., Neu, J., Ren, L., Tas, E.N.: Optimal good-case latency for sleepy consensus. arXiv preprint arXiv:2510.06023 (2025)
15. Efron, Y., Tas, E.N.: Dynamically available common subset. *Cryptology ePrint Archive* (2025)
16. Feldman, P., Micali, S.: Optimal algorithms for byzantine agreement. In: Proceedings of the twentieth annual ACM symposium on Theory of computing. pp. 148–161 (1988)
17. Fitzi, M., Ga, P., Kiayias, A., Russell, A.: Parallel chains: Improving throughput and latency of blockchain protocols via parallel composition. *Cryptology ePrint Archive* (2018)
18. Gao, Y., Lu, Y., Lu, Z., Tang, Q., Xu, J., Zhang, Z.: Dumbo-ng: Fast asynchronous bft consensus with throughput-oblivious latency. In: Proceedings of the 2022 ACM

- SIGSAC Conference on Computer and Communications Security. pp. 1187–1201 (2022)
19. Goyal, V., Li, H., Raizes, J.: Instant block confirmation in the sleepy model. In: International Conference on Financial Cryptography and Data Security. pp. 65–83. Springer (2021)
 20. Guo, B., Lu, Y., Lu, Z., Tang, Q., Xu, J., Zhang, Z.: Speeding dumbos: Pushing asynchronous bft closer to practice. In: Network and Distributed Systems Security (NDSS) Symposium 2022. pp. 1–18 (2022)
 21. Keidar, I., Kokoris-Kogias, E., Naor, O., Spiegelman, A.: All you need is dag. In: Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing. pp. 165–175 (2021)
 22. Li, S., Tse, D.: Taiji: longest chain availability with bft fast confirmation. arXiv preprint arXiv:2011.11097 (2020)
 23. Losa, G., Gafni, E.: Consensus in the unknown-participation message-adversary model. arXiv preprint arXiv:2301.04817 (2023)
 24. Loss, J., Nielsen, J.B.: Early stopping for any number of corruptions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 457–488. Springer (2024)
 25. Lu, Y., Lu, Z., Tang, Q., Wang, G.: Dumbos-mvba: Optimal multi-valued validated asynchronous byzantine agreement, revisited. In: Proceedings of the 39th symposium on principles of distributed computing. pp. 129–138 (2020)
 26. Malkhi, D., Momose, A., Ren, L.: Byzantine consensus under fully fluctuating participation. IACR Cryptol. ePrint Arch. **2022**, 1448 (2022)
 27. Malkhi, D., Momose, A., Ren, L.: Towards practical sleepy bft. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. pp. 490–503 (2023)
 28. Micali, S., Rabin, M., Vadhan, S.: Verifiable random functions. In: 40th annual symposium on foundations of computer science (cat. No. 99CB37039). pp. 120–130. IEEE (1999)
 29. Miller, A., Xia, Y., Croman, K., Shi, E., Song, D.: The honey badger of bft protocols. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. pp. 31–42 (2016)
 30. Momose, A., Ren, L.: Constant latency in sleepy consensus. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 2295–2308 (2022)
 31. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
 32. Pass, R., Shi, E.: The sleepy model of consensus. In: Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3–7, 2017, Proceedings, Part II 23. pp. 380–409. Springer (2017)
 33. Pass, R., Shi, E.: Thunderella: Blockchains with optimistic instant confirmation. In: Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29–May 3, 2018 Proceedings, Part II 37. pp. 3–33. Springer (2018)
 34. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2), 120–126 (1978)
 35. Shoup, V., Sliwinski, J., Vonlanthen, Y.: Kudzu: Fast and Simple High-Throughput BFT. In: 39th International Symposium on Distributed Computing (DISC 2025). vol. 356, pp. 42:1–42:19 (2025)

- 36. Spiegelman, A., Girdharan, N., Sonnino, A., Kokoris-Kogias, L.: Bullshark: Dag bft protocols made practical. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 2705–2718 (2022)
- 37. Vonlanthen, Y., Sliwinski, J., Albarello, M., Wattenhofer, R.: Banyan: Fast rotating leader bft. In: Proceedings of the 25th international middleware conference. pp. 494–507 (2024)
- 38. Yin, M., Malkhi, D., Reiter, M.K., Gueta, G.G., Abraham, I.: Hotstuff: Bft consensus with linearity and responsiveness. In: Proceedings of the 2019 ACM symposium on principles of distributed computing. pp. 347–356 (2019)