Byzantine Consensus under Fully Fluctuating Participation

Dahlia Malkhi[†], Atsuki Momose[*], and Ling Ren[*]

[†]Chainlink Labs
[*]University of Illinois at Urbana-Champaign

October 2022

Abstract

The longest-chain paradigm introduced by the Bitcoin protocol allows Byzantine consensus with fluctuating participation where nodes can spontaneously become active and inactive anytime. Since then, there have been several follow-up works that aim to achieve similar guarantees without Bitcoin's computationally expensive proof of work. However, existing solutions do not fully inherit Bitcoin's dynamic participation support. Specifically, they have to assume malicious nodes are always active, i.e., no late joining or leaving is allowed for malicious nodes, due to a problem known as costless simulation. Another problem of Bitcoin is its notoriously large latency. A series of works try to improve the latency while supporting dynamic participation. The work of Momose-Ren (CCS 2022) eventually achieved constant latency, but its concrete latency is still large. This work addresses both of these problems by presenting a protocol that has 3 round latency, tolerates one-third malicious nodes, and allows fully dynamic participation of both honest and malicious nodes. We also present a protocol with 2 round latency with slightly lower fault tolerance.

1 Introduction

Byzantine consensus [22], a decade-old problem in distributed computing and cryptography, allows a group of nodes to reach an agreement in the presence of corrupted participants. Classic consensus research has focused on the static participation model where every honest node is assumed to be always online [22, 12, 5, 15, 13, 6]. Recently, due to the rise of cryptocurrencies, Byzantine consensus has been studied under the *dynamic and unknown participation model*, where nodes can become active and inactive at any time. The celebrated Bitcoin protocol [27], with its underlying longest-chain paradigm, is the first consensus protocol that supports dynamic participation. Since then, there have been many efforts to improve or generalize the longest-chain paradigm [28, 10, 8]; there have also been works that extend classic quorum-based consensus protocols to the dynamic and unknown participation model [20, 18, 26].

Constraints on dynamic participation. However, all existing solutions fall short of achieving fully dynamic participation like Bitcoin does. The dynamic participation in Bitcoin is natural and elegant: both honest nodes and malicious nodes can fluctuate dynamically as long as we have an honest majority. However, once we remove Bitcoin's computationally expensive proof-of-work, we lose an important feature that computation effort is not reusable. Without proof-of-work existing

protocols assume that *faulty nodes are always active* [28, 26, 9]. In other words, faulty nodes cannot dynamically leave or join the system at all. This assumption is hard to justify in practice. Suppose only dozens of nodes are active at the beginning, but a million nodes are active a few years later when a system becomes popular. At that later time, the above protocols still require that only dozens of nodes are faulty out of the one million nodes!

The difficulty in handling faulty nodes' fluctuation is due to a problem called *costless simulation* [11]. To elaborate, when a faulty node becomes active, it can pretend to have always been active in the past. It can fabricate messages that were supposed to be sent when it was not active and try to alter the consensus results in the past. For example, this directly affects the safety of proof-of-stake longest-chain protocols as faulty nodes can create a longer chain than the honest one retroactively, using the stake (and the voting power that comes with it) they had before they became active.

Besides such a "backward" simulation, faulty nodes can also engage in "forward" simulation by giving their secret keys to other faulty nodes before going inactive. This way, other faulty nodes that are active in the future can send messages on their behalf, and it is as if those faulty nodes never went inactive. Some works incorporate strong assumptions such as verifiable delay functions (VDF) [11] or a trusted list of participants [8, 10] to defeat backward simulation. But even these strong assumptions do not help prevent forward simulation.

We also point out that some prior works, especially the ones with improved efficiency, do not even support fully dynamic participation of honest nodes [18, 26]. Goyal et al. require a minimum number (linear in the security parameter) of nodes to be active at all times. The work of Momose-Ren requires the number of honest nodes to eventually stabilize in order to make progress.

The primary goal of this work is to support fully dynamic participation of both honest and faulty nodes without the enormous energy consumption of proof-of-work.

Long latency. Another drawback of existing solutions is the long latency. Longest-chain consensus protocols have long latency because their latency depends on the security parameters (and some other factors). There have been significant efforts to remove these dependencies [4, 16, 24, 18], eventually leading to the work of Momose-Ren [26] which achieves $O(\Delta)$ latency (Δ is the bound on communication delay). Despite being asymptotically optimal, the concrete latency in Momose-Ren is still quite large, at least 16Δ, whereas classic BFT protocols (without dynamic participation support) can make decisions in as low as Δ time [2].

Main result. This work addresses both of the above problems. Concretely, the consensus problem we study is *Byzantine atomic broadcast* [7], the underlying problem of blockchain, where nodes agree on a sequence of values (i.e., a ledger). As for the model, our starting point is the *sleepy model* of Pass-Shi [28]. The original sleepy model assumes that the number of active *honest* nodes can fluctuate dynamically at the adversary's control, but all faulty nodes are always awake. We extend the model to allow the number of faulty nodes to fluctuate as well, which we call the sleepy model with *fluctuating adversary*. Our main result below supports fully dynamic participation of both honest nodes and faulty nodes and achieves a concretely small latency while sacrificing the fault tolerance.

Theorem 1 (informal)**.** *Assuming authenticated channels and verifiable random functions (VRF), there exists a Byzantine atomic broadcast protocol in the sleepy model with fluctuating adversary with (best-case)* 3Δ *latency that allows less than* $1/3$ *of active nodes at any time to be faulty.*

Here, we assume authenticated channels do not deliver any stale messages (formalized in Section 3). Thus, faulty nodes cannot send future messages unless they are active at that future time.

In practice, authenticated channels are usually established using a public key infrastructure (PKI) and digital signatures. Thus, digital signatures plus PKI is traditionally considered to be a stronger assumption than authenticated channels. Observe that Byzantine consensus in the authenticated channels (without signatures) tolerates less than one-third of Byzantine faults [22]; with signatures, the fault tolerance can be up to one-half or even 99% [12].

