Safe Permissionless Consensus

Youer Pu
Cornell University, Ithaca, NY, USA

Lorenzo Alvisi
Cornell University, Ithaca, NY, USA

Ittay Eyal
Technion, Haifa, Israel

Abstract

Nakamoto's consensus protocol works in a permissionless model, where nodes can join and leave without notice. However, it guarantees agreement only probabilistically. Is this weaker guarantee a necessary concession to the severe demands of supporting a permissionless model? This paper shows that, at least in a benign failure model, it is not. It presents Sandglass, the first permissionless consensus algorithm that guarantees deterministic agreement and termination with probability 1 under general omission failures. Like Nakamoto, Sandglass adopts a *hybrid synchronous* communication model, where, at all times, a majority of nodes (though their number is unknown) are correct and synchronously connected, and allows nodes to join and leave at any time.

2012 ACM Subject Classification Computer systems organization → Dependable and fault-tolerant systems and networks

Keywords and phrases Consensus, Permissionless, Nakamoto, Deterministic Safety

Digital Object Identifier 10.4230/LIPIcs.DISC.2022.33

Related Version *Full Version*: `https://eprint.iacr.org/2022/796.pdf` [25]

Funding This work was supported in part by the NSF grant CNS-CORE 2106954, BSF and IC3.

1 Introduction

The publication of Bitcoin's white paper [22], besides jumpstarting an industry whose market is expected to reach over $67B by 2026 [27], presented the distributed computing community with a fundamental question [12]: how should the agreement protocol at the core of Nakamoto's blockchain construction (henceforth, *Nakamoto's Consensus* or *NC*) be understood in light of the combination of consensus and state machine replication that the community has studied for over 30 years? The similarities are striking: in both cases, the goal is to create an append-only distributed ledger that everyone agrees upon, which NC calls a *blockchain*. But so are the differences. Unlike traditional consensus algorithms, where the set of participants n is known and can only be changed by running an explicit reconfiguration protocol, Nakamoto's consensus is *permissionless*: it does not enforce access control and allows the number and identity of participants to change without notice. It only assumes that the computing power of the entire system is bounded, which effectively translates to assuming the existence of an upper bound $\mathcal{N}$ on the number of participants. [1]

To operate under these much weaker assumptions, NC adopts a new mechanism for reaching agreement: since the precise value of n is unknown, NC forsakes explicit majority voting and relies instead on a *Proof of Work* (PoW) lottery mechanism [22], designed to

[1] The bound can be trivial, e.g., equal to the number of atoms in the Universe, but it needs to exist; otherwise, if it would be possible for a large, unknown group of nodes to be secretly adding blocks onto a different branch of the blockchain, and Nakamoto's decisions would never be, even probabilistically, safe.

36th International Symposium on Distributed Computing (DISC 2022).
Editor: Christian Scheideler; Article No. 33; pp. 33:1–33:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

drive agreement towards the blockchain whose construction required the majority of the computational power of all participants. Finally, whereas traditional consensus protocols guarantee agreement deterministically, NC can do so only probabilistically; furthermore, that probability approaches 1 only as termination time approaches infinity. Is settling for these weaker guarantees the inevitable price of running consensus in a permissionless setting?

In this paper we show that, at least in a benign failure model, one can do much better. We present *Sandglass*, a permissionless consensus algorithm that guarantees *deterministic* agreement and terminates with probability 1. It operates in a model based on Nakamoto's. Our model allows an arbitrary number of participants to join and leave the system at any time and stipulates that at no time the number of participants exceeds an upper bound $\mathcal{N}$ (though the *actual* number n of participants at any given time is unknown). Further, like Nakamoto's, it is *hybrid synchronous*, in that, at all times, a majority of participants are correct and able to communicate synchronously with one another. We call these participants *good*; our protocol's safety and liveness guarantees apply to them. Participants that are not good (whether because they crash, perform omission failures, and/or experience asynchronous network connections) we call *defective.* Sandglass proceeds in asynchronous rounds, with a structure surprisingly reminiscent of Ben-Or's classic consensus protocol [3]. Let's review it. Nodes propose a value by broadcasting it; in the first round, each node proposes its initial value; in subsequent rounds, nodes propose a value chosen among those received in the previous round. Values come with an associated priority, initialized to 0. The priority of v depends on the number of consecutive rounds during which v was the only value received by the node proposing v – whenever a node receives a value other than v, it resets v's priority back to 0. When proposing a value in a given round, node p selects the highest priority value received in the previous round; if multiple values have the same priority, then it selects randomly among them. A node can safely decide a value v after sufficiently many consecutive rounds in which the proposals it receives unanimously endorse v (i.e., when v's priority is sufficiently high); and termination follows from the non-zero probability that the necessary sequence of unanimous, consecutive rounds will actually eventually occur.

Of course, embedding this structure in a permissionless setting introduces unprecedented challenges. Consider, for example, how nodes decide. In Ben-Or, a node decides v after observing two consecutive, unanimous endorsements of v; it can do so safely because any two majority sets of its fixed set of n nodes intersect in at least one correct node. This approach is clearly no longer feasible in a permissionless setting, where n is unknown and the set of nodes can change at any time.

Instead, Sandglass's approach to establish safety is inspired by one of the key properties of Nakamoto's PoW: whatever the value of n, whatever the identity of the nodes participating in the protocol at any time, the synchronously connected majority of *good nodes* will, in expectation, be faster than the remaining nodes in adding a new block to the blockchain.

Think now of adding a block b at position i of the blockchain as implicitly starting a new round of consensus for all the chain's positions that precede i; for each position, the new round proposes the corresponding block in the hash chain that ends at b. In this light, the greater speed in adding blocks that PoW promises to the majority of connected nodes translates into these nodes moving faster from one asynchronous round to the next in each of the consensus instances.

This insight suggests an alternative avenue for achieving deterministic consensus among good nodes – without relying on quorum intersection. Node p should decide on a value v only after it has seen v unanimously endorsed for sufficiently many rounds that, if p is good, the lead p (and all other good nodes) have gained over any defective node q proposing some other value is so large that q's proposals can no longer affect the proposal of good nodes.

