The Generals' Scuttlebutt: Byzantine-Resilient Gossip Protocols

Sandro Coretti
IOHK
sandro.coretti@iohk.io

Aggelos Kiayias
University of Edinburgh and IOHK
aggelos.kiayias@iohk.io

Alexander Russell
University of Connecticut and IOHK
alexanderrussell@uconn.edu

Cristopher Moore
Santa Fe Institute
moore@santafe.edu

ABSTRACT

One of the most successful applications of peer-to-peer communication networks is in the context of blockchain protocols, which—in Satoshi Nakamoto's own words—rely on the "nature of information being easy to spread and hard to stifle." Significant efforts were invested in the last decade into analyzing the security of these protocols, and invariably the security arguments known for longest-chain Nakamoto-style consensus use an idealization of this tenet. Unfortunately, the real-world implementations of peer-to-peer gossip-style networks used by blockchain protocols rely on a number of ad-hoc attack mitigation strategies that leave a glaring gap between the idealized communication layer assumed in formal security arguments for blockchains and the real world, where a wide array of attacks have been showcased.

In this work we bridge this gap by presenting a Byzantine-resilient network layer for blockchain protocols. For the first time we quantify the problem of network-layer attacks in the context of blockchain security models, and we develop a design that thwarts *resource restricted* adversaries. Importantly, we focus on the proof-of-stake setting due to its vulnerability to Denial-of-Service (DoS) attacks stemming from the well-known deficiency (compared to the proof-of-work setting) known as *nothing at stake*.

We present a Byzantine-resilient gossip protocol, and we analyze it in the Universal Composition framework. In order to prove security, we show novel results on expander properties of random graphs. Importantly, our gossip protocol can be based on any given bilateral functionality that determines a desired interaction between two "adjacent" peers in the networking layer and demonstrates how it is possible to use application-layer information to make the networking-layer resilient to attacks. Despite the seeming circularity, we demonstrate how to prove the security of a Nakamoto-style longest-chain protocol given our gossip networking functionality, and hence, we demonstrate constructively how it is possible to obtain provable security across protocol layers, given only bare-bone point-to-point networking, majority of honest stake, and a verifiable random function.

1 INTRODUCTION

1.1 Gossip Protocols and Byzantine Attackers

Gossip protocols. Gossip protocols [12, 18] provide an efficient mechanism to distribute information to a large set of parties. The key feature of such algorithms is their peer-to-peer operation that load balances the effort of information propagation in a way that individual nodes are only investing a modicum of effort when contributing to the delivery of a message network-wide.

As communication infrastructure for multiparty cryptography, gossip protocols have recently found wide-spread application in the context of blockchain protocols, notably with the introduction of the Bitcoin blockchain [25]. Among other things, gossip protocols are used by blockchain participants to diffuse newly found blocks. In the words of Nakamoto, blockchain protocols rely on the "nature of information being easy to spread and hard to stifle," underscoring the relevance of gossip as the underlying communication layer.[1]

The "security" of gossip protocols. Deploying gossip protocols as the communication layer of blockchain protocols adds a crucial new dimension to their design: their "security."

At the very least, gossip protocols underlying blockchains must deal with the fact that the resources (e.g., network bandwidth, computation time and space) of each peer are limited, and exhausting them will lead to denial-of-service (DoS) attacks. As such, the aforementioned consideration of keeping the complexity for each peer small (sublinear, preferably constant, in the number of total participants) is therefore not only relevant for efficiency but also essential for security.

For this reason, gossip protocols currently used in practice incorporate an array of (typically ad-hoc) measures to protect information propagation against DoS attacks. In the setting of blockchain protocols in particular, the most salient feature of such DoS mitigations is the fact that the adversary is resource-bounded (e.g., has limited hashing power, in the context of the Bitcoin protocol, or stake, in the context of proof-of-stake protocols), and peers can exploit this to manage network-wide message propagation. Techniques include rejecting previously seen proof-of-work messages and skipping content downloads that will not result in a local state update (such as skipping the download of a block's contents when the block header indicates that it cannot be adopted based on the local state of the client).

It should be stressed that such measures are far from perfect, as exemplified by a number of attacks that have been described, including eclipse attacks [17] and routing attacks [2]. Moreover, the proof-of-stake setting poses additional difficulties stemming from the possibility of reusing keys to issue numerous conflicting messages [26] and the fact that the whole stakeholder set should be at hand for proof-of-stake verification to work, in sharp contrast to proof of work, which only needs the current difficulty level.

However, the security issues extend far beyond just the per-peer complexity. Most crucially, the design, setup, and maintenance of

[1]Note that the security guarantees of gossip protocols are weaker than Byzantine-resilient Broadcast aka the Byzantine Generals problem [23] because they do not guarantee any kind of agreement on or consistent ordering of gossiped messages among parties.

the overlay used for gossiping must be such that it resists *Byzantine attackers*—who make participants actively and maliciously deviate from the prescribed protocol—that command a large number of peers in the network. Despite all of the above shortcomings, in practice the information-propagation guarantees of the deployed networking layers of blockchain protocols are generally postulated to be sufficient for the higher-level protocol to maintain its security and correctness.

On the theory side, blockchain consensus protocols—be they Nakamoto-style (e.g., [11, 25]) or inspired by Byzantine fault-tolerant computing (e.g., [7]), based on proof of work (PoW) or proof of stake (PoS)—all crucially rely on the reliable and timely delivery of protocol messages (blocks, votes, etc.) to achieve liveness and many also to be safe. However, while the consensus layers of all these protocols have received considerable attention, with a number of them achieving provable security against Byzantine attackers, the design and security of the network layer are usually an afterthought at best.

For example, previous formal security analyses of PoS protocols (e.g., [7, 10, 11]) use (over-)idealized message-passing abstractions that essentially promise that honest messages are distributed undisturbed to all honest parties within a reasonable delay window and ignore the fact that these abstractions must be implemented in the real world.

More broadly, there exists surprisingly little published work that considers the problem of extending Byzantine resilience to the communication layer of blockchains or gossip protocols in general.

The above state of affairs highlights serious shortcomings of the approaches taken both in theory and in practice and also suggests a significant gap between the two. Given that gossip is a critical piece of the protocol stack for any permissionless distributed-ledger protocol, the lack of a thorough, formal security treatment of its properties is a critical deficiency in the understanding of the security of these protocols. This the main motivation behind the present paper.

1.2 Our Results

This work takes a systematic and principled approach to alleviating the issues explained above and provides a novel design for Byzantine-resilient gossiping in the context of PoS blockchain protocols. Note, however, that the ideas generalize to settings other than PoS. The results are presented in the Universal Composability framework [5].

A Byzantine-resilient network layer for blockchains. The first main contribution of this work is a protocol for "synchronizing chains" globally among participants of PoS blockchain consensus.[2] The protocol is designed to work over a standard, Internet-like network with (bounded-delay) message passing.

Crucially, the security of the network layer is based on the same assumption as that of the consensus layer, namely that the majority of stake in the system is controlled by honest parties. This may seem circular at first, as the proper operation of the network layer is conditioned on agreement on the stake distribution. The way to break this "cycle" is as follows: Commonly, Nakamoto-style PoS blockchains anyway split the execution of the consensus protocol into epochs and use the stake distribution SD_{i-1} at the end of an epoch $i-1$ as a basis for consensus in epoch $i+1$ (under the assumption of bounded stake drift during epochs). The same approach can be taken for the network layer, i.e., SD_{i-1} underpins the execution of the network layer in epoch $i+1$. Specifically, in the new protocol parties use a verifiable random function (VRF) to, based on SD_{i-1}, create a stake-weighted random-graph overlay in which the degree of each party is constant in the number of participants. The use of VRFs allows parties to reject connection requests from participants that are not supposed to be in their neighbor set.[3]

Because the node degrees in the network protocol are constant, it is easy for an (adaptive) attacker to isolate a (bounded) fraction of the stake by corrupting all neighbors of parties making up said stake. To that end, edges in the graph have an expiration time, at which point replacements are sampled. In addition to helping parties recover from eclipses, this also allows the overlay to gradually adapt to changing stake distributions.