However, the situation is quite different in the sleepy model. In the sleepy model, authenticated channels are stronger in some ways than digital signatures plus PKI. With the latter, it seems hard (if not impossible) to defeat forward simulation. A faulty node can give its secret key to the adversary (or other faulty nodes) before going to sleep, allowing the adversary to act on its behalf after it goes to sleep. It is then as if that faulty node never left.

Because of this, under the PKI settings, we have to consider a slightly weaker adversary. Specifically, our protocol allows an increasing number of faulty nodes but they are not allowed to go from active to inactive. We call it the sleepy model with *growing adversary*. In other words, we cannot handle forward simulation but can still handle backward simulation. Under this model, we show the following result:

Theorem 2 (informal)**.** *Assuming digital signatures, public key infrastructure (PKI), and verifiable random functions (VRF), there exists a Byzantine atomic broadcast protocol in the sleepy model with growing adversary with (best-case)* 3Δ *latency that allows less than* $1/3$ *of active nodes at any time to be faulty.*

In fact, we believe the downgrade to the growing adversary can be an artifact in the modeling itself. In practice, even malicious nodes are mutually distrustful and would not share keys with each other. If we have a way to formally model faulty nodes going to sleep without giving away their secret keys, then our protocol will likely work fine against fluctuating faulty nodes. We leave this direction to future work.

Technical overview. We now explain at a high level how we address the two problems above. Our starting point is the protocol by Momose-Ren [26], which is constructed in two steps: they first construct a graded agreement (GA) subroutine [14, 19, 1, 25] in the sleepy model. The second step uses the GA to build an atomic broadcast.

First, by slightly sacrificing the fault tolerance from $1/2$ to $1/3$, we design an extremely simple GA with one round of communication. Our technique somewhat resembles the reliable broadcast protocol of Khanchandani-Wattenhofer [20], though their work is presented in an unknown but static participation model. Notably, our GA protocol is secure without digital signatures (assuming authenticated channel), and hence trivially defeats forward simulation (as there is no secret key to reveal).

However, this is not the end of the story. Naïvely plugging the one-round GA into the Momose-Ren atomic broadcast does not work for either of the two settings above. To delve deeper into the issue, the atomic broadcast of Momose-Ren is composed of sequential invocations of GA where all past GAs potentially affect the current decision. In the case of authenticated channels without digital signatures, messages from past rounds (and hence the results of past GAs) cannot be reliably recovered. In the PKI plus signature case, past signed messages can be recovered, but this allows

faulty nodes to concoct messages of past rounds (back when they were inactive), again opening up the vulnerability of backward simulation.

The main technical contribution of this work is to construct an atomic broadcast protocol in which only the messages from the immediate last round have an impact on the current round. The outcome is a remarkably simple protocol consisting of only two single-round GA instances per decision. This is also how we get a significant latency improvement over the five-GA construction of Momose-Ren.

Additional advantages and assumption. In addition to the two highlighted features, our protocol has several other advantages over the state-of-the-art protocol of Momose-Ren [26]. First, communication cost per decision is reduced from cubic to quadratic in the number of active nodes. Another advantage of our protocol is that it has *perfect* safety (with authenticated channels), whereas all previous protocols, longest-chain and sleepy BFT alike, only guarantee safety with overwhelming probability.

Towards optimal latency. After giving the two results above, we explore further latency reduction towards optimal latency. It is easy to show that any protocol in the sleepy model cannot make a decision in less than Δ time (Appendix A). With this lower bound in mind, we show the following result:

Theorem 3 (informal)**.** *Assuming digital signature and public-key infrastructure (PKI) a verifiable random function (VRF), there exists a Byzantine atomic broadcast protocol in the sleepy model with growing adversary with (best-case)* 2Δ *latency that tolerates up to 1/4 of active nodes being faulty.*

The one-round latency improvement is attributed to reducing the number of GA invocations from 2 to 1. Intuitively, our first protocol (and Momose-Ren) implements a three-graded agreement with two invocations of a standard two-graded agreement. Our key insight is that further downgrading the fault tolerance to 1/4 allows us to achieve a three-graded agreement in one round of communication.

To summarize, this paper presents two protocols as follows:

1. Byzantine atomic broadcast with 3Δ latency and 1/3 fault tolerance (Section 5), which is built on a graded agreement (Section 4)

2. Byzantine atomic broadcast with 2Δ latency and 1/4 fault tolerance (Section 6).

2 Related Work

Byzantine consensus has been studied for a few decades, mostly in the static and known participation model [22, 12, 5, 15, 13, 6]. The Bitcoin protocol [27] inspired a new area of research in Byzantine consensus that considers unknown and dynamic participation. The unknown and dynamic participation model was later formalized as the sleepy model [28]. Below, we review the related works in sleepy consensus research.

Longest-chain paradigm. Early research on sleepy consensus naturally adopted Bitcoin's longest-chain paradigm. A number of works generalized the longest-chain paradigm by replacing the computationally expensive proof-of-work with proof-of-stake [28, 21, 3, 8]. A major drawback of a longest-chain protocol is its long latency. Specifically, a plain longest-chain protocol (e.g., Bitcoin)

has a latency of $O(\frac{\kappa\Delta}{\gamma})$ where κ is the desired security level, γ is the active participation level (i.e., the fraction of active nodes compared to the total nodes), and Δ is the bound on network delay. A number of works try to remove some of the factors that lead to the long latency. Prism [4], Parallel Chain [16], and Taiji [24] remove the dependency on κ using many parallel instances of longest chains, but could not remove the dependency on γ. A recent work by Deb et al. [11] achieves $O(\Delta)$ latency under optimistic conditions where the participation level is high, but it still has the same latency as a longest-chain protocol with low participation

The classic BFT paradigm. Another line of work tries to adapt the classic BFT paradigm from the traditional known and static participation model to the sleepy model. Goyal et al. [18] removes the dependency on γ by extending Algorand [17], but the dependency on κ remains. Besides, due to the use of a static quorum threshold, it places a constraint on honest nodes' fluctuation; it requires $\Omega(\kappa)$ awake honest nodes at all times. This was resolved in Momose-Ren [26] using dynamic quorum thresholds that adapt to the actual level of participation. Momose-Ren [26] also removes the latency's dependencies on both the security parameter κ and actual participation level γ to achieve $O(\Delta)$ latency. But their concrete latency is still quite high, at least 16Δ. Moreover, the liveness of their protocol is only guaranteed when participation is stable for long enough periods. We also mention the work of Khanchandani-Wattenhofer [20]. While their protocols assume an unknown but static participation model, some of their techniques seem applicable to the unknown and dynamic participation model.