Why can't the same approach be used to achieve deterministic consensus in Nakamoto's original protocol? Because Nakamoto's PoW mechanism, notwithstanding its name, is an indirect and imperfect vehicle for proving work. As evidence of performed work, Nakamoto presents the solution to a puzzle: this solution, however, could just have been produced as a result of a lucky guess. Thus, however unlikely, it is always possible in NC for defective nodes proposing a value other than v to catch up with, or even overtake, good nodes and reverse their decisions.

To avoid this danger, Sandglass relies on a different PoW mechanism, which ties the ability to propose a value to a *deterministic* amount of work. In particular, Sandglass nodes can propose a value in any round other than the first only after they have received a specific threshold of messages from the previous round. Therefore, each proposed value implicitly represents all the work required to generate the messages needed to clear the threshold. The threshold value is chosen as a function of the upper bound $\mathcal{N}$ on the number of nodes that at any time run the protocol, in such a way that, whatever is their actual number n, any node that does not receive messages from good nodes will inevitably take longer than them in moving from round to round.

The full power of this PoW mechanism, however, comes from pairing it with the idea, which we borrow from Ben Or, of associating a priority with the values being proposed. With a fixed set of n nodes, Ben Or leverages priorities and quorum intersection to safely decide a value v once it has reached priority 2, because it can guarantee that henceforth every node executing in the same round as a correct node will propose v. In a permissionless setting, we show that the combination of priorities and our PoW mechanism allows Sandglass to offer good nodes the same guarantee (though, as we will see, v will be required to reach a significantly higher priority value!). Intuitively, by the time v reaches the priority necessary to decide, any node q that manages not to fall behind (and thus become irrelevant) to the unanimous majority of good nodes who have kept proposing v must have received *some* of the messages proposing v from some good nodes. Furthermore, to keep up, q must have received such messages often enough that, given how the priority of received values determines what a node can propose, it would be impossible for q to propose any value other than v.

In summary, this paper makes the following contributions: (i) it formalizes Nakamoto's permissionless model in the vocabulary of traditional consensus analysis; (ii) it introduces novel proof strategies suitable for this new model; (iii) it exposes the connection between PoW and a voting mechanism that can be implemented by message passing; and (iv) it introduces Sandglass, the first protocol that achieves deterministic agreement in a permissionless setting under hybrid synchrony.

2 Related work

The consensus problem has been studied for decades, covering both benign and Byzantine faults under different synchrony assumptions. Common across these classic works is the assumption that the set of nodes that participate in running the protocol is either constant or changes through an agreement among the incumbents (*reconfiguration*). In contrast, Sandglass allows for participants to change arbitrarily and without any coordination, as long as at all times a majority of nodes is correct and synchronously connected. More recent papers also explore models where participants can change dynamically at any time, subject to guarantees of well-behaved majority; unlike Sandglass, those works achieve only probabilistic safety guarantees. We briefly review related prior work in more detail below.

The permissionless nature of our model implies that consensus solutions for classical models (e.g., [13]) do not apply. For synchronous networks, previous solutions rely on the fact that the number of failures is bounded in a period of time. They tolerate up to $(n-1)$ benign failures [29] or Byzantine failures with authentication [7, 18]. For an asynchronous network, Fisher, Lynch, and Paterson [8] show that it is impossible to solve consensus with deterministic safety and liveness even with a single crash failure. Various protocols (e.g., [16, 26, 28]) thus either solve asynchronous consensus with weaker liveness guarantees than deterministic termination, or provide deterministic termination after a Global Stabilization Time (GST) (e.g., [4]). They use logical rounds, and for each round collect messages from a sufficient number of (authenticated) nodes, tolerating fewer than $\frac{n}{2}$ failures in a benign failure model [3, 17], and $\frac{n}{3}$ failures with Byzantine failures and authentication [4, 31]. Although our model is not directly comparable, we note that our protocol matches the $(n/2)$ bound of a benign model in an asynchronous network, despite assuming synchrony among good nodes.

Aspnes et al. [2] explore the consensus problem in an asynchronous benign model where an unbounded number of nodes can join and leave [9], but where at least one node is required to live forever, or until termination. It is easy to see that in their model, but without this latter assumption, deterministic safety is impossible. In contrast, Sandglass, in a hybrid synchronous model, guarantees deterministic safety while allowing *all* nodes to freely join and leave.

Consider two groups of good nodes with different initial values running from $t = 0$, with messages within the groups delivered immediately, but messages between the two groups are delayed until at least one in each group decides. By validity of consensus, the two groups will decide on different values, which violates agreement.

A newer line of work, starting with Nakamoto [22], studies systems where principals can unilaterally join or leave without notifying previous participants. These protocols (e.g., [21, 30]) are based on probabilistic assumptions and provide probabilistic guarantees. Specifically, participation is based on probabilistic proofs of work, and the assumption that no minority can find most proofs of work in a long period. They provide safety with high probability, given a sufficiently long running time (latency) [6, 15, 23, 10]. Nonetheless, they are all based on probabilistic techniques and provide probabilistic guarantees, which cannot be directly translated to deterministic guarantees.

Several protocols, inspired by the PoW approach, achieve consensus among a large group of principals while requiring the active participation of only a subset of them. In the Sleepy Model [24] participants join and leave ("sleep"); the assumptions and guarantees of the consensus protocol presented for this model are as probabilistic as those of pure proof of work. Momose and Ren [20] present a consensus protocol in the Sleepy Model with constant latency and deterministic agreement; however, their protocol does not guarantee progress until the participation is stable. Ouroboros [14, 5] forms a chain in the spirit of PoW but using internal tokens for the random choice of participants, again leading to probabilistic guarantees. In Algorand [11], committees elect one another in a series of reconfigurations, with assumptions and guarantees similar to classical consensus, except that participants are chosen at random from a large pool, with a negligible probability of a Byzantine majority – again, providing probabilistic guarantees.