It may seem tempting to now simply perform run-of-the-mill block gossiping over the above overlay. Such an approach, however, seems to be unsuitable (at least) for Nakamoto blockchains: For example, it is an impractical and insecure design to ask parties to keep all received blocks around, as many of them could be adversarial in the PoS setting (an attacker may—in principle—generate as many blocks as they like, in a particular as slot leader). Therefore, with block gossip, a party should delete blocks unless they extend its current local chain. During a fork event, however, the party may require previously deleted blocks for which the gossip is "over." Similar issues apply if a party misses blocks, e.g., due to an eclipse. In general, which blocks are needed by a particular party to synchronize with the system is highly dependent on the party's local state.

Therefore, stateless solutions, i.e., solutions in which neighbors do not share state, do not appear to be a good fit and will not yield DoS-resilient and scalable blockchain protocols in practice. A more suitable approach is to have each pair of neighbors run a *bilateral chain synchronization (chain sync) protocol*, which allows them to keep each other informed about their locally preferred chains. When a party discovers a better chain in one of these chain-sync instances or when it produces a better one, it informs all of its current chain-sync instances of the new chain.

This work abstracts this bilateral chain sync as a functionality $\mathcal{F}_{\mathsf{bilateral}}$. The actual implementation of such a protocol is outside the scope of and irrelevant for this work.[4] It is merely important to note that chain sync is *stateful* and instances thereof take time to set up in the real world (establishing the connection, initial synchronization of blocks, etc.). Hence bilateral chain sync is intended to run between two parties for an extended (albeit bounded) amount of time. These facts are captured by $\mathcal{F}_{\mathsf{bilateral}}$ in that there is an initial synchronization delay δ_{init}, which may be much larger than

[2] This work focuses on synchronizing chains; similar approaches can be taken to synchronize other protocol messages, such as votes, transactions, etc.

[3] Non-Nakamoto blockchains sometimes do not have the notion of an epoch built into the consensus protocol, e.g., because they *finalize* blocks and use the stake distribution of the finalized block as a basis for agreeing on the next block. However, for the purpose of equipping them with the network design presented here, such a notion can easily be added to them. Moreover, note that it seems that any reasonable design of a network layer for PoS protocols needs to assume bounded stake drift.

[4] Secure and practical implementations of chain sync exist, e.g., [9] and [8, Section 3.7].

the delay δ_{sync} occurring once a chain sync instance has been set up.

Finally, observe that in the blockchain context particularly (but also in general), the attacker must be prevented from interrupting the propagation of any specific message. Otherwise, the attacker can, e.g., prevent honest chain updates from spreading through the network. Unfortunately, with an efficient gossip protocol, in which nodes have constant degrees, the (adaptive) attacker can simply corrupt all neighbors of a block leader and thereby halt propagation. It is therefore unavoidable to consider a model in which corruption requests by the attacker only take effect after a certain amount of time.

The ideal global chain-sync functionality. The security guarantees of the protocol above are captured by a functionality F_{sync}, the second main contribution of this work. Similarly to the network functionalities assumed in prior work, F_{sync} provides global chain "propagation" within some time bound Δ_{sync}. However, there are several important differences stemming from the fact F_{sync} is implemented and not assumed. The two most crucial ones are the following:

- F_{sync} allows the attacker to "eclipse" parties and exclude them from the provided guarantees, as long as the fraction of eclipsed stake does not exceed a certain bound. Note that the adversary is allowed to be "mobile" w.r.t. which parties are eclipsed, i.e., every party can potentially be eclipsed at some point during the execution of the protocol. Note that there is an eclipse delay $\xi_{\mathsf{ecl}} \geq \Delta_{\mathsf{sync}}$ in F_{sync}. This ensures that the attacker cannot stop the propagation of specific chains.
- Due to the use of chain sync, F_{sync}'s guarantees are slightly weaker than those offered by the assumed network functionalities in prior work: instead of a particular chain C "propagating" through the network within Δ, F_{sync} may also instead deliver different chains C' that are not worse (i.e., equal length or longer) than C.

Security proof. In order to show that the new network protocol securely realizes ideal functionality F_{sync}, this work derives a new result on expander graphs: Consider the stake-weighted random graph formed by parties in the protocol. Then, even after removing all corrupted nodes form the graph, leaving behind some fraction α of honest stake, there exists a subgraph of honest parties, the *backbone*, corresponding to at least an $(\alpha - \beta)$-fraction of the total stake, for some β, and this backbone is an expander graph (with overwhelming probability). Most importantly, the result holds even if the attacker chooses which nodes to remove *with full knowledge of the entire graph.* The expander property guarantees that the diameter in the backbone is small, and therefore timely chain "propagation" is possible therein.

Fully Byzantine-resilient PoS. As a final contribution, this work demonstrates how to utilize the synchronization functionality in the context of a proof-of-stake protocol. Specifically, [11] is used to illustrate the result. First, note that the original analysis is insufficient: despite the fact that the networking model of [11] allows a Byzantine adversary controlling a minority of stake, F_{sync} permits a "mobile" eclipsing strategy that would deplete the adversarial budget of any straightforward reduction to the adversary of [11]. To circumvent this issue, a revamped analysis is presented showing that adaptive eclipsing does not disturb the forkable-string analysis of [11], which can be recovered to demonstrate that the protocol remains secure even against an adversary that exploits the enhanced capabilities of the adversarial interface of F_{sync}. This gives rise to a new analysis of [11] in an adversarial setting where in addition to party corruption, some degree of message suppression is also permitted.

1.3 Related Work

The use of gossip or epidemic algorithms in the context of distributed systems was put forth in [13] and explored at length both in networking systems [20] but also from theoretical angles [19]. The study of Byzantine fault tolerance in a network of a bounded degree was initiated in [14] and further refined in [28], where it is shown that if the adversary is bounded by $O(n/\log n)$ or $O(n)$, respectively, there exist graphs of bounded degree that facilitate broadcast.

In terms of total communication, it is known that the communication complexity of Byzantine broadcast is $\Omega(n^2)$, and assuming some type of delay in the corruption model is necessary to break the communication complexity barrier to sublinear [1].

In the context of peer-to-peer networking for blockchains, a "structured" approach in the organization of the peer-to-peer network can be used to reduce communication complexity further, but at the expense of adaptive security [27].

Our protocols, viewed from the lens of multiparty computation, exhibit a "communication locality," which has also been studied more broadly in the context of general secure multiparty computation [6]. We also remark that message suppression as an enhanced adversarial capability was also studied in the general secure MPC setting in the context of "omission corruptions" [29].

Finally, concurrently and independently of our work, [24] studied the related problem of stateless flooding amongst a set of participants, also motivated in part by the blockchain setting. Their flooding protocol has peers connecting to a bounded-size, randomly chosen neighborhood for each message transmission. This can be seen as creating a separate random graph for every message. However, as mentioned above, stateless flooding is unsuitable in practice for blockchain protocols. Moreover, their protocol does not use any mechanism to resource bound the adversary (e.g., as a VRF) and hence there is no mitigation that can prevent honest parties from being flooded with messages by the adversary.