3 Model and Preliminaries

We study Byzantine atomic broadcast problem in the sleepy model. We consider a system of N total nodes communicating over a synchronous network (note that network synchrony is necessary in the sleepy model [28]). We use Δ to denote the bound on communication delay. For simplicity, we assume a completely synchronous clock, i.e., nodes have access to a common global clock. We can extend our results to a model with bounded clock skew by applying the round transformation technique in [26] (with a slight latency increase).

The adversary is adaptive and can corrupt nodes at any time. Faulty nodes are Byzantine and deviate from the protocol arbitrarily. Non-faulty nodes are said to be *honest* and behave as instructed by the protocol.

Sleepy model. The original sleepy model is introduced by Pass and Shi [28]. A node is in one of two states: *awake* or *asleep*. Awake nodes actively participate in the execution, while asleep nodes neither execute any code nor send/receive any message. The number of awake nodes at each time t is denoted $0 < n_t \leq N$; The status of each node can change at the adversary's control without any advance notice. As for the faulty nodes' dynamic participation, we consider two types of adversarial assumptions.

1. **Fluctuating adversary.** faulty nodes can wake up and go to sleep at any time.
2. **Growing adversary.** Asleep faulty nodes can become awake but cannot go back to sleep ever after. In other words, awake faulty nodes are non-decreasing.

Our protocols tolerate one of the adversaries above depending on the communication model. With authenticated channels, we tolerate the fluctuating adversary. With digital signature with PKI, we tolerate the growing adversary.

1. **Authenticated channels.** Each pair of nodes has an independent bidirectional channel to which no other nodes can send any message. If an awake node p sends a message x at time t to another node q who is awake at time $t + \Delta$, then q will receive x by time $t + \Delta$. We also assume any message not received within Δ will be lost including those from faulty nodes. Therefore, even faulty nodes cannot send any future message unless it is awake in the future time.

2. **Digital signature with PKI.** Communication channels are unauthenticated, i.e., no one knows the origin of any message, but each node has access to digital signature with public-key infrastructure (PKI). In this case, messages failed to be received can be transferred by other awake nodes. So, we assume (for simplicity) messages not received within Δ are not lost and delivered later (this is also assumed in the original sleepy model [28]). More precisely, if an awake node p sends a message x at time t to another node q who is awake at time $t' \geq t + \Delta$, then q will receive x by time t'. In practice, only messages with necessary information must be recovered; We will discuss this later in this paper (Section 7).

In both cases, we use $\langle x \rangle_p$ to denote a message from p. In the former case, it simply means the message is sent by p. In the latter case, it is a message signed by p's secret key.

Fault threshold. We define the fault threshold in the following way. Let $\mathcal{F}_t$ be the set of faulty nodes at time t, and let $f_{t,T} = |\cup_{t \leq \tau \leq t+T} \mathcal{F}_\tau|$, i.e., $f_{t,T}$ is the total number of faulty nodes that are ever awake at some point during the time interval $[t, t + T]$. Section 4 and 5 assume $f_{t,\Delta} < n_t/3$ for any t. Section 6 assumes $f_{t,2\Delta} < n_t/4$ for any t. In other words, our fault threshold condition requires that the number of faulty nodes that are ever awake in an interval of length Δ (or 2Δ) is less than one-third (or one-fouth) of the total number of awake nodes at a given point in time. We will explain why the fault threshold is defined this way in Section 4. For now, we remark that the way we define fault threshold is a little stronger than what the Bitcoin protocol assumes. Bitcoin simply requires a minority fault at any time (no time interval required) — which can be thought of as $f_{t,0} < n_t/2$ (again this is because computation effort is not reusable in proof-of-work). But all prior works on sleep consensus [28, 18, 26] require the much stronger assumption that all faulty nodes are always awake — essentially $f_{t,\infty} < n_t/2$.

Byzantine atomic broadcast. Byzantine atomic broadcast allows nodes to agree on a growing sequence of values $[x_0, x_1, x_2, ...]$ called a *log* such that:

1. *Safety.* If two honest nodes decide logs $[x_0, x_1, .., x_j]$ and $[x'_0, x'_1, .., x'_{j'}]$, then $x_i = x'_i$ for all $i \leq \min(j, j')$.

2. *Liveness.* If an honest node inputs a value x, then there exists a time t such that all honest nodes awake at t decide a log containing x.

Here, we do not care what the values are. It might be from a finite class depending on the application built on top of the atomic broadcast.

Log format. For simplicity, we assume values input by nodes are batched into a block. A log is represented as an ordered set of blocks $[b_1, b_2, .., b_k]$. We say a log Λ *extends* Λ' if Λ' is a prefix of Λ. We say two logs *conflict* if they do not extend one another.

Verifiable random function. We use verifiable random function (VRF) for liveness. Each node p with its secret key can evaluate $(\rho, \pi) \leftarrow \mathsf{VRF}_p(\mu)$ on any input μ. The output is a deterministic

pseudorandom value ρ along with a proof π. Using π and the public key of node p, anyone can verify whether ρ is a correct evaluation of VRF_p on input μ.

4 Graded Agreement with 1/3 Fault Tolerance

This section presents a graded agreement (GA) tolerating $f_{t,\Delta} < n_t/3$ faulty nodes.

Graded agreement. Graded agreement (GA) is a weak form of agreement problem that has been used widely (either implicitly or explicitly) as a stepping stone to full consensus. At the beginning of the GA protocol, nodes input logs[1]; at the end of the protocol, each node outputs a set of logs, with each log assigned a *grade* bit. Informally, it provides the following guarantees:

1. *Graded consistency.* If an honest node outputs a log with grade 1, then all honest nodes output the log with at least grade 0.
2. *Integrity.* If an honest node outputs a log with any grade, then there exists an honest node that inputs the log.
3. *Validity.* Nodes output with grade 1 the longest common prefix among honest nodes' input logs.
4. *Uniqueness.* If an honest node outputs a log with grade 1, then no honest node outputs any conflicting log with grade 1.
5. *Bounded divergence.* Each honest node outputs at most 2 conflicting logs (of course with grade 0).