In contrast, despite its permissionless model, our protocol guarantees deterministic safety and terminates with probability 1. We note other differences: Sandglass's failure model assumes only benign failure and is thus stronger than the Byzantine model adopted by many of these works, but its network assumptions are weaker, as defective nodes can experience asynchronous communication, and all nodes can join or leave instantaneously.

Abraham and Malkhi [1] formalize Nakamoto's Consensus within a classical disturbed systems framework, and in particular abstract the PoW primitive as a Pre-Commit, Non-Equivocation, Leader Election (PCNELE) Oracle. However, the power of this probabilistic oracle is similar to that of Nakamoto's PoW and yields a consensus protocol that provides only probabilistic guarantees as Nakamoto.

Lewis-Pye and Roughgarden [19] show that deterministic consensus cannot be achieved in a permissionless synchronous model with Byzantine nodes, let alone in a partially synchronous model (where communication becomes synchronous only after some point unknown to the processes). Sandglass shows, for the first time, that deterministic safety and termination with probability 1 can be achieved in a permissionless model, though the network is hybrid-synchronous rather than synchronous. The exploration of a Byzantine model remains for future work.

3 Model

The system comprises an infinite set of nodes $p_1, p_2, \ldots$. Time progresses in discrete steps; in each step, a subset of the nodes is *active* and the rest are *inactive*. At each step, active nodes are partitioned into *good* and *defective* subsets.

We assue a hybrid synchronous model. *Good* nodes are correct, and the network that connects them to one another is synchronous; at all times, a majority of active nodes are good. *Defective* nodes may suffer from benign failures, such as crashes and omission failures, or simply lack a synchronous connection with some good node.

The system progress is orchestrated by a *scheduler*. In each step, the scheduler can activate any inactive node p_i (we say that p_i has *joined* the system) and deactivate an active node (which then *leaves* the system). The scheduler chooses which nodes to activate and deactivate arbitrarily, subject only to the following three constraints: (i) The upper bound of active nodes in any step is $\mathcal{N}$; (ii) there is at least one active node in every step; and (iii) in every step the majority of active nodes is good.

In each step where it is active, each node p_i executes the stateful protocol shown as procedure *Step* in Sandglass's pseudocode (see Algorithm 1). It can execute computations, update its state variables, and communicate with other nodes with a broadcast network. In particular, since Sandglass assumes benign failures, every active node, whether good or defective, waits for a full step to elapse before sending its next message.

The network allows each active node to *broadcast* and *receive* unauthenticated messages. Node p_i broadcasts a message m with a $\mathit{Broadcast}_i(m)$ instruction and receives messages broadcast by itself and others with a $\mathit{Receive}_i$ instruction. The network does not generate or duplicate messages, *i.e.*, if in step t a node p_i receives message m with $\mathit{Receive}_i$, then m was sent in some step $t' < t$.

The communication model is designed to capture the design of Nakamoto's consensus, which relies on an underlying network layer to propagate and store blocks. Nakamoto's network layer provides a shared storage of data structures, called blocks, and guarantees delivery of published blocks within a bounded time. Each block includes cryptographically secure references to all blocks seen by its creator. This allows a newly joined node to receive and validate the entire history of published blocks. Thus, in our model, the scheduler determines when each message is delivered to each node under the following constraints.

First, propagation time is bounded between any pair of good nodes. Formally: if a good node p_i calls $\mathit{Broadcast}_i(m)$ in step t, and if a good node p_j calls $\mathit{Receive}_j$ in step $t' > t$, then m is returned, unless it was already received by p_j in an earlier call to $\mathit{Receive}_j$. Thus, a newly activated good node is guaranteed, upon executing its first *Receive*, to receive all messages from other good nodes broadcast in the steps prior to its activation.

Second, the network is reliable, but there is no delivery bound unless both nodes are good. Formally: For any two nodes p_i and p_j, where at least one of p_i and p_j is defective, and for a message m broadcast by p_i, if node p_j calls $Receive_j$ infinitely many times, then m is eventually delivered.

Each node is initiated when joining the system with an initial value $v_i \in \{\mathsf{a}, \mathsf{b}\}$. An active node p_i can decide by calling a $Decide_i(v)$ instruction for some value v. The goal of the nodes is to reach a consensus based on these values:

▶ **Definition 1** (Agreement). *If a good node decides a value v, then no good node decides a value other than v.*

▶ **Definition 2** (Validity). *If all nodes that ever join the system have initial value v and any node (whether good or defective) decides, then it decides v.*

▶ **Definition 3** (Termination). *Every good node that remains active eventually decides.*

4 Protocol

To form an intuition for the mechanics of Sandglass, it is useful to compare and contrast it with Ben-Or. From a distance, the high-level structure of the two protocols is strikingly similar: execution proceeds in asynchronous rounds; progress to the next round depends on collecting a threshold of messages sent in the current round; safety and liveness depend on the correctness of a majority of nodes; and nodes decide a value v when, for sufficiently many consecutive rounds, all the messages they collect propose v. But looking a little closer, the differences are equally striking. On the one hand, Sandglass's notion of node correctness and its hybrid synchronous model are stronger than Ben-Or's. Sandglass assumes a majority of good nodes that are not only free from crashes and omissions, but also synchronously connected to one another. On the other hand, in Sandglass, unlike Ben-Or, the number n of nodes running the protocol is not only unknown, but may be changing all the time. These differences motivate four key aspects that separate the two protocols:

Choosing a threshold In Ben-Or, a node advances to a new round only after having received a message from a majority of nodes. This strict condition for achieving progress is critical to how Ben-Or establishes Agreement. Any node that, from a majority of the nodes in round r, receives a set of messages that unanimously propose v, can be certain that (i) there cannot exist in r also a unanimous majority proposing a value other than v and (ii) no node can proceed to round $(r + 1)$ unaware that v is among the values proposed in round r. Nodes that isolate themselves from a majority simply do not make any progress; and since all majority sets intersect, nodes cannot make contradictory decisions.