2 PRELIMINARIES AND NOTATION

UC security. Protocols are described and proven secure in the Universal Composability (UC) framework [5]. In UC, the security of a particular protocol is captured by comparing a real-world execution of the protocol to an ideal-world execution in which the protocol is replaced by an ideal functionality. In rough terms, a protocol π securely realizes a functionality F, if for every real-world attacker $\mathcal{A}$ attacking π, there exists an ideal-world simulator

$\mathcal{S}$ attacking F such that the real and ideal experiments become indistinguishable to all environments $\mathcal{Z}$.[5]

A protocol (in the real world) may itself make calls to so-called hybrid functionalities. These hybrid functionalities may serve to either model assumptions (e.g., F_{net} below) or are themselves realized by protocols. The UC framework guarantees that the security of a protocol is maintained when hybrids are replaced by the protocols that realize them.

Round structure. All functionalities/protocols proceed in rounds (not made explicit) and are assumed to have access to the current time, denoted $\mathbb{T}$. Generally, the round structure is such that parties first use a fetch-type command to retrieve information, followed by a send/set-type command to distribute information.

Attacker and corruption delay. The attacker considered is polynomially bounded and may corrupt parties, thereby learning their internal state, and make them deviate from the prescribed protocol arbitrarily. The attacker is adaptive, i.e., it may choose whom to corrupt on the fly during the execution of the protocol and based on all the information observed. However, there is a corruption delay of ξ_{corr}, i.e., a corruption request issued by the attacker takes effect after a delay of ξ_{corr} rounds only.

Underlying network. This work considers parties P with so-called *relays*, identified by their IP addresses IP. Having the actual node—holding the key material—firewalled by relays is common practice as a first line of defense against intrusion attacks.[6]

Communication between relays is modeled by functionality F_{net} (cf. Figure 1), which captures a simple, Internet-like network. Parties can obtain (unique) IP addresses for their relays. F_{net} guarantees bounded-delay message transmission between any two relays. The attacker sees all messages sent and may send messages on behalf of any relay IP owned by a corrupted party; it is, however, not permitted to interfere with message transmission between honest relays (beyond inducing a bounded delay).

Master index. A *master index* MI is made up of:

- the network directory ND, which consists of tuples (P, IP, v), where P is a party ID, IP is an IP address, and v is a VRF public key (see below);
- the stake distribution SD, which consists of tuples (P, α), where $\alpha \in (0, 1]$ denotes P's stake fraction; and
- a seed value R.

A master index is *valid* if (a) the same parties appear in ND and SD, and each party appears at most once, (b) values IP and v appear only once in ND, and (c) the values α in SD sum up to 1. All master indices appearing in this work are tacitly assumed to be valid.

Observe that the MI format defined above restricts each party to having only one IP address. This choice was made for simplicity; the definition of master indices as well as all protocols and functionalities can easily be adapted to allow multiple IP addresses per party, thereby modeling the fact that in practice parties often have multiple relays.

[5] The environment both acts as distinguisher and controls the attacker/simulator as well as the inputs to the parties.

[6] Note, however, that for the purposes of this work, the attacker corrupts parties, at which point it gains control over (all of) their relays.

Functionality F_{net}

Parameters: δ_{net}: maximum delay.

Variables: The functionality keeps track of the following variables, initialized to the following values: array $\mathsf{IPs}[P] := \emptyset$: set of IPs owned by parties P; array $\mathcal{M}[\text{mid}]$: message records, indexed by message IDs (MIDs) mid.

IP addresses: Upon (getIP) from P: Output (getIP, P) to $\mathcal{S}$ and ask $\mathcal{S}$ for a unique address IP. Add IP to $\mathsf{IPs}[P]$ and output (getIP, IP) to P.

Malicious IPs Registration: Upon (regIP, IP) from $\mathcal{S}$: if IP is unique, add it to $\mathsf{IPs}[\mathcal{S}]$.

Message send: Upon (send, IP, IP', m) from P with $\mathsf{IP} \in \mathsf{IPs}[P]$:

(1) Store $\mathcal{M}[\text{mid}] := (\text{mid}, \mathsf{IP}, \mathsf{IP}', \mathbb{T}, m)$.
(2) Output (send, mid, IP, IP', m) to $\mathcal{S}$.

Malicious message send: Upon (send, IP, IP', m) from $\mathcal{S}$ where $\mathsf{IP} \in \mathsf{IPs}[\mathcal{S}]$ or P with $\mathsf{IP} \in \mathsf{IPs}[P]$ is corrupted: Pick a unique mid and store $\mathcal{M}[\text{mid}] := (\text{mid}, \mathsf{IP}, \mathsf{IP}', \infty, m)$. Output (send, mid) to $\mathcal{S}$.

Fetching: Upon (fetch, IP) from P with $\mathsf{IP} \in \mathsf{IPs}[P]$:

(1) Output (fetch, P, IP) to $\mathcal{S}$ and ask $\mathcal{S}$ for a set M of MIDs.
(2) Let M' be the set of MIDs corresponding to records $(\cdot, \cdot, \mathsf{IP}, t, \dot{})$ with $\mathbb{T} - t \geq \delta_{\text{net}}$.
(3) Let $\tilde{\mathcal{M}} := \mathcal{M}[M \cup M']$ and set $\mathcal{M}[M \cup M'] := \bot$.
(4) Output (fetch, $\tilde{\mathcal{M}}'$) to P, where $\tilde{\mathcal{M}}'$ is the set $\tilde{\mathcal{M}}$ with MID and time stamp removed in each tuple.

Figure 1: Network/Internet functionality F_{net}.

Verifiable random functions. A *verifiable random function (VRF)* is a cryptographic primitive that allows a party P to create key pair consisting of a secret evaluation key and a public verification key such that: (a) with the secret key, P can evaluate the VRF at any input x, obtaining a random-looking output y and a proof π; (b) given the public key of P, anyone is able to verify, using π, that y is indeed the output corresponding to x. Importantly, even a malicious P cannot bias the output of the VRF on any particular input x (for any fixed public key).

The above guarantees are abstracted by an idealized functionality F_{vrf} (cf. Figure 2). The main commands offered by F_{vrf} are (i) (GetKey), answered by (GetKey, v) for an (adversarially chosen) idealized public key v, (ii) (eval, v, x), answered by (eval, y, π), and (iii) (Verify, v, x, y, π), answered by (Verify, ϕ) with $\phi \in \{0, 1\}$ indicating whether π is a valid proof for the input/output pair (x, y) under public key v.

3 BYZANTINE-RESILIENT NETWORKING

3.1 Overview

One of the main contributions of this work is to define and realize a *synchronization functionality* F_{sync} that can be used by participants of proof-of-stake (PoS) consensus layers to globally synchronize their blockchains. A most crucial feature of F_{sync} is that it is realizable *under the same assumptions* as the consensus protocol that builds on top of it.

Functionality F_{vrf}

Variables: The functionality keeps track of the following variables, initialized to the values below:
(1) an array $\mathsf{Keys}[P] := \emptyset$: set of keys owned by P;
(2) an array $T[v, x] := \bot$, where v is a key and x a domain value: pair (y, S), where y is a range value and S a set of proofs π;
(3) a set $E := \emptyset$: contains triples (v, x, y) to keep track of all VRF evaluations.

Keys: Upon ($\textsc{getKey}$) from P: Output ($\textsc{getKey}$, P) to $\mathcal{S}$ and ask $\mathcal{S}$ for a unique key v. Add v to $\mathsf{Keys}[P]$ and output ($\textsc{getKey}$, v) to P.

Register Keys: Upon ($\textsc{regKey}$, v) from $\mathcal{S}$: if v is unique, add it to $\mathsf{Keys}[\mathcal{S}]$.

Evaluation: Upon ($\textsc{Eval}$, v, x) from P with $v \in \mathsf{Keys}[P]$: Output ($\textsc{Eval}$, P, v, x) to $\mathcal{S}$, and upon obtaining ($\textsc{Eval}$, π) from $\mathcal{S}$:
(1) If π is not unique, exit the procedure.
(2) If $T[v, x]$ is undefined, pick y uniformly at random from the range and set $S := \emptyset$; otherwise, let $(y, S) := T[v, x]$.
(3) Set $T[v, x] := (y, S \cup \{\pi\})$ and add (v, x, y) to E.
(4) Output ($\textsc{Eval}$, y, π) to P.