We remark that our GA definition above is a little weaker than the standard GA definition in which each node outputs only one value (and hence uniqueness and bounded divergence hold by definition in standard GA). Both properties are useful to our atomic broadcast construction (see more in Section 5).

Our protocol. Our GA protocol is described in Figure 1. It takes one round of communication. At the beginning of the round (time $t = 0$), awake nodes vote for their own input logs. At the end of the round (time $t = \Delta$), awake nodes tally votes, and decide the outputs. Here, note that a node does not know the actual participation level (i.e., how many nodes are awake), and furthermore, different nodes can hear from different sets of nodes. So, each node makes a decision based on its own *perceived* participation. If a log is voted by more than 2/3 of voters (the node hears from), the node outputs the log with grade 1; and if the log is voted by more than 1/3 but less than 2/3 of voters, then the node outputs the log with grade 0. Here, a vote for a log is also considered as vote for all its prefixes. Also, votes for different logs from the same node (an obvious Byzantine behavior) are simply ignored. Note that the set of nodes awake at the beginning of a round may be different from the set of nodes awake at the end of a round.

Proof sketch. This simple protocol achieves the above-mentioned properties by the following logic. First, although the number of voters is unknown, more than 2/3 of voters are honest and every honest node hears from them; validity holds. If $> 2/3$ of voters (that a node hears from) vote

[1] GAs are usually defined on a single value, but we define it on logs

for the same log, all other nodes at least receive votes for the log from $> 1/3$ of voters (that they hear from); graded consistency holds. Moreover, these $> 1/3$ voters are in fact honest, so no other log can collect $> 2/3$ votes; uniqueness holds. Integrity follows from the fact that $> 1/3$ of voters must include one honest node. Finally, as different vote messages from the same node are ignored, bounded divergence holds.

Remark on fault threshold. Recall we define the fault threshold slightly differently from the most straightforward way (see Section 3). This is due to a subtle technical reason. Counting faulty nodes in a time interval makes sure a large enough fraction of voters are honest nodes. Otherwise, faulty nodes could be replaced arbitrarily many times during one round of voting, and could easily outnumber honest nodes. To elaborate, suppose the fault threshold is simply defined for each point in time without using an interval, e.g. $f_t < n_t/3$ for all t. Then, an adversary can launch the following attack: a faulty node wakes up, votes, and goes to sleep immediately; then another faulty node wakes up, votes, and goes to sleep immediately; and this process repeats arbitrarily many times. This way, at no point in time did the adversary have many faulty nodes awake, but the total number of faulty votes that can be produced this way is arbitrarily large and easily outnumbers the honest nodes.

Node p runs the following algorithm if it is awake at time t.

1. time $t = 0$. Send $\langle \mathsf{vote}, \Lambda \rangle_p$ where Λ is the input log.
2. time $t = \Delta$. Tally vote messages and decides the outputs as follows. Let m be the number of vote messages received.
   (a) For any Λ voted by $> 2m/3$ nodes, output $(\Lambda, 1)$
   (b) For any Λ voted by $> m/3$ nodes (but $\leq 2m/3$), output $(\Lambda, 0)$

Here, if Λ' extends Λ, then $\langle \mathsf{vote}, \Lambda' \rangle$ is also counted as a vote for Λ. Two different vote messages from the same node are ignored.

Figure 1: GA: our graded agreement protocol

4.1 Correctness Proof

Our GA protocol provides the following formal guarantees. Note that all of the proofs below hold assuming either authenticated channels or digital signatures plus PKI.

Lemma 1 (graded consistency)**.** *If an honest node outputs* $(\Lambda, 1)$*, then all honest nodes awake at time* $t = \Delta$ *output* $(\Lambda, *)$

Proof. Suppose an honest node p outputs $(\Lambda, 1)$, then it sees more than $2m/3$ votes for Λ where m is the number of vote messages p receives by time $t = \Delta$. Let n be the number of nodes awake at time $t = 0$. Then, more than $2n/3$ honest nodes are awake at time $t = 0$, and these nodes will vote. Given the way we define the fault threshold, there are less than $n/3$ faulty nodes that are ever awake during the time interval $[0, \Delta]$. Thus, the m messages p receives consist of $h > 2n/3$

messages from honest nodes and $\alpha < n/3$ messages from faulty nodes. By the same argument, another honest node q awake at time $t = \Delta$ receives $m' = h + \alpha'$ messages where $\alpha' < n/3$ messages are from faulty nodes. Now, the number of vote messages for Λ that p receives from honest nodes is more than

$$\begin{aligned} 2m/3 - \alpha &= 2(h+\alpha)/3 - \alpha = 2h/3 - \alpha/3 \\ &> h/3 + \alpha'/3 = m'/3. \end{aligned}$$

The inequality step uses the fact that $h > \alpha + \alpha'$. These honest nodes must also send the same vote messages to q. Therefore, out of the m' votes q receives, $> m'/3$ votes are for Λ, and output $(\Lambda, *)$. □

Lemma 2 (integrity)**.** *If an honest node outputs* $(\Lambda, *)$*, then at least an honest node inputs a log that extends* Λ*.*

Proof. Suppose an honest node p outputs $(\Lambda, *)$, then it sees $> m/3$ votes for Λ where m is the number of vote messages p receives by time $t = \Delta$. The m messages include $h > 2n/3$ messages from honest nodes and $\alpha < n/3$ messages from faulty nodes, where n is the number of nodes awake at time $t = 0$. So the number of vote messages voting for Λ that p receives from honest nodes is more than

$$m/3 - \alpha = (h+\alpha)/3 - \alpha = (h - 2\alpha)/3 > 0.$$

This implies at least one honest node must have input a log that extends Λ. □

Lemma 3 (validity)**.** *Let* Λ *be the longest common prefix of honest nodes' inputs. Then all honest nodes awake at* $t = \Delta$ *output* $(\Lambda, 1)$*.*