Unfortunately, this approach is unworkable in Sandglass: when the cardinality and membership of the majority set can change at any time, receiving messages from a majority can no longer serve as a binary switch to trigger progress. More generally, thresholds based on the cardinality of the set of nodes from which one receives messages become meaningless. Instead, Sandglass allows nodes to broadcast multiple messages during a round, one in each of the round's steps, and lets nodes move to round $(r + 1)$ once they have collected a specified threshold of messages sent in round r.

Think of the threshold $\mathcal{T}$ of messages that allows a node to move to a new round as the number of grains of sands in a sandglass: a node (figuratively) flips the sandglass at the beginning of a round, and cannot move to the next until all $\mathcal{T}$ sand grains have moved to the bottom bulb. The value of $\mathcal{T}$ is the same for all nodes; the speed at which messages are collected, however – the width of their sandglass's neck – is not, and can change from step to step: if all nodes broadcast messages at the same rate, the larger the number of nodes that one receives messages from in a timely fashion, the faster it will be to reach the threshold. Thus, while in Sandglass setting a threshold cannot altogether prevent nodes that don't receive messages from a majority from making progress, it ensures that they will progressively fall behind those who do.

Exchanging messages In each step of the protocol, a node currently in round r (i) determines, on the basis of the messages received so far, what is the largest round $r_{max} \geq r$ for which it has received the required threshold of messages and (ii) broadcasts a message for round r_{max}, which includes the node's current proposed value, as well as the critical *metadata* discussed below.

Keeping history Unlike Ben-Or, Sandglass allows nodes to join the system at any time. To bring a newly activated node up to speed, each message broadcast by a node p in round r carries a *message coffer* that includes (i) the set of messages (at least $\mathcal{T}$ of them) p collected in round $r-1$ to advance to round r; (ii) recursively, the set of messages in those messages' coffers; and (iii) the set of messages p collected so far for round r.

Respecting priority In Ben-Or, a node decides v if, for two consecutive rounds, v is the only value it collects from a majority set. To ensure the safety of that decision, Ben-Or assigns a *priority* to the value v that a node p proposes: if v was unanimously proposed by all the messages p collected in the previous round, it is given priority 1; otherwise, 0. Nodes that collect more than one value in round r, propose for round $r+1$ the one among them with the highest priority, choosing by a coin flip in the event of a tie. Sandglass uses a similar idea, although its different threshold condition requires a much longer streak of consecutive rounds where v is unanimously proposed before v's priority can be increased. To keep track of the length of that streak, every message sent in a given round r carries a *unanimity counter*, which the sender computes upon entering r.

4.1 Selecting the Threshold

Unlike Ben-Or, Sandglass's threshold condition can not altogether prevent nodes from making progress. It is perhaps surprising that, by leveraging only the assumption that at all times a majority of nodes are good (i.e., correct and synchronously connected with each other) without ever knowing precisely how many they actually are, Sandglass retains enough of the disambiguating power of intersecting majorities to ultimately yield deterministic agreement.

In essence, Sandglass succeeds by causing defective nodes that isolate themselves from the majority of nodes in the systen to fall eventually so far behind that they no longer share the same round with good nodes. At the same time, it ensures that, once some good node has decided on a value v, nodes that manage to keep pace with good nodes will never propose anything other than v.

Of course, to obtain this outcome it is critical to set $\mathcal{T}$ appropriately. Consider two nodes, one good and one defective, and suppose they flip their sandglass at the same time – *i.e.*, they enter a new round in the same step. We want that, independent of how the number of active nodes may henceforth vary at each step, if the defective node only receives messages from other defective nodes (*i.e.*, if it fails to hear from a majority of nodes), it will reach the threshold $\mathcal{T}$ at least one step later than the good node will. The following lemma shows that setting $\mathcal{T}$ to $\lceil \frac{\mathcal{N}^2}{2} \rceil$ (where $\mathcal{N}$ is the upper bound on the maximum number of nodes active in any step) does the trick.

▶ **Lemma 4.** *For any k, consider any time interval comprising $(k+1)$ consecutive steps. Let the number of messages generated by good nodes and defective nodes in each step of the interval be, respectively, $g_0, g_1, ..., g_k$ and $d_0, d_1, ..., d_k$. Setting the threshold $\mathcal{T}$ to $\lceil \frac{\mathcal{N}^2}{2} \rceil$ ensures that, if $\Sigma_{i=1}^{i=k-1} g_i < \mathcal{T}$, then $\Sigma_{i=0}^{i=k} d_i < \mathcal{T}$.*

Proof. Note how the lemma does not count the messages generated by good nodes in the steps at the two ends of the interval. Recall that moving from the current round to the next requires a node to receive at least a threashold $\mathcal{T}$ of messages sent in the current round. Thus, we drop good messages from step 0 because good nodes that in step 0 enter a new round r are unable to count against the threshold for round r messages generated by good node that in step 0 are still in round $r-1$. And we similarly drop step k because good nodes may only need one of the messages sent by good nodes in step k to move to a new round – and have no use for the remaining messages in g_k.

We begin by observing that, when k is either 0 or 1, the lemma trivially holds, since in all steps defective nodes generate fewer than $\mathcal{N}$ messages. For example, when $k = 1$, $d_0 + d_1 < \frac{\mathcal{N}}{2} + \frac{\mathcal{N}}{2} = \mathcal{N} \leq \lceil \frac{\mathcal{N}^2}{2} \rceil$. We then prove the lemma for $k \geq 2$.