Malicious evaluation: Upon ($\textsc{Eval}$, v, x) from $\mathcal{S}$:
- *Case 1:* There exists an *uncorrupted* P with $v \in \mathsf{Keys}[P]$: If $(y, S) := T[v, x]$ is defined, return ($\textsc{Eval}$, y) to $\mathcal{S}$; otherwise, do nothing.
- *Case 2:* There exists a *corrupted* P with $v \in \mathsf{Keys}[P]$ *or* $v \in \mathsf{Keys}[\mathcal{S}]$: If $T[v, x]$ is not defined, pick y uniformly at random from the range, let $S := \emptyset$ and set $T[v, x] := (y, S)$; otherwise, let $(y, S) := T[v, x]$. Return ($\textsc{Eval}$, y) to $\mathcal{S}$.
- *Else:* Do nothing.

Verification: Upon ($\textsc{Verify}$, v, x, y, π) from any ITI: Send ($\textsc{Verify}$, v, x, y, π) to $\mathcal{S}$, and upon receiving ($\textsc{Verify}$, ϕ) from $\mathcal{S}$:
- If $v \in \mathsf{Keys}[\cdot]$ and $T[v, x] = (y, S)$ is defined:
  (1) If $\pi \in S$, set $f := 1$.
  (2) Else, if $\phi = 1$ and π is unique—i.e., if for all $(v', x') \neq (v, x)$ with $T[v', x'] = (\cdot, S)$, $\pi \notin S$—set $T[v, x'] := (y, S \cup \{\pi\})$ and $f := 1$.
  (3) Else, set $f := 0$.
- Else, set $f := 0$.

Output ($\textsc{Verify}$, f) to P.

Adversarial leakage: Upon ($\textsc{Leak}$) from $\mathcal{S}$: return ($\textsc{Leak}$, E) to $\mathcal{S}$.

Figure 2: VRF functionality F_{vrf}.

In rough terms, the idea underpinning the protocol realizing F_{sync} is as follows: Each consensus participant P samples $\Theta_P \cdot d$ neighbors in a stake-based fashion based on the output of a verifiable random function (VRF); d is a small constant, and Θ_P is a multiplier that depends on the amount of stake of P itself. The use of a VRF guarantees that parties can verify whether they were indeed supposed to be chosen as neighbors by other participants.

Once the neighbors are sampled, chains are synchronized globally by each party engaging in bilateral chain synchronization with all its neighbors. Due to the fact that the resulting communication overlay is essentially a *random graph*, using expander theory, one can show that even with all adversarial nodes removed, there remains a *large connected backbone* with *small diamater.* This enables *timely synchronization (TS)*, which is crucial to the security of many consensus protocols.

Of course, with a constant number of neighbors, it is unavoidable that some parties end up being connected solely to corrupted parties, which means that they are *eclipsed.* Consequently, F_{sync} will have to grant eclipsing power to the adversary and can only guarantee TS to non-eclipsed parties.

Functionality F_{sync} will be realized (cf. Section 3.4) from a functionality $F_{\mathsf{bilateral}}$ (cf. Section 3.2), which models bilateral chain synchronization, as well as from network functionality F_{net} (cf. Section 2) and (standard) functionality F_{vrf} (cf. Section 2).

Functionality $F_{\mathsf{bilateral}}$

Parameters: δ_{init}: initial delay; δ_{sync}: synchronization delay; $\mathsf{prefer}(\cdot, \cdot)$: strict partial order.

Parties and session ID: Involves two parties A and B. Below, whenever $P \in \{A, B\}$, P' refers to the other party. The parties use $\mathsf{sid} = (A, \mathsf{IP}, B, \mathsf{IP}', t, j)$ as session ID (for some t and j).

Variables: The functionality keeps track of the following variables, initialized to the values below for both parties $P \in \{A, B\}$ and all rounds t:
(1) $C[P, t] := \bot$: local chain of P in round t;
(2) $\mathsf{Ptr}[P] := -\infty$: time pointer of P into $C[P', \cdot]$;
(3) $t_{\mathsf{start}}[P]$ (derived from C and Ptr): smallest t such that $C[P, t] \neq \bot$.

Fetch chain: Upon ($\textsc{fetch}$) from P:
(1) Output ($\textsc{fetch}$, P) to $\mathcal{S}$ and ask $\mathcal{S}$ for time pointer $\mathsf{Ptr} < \mathbb{T}$.
(2) If $\mathbb{T} - \max\{t_{\mathsf{start}}[P], t_{\mathsf{start}}[P']\} < \delta_{\mathsf{init}}$, set $d := \infty$; else, set $d := \delta_{\mathsf{sync}}$.
(3) Let $\mathsf{Ptr}[P] := \max\{\mathsf{Ptr}, \mathsf{Ptr}[P], \mathbb{T} - d\}$.
(4) Output ($\textsc{fetch}$, $C[P', \mathsf{Ptr}[P]]$) to P.

Set local chain: Upon ($\textsc{set}$C, C) from P: if $\mathsf{prefer}(C, C[P, \mathbb{T} - 1])$, set $C[P, \mathbb{T}] := C$.

Figure 3: Bilateral chain-sync functionality $F_{\mathsf{bilateral}}$.

3.2 Bilateral Chain Synchronization

Functionality $F_{\mathsf{bilateral}}$ (cf. Figure 3) models bilateral chain synchronization between two parties A and B. In $F_{\mathsf{bilateral}}$, each party has a *local chain*, and the functionality allows them to learn their counterparty's chain as it evolves. A party's local chain evolves by use of the $\textsc{set}$C command, under the restriction that local chains are only replaced by "better" chains as determined by a strict partial order $\mathsf{prefer}(\cdot, \cdot)$, where $\mathsf{prefer}(C, C')$ (also denoted by $C > C'$) if and only if C is strictly preferable to C';[7] prefer is a parameter of the functionality.

[7] Commonly, C is strictly preferable to C' if it is longer.

Functionality F_{sync}

Parameters: Δ_{init}: initial delay; Δ_{sync}: synchronization delay; $\xi_{\mathsf{ecl}} \geq \Delta_{\mathsf{sync}}$: eclipse delay; $\mu \geq \Delta_{\mathsf{sync}}$: non-free lookback parameter; λ: maximum fraction of non-free honest stake; prefer$(\cdot, \cdot)$: strict partial order.

Admin

Variables: The functionality keeps track of the following variables, initialized to the values below for all parties P and rounds t:

(1) $\mathsf{IPs}[P] := \emptyset$: set of IPs owned by P;
(2) $\mathsf{Keys}[P] := \emptyset$: set of keys owned by P;
(3) $C[P, t] := \perp$: local chain of P in round t;
(4) $E[P, t] :=$ false: eclipse status of P in round t;
(5) $\mathsf{MI}[P] := \perp$: master index as seen by P; recall that $\mathsf{MI} = (\mathsf{ND}, \mathsf{SD}, \mathsf{R})$ (cf. Section 2).

Setup: Upon (SETUP, MI, C) from P (in round $-\Delta_{\mathsf{init}}$): set $C[P, < 0] := C$ and $\mathsf{MI}[P] := \mathsf{MI}$.

IP addresses: Upon (GETIP) from P: Output (GETIP, P) to $\mathcal{S}$ and ask $\mathcal{S}$ for a unique address IP. Add IP to $\mathsf{IPs}[P]$ and output (GETIP, IP) to P.

Keys: Upon (GETKEY) from P: Output (GETKEY, P) to $\mathcal{S}$ and ask $\mathcal{S}$ for a unique key v. Add v to $\mathsf{Keys}[P]$ and output (GETKEY, v) to P.