Proof. Let m be the number of messages an honest node p awake at time $t = \Delta$ receives. The m messages consist of $h > 2n/3$ messages from honest nodes and $\alpha < n/3$ messages from faulty nodes, where n is the number of nodes awake at time $t = 0$. As all h honest nodes awake at time $t = 0$ must vote for Λ, p must receive $h > 2m/3$ votes for Λ, and output $(\Lambda, 1)$. □

Lemma 4 (uniqueness)**.** *If an honest node outputs* $(\Lambda, 1)$ *and another honest node outputs* $(\Lambda', 1)$*, then* B *and* Λ' *do not conflict with each other.*

Proof. Suppose an honest node p outputs $(\Lambda, 1)$, then it sees $> 2m/3$ votes for Λ where m is the number of vote messages p receives. By the same logic in the proof of graded consistency, we have that out of the m' votes received by an honest node q awake at $t = \Delta$, $> m'/3$ votes are for Λ and come from honest nodes. So q cannot see more than $2m'/3$ votes for a log conflicting with Λ and cannot output such a log with grade 1. □

Lemma 5 (bounded divergence)**.** *Each honest node outputs (with grade 0) at most two conflicting logs.*

Proof. In order to be an output, a log must be voted by $> m/3$ nodes out of the m votes received. Recall that conflicting vote messages from the same node are ignored. Thus, each node outputs at most two conflicting logs. □

5 Atomic Broadcast with 1/3 Fault Tolerance

This section presents an atomic broadcast protocol building on the GA protocol described in Section 4. Our protocol tolerates $f_{t,\Delta} < n_t/3$ faulty nodes controlled by either a fluctuating or a growing adversary (depending on the communication model).

View-based execution. Our protocol is described in Figure 2. Time is divided into *views*. Each view lasts 2Δ time, but the first view (view 0) lasts only Δ. Thus, view $v \geq 1$ starts at time $(2v-1)\Delta$. For convenience, we also say "time $t = \tau$ of view v" to refer to time $t = (2v-1)\Delta + \tau$.

Propose. Nodes propose logs for view v in the second round of view $v-1$ (i.e., time $[\Delta, 2\Delta]$ of view $v-1$). They also include VRF evaluation on the view number $\mathsf{VRF}_p(v)$ in their proposals, which work as leader election lotteries. At the beginning of view v (time 0 of view v), a node chooses a proposal with the highest valid VRF evaluation and treats the node who sent it as the *leader*. The very first view $v = 0$ serves only as the "propose" step for view $v = 1$, which is why it takes Δ.

Decide A log is decided after two sequential invocations of GA. A node inputs the leader's log to the first GA (denoted $\mathsf{GA}_{v,1}$) at time 0. An output log by the first GA with grade 1 is input to the second GA (denoted $\mathsf{GA}_{v,2}$) at time Δ. Recall that GA always outputs a (possibly empty) log with grade 1 (by validity). Finally, an output log by the second GA with grade 1 is decided at time 2Δ. The two GA invocations also update two key variables respectively: $\mathcal{C}_v$ and $\mathcal{L}_v$, which stand for "candidate" and "lock", respectively. We explain how they help achieve safety and liveness below.

1. **Safety.** $\mathcal{L}_v$ is set to the longest log that $\mathsf{GA}_{v,2}$ outputs (with either grade 0 or 1). $\mathcal{L}_v$ is then used to perform a consistency check on a proposed log at time 0 of the next view (view $v+1$). A log is input to $\mathsf{GA}_{v+1,1}$ only if it does not conflict with $\mathcal{L}_v$, i.e., the node "locks" on the log. If a log Λ is decided, all honest nodes update $\mathcal{L}_v$ to be Λ (due to graded consistency and uniqueness). So no honest node will input a log conflicting with Λ to GA in the next view, and any decision will be consistent with Λ (due to the integrity of GA).

2. **Liveness.** $\mathcal{C}_v$ is set to the longest log that $\mathsf{GA}_{v,1}$ outputs (with either grade 0 or 1). Each node proposes a log extending $\mathcal{C}_v$, i.e., a "candidate" for the next decision. In order for an honest leader's proposal to get decided, the honest leader's "candidate" $\mathcal{C}_v$ must be consistent with honest nodes' locks $\mathcal{L}_{v-1}$; Otherwise, it cannot pass the consistency check. We will show this "good" event happens with probability at least 1/2.

A key feature of our protocol is that every step of the protocol depends only on the immediate last round. More concretely, at time $t = 0$, an input to $\mathsf{GA}_{v,1}$ depends only on the $\mathsf{propose}$ messages and vote messages (from $\mathsf{GA}_{v-1,2}$) sent during $[\Delta, 2\Delta]$ of view $v-1$. Likewise, at time $t = \Delta$, an input to $\mathsf{GA}_{v,2}$ depends only on the vote messages (from $\mathsf{GA}_{v,1}$) sent during $[0, \Delta]$ of view v. This feature is critical to supporting a fluctuating or growing adversary. First of all, this is necessary when we assume authenticated channels where messages are lost after one round (Δ time). When we assume digital signatures and PKI in case of a growing adversary, this feature allows us to tolerate backward simulation because faulty nodes who fabricate messages of past rounds now have no influence on the current actions of honest nodes.

$\mathcal{L}_0$ and $\mathcal{C}_0$ are defined as an empty log []. Node p runs the following algorithm if it is awake at time t. View 0 lasts Δ time. At time $t = 0$ of view 0, send $\langle \mathsf{propose}, \Lambda, \mathsf{VRF}_p(1) \rangle_p$ to propose the first log $\Lambda := [b]$. All later views $v \geq 1$ each takes 2Δ and work as follows.