Let $\bar{g} = \frac{\Sigma_{i=1}^{i=k-1} g_i}{k-1}$ and $\bar{d} = \frac{\Sigma_{i=1}^{i=k-1} d_i}{k-1}$ denote, respectively, the average number of messages generated by good nodes and by defective nodes during the $k-1$ steps that include all but the interval's first and last step. Expressed in terms of $\bar{g}$ and $\bar{d}$, the lemma requires us to show that, if $\bar{g} \cdot (k-1) < \mathcal{T}$, then $\Sigma_{i=0}^{i=k} d_i = d_0 + \bar{d} \cdot (k-1) + d_k < \mathcal{T}$ when $\mathcal{T}$ is chosen to equal $\lceil \frac{\mathcal{N}^2}{2} \rceil$.

Assume $\bar{g} \cdot (k-1) < \mathcal{T}$; then $k - 1 < \frac{\mathcal{T}}{\bar{g}}$. Substituting for $(k-1)$ in the formula that computes the messages generated by defective nodes, we have:

$$\begin{aligned}
\Sigma_{i=0}^{i=k} d_i &= \bar{d} \cdot (k-1) + d_0 + d_k \\
&< \bar{d} \cdot \frac{\mathcal{T}}{\bar{g}} + d_0 + d_k \quad (\text{since } (k-1) < \frac{\mathcal{T}}{\bar{g}}) \\
&\leq \bar{d} \cdot \frac{\mathcal{T}}{\bar{g}} + \frac{\mathcal{N}-1}{2} + \frac{\mathcal{N}-1}{2} \quad (\text{since defective nodes are always a minority}) \\
&\leq \bar{d} \cdot \frac{\mathcal{T}}{\bar{g}} + \frac{\mathcal{T}}{\frac{\mathcal{N}^2}{2}}(\mathcal{N}-1) \quad (\text{since } \mathcal{T} = \lceil \frac{\mathcal{N}^2}{2} \rceil \geq \frac{\mathcal{N}^2}{2}) \\
&= \mathcal{T}(\frac{\bar{d}}{\bar{g}} + \frac{2(\mathcal{N}-1)}{\mathcal{N}^2}).
\end{aligned}$$

Then, to establish that $\Sigma_{i=0}^{i=k} d_i < \mathcal{T}$, it suffices to prove that $\frac{\bar{d}}{\bar{g}} + \frac{2(\mathcal{N}-1)}{\mathcal{N}^2} < 1$.

Since for any i, $d_i \leq g_i - 1$ and $d_i + g_i \leq \mathcal{N}$, we know that $\bar{d} \leq \bar{g} - 1$ and $\bar{d} + \bar{g} \leq \mathcal{N}$. Dividing both inequalities by $\bar{g}$ yields $\frac{\bar{d}}{\bar{g}} \leq \min(1 - \frac{1}{\bar{g}}, \frac{\mathcal{N}}{\bar{g}} - 1)$. Note that the largest value of $\min(1 - \frac{1}{\bar{g}}, \frac{\mathcal{N}}{\bar{g}} - 1)$ occurs when $1 - \frac{1}{\bar{g}} = \frac{\mathcal{N}}{\bar{g}} - 1$; solving for $\bar{g}$ and plugging the solution back in, gives us: $\min(1 - \frac{1}{\bar{g}}, \frac{\mathcal{N}}{\bar{g}} - 1) \leq \frac{\mathcal{N}-1}{\mathcal{N}+1}$.

Therefore, we have that $\frac{\bar{d}}{\bar{g}} + \frac{2(\mathcal{N}-1)}{\mathcal{N}^2} \leq \frac{\mathcal{N}-1}{\mathcal{N}+1} + \frac{2(\mathcal{N}-1)}{\mathcal{N}^2} = \frac{\mathcal{N}^3+\mathcal{N}^2-2}{\mathcal{N}^3+\mathcal{N}^2} < 1$. ◀

4.2 Protocol Mechanics

Protocol 1, besides showing how Sandglass initializes its key variables, presents the code that node p_i executes to take a step. Every step begins with adding all received messages, as well as the messages in their message coffers, to a single set, Rec_i (lines 4 - 5). Going over the elements of that set, p_i determines the largest round r_{max} for which it has received at least a threshold $\mathcal{T}$ of messages, and, if the condition at line 6 holds, sets the current round to $(r_{max} + 1)$ (line 7). Upon entering a new round, p_i does four things. First, after resetting

Algorithm 1 Sandglass: Code for node p_i.

```
 1: procedure INIT($input_i$)
 2:     $v_i \leftarrow input_i$; $priority_i \leftarrow 0$; $uCounter_i \leftarrow 0$; $r_i = 1$; $M_i = \emptyset$; $Rec_i = \emptyset$; $uid_i = 0$
 3: procedure STEP
 4:     for all $m = (\cdot,\cdot,\cdot,\cdot,\cdot,M)$ received by $p_i$ do
 5:         $Rec_i \leftarrow Rec_i \cup \{m\} \cup M$
 6:     if $\max_{|Rec_i(r)|\geq\mathcal{T}}(r) \geq r_i$ then
 7:         $r_i = \max_{|Rec_i(r)|\geq\mathcal{T}}(r) + 1$
 8:         $M_i = \emptyset$
 9:         for all $m = (\cdot, r_i - 1, \cdot,\cdot,\cdot, M) \in Rec_i(r_i - 1)$ do
10:             $M_i \leftarrow M_i \cup \{m\} \cup M$
11:         Let $C$ be the multi-set of messages in $M_i(r_i - 1)$ with the largest priority.
12:         if all messages in $C$ have the same value $v$ then
13:             $v_i \leftarrow v$
14:         else
15:             $v_i \leftarrow$ one of$\{a, b\}$, chosen uniformly at random
16:         if all messages in $M_i(r_i - 1)$ have the same value $v_i$ then
17:             $uCounter_i \leftarrow 1 + \min\{uCounter | (\cdot, r_i - 1, v_i, \cdot, uCounter, \cdot) \in M_i(r_i - 1)\}$
18:         else
19:             $uCounter_i \leftarrow 0$
20:         $priority_i \leftarrow \max(0, \lfloor \frac{uCounter_i}{\mathcal{T}} \rfloor - 5)$
21:         if $priority_i \geq 6\mathcal{T} + 4$ then
22:             Decide$_i(v_i)$
23:     $uid_i \leftarrow uid_i + 1$;
24:     $M_i \leftarrow M_i \cup Rec_i(r_i)$
25:     broadcast $(p_i, uid_i, r_i, v_i, priority_i, uCounter_i, M_i)$
```