Fetch & Set

Fetch chains: Upon (FETCH) from P:

(1) Output (FETCH, P) to $\mathcal{S}$ and ask $\mathcal{S}$ for a set D of chains.
(2) For any C with $C[P', \mathbb{T} - \Delta_{\mathsf{sync}}] = C$ for some $\mathbb{T}$-core node P': if P is a $\mathbb{T}$-core node and prefer$(C, C[P, \mathbb{T} - 1])$ as well as prefer(C, C') for all chains C' in D, add C to a set D'.
(3) Output (FETCH, $D \cup D'$) to P.

Set local chain: Upon (SETC, C) from P: if prefer$(C, C[P, \mathbb{T} - 1])$, set $C[P, \mathbb{T}] := C$.

Adversarial

Eclipsing: Upon (ECLIPSE, P) from $\mathcal{S}$—after all honest parties have set master index in current round $\mathbb{T}$: If eclipsing P in round $\mathbb{T}$ would result in the fraction of $(\mathbb{T} + \xi_{\mathsf{ecl}} + 1)$-*non-free* stake remaining below λ, set $E[P, \mathbb{T} + \xi_{\mathsf{ecl}}] :=$ true.

Register IPs: Upon (REGIP, IP) from $\mathcal{S}$: if IP is unique, add it to $\mathsf{IPs}[\mathcal{S}]$.

Register keys: Upon (REGKEY, v) from $\mathcal{S}$: if v is unique, add it to $\mathsf{Keys}[\mathcal{S}]$.

Notions

Representation: P is *represented* if $(P, \mathsf{IP}, v) \in \mathsf{ND}$ for $\mathsf{IP} \in \mathsf{IPs}[P]$ and $v \in \mathsf{Keys}[P]$.

Eclipsed: P is *eclipsed* in round t if $E[P, t] =$ true.

Core and free parties: P is a *t-core party* (resp. *t-free party*) if it was *not* eclipsed in rounds $t - \Delta_{\mathsf{sync}}, \ldots, t - 1$ (resp. $t - \mu, \ldots, t - 1$).

Valid setup: F_{sync} has *valid setup* if (a) all honest parties use SETUP with (i) chains originating from the same genesis block and (ii) the same MI, and (b) all honest parties are represented.

If F_{sync}'s setup is not valid, it halts.

Figure 4: Functionality F_{sync} for global chain synchronization among participants of PoS consensus.

Both parties are informed about changes to their counterparty's local chain with a delay of at most δ_{sync}. However, initially, i.e., until both parties have used SETC at least once (thereby indicating that they are ready to start chain synchronization), the delay may be up to δ_{init}; this models the fact that in reality chain synchronization can take significant time between two parties who have just established a connection.

This work leaves the exact mechanism employed to realize $F_{\mathsf{bilateral}}$ open and simply assumes $F_{\mathsf{bilateral}}$ as a hybrid. For an example of a protocol used in practice, see [9] and [8, Section 3.7].

3.3 Synchronization Functionality

Functionality F_{sync}, whose realization is the main focus of this work, allows parties to synchronize their chains with the rest of the participants. It is parametrized by an initial delay Δ_{init}, a synchronization delay Δ_{sync}, an eclipse delay $\xi_{\mathsf{ecl}} \geq \Delta_{\mathsf{sync}}$, a "lookback" parameter $\mu \geq \Delta_{\mathsf{sync}}$, as well as an upper bound λ on the amount of eclipsed honest stake.

Note that parties have to agree on the round number in which they start using F_{sync}. For convenience, in this section, initialization of F_{sync} begins in round $-\Delta_{\mathsf{init}}$, and parties actually start using F_{sync} in round 0.

IP and key management. Since F_{sync} is realized from F_{net} and F_{vrf}, interfaces for IP and key registration are also provided by F_{sync}. The reason these are not abstracted away is that IPs and VRF keys must be known by the (higher-level) consensus protocol (e.g., to generate the genesis block).

Master index and setup. As previously mentioned, in the synchronization protocol parties will sample neighbors based on their stake. The stake distribution, however, is an object that emanates from the consensus layer, which itself relies on the synchronization functionality. This apparent cycle is broken as follows: F_{sync} expects each party to input its view on the *master index* (cf. Section 2) in the beginning via command SETUP, and F_{sync} only provides guarantees if (a) all honest parties (i) agree on the master index and (ii) input chains originating from the same genesis block, and (b) all honest parties are *represented* in the master index $\mathsf{MI} = (\mathsf{ND}, \mathsf{SD}, \mathsf{R})$ they input; a party is represented in MI if one its IP addresses IP and one of its keys v appear in ND, i.e., $(P, \mathsf{IP}, v) \in \mathsf{ND}$.

When all of the above conditions are satisfied, F_{sync} has *valid setup*. If the setup is not valid, F_{sync} shuts down. Note that it is the responsibility of the consensus protocol (i.e., the environment in the diffusion context) to ensure that the setup is valid. This is commonly achieved via the genesis block (which is assumed to be available to all parties) initially and later on based on the epoch-wise consensus properties of the protocol. More details on this relationship are provided in Section 5.

Eclipsing. Due to the fact that in practically feasible and scalable protocols for realizing F_{sync}, each party is connected only to a small subset of all participants, it is unavoidable that the adversary may eclipse certain honest parties by (adaptively) corrupting all of their neighbors. Consequently, F_{sync} offers an ECLIPSE command by which the attacker can exclude any honest party from the TS guarantees (see below).

There are two limitations on the attacker's use of the ECLIPSE command. The first one is that eclipse commands only take effect after a delay of ξ_{ecl}. The second limitation is based on the following notion of *t-free party*:

Definition 3.1. A *t-free party* is an honest party that was not eclipsed during rounds $t-1, t-2, \ldots, t-\mu$.

The eclipse restriction is that at any time t, the fraction of stake corresponding to *t-non-free* honest parties may not exceed parameter λ.

Formulating an eclipse restriction in this way prevents the attacker from eclipsing a completely different subset of parties in each round since a party eclipsed in round t essentially blocks a fraction corresponding to its stake in the attacker's eclipse budget λ for μ slots. Note, however, that it is absolutely possible for the adversary to eclipse *a particular* party for an *unlimited* amount of time. This must be dealt with by the higher-level consensus protocol using F_{sync}.

Main operation and timely synchronization (TS). Similarly to $F_{\mathsf{bilateral}}$, a party uses SETC to switch their local chain to a strictly better one according to a predicate prefer (cf. Section 3.2). Moreover, parties use FETCH in order to receive information about chains of other participants.

The TS guarantee offered by F_{sync} is based on the following notion of *t-core*:

Definition 3.2. A *t-core party* is an honest party that was not eclipsed during rounds $t-1, t-2, \ldots, t-\Delta_{\mathsf{sync}}$.

Thus, the t-core notion is very similar to that of t-free parties, except with less "lookback" (as $\Delta_{\mathsf{sync}} \leq \mu$).[8]

The TS guarantee is now the following: Suppose an honest party P sets its local chain to some chain C at time t. Then, by time $t+\Delta_{\mathsf{sync}}$, provided P is in the $(t+\Delta_{\mathsf{sync}})$-core, the following holds for all $(t+\Delta_{\mathsf{sync}})$-core parties P',

- either P' has already switched their local chain to C', or
- the FETCH command returns C',

where C' is a chain with $\neg\mathsf{prefer}(C, C')$, i.e., incomparable to or better than C.[9]

3.4 Synchronization Protocol

Overview. Protocol π_{sync} (cf. Figure 5), realizing F_{sync} in the hybrid model with F_{net}, F_{vrf}, and $F_{\mathsf{bilateral}}$, follows the stake-weighted random-graph approach outlined in Section 3.1. The protocol uses a VRF to determine the random graph and to ensure that honest parties only accept connections from parties they are neighbors of in the graph. A party runs instances of $F_{\mathsf{bilateral}}$ with each neighbor in order to keep track of their local chain. Whenever a party learns of a *valid* chain C' better than their current local one, the idea is that they switch to C'. However, as elaborated below, validity checks and the decision to switch occur in the environment; therefore, π_{sync} will only realize F_{sync} w.r.t. to a (very reasonably) restricted class of environments.