1. time $t = 0$. Start $\mathsf{GA}_{v,1}$; The input is a log in the $\mathsf{propose}$ message with the largest valid VRF on v that does not conflict with $\mathcal{L}_{v-1}$.
2. time $t = \Delta$.
   - Start $\mathsf{GA}_{v,2}$; The input is the longest log Λ such that $\mathsf{GA}_{v,1}$ outputs $(\Lambda, 1)$.
   - Let $\mathcal{C}_v$ be the longest log $\mathcal{C}$ such that $\mathsf{GA}_{v,1}$ outputs $(\mathcal{C}, *)$. (If there are two such $\mathcal{C}$, pick one at random.) Then, send $\langle \mathsf{propose}, \Lambda', \mathsf{VRF}_p(v+1) \rangle_p$ where the new log is $\Lambda' := b \mid \mathcal{C}_v$.
3. time $t = 2\Delta$.
   - If $\mathsf{GA}_{v,2}$ outputs $(\Lambda, 1)$, decide Λ.
   - Set $\mathcal{L}_v$ to the longest log Λ' such that $\mathsf{GA}_{v,2}$ outputs $(\Lambda', *)$.

Note that $t = 2\Delta$ of view v matches $t = 0$ of view $v + 1$.

Figure 2: Atomic broadcast with 1/3 fault tolerance and 3Δ latency

5.1 Correctness Proof

We prove safety and liveness of our protocol.

Lemma 6 (safety)**.** *If two honest nodes decide logs Λ and Λ', then Λ and Λ' do not conflict with each other.*

Proof. Without loss of generality, suppose Λ is decided in view v and Λ' is decided in view $v' \geq v$. The honest node who decides Λ must have output $(\Lambda, 1)$ in $\mathsf{GA}_{v,2}$. If $v = v'$, then the lemma follows from the uniqueness of $\mathsf{GA}_{v,2}$. So we consider $v' > v$. Due to graded consistency of $\mathsf{GA}_{v,2}$, any honest node p awake at time $t = 2\Delta$ of view v must have output $(\Lambda, *)$ in $\mathsf{GA}_{v,2}$. We also observe that p could not have output any log conflicting with Λ in $\mathsf{GA}_{v,2}$; otherwise, two conflicting logs must have been output with grade 1 from $\mathsf{GA}_{v,1}$ by honest nodes, which violates GA uniqueness. Therefore, p must have set $\mathcal{L}_v$ to a log extending Λ, and hence input to $\mathsf{GA}_{v+1,1}$ a log extending Λ. Due to integrity, $\mathsf{GA}_{v+1,1}$, and inductively all later GAs, must output logs extending Λ. Hence, Λ' must extend Λ. □

Lemma 7 (liveness)**.** *If an honest node inputs a value x to the atomic broadcast protocol, then there exists a time t such that all honest nodes awake at t decide a log containing x.*

Proof. If an honest node has the highest VRF in view v, we call that node the honest leader. If all honest nodes awake at time $t = 0$ of view v input to $\mathsf{GA}_{v,1}$ a log proposed by an honest leader, then the log must be decided. The only reason an honest node does not input the leader's proposal to $\mathsf{GA}_{v,1}$ is that it conflicts with its lock $\mathcal{L}$. We prove all awake honest node *accept* (i.e., input to $\mathsf{GA}_{v,1}$) the leader's proposal with probability more than 1/2.

Let Λ be the longest log among the locks (i.e., variable $\mathcal{L}_{v-1}$) of honest nodes awake at time $t = 0$ of view v. Then, there exists an honest node p who outputs $(\Lambda, *)$ from $\mathsf{GA}_{v-1,2}$. Then, at least an honest node q must have input to $\mathsf{GA}_{v,2}$ a log extending Λ after outputting $(\Lambda, 1)$ in $\mathsf{GA}_{v-1,1}$. Due to graded consistency, the leader must have output $(\Lambda, 0)$ in $\mathsf{GA}_{v-1,1}$. Due to bounded divergence, the leader outputs at most one other conflicting log from $\mathsf{GA}_{v-1,1}$. Therefore, with probability at least $1/2$, the leader proposes a log extending Λ. Due to uniqueness, all honest nodes awake at time $t = 0$ of view v set their lock values $\mathcal{L}_{v-1}$ to logs extending Λ (as we are assuming Λ is the longest among them). Therefore, with probability at least $1/2$, the leader's proposal is accepted by all honest nodes awake at time $t = 0$ of view v.

Now, as more than $3/4$ of awake nodes are honest at all times, an honest leader exists in each view with probability at least $3/8$. Hence, a log is decided in each view with probability at least $3/8$. All values input by honest nodes to the atomic broadcast are eventually included in a decided log. □

6 Atomic Broadcast with 1/4 fault tolerance

This section presents an atomic broadcast protocol with 2Δ latency that tolerates $f_{t,2\Delta} < n_t/4$ faulty nodes controlled by the growing adversary. The protocol is described in Figure 3.

The basic structure of the protocol is very similar to our previous protocol in Figure 2. In each view, logs are proposed by nodes along with VRFs, and an elected leader's proposal is then voted and decided after collecting enough votes. The difference is that only one voting round is used instead of two (the two GA invocations) to reduce the latency of the protocol to 2Δ.

To aid understanding, we point out the connection between the two protocols below. The previous protocol, in fact, implements a *multi-graded* agreement (in fact 3 grades) using two sequential invocations of a single-graded agreement. We can re-interpret grade 0 of GA_1 as grade 0, grade 0 of GA_2 as grade 1, and grade 1 of GA_2 as grade 2. Then, we have the following informal guarantee: *if an honest node outputs a value with grade g, then all honest nodes output that value with grade $g - 1$.* This property is the multi-graded version of graded consistency and we crucially relied on it to achieve safety and liveness. The protocol in this section implements a multi-graded agreement using a single voting round as follows: if a value is voted by $3/4$ of voters, it is assigned grade 2; likewise, $1/2$ votes result in grade 1, and $1/4$ votes result in grade 0. It is not hard to verify multi-graded consistency. Finally, as done in the previous protocol, a log is decided if it is assigned grade 2, and logs with grades 1 and 0 updates $\mathcal{L}_v$ and $\mathcal{C}_v$, respectively. We can then argue safety and liveness in the same way.