its message coffer M, p_i collects in the coffer all the messages it received from the previous round – as well as the messages stored in the coffers of those messages (lines 8 - 10). Second, p_i chooses the value v that it will propose in the current round (lines 11 - 15): it picks the highest-priority value among those collected in its coffer for the previous round ; if more than one value qualifies, it chooses among them uniformly at random. Third, p_i computes the unanimity counter and the priority for all messages that p_i will broadcast during the current round (lines 16 -20). The counter represents, starting from the previous round and going backwards, the longest sequence of rounds for which all corresponding messages in p_i's coffer unanimously proposed v. The priority is simply a direct function of the value of the unanimity counter: we maintain it explicitly because it makes it easier to describe how Sandglass works. Finally, if v's priority is high enough, p_i decides v (lines 21- 22). Whether or not it starts a new round, p_i ends every step by broadcasting a message (line 25): before it is sent, the message is made unique (line 23) and p_i adds to the message's coffer all messages received for the current round (line 24).

5 Correctness: Overview

Sandglass upholds the definitions of Validity, Agreement, and Termination (with probability 1) given in Section 3. We overview the proof below, as its approach differs from proofs of classical, permissioned protocols. We defer the proof to the full version of this report [25], which includes the formal statements of the lemmas we informally state below.

Validity is easily shown by induction on the round number, since if all nodes that join have the same value, there is only one value that can be sent in each round. Establishing Agreement and Termination is significantly more involved, and hinges on a precise understanding of the kinematics of good and defective nodes – and how that interacts with the ability of good nodes to converge on decision value and on the number of rounds necessary to do so safely. How clustered are good nodes as they move from round to round? At what rate do good nodes gain ground over defective nodes that cut themselves out from receiving messages from good nodes? How often do defective nodes need to receive messages from good nodes to be in turn able to have their messages still be relevant to good nodes?

The answer to these and similar questions constitute the scaffolding of lemmas and corollaries on which the proofs of Agreement and Termination rely. We discuss it in greater detail below, before moving on to the proofs.

5.1 The Scaffolding

The protocol achieves several properties that facilitate the consensus proof.

First, it keeps good nodes close together as they move from round to round. Specifically, the following lemmas hold:

▶ **Lemma 5.** *In any step two good nodes are at most one round apart.*

▶ **Lemma 6.** *If in any step a good node is in round r, then by the next step all good nodes are guaranteed to be at least in round r.*

A key intuition that guides the proof is that defective nodes move from round to round slower than good ones. There is a complication, however: this intuition holds only when defective nodes receive messages only from their own kind; in fact they can actually advance *faster* than good ones by combining messages from good nodes with messages from defective nodes that do not reach the good nodes. Nonetheless, we can show that

▶ **Lemma 7.** *At any step a defective node is at most one round ahead of any good node.*

Second, the protocol guarantees information sharing among good nodes. This may appear trivial to establish, since good nodes are correct and synchronously connected, but the *laissez-faire* attitude of the permissionless model, with nodes joining and leaving without coordination at any step, complicates matters significantly, making it impossible to prove seemingly basic properties. For example, consider a good node p that, in round r and step T, proposes a value v with a positive *uCounter*. It would feel natural to infer that all good nodes must have proposed v in the previous round – but it would also be wrong. If p just entered r in step T, it would in fact ignore any value proposed by good nodes that newly joined the systems in step T, but are still in round $r - 1$. Fortunately, we show that a much weaker form of information sharing among good nodes is sufficient to carry the day. As a matter of terminology, let's say that a node *collects* a message in a round if it receives the message and does not ignore it (messages originated from a lower round number are ignored). We then prove the following facts about collected messages:

▶ **Lemma 8.** *In any round, a good node collects at least one message from a good node.*

▶ **Corollary 9.** *For any round, there exists a message from a good node that is collected by all good nodes.*

Third, it allows us to establish the basis for a key insight about the kinematics of Sandglass nodes that will be crucial for proving Agreement and Termination: in the long run, the only values proposed by defective nodes that remain relevant to the outcome of consensus are those that have been, in turn, recently influenced by values proposed by good nodes. This insight stands on a series of intermediate results. We already saw (Lemma 4) that, given any sequence of steps, if good nodes cannot generate enough messages to get into the next round, neither can defective nodes, even if they, unlike good nodes, are allowed to count messages generated in the two steps at the opposite ends of the period. It follows that

▶ **Lemma 10.** *During the steps that good nodes spent in a round, defective nodes can generate fewer than the $\mathcal{T}$ messages necessary to move to the next round.*

It all ultimately leads to the following lemma, which quantifies the slowdown experienced by defective nodes that don't allow themselves to be contaminated by good nodes:

▶ **Lemma 11.** *Defective nodes that do not collect any message from good nodes for $k\mathcal{T}$ consecutive rounds fall behind every good node by at least $(k-1)$ rounds.*

5.2 Agreement

The intuition behind our proof of Agreement is simple. To each value v proposed and collected by Sandglass nodes is associated a *uCounter*, which records the current streak of consecutive rounds for which all the messages collected by the proposer of v were themselves proposing v. Once v's *uCounter* reaches a certain threshold, v's *priority* increases; and once the value v proposed by a node reaches a given priority threshold, then a node decides v (see Algorithm 1, line 21). Since, as we saw, good nodes share information from round to round (recall Corollary 9), proving Agreement hinges on showing that, once a good node decides v, no good node will ever propose a value other than v. To prove that, we must in turn leverage what we learned about the kinematics of Sandglass nodes to identify a priority threshold that makes it safe for good nodes to decide. It should be large enough that, after it is reached, it becomes impossible for a defective node to change the proposal value of any good node.