Admin. Protocol π_{sync} handles requests for IPs and keys by simply forwarding them to F_{net} and F_{vrf}, respectively, and the subsequent replies back to the environment.

Upon receiving $(\text{SETUP}, \mathsf{MI}', C)$ from the environment, π_{sync} stores these values internally for later use.

Overlay. The most crucial part of π_{sync} deals with establishing and updating the random-graph overlay. A party P initially samples Θ_P neighbors for each time stamp $t = -(d-1)r, -(d-2)r, \ldots, -r, 0$ independently and based on their stake, where Θ_P is a multiplier that depends on the stake α_P of P itself; specifically,

$$\Theta_P := \lceil \alpha_P / \alpha_{\mathsf{min}} \rceil ,$$

for some parameter α_{min}.

Connections to neighbors expire after dr slots; thus, in each round that is a multiple of r, Θ_P new neighbors are sampled. Updating the overlay in this fashion helps parties recover from eclipse events in practice; in the context of this work, however, the adaptive attacker considered here may simply corrupt all new neighbors of a party. Hence, with such a powerful adversary, there is no upper bound on the duration of an eclipse for a particular party (short of the adversary exhausting its corruption budget, of course). Furthermore, refreshing neighbors also allows gradual adaptation of the overlay to changing stake distributions.

The actual sampling performed by a party P is described as a procedure $\mathsf{SamCon}(t, j)$ in Figure 5, where t is a time stamp and $j = 1, \ldots, \Theta_P$. The procedure uses a subroutine $\mathsf{pick}(y)$ which, using VRF output y as random coins, chooses a party P' proportionally to its stake in SD. Note that the inputs to the VRF are (apart from P's public key) the random nonce R (part of MI), t, as well as j.

Subsequent to the above sampling, P sends a connection request to P' via F_{net} and immediately begins a corresponding instance of $F_{\mathsf{bilateral}}$. Conversely, P' will start the instance upon receiving the connection request (after performing the obvious checks).

Chain management. Upon receiving (SETC, C) from the environment, if C is better than P's current local chain, P updates the corresponding variable and inputs (SETC, C) to all running $F_{\mathsf{bilateral}}$ instances.

Upon receiving (FETCH) from the environment, P sends (FETCH) to all running $F_{\mathsf{bilateral}}$ instances and collects the answer chains in a set D; all chains in D preferable to P's current local chain are returned to the environment.

Chain validity and switching. With the application of blockchain consensus in mind, observe that in the context of π_{sync} (or, in the ideal world, of F_{sync}) there is no notion of "chain validity." This is a concept from the higher-level consensus layer. Consequently, π_{sync} (F_{sync}) cannot possibly check a chain for its validity and "automatically" switch to the best valid chain. This task therefore falls upon the environment.

At this point it is also important to observe that successful chain "propagation" through the network crucially depends on the environment—in each round—using SETC to make each party

[8] The reason for having two parameters Δ_{sync} and μ here is that F_{sync} is more "useful" the smaller Δ_{sync} is and the larger μ is.

[9] Note that since prefer is a strict partial order, $C' = C$ is also possible.

Protocol π_{sync}

Parameters: d: degree paremeter; r: refresh parameter; α_{min}: minimum core-party stake; prefer$(\cdot,\cdot)$: strict partial order.

Hybrids: F_{net}: network functionality; F_{vrf}: VRF functionality; $F_{\mathsf{bilateral}}$: functionality for bilateral chain synchronization.

Admin

Variables: The protocol is described from the point of view of a party P; it keeps track of the following items, initialized to the values below:

(1) $\mathsf{IPs} := \emptyset$: set of IPs owned by P;
(2) $\mathsf{Keys} := \emptyset$: set of keys owned by P;
(3) $C_{\mathsf{local}} := \bot$: local chain of P;
(4) $\mathsf{MI} := \bot$: master index; recall that $\mathsf{MI} = (\mathsf{ND}, \mathsf{SD}, \mathsf{R})$ (cf. Section 2).
(5) (implicitly) instances of $F_{\mathsf{bilateral}}$ involving P.

Setup: Upon $(\textsc{setup}, \mathsf{MI}', C)$ from P (before round 0): set $C_{\mathsf{local}} := C$ and $\mathsf{MI} := \mathsf{MI}'$. Moreover, let IP and v be P's IP and VRF keys in ND, i.e., $(P, \mathsf{IP}, v) \in \mathsf{ND}$.

IP addresses: Upon $(\textsc{getIP})$ from $\mathcal{Z}$: Output $(\textsc{getIP}, P)$ to F_{net}, receive $(\textsc{getIP}, \mathsf{IP})$ from F_{net}, add IP to IPs, and output $(\textsc{getIP}, \mathsf{IP})$ to $\mathcal{Z}$.

Keys: Upon $(\textsc{getKey})$ from P: Output $(\textsc{getKey}, P)$ to F_{net}, receive $(\textsc{getKey}, v)$ from F_{vrf}, add v to Keys, and output $(\textsc{getKey}, v)$ to $\mathcal{Z}$.

Fetch & Set

Fetch chains: Upon $(\textsc{fetch})$ from $\mathcal{Z}$:

(1) Let $D := \emptyset$. For all instances of $F_{\mathsf{bilateral}}$: output $(\textsc{fetch})$ to $F_{\mathsf{bilateral}}$ and receive $(\textsc{fetch}, D')$ from $F_{\mathsf{bilateral}}$; add D' to D.
(2) Let D_{pref} be all $C \in D$ with prefer(C, C_{local}). Output $(\textsc{fetch}, D_{\mathsf{pref}})$ to $\mathcal{Z}$.

Set local chain: Upon $(\textsc{setC}, C)$ from $\mathcal{Z}$: If prefer(C, C_{local}), set $C_{\mathsf{local}} := C$. For all instances of $F_{\mathsf{bilateral}}$, output $(\textsc{setC}, C)$ to $F_{\mathsf{bilateral}}$.

Overlay Management

Multiplier: The degree multiplier for P is defined as $\Theta_P := \lceil \alpha_P / \alpha_{\mathsf{min}} \rceil$, where α_P is P's stake according to SD.

Initialization: (Run as part of the $\textsc{setup}$ command.) For $t = -(d-1)r, -(d-2)r, \ldots, -r, 0$ and $j = 1, \ldots, \Theta_P$, run $\mathsf{SamCon}(t, j)$.

Refresh Operation: (Run as part of the $\textsc{fetch}$ command.) When $\mathbb{T}$ is a positive multiple of r: Stop $F_{\mathsf{bilateral}}$ instances with time stamps t that have expired (i.e., $\mathbb{T} - t = dr$). For $j \in 1, \ldots, \Theta_P$: run $\mathsf{SamCon}(\mathbb{T}, j)$.

Procedure $\mathsf{SamCon}(t, j)$**:** Proceed as follows:

(1) Output $(\textsc{eval}, v, \mathsf{R}||t||j)$ to F_{vrf} and receive $(\textsc{eval}, y, \pi)$ from F_{vrf}.
(2) Let $(P', \mathsf{IP}') := \mathsf{pick}(y)$ and let IP be the IP with $(P, \mathsf{IP}, \cdot) \in \mathsf{ND}$. Send message $(P, \mathsf{IP}, P', \mathsf{IP}', t, j, y, \pi)$ to IP' using F_{net}. Start $F_{\mathsf{bilateral}}$ instance with $\mathsf{sid} = (P, \mathsf{IP}, P', \mathsf{IP}', t, j)$.