6.1 Correctness Proof

Lemma 8 (safety)**.** *If two honest nodes decide Λ and Λ', then Λ and Λ' do not conflict with each other.*

Proof. Without loss of generality, suppose Λ is decided in view v and Λ' is decided in view $v' \geq v$. The node p who decides the log Λ must see votes for Λ from more than $3m/4$ nodes (at time $t = 2\Delta$ of view v) where m is the number of vote messages of view v (i.e., $\langle \mathsf{vote}, *, v \rangle$) that p received. The m messages consist of $h > 3n/4$ messages from honest nodes and $\alpha < n/4$ messages from faulty nodes, where n is the number of awake nodes at time $t = \Delta$ of view v. Similarly, another honest node q awake at time $t = \Delta$ of view $v + 1$ receives $m' = h + \alpha$ messages, out of which α' messages

$\mathcal{L}_0$ and $\mathcal{C}_0$ are defined as an empty log []. Each view $v \geq 0$ takes 2Δ. Note that $t = 2\Delta$ of view v is also $t = 0$ of view $v + 1$. Node p runs the following algorithm at each time t if it is awake.

1. time $t = 0$. Send $\langle \mathsf{propose}, \Lambda, \mathsf{VRF}_p(v) \rangle_p$ where the log is $\Lambda := b \mid \mathcal{C}_{v-1}$.
2. time $t = \Delta$.
   - Let m be the number of $\langle \mathsf{vote}, *, v - 1 \rangle$ received. Set $\mathcal{L}_{v-1}$ to longest log that is voted by $> m/2$ nodes.
   - Let Λ be the log in the $\mathsf{propose}$ message with the largest valid VRF on v that extends $\mathcal{L}_{v-1}$, or $\Lambda := \mathcal{L}_{v-1}$ if no such block exists. Send $\langle \mathsf{vote}, \Lambda, v \rangle_p$.
3. time $t = 2\Delta$. Let m be the number of $\langle \mathsf{vote}, *, v \rangle$ and update variables.
   - Set $\mathcal{C}_v$ to the longest log that is voted by $> m/4$ nodes.
   - Decide any log that is voted by $> 3m/4$ nodes.

If Λ' extends Λ, then $\langle \mathsf{vote}, \Lambda' \rangle$ is also counted as a vote for Λ. Two different vote messages from the same node are ignored.

Figure 3: Atomic broadcast with 1/4 fault tolerance and 2Δ latency

are from faulty nodes. So the number of vote messages of view v voting for Λ that p receives from honest nodes is

$$\begin{aligned} 3m/4 - \alpha &= 3(h + \alpha)/4 - \alpha = 3h/4 - \alpha/4 \\ &> h/2 + \alpha'/2 = m'/2. \end{aligned}$$

These honest nodes must also send the same vote messages to all other honest nodes. Therefore, q also sees at least $> m'/2$ votes for Λ, and sets $\mathcal{L}_v$ to a log that extends Λ. The rest of the proof is the same as that of Lemma 6. □

Lemma 9 (liveness)**.** *If an honest node inputs a value x, then there exists a time t such that all honest nodes awake at t decide a log containing x.*

Proof. We prove all honest nodes awake at time $t = \Delta$ of view v send vote for a log proposed by an honest leader with probability 1/4; rest of the proof is the same as that of Lemma 7. Again, the only reason an honest node p does not vote for an honest leader's proposal is it conflicts with its lock $\mathcal{L}_{v-1}$. We prove all honest nodes awake at time $t = \Delta$ of view v accept the honest leader's proposal with probability at least 1/3.

Let Λ be the longest log among the locks (i.e., value of $\mathcal{L}_{v-1}$) of honest nodes awake at time $t = \Delta$ of view v. Then, there exists an honest node p who observes $> m/2$ votes for Λ; m is the number of vote of view v (i.e., $\langle \mathsf{vote}, *, v - 1 \rangle$) p receives by time $t = \Delta$ of view v, which include $h > 3n/4$ messages from honest nodes and $\alpha < n/4$ messages from faulty nodes, where n is the number of awake nodes at time $t = \Delta$ of view $v - 1$. Similarly, the honest leader (say q) receives vote of view v from m' nodes by time $t = 0$ of view v. The number of vote of view v for Λ that p receives from honest nodes is

$$m/2 - \alpha = (h + \alpha)/2 - \alpha = h/2 - \alpha/2$$
$$> h/4 + \alpha'/4 = m'/4.$$

These honest nodes must also send the same vote messages to all other honest nodes. Therefore, the honest leader q observe $> m'/4$ votes for Λ. As two different vote messages of the same view from the same node are ignored, the honest leader q observes $> m'/4$ votes for at most two logs conflicting with Λ. Therefore, with probability at least $1/3$, the honest leader q chooses to propose a log extending Λ, and all honest nodes awake at time $t = \Delta$ of view v accept the proposal. □

7 Practical Recovery

Recall for the PKI setting, we have been assuming that messages are buffered for an arbitrarily long time and delivered when recipients wake up (Section 3). This assumption is only made for ease of presentation. In fact, the protocols in Section 4 and 5 do not even need this assumption. The protocol in Section 6 uses a much weaker assumption: messages are buffered 2Δ time; this is because vote messages sent in $t = \Delta$ of view v are used to compute $\mathcal{L}_v$ in $t = \Delta$ of the next view $v + 1$. Thus, in theory, we can easily get rid of this assumption. However, an important point to note is that our protocols presented thus far assume that every vote or propose message carries the entire sequence of blocks in a log. This is impractical as the size of a log is unbounded. In practice, these messages contain only a hash digest of the log, and the blocks in the log must be disseminated separately. A node may be temporarily missing some blocks if it goes to sleep for some time during the protocol. These blocks must be recovered from other awake nodes after it wakes up. For this process, we can adopt the model formalized in [26]. Basically, there is a recovery period when an honest node wakes up from its sleep; an honest node is considered awake only after it has recovered the previously decided blocks from other nodes.

8 Conclusion

We have presented Byzantine atomic broadcast protocols in the sleepy model that have concretely small (3Δ or 2Δ) latency and tolerate fluctuating or growing faulty nodes' participation. Our protocols use messages from the immediate last round (or 2 rounds) to decide the next move, which protects from costless simulation and allows us to handle faulty nodes' dynamic participation. We finalize this paper with a few open questions below:

Optimal latency. An interesting question is the tightness of the latency lower bound (i.e., Δ). As a propose-then-vote style protocol necessarily takes 2 rounds, we have to make at least one of these rounds independent of the Δ waiting step, which is technically challenging.