The technical core of the Agreement proof then consists in establishing the truth of the following claim:

▷ Claim 12. Let p_d be the earliest good node to decide, in round r_d at step T_d. Suppose p_d decides v_d. Then, any good node p_g that in any step (whether before, at, or after T_d) finds itself in a round r_g that is at least as large as r_d, proposes v_d for r_g with *priority* at least 1. [2]

It is easy to see that if the above claim holds, then Agreement follows. Say that T_d is the earliest step in which a good node p_d, currently in round r_d, decides v_d. The claim immediately implies that no good node can decide a value other than v_d in a round greater or equal to r_d, since, from r_d on, every good node proposes v_d. Recall that, since good nodes are never more that one round apart at any step (Lemma 5), the earliest round a good node can find itself at T_d is $(r_d - 1)$; and that, by Lemma 6, every good node is guaranteed to be at least in round r_d by step $(T_d + 1)$. All that is left to show then is that no good node p', which at T_d found itself in round $(r_d - 1)$, can decide some value v' other than v_d. To this end, we leverage the information sharing that we proved exists among good nodes.

[2] Although proving Agreement does not require that v_d be proposed with priority at least 1, it makes proving the claim easier.

By Corollary 9, there is at least a message m generated in round $(r_d - 2)$ by a good node that is collected by all the good nodes. Since p_d at T_d has reached the priority threshold required to decide v_d, m must have proposed v_d; but if so, it would be impossible for good node p', which also must have collected m, to have reached the priority threshold required to decide a different value v'.

Proving Claim 12 is non trivial. The core of the proof consists in showing that any node that proposes a value v' other than the decided value v_d must find itself, at T_d, in a much earlier round than the earliest round occupied by any good node. In fact, we show something stronger: we choose a priority threshold large enough that any node, whether good or defective, that at T_d or later is within earshot of a good node (*i.e.*, whose message m can be collected by a good node), not only proposes v_d, but it does so with a *uCounter* large enough that allows whoever collects m to propose v_d with priority at least 1.

To see why those who propose v' are so far behind good nodes, note that the good node p_d that decided v_d at T_d must have received only messages proposing v_d for a long sequence of rounds, so long as to push v_d's priority over the $(6\mathcal{T} + 4)$ threshold required for a decision. Let's zoom in on that sequence of rounds. It took $6\mathcal{T}$ unanimous rounds for v_d to reach priority 1 (see Algorithm 1, line 20); after clearing that first hurdle, v_d's priority increased by 1 every $\mathcal{T}$ rounds.

Consider now the set S of messages collected by p_d during the long climb that took v_d's priority from 1 to $(6\mathcal{T} + 4)$. Any node p' that during this climb proposes something other than v_d faces a dilemma. It can either refuse to collect any message in S – but if it does so, it will advance more slowly than good nodes, and, by the time v_d's priority reaches the decision threshold, it will be so far behind that no good node will collect its messages. Or p' can try to keep up by collecting messages from S – but, if it wants to keep proposing $v' \neq v_d$, it can do so in at most one round during the entire climb: since the first message collected from S would reset v' priority to 0, any further message from S collected by p' in later rounds would have higher priority than the one of v', forcing p' to henceforth propose v_d instead of v'.

In short, since p' can collect messages from S in at most one round, to ensure that any node that in round r_d is within earshot of good nodes will propose v_d it suffices to choose a large enough priority threshold for deciding. In particular, setting the threshold to $(6\mathcal{T} + 4)$ ensures that (i) all messages collected by good nodes for round $(r_d - 1)$ will propose v_d, and (ii) v_d's *uCounter* in all these messages is at least $(6\mathcal{T} - 1)$, ensuring that all good nodes in round r_d will propose v_d with *uCounter* at least $6\mathcal{T}$, *i.e.*, with priority at least 1.

Finally, a simple induction argument shows that, if all good nodes propose v_d with priority at least 1 from r_d on, then any node that, from step $(T_d + 1)$ on, continues to propose a value other than v_d, will fall ever more behind good nodes, as it will be allowed to collect messages from good nodes only once every $6\mathcal{T}$ rounds, on pain of being forced to switch its proposed value to v_d.

5.3 Termination

The Termination property requires good nodes that stay active to eventually decide. Sandglass's Termination guarantee is probabilistic: for Termination to hold, Sandglass needs to be lucky, so that it can build a sequence of consecutive rounds during which all messages collected by good nodes propose the same value; long enough that the value will reach the priority required for a node to decide. Luck is required because Sandglass allows some randomness in the values that a node proposes: nodes are required to propose the highest priority value from any message collected in the previous round, but, if they receive multiple values with the same priority, they can choose among them uniformly at random.

To help us prove that luck befalls Sandglass with probability 1, we introduce the interdependent notions of *lucky period*, *lucky value*, and *lucky round.* Intuitively, a lucky period is a sequence of steps that leads to a decision: all nodes that are active in the step that immediately follows the end of the lucky period are guaranteed to decide in that step, if not earlier. A lucky round is simply the first round of a lucky period. What is more interesting is the quality that makes a period lucky: during a lucky period, whenever Sandglass allows nodes to use randomness in picking which value they will propose in the current round, they select the same value – the *lucky value* for that round.

A minimum requirement for a round's lucky value is that it should be a *plausible* value on which good nodes may converge, in the sense that it should not explicitly go counter the value that some good node is required to propose in that round. Concretely, if the messages collected by a good node require it to propose v and all other nodes can randomly choose between v and $\overline{v}$, then the round's lucky value better not be $\overline{v}$. In addition, to encourage the possibility of a lucky period, the lucky value should be *sticky*: we would like random choices to consistently pick the same value, round after round, unless doing so would make the value implausible.

In the end, Sandglass adopts a definition of lucky value (see [25]) that, in addition to upholding plausibility, has two additional properties that express its stickiness.