Procedure $\mathsf{pick}(y)$**:** Using y as random coins, pick a party P' proportionally to its stake in SD. Output (P', IP'), where IP' is such that $(P', \mathsf{IP}', \cdot) \in \mathsf{ND}$.

Handling incoming connections: (Run as part of the $\textsc{fetch}$ command.) Proceed as follows:

(1) Let IP be the IP with $(P, \mathsf{IP}, \cdot) \in \mathsf{ND}$.
(2) Fetch messages from F_{net}. For each message $(P', \mathsf{IP}', P, \mathsf{IP}, t, j, y, \pi)$:
  (a) Check that t has not expired (i.e., $\mathbb{T} - t < dr$) and is not from the future or non-positive (i.e., $t \le \mathbb{T} \vee t \le 0$).
  (b) Check $j \in \{1, \ldots, \Theta_{P'}\}$.
  (c) Output $(\textsc{verify}, v', \mathsf{R}||t||j, y, \pi)$ to F_{vrf}, where $(P', \cdot, v') \in \mathsf{ND}$, and check that F_{vrf} answers with $(\textsc{verify}, 1)$. Check that $\mathsf{pick}(y) = (P, \mathsf{IP})$.

  If all checks pass, run $F_{\mathsf{bilateral}}$ instance with $\mathsf{sid} = (P', \mathsf{IP}', P, \mathsf{IP}, t, j)$.

Figure 5: Protocol π_{sync}, implementing functionality F_{sync}.

switch to the best *valid* chain received so far. In order to circumvent the application-specific nature of the notion of "validity," the following proxy for it is used:

Definition 3.3. In an execution of π_{sync} (or F_{sync}), a chain C is *endorsed* if the environment at some point inputs $(\textsc{setC}, C)$ on behalf of an honest party P.

The notion of endorsed chains allows to define the class of environments $\mathcal{Z}$ w.r.t. which π_{sync} must realize F_{sync}:

Definition 3.4. An environment $\mathcal{Z}$ is *H-improving* if in each round t, every honest party P inputs $(\textsc{setC}, C)$ for a chain C which is maximal w.r.t. prefer among all endorsed chains received by P via $\textsc{fetch}$ in rounds up to t.

Theorem 3.5. *Let*

- $n \in \mathbb{N}$,
- $\alpha > \beta > 0$, $\delta > 0$, c_{min}, *and* c_{max} *be constants satisfying* $0 < c_{\mathsf{min}} < 1 < c_{\mathsf{max}}$ *and* $(\alpha - \beta)/2 \ge \beta + c_{\mathsf{min}}/n$, *and*
- $r \in \mathbb{N}$.

Then, for all sufficiently large d, there exists a constant $\gamma > 0$ such that protocol $\pi_{\mathsf{sync}}[d+1, r, c_{\mathsf{min}}/n]$ ε-securely realizes $F_{\mathsf{sync}}[\Delta_{\mathsf{init}}, \Delta_{\mathsf{sync}}, \mu, \xi_{\mathsf{ecl}}, \lambda]$ in the $\{F_{\mathsf{net}}[\delta_{\mathsf{net}}], F_{\mathsf{vrf}}, F_{\mathsf{bilateral}}[\delta_{\mathsf{init}}, \delta_{\mathsf{sync}}]\}$-hybrid model w.r.t. H-improving environments that (1) input an MI *with n parties in which no party owns more than a c_{max}/n fraction of stake, (2) corrupt at most an α-fraction of stake, (3) run for at most L rounds, where*

- $\Delta_{\mathsf{init}} = \delta_{\mathsf{net}} + \delta_{\mathsf{init}}$,
- $\Delta_{\mathsf{sync}} = 8\ell$ *for* $\ell := \lceil -\log_{1+\gamma}(2c_{\mathsf{min}}/n) + 1 \rceil$,
- $\Delta_{\mathsf{sync}} \le \xi_{\mathsf{ecl}} \le \min(\xi_{\mathsf{corr}}, r)$,
- $\mu = r$,
- $\delta_{\mathsf{init}} \le r$,
- $\lambda = 2(\beta + c_{\mathsf{min}})$, *and*
- $\varepsilon = L/r \cdot e^{-\delta n}$.

Remarks. A simple Chernoff bound can be used to show that if there are no small-stake nodes (with less than c_{min}/n stake), the degrees of all parties are constant. In case there are many small-stake nodes, the degrees of large nodes are beyond constant. However, one can show that by having small-stake node pick their peers uniformly (instead of based on stake), all node degrees go back to constant again.

The security error ε is exponentially small in the number of parties n. It is, of course, important in practice to ensure by suitable means that there are sufficiently many parties participating. One possible way to achieve this by having so-called stake-pool operators (SPOs), to which parties can delegate stake, run the blockchain and use incentive mechanisms to control the total number of SPOs.

Furthermore, there are also several ways of enforcing the maximum-stake restriction in Theorem 3.5. In a system with SPOs, one may restrict the maximum delegated stake on the consensus level. Alternatively, one can again set incentives in such a way that SPOs do not attract more than a certain amount of stake. For more information, see [4].

4 SECURITY PROOF

This section presents the security proof of the synchronization protocol π_{sync} (cf. Section 3.4).

The most crucial part of the security argument is a new result on stake-based expander graphs. Specifically, given a graph G wherein each vertex v has assigned to it some stake $\alpha_v \in (0,1]$, where $\sum_v \alpha_v = 1$, and each vertex chooses its neighbors in the stake-based fashion adopted by protocol π_{sync}, the results in Section 4.1 show that even after removing all adversarial nodes from G, leaving behind at least some α-fraction of honest stake, there exists an honest "backbone" holding at least $\alpha - \beta$ stake, for some β, such that the backbone is an expander graph.

As shown in Section 4.2, the above translates to π_{sync} realizing F_{sync} with roughly a β-fraction of honest stake being eclipsed,[10] while due to the expander property of the backbone, the remaining "core" of honest parties are at most $O(\log n)$ hops apart from each other, where n is the total number of parties. It should be noted that the this core only approximately corresponds to the backbone above, the reason for this being that (a) the backbone cannot be efficiently computed (and thus, an approximation has to be used), and (b) backbone nodes with less than c_{min}/n stake need to be excluded from the core (because the expander property cannot be used to bound their distance from other nodes in the backbone).

4.1 Expanders Resisting Vertex Deletion

It is a well-known fact that random graphs—with a wide variety of edge distributions—form expanders with high probability and hence have small diameter and other desirable properties. This section shows that the random graphs produced by the protocol possess such strong properties *even if an adversary with full knowledge of the graph is permitted to remove a constant fraction of the nodes.*

As outlined above, we show that for any two constants $\alpha > \beta > 0$, there is a degree parameter d for which the following holds: so long as an α fraction of nodes remain after adversarial deletion, there is a subset (the "backbone") consisting of an $\alpha - \beta$ fraction of the nodes that is a strong expander. Typically, one would choose $\beta \ll \alpha$ so that the backbone consists of almost all of the remaining vertices.

These results are motivated by a classical theorem of Upfal [28], which uses different methods to establish a similar property of Ramanujan graphs (which achieved fixed constants α and β). Thus, our results expand on this theory by (i) handling random graphs, and (ii) establishing that *any* constants can be achieved by appropriate choice of d. A final remark before transitioning to the technical survey: we also work directly with weighted graphs (with a weighting corresponding to stake) so that we can define and treat a natural notion of "stake-weighted" expansion.

Our protocol directly motivates the following family of distributions of random directed graphs.