Deterministic solution. Our protocol in Section 5 achieves perfect safety but it still uses randomness (from VRF) and its liveness is not a perfect guarantee. It is an interesting open question whether a deterministic protocol exists or not. We conjecture it is impossible to have a fully deterministic protocol, and the impossibility proof for the *permissionless model* [23, 29] might be applied to the sleepy model as well.

Modeling a fluctuating adversary with PKI. Handling forward simulation in the PKI settings is hard. In the real world, however, malicious nodes do not trust each other and are not willing to reveal their secret keys before going down. It is interesting to formalize malicious nodes going down while hiding their keys.

Acknowledgement

We would like to thank Lorenzo Alvisi, Ittay Eyal, Jacob Leshno, Kartik Nayak, Youer Pu, for valuable discussions. This work is supported in part by NSF award 2143058.

References

[1] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous byzantine agreement with expected $o(1)$ rounds, expected $o(n^2)$ communication, and optimal resilience. In *Financial Cryptography and Data Security (FC)*, pages 320–334. Springer, 2019.

[2] Ittai Abraham, Kartik Nayak, Ling Ren, and Zhuolun Xiang. Good-case latency of byzantine broadcast: A complete categorization. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 331–341, 2021.

[3] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 913–930, 2018.

[4] Vivek Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. Prism: Deconstructing the blockchain to approach physical limits. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 585–602, 2019.

[5] Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *Journal of the ACM (JACM)*, 32(4):824–840, 1985.

[6] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *3rd Symposium on Operating Systems Design and Implementation (OSDI)*, pages 173–186. USENIX, 1999.

[7] Flaviu Cristian, Houtan Aghili, Ray Strong, and Danny Dolev. Atomic broadcast: From simple message diffusion to byzantine agreement. *Information and Computation*, 118(1):158–179, 1995.

[8] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In *Financial Cryptography and Data Security (FC)*, pages 23–41. Springer, 2019.

[9] Francesco D'Amato, Joachim Neu, Ertem Nusret Tas, and David Tse. No more attacks on proof-of-stake ethereum? *arXiv preprint arXiv:2209.03255*, 2022.

[10] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 66–98. Springer, 2018.

[11] Soubhik Deb, Sreeram Kannan, and David Tse. Posat: proof-of-work availability and unpredictability, without the work. In *International Conference on Financial Cryptography and Data Security*, pages 104–128. Springer, 2021.

[12] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[13] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, 1988.

[14] Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 148–161, 1988.

[15] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[16] Matthias Fitzi, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Parallel chains: Improving throughput and latency of blockchain protocols via parallel composition. *IACR Cryptology ePrint Archive, Report 2018/1119*, 2018.

[17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *26th Symposium on Operating Systems Principles (SOSP)*, pages 51–68, 2017.

[18] Vipul Goyal, Hanjun Li, and Justin Raizes. Instant block confirmation in the sleepy model. In *Financial Cryptography and Data Security (FC)*, 2021.

[19] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. *Journal of Computer and System Sciences*, 75(2):91–112, 2009.

[20] P. Khanchandani and R. Wattenhofer. Byzantine agreement with unknown participants and failures. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 952–961. IEEE Computer Society, 2021.

[21] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference (CRYPTO)*, pages 357–388. Springer, 2017.

[22] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[23] Andrew Lewis-Pye and Tim Roughgarden. Byzantine generals in the permissionless setting. *arXiv preprint arXiv:2101.07095*, 2021.

[24] Songze Li and David Tse. Taiji: Longest chain availability with bft fast confirmation. *arXiv preprint arXiv:2011.11097*, 2020.

[25] Atsuki Momose and Ling Ren. Optimal communication complexity of authenticated byzantine agreement. In *International Symposium on Distributed Computing (DISC)*, 2021.

[26] Atsuki Momose and Ling Ren. Constant latency in sleepy consensus. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022.

[27] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[28] Rafael Pass and Elaine Shi. The sleepy model of consensus. In *Annual International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 380–409. Springer, 2017.

[29] Youer Pu, Lorenzo Alvisi, and Ittay Eyal. Safe permissionless consensus. *IACR Cryptology ePrint Archive, Report 2022/796*, 2022.

A Latency Lower Bound

We show that any Byzantine atomic broadcast cannot make a decision (even in a good-case) in less than Δ time.

Definition of latency. We first need to define the latency in compliance with the measure applied to our protocols. We say an atomic broadcast protocol has a latency of τ if the following holds: If all honest nodes awake at time 0 receive a value x at time 0, all honest nodes decide x at a log position by time τ with non-negligible probability.

With the definition above, we show the lower bound as follows. Note that it holds even for the original sleepy model that does not allow faulty nodes' dynamic participation.

Theorem 4. *There does not exist a Byzantine atomic broadcast protocol in the sleepy model that has a latency of less than Δ.*

The proof basically follows the proof of the necessity of a message delay upper bound Δ in [28].

Proof. Suppose, for contradiction, there exists such a protocol. We consider two sets of nodes P and Q. Consider three executions as follows:

W1. Nodes in P are always awake and honest, but nodes in Q are asleep. All messages are delivered instantly. By definition, nodes in P decide a log Λ in less than Δ with non-negligible probability.

W2. Symmetric to W1. Nodes in Q are always awake and honest, but nodes in P are asleep. All messages are delivered instantly. By definition, nodes in P decide a log Λ' in less than Δ with a non-negligible probability.

W3. Nodes in P and Q are both honest and awake. Messages across P and Q are delivered with delay Δ, and those sent inside each set of nodes are delivered instantly.

In W3, if nodes in P have the same input values as W1, they cannot distinguish from W1 until time Δ, so they decide Λ with non-negligible probability. Likewise, if Q have the same input values as W2, they cannot distinguish from W2 until time Δ, so they decide Λ' with non-negligible probability. If P and Q input different sets of values, Λ and Λ' will conflict with each other; safety is violated with non-negligible probability. □