▶ **Property 13.** *In every round good nodes collect at least one message that proposes the lucky value of the previous round.*

▶ **Property 14.** *For the lucky value in the current round to change, some good node must have collected a different value with priority at least 1 from the previous round.*

Property 13 guarantees that under no circumstances the previous round's lucky value will simply be forgotten when moving to a new round; building on Property 13, Property 14 establishes that lucky values don't flip easily.

To prove that Sandglass guarantees Termination with probability 1, we then proceed in two steps.

First, we show that the *uCounter* of all good nodes active in the step that immediately follows the end of the lucky period reaches a value that allows these good nodes to decide. To this end, we begin by proving that, in any lucky period, the lucky value after a while becomes *locked*: specifically, we show that the lucky value v_ℓ at round $6\mathcal{T}$ in the lucky period remains the lucky value until the end of the lucky period, and, further, that after that round all good nodes propose v_ℓ. Then, leveraging techniques similar to those used to prove Agreement, we show that any node p' that proposes a value v' other that v_ℓ must fall behind good nodes during the lucky period. The reason is that, once v_ℓ is locked, p' can collect a message from a good node only every $6\mathcal{T}$ rounds. If it did it more often, p' would collect a message proposing v_ℓ from a good node while v' has priority 0, which would force p' to change its proposal to v_ℓ – even if v' and v_ℓ both had priority 0, and p' could choose randomly among them, it would have to propose v_ℓ in the next round, since v_ℓ is the lucky value. Thus, by choosing a sufficiently long lucky period, we ensure that nodes that propose values other that v_ℓ fall so far behind good nodes that v_ℓ's priority, for any good node that is active in the step right after the end of a lucky period, reaches the threshold necessary for deciding.

Second, we show that lucky periods occur with non-zero probability, since the probability of a certain outcome of random choices for a finite number of nodes during a finite number of steps is non-zero. Since in any infinite execution lucky periods appear infinitely often, it follows that any good node that stays active, no matter when it joins, is guaranteed to eventually decide.

6 Conclusion

Sandglass shows, for the first time, that it is possible to obtain consensus with deterministic safety in a permissionless model. This result suggests that it is the probabilistic nature of its PoW mechanism, rather that its permissionless model, that prevents deterministic safety in Nakamoto's consensus. It also opens up several additional interesting questions. First among them is to understand how the interplay between permissionlessness and the hybrid synchronous model shape the boundaries of what is possible, and at what cost. As we noted, Sandglass matches the $(n/2)$ bound of a benign model in an asynchronous network, even though a majority of its nodes are synchronously connected. Perhaps at the root of this result is that in both an asynchronous model and a permissionless hybrid one it is impossible for a node to know when it has received all the messages that were intended for it. Regardless, whether there exists a protocol that achieves deterministic safety and termination in a hybrid synchronous model remains an open question. Another natural question is whether there exists a deterministic solution to consensus in a hybrid-synchronous model with Byzantine failures. Answering these questions might pave the way to a qualitative improvement of permissionless systems that would provide deterministic guarantees; or, at the very least, give us more insight about the nature of consensus.

References

1 Ittai Abraham, Dahlia Malkhi, et al. The blockchain consensus layer and bft. *Bulletin of EATCS*, 3(123), 2017.

2 James Aspnes, Gauri Shah, and Jatin Shah. Wait-free consensus with infinite arrivals. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 524–533, 2002.

3 Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*, pages 27–30. ACM, 1983.

4 Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

5 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*, pages 66–98. Springer, 2018.

6 Amir Dembo, Sreeram Kannan, Ertem Nusret Tas, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. Everything is a race and Nakamoto always wins. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 859–878, 2020.

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

8 Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. Technical report, Massachusetts Inst. of Tech., Cambridge Lab for Computer Science, 1982.

9 Eli Gafni, Michael Merritt, and Gadi Taubenfeld. The concurrency hierarchy, and algorithms for unbounded concurrency. In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, pages 161–169, 2001.

10 Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015.

11 Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th symposium on operating systems principles*, pages 51–68, 2017.

12 Maurice Herlihy. Blockchains and the future of distributed computing. In *Proceedings of the 2017 ACM Symposium on Principles of Distributed Computing (PODC '17)*, page 155, August 2017. Keynote Address.

13 Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. All you need is DAG. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 165–175, 2021.

14 Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer, 2017.

15 Lucianna Kiffer, Rajmohan Rajaraman, and Abhi Shelat. A better method to analyze blockchain consistency. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 729–744, 2018.

16 Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. Zyzzyva: speculative Byzantine fault tolerance. *Communications of the ACM*, 51(11):86–95, November 2008.

17 Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems (TOCS)*, 16(2):133–169, 1998.

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

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

20 Atsuki Momose and Ling Ren. Constant latency in sleepy consensus. *Cryptology ePrint Archive*, 2022.

21 Tal Moran and Ilan Orlov. Simple proofs of space-time and rational proofs of storage. In *Annual International Cryptology Conference*, pages 381–409. Springer, 2019.

22 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, December 2008. Accessed: 2015-07-01. URL: https://bitcoin.org/bitcoin.pdf.

23 Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 643–673. Springer, 2017.

24 Rafael Pass and Elaine Shi. The sleepy model of consensus. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 380–409. Springer, 2017.

25 Youer Pu, Lorenzo Alvisi, and Ittay Eyal. Safe permissionless consensus. Cryptology ePrint Archive, Paper 2022/796, 2022. URL: https://eprint.iacr.org/2022/796.

26 Michael O Rabin. Randomized byzantine generals. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 403–409. IEEE, 1983.

27 Research and Markets. Blockchain market with covid-19 impact analysis, by component (platforms and services), provider (application, middleware, and infrastructure), type (private, public, and hybrid), organization size, application area, and region - global forecast to 2026. https://www.researchandmarkets.com, November 2021.

28 Yee Jiun Song and Robbert van Renesse. Bosco: One-step Byzantine asynchronous consensus. In *International Symposium on Distributed Computing*, pages 438–450. Springer, 2008.

29 TK Srikanth and Sam Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2(2):80–94, 1987.

30 Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

31 Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356, 2019.