Definition 4.1. Let $\overline{d}$ and n be positive integers and let $\mathcal{D}$ be a probability distribution on $[n]$ with the property that $d_v = n\overline{d}\mathcal{D}(v)$ is an integer for every vertex v. We let $\mathcal{G}_{n,\overline{d};\mathcal{D}}$ denote the probability law on directed multigraphs with $V = [n]$ obtained by selecting, for each v, d_v outgoing neighbors $w_1, \dots, w_{d_v}$ independently according to $\mathcal{D}$ and defining the multiset of directed edges to be $E = \bigcup_v \bigcup_{i=1}^{d_v} (v, w_i)$.[11]

We let $\mathcal{G}_{n,d}$ denote the special case when $\mathcal{D}$ is the uniform distribution, in which case $d_v = d$ for all v.

We wish to show that $\mathcal{G}_{n,\overline{d};\mathcal{D}}$ is typically a "stake expander": that is, that sets $S \subset V$ have a number of neighbors outside them, or edges leaving them, proportional to their total stake.

Definition 4.2. Let $G = (V, E)$ be a directed graph. For a subset $S \subset V$, we define the (outer) boundary of S as

$$\partial(S) = \{w \notin S \mid \exists s \in S : (s, w) \in E \text{ or } (w, s) \in E\}\,.$$

Let $\mathcal{D}$ be a distribution on the set V and $\gamma > 0$. We say that G is a $(\mathcal{D}, \gamma)$-expander if for every subset of vertices $S \subset V$ for which $\mathcal{D}(S) \leq 1/2$,

$$\mathcal{D}(\partial(S)) \geq \gamma \mathcal{D}(S)\,,$$

where we use the notation $\mathcal{D}(S)$ to denote $\sum_{s \in S} \mathcal{D}(s)$.

Theorem 4.3. *Let $\alpha > \beta > 0$ and $\delta > 0$ be positive constants, and let c_{min} and c_{max} be positive constants satisfying $c_{\mathsf{min}} < 1 < c_{\mathsf{max}}$. For sufficiently large d there is a constant $\gamma > 0$ for which the following holds: Let $\mathcal{D}$ be a distribution on $V = [n]$ for which $c_{\mathsf{min}}/n \leq \mathcal{D}(v) \leq c_{\mathsf{max}}/n$ and $d_v \triangleq n d \mathcal{D}(v)$ is an integer for each $v \in V$. Consider $G = (V, E)$ drawn according to $\mathcal{G}_{n,d;\mathcal{D}}$. Then, except with probability $p_{\mathsf{fail}} \leq \mathrm{e}^{-\delta n}$, for every subset $H \subseteq V$ for which $\mathcal{D}(H) \geq \alpha$, there is a subset $H' \subset H$ for which $\mathcal{D}(H') \geq \alpha - \beta$ and the subgraph induced by H' is a $(\mathcal{D}', \gamma)$-expander, where $\mathcal{D}'$ is the distribution $\mathcal{D}$ scaled by $1/\mathcal{D}(H')$.*

See Appendix A for the proof.

Remark. The condition in Definition 4.1 and Theorem 4.3 that $n d \mathcal{D}(v)$ is an integer for all v is a mere convienience so that the out-degree of each vertex will be a fixed integer d_v. We can also define $d_v = \lceil n d \mathcal{D}(v) \rceil$. This changes the effective stake of each player by a factor of $1 \pm O(1/d)$. In fact we only need an upper bound on the stake each vertex has, as we can hand over all the vertices with very low stake to the adversary (cf. Lemma A.5).

4.2 Security of the Synchronization Protocol

This section uses the results from Section 4.1 to finally prove the security of protocol π_{sync}.

[10] More precisely, the non-free parameter λ is roughly β (cf. Section 3.3).

[11] In our main application, we are actually interested in the properties of the *undirected* version of such graphs; however, for analytic purposes it is convenient to distinguish the source and sinks for each edge.

Simulation basics. Simulator $\mathcal{S}$ internally simulates instances of the protocol and of hybrids F_{net}, F_{vrf}, and $F_{\mathsf{bilateral}}$ as well as the adversary $\mathcal{A}$ (which acts as the interface to the environment); in the following, this ensemble is referred to as the *simulated real world (SRW)*. Specifically, $\mathcal{S}$ reacts as follows when receiving messages from F_{sync}:

- Upon ($\textsc{setup}, P, \mathsf{MI}, C$): in the SRW, input ($\textsc{setup}, \mathsf{MI}, C$) on behalf of P.
- Upon ($\textsc{getIP}, P$): in the SRW, input ($\textsc{getIP}$) on behalf of P; wait to receive ($\textsc{getIP}, \mathsf{IP}$) on behalf of P; return IP to F_{sync}.
- Upon ($\textsc{getKey}, P$): in the SRW, input ($\textsc{getKey}$) on behalf of P; wait to receive ($\textsc{getKey}, v$) on behalf of P; return v to F_{sync}.
- Upon ($\textsc{fetch}, P$): in the SRW, input ($\textsc{fetch}$) on behalf of P; wait to receive ($\textsc{fetch}, \tilde{D}$); return $\tilde{D}$ to F_{sync}.
- Upon ($\textsc{setC}, P, C$): in the SRW, input ($\textsc{setC}, C$) on behalf of P.

Messages from $\mathcal{A}$ are handled as follows:

- Upon ($\textsc{regIP}, \mathsf{IP}$): input ($\textsc{regIP}, \mathsf{IP}$) to F_{sync}.
- Upon ($\textsc{regKey}, v$): input ($\textsc{regKey}, v$) to F_{sync}.

Notation. Fix the master index $\mathsf{MI} = (\mathsf{ND}, \mathsf{SD}, \mathsf{R})$ input by the honest parties at the beginning; recall that n denotes the number of parties in MI. In the following, for a subset S of the parties in MI, denote by $\alpha_S := \mathsf{SD}(S)$ the amount of stake held by parties in S.

Determining whom to eclipse. A crucial part of $\mathcal{S}$ is to ensure that when handling $\textsc{fetch}$ commands, F_{sync} does not enforce delivery guarantees that contradict the SRW. To that end, $\mathcal{S}$ determines which honest parties are "eclipsed" and relays this information to F_{sync}. Recall that F_{sync} will not offer any guarantees to non-core parties, where a party is in the core in round t if and only if it has not been eclipsed in rounds $t-1, t-2, \ldots, t-\Delta_{\mathsf{sync}}$.

Towards understanding which parties to eclipse at a particular time t, consider the graph G formed by all parties in MI and the connections implied by the use of the VRF at time t, but ignore the edges added to it during the latest refresh (i.e., in the largest round $t' \le t$ that is a multiple of r). This graph follows the distribution $\mathcal{G}_{n,d;\mathsf{SD}}$.

Let H denote the set of honest parties in at time t. By Theorem 4.3, there exists a set $H' \subseteq H$ with $\alpha_{H'} \ge \alpha_H - \beta$, such that the subgraph of G induced by H' is an (SD', γ)-expander, where SD' is SD scaled by $1/\alpha_{H'}$. This expander property allows to bound the diameter between parties in H' with a certain minimum amount of stake, a fact used to show:

Claim 4.4. *From each party in H' with stake at least c_{min}/n in* SD*, one can reach more than $(\alpha_H - \beta)/2$ of honest stake in at most ℓ steps.*

Proof. Consider a party $P \in H'$ with stake $\alpha_P \ge c_{\mathsf{min}}/n$. Its stake according to SD' is equal to

$$\alpha_P/\alpha_{H'} \;\ge\; \frac{c_{\mathsf{min}}}{\alpha_{H'} n}\,.$$

By the expander property on H', one can reach more than $1/2$ of the stake in H' from P (according to SD') in ℓ' steps if

$$\frac{c_{\mathsf{min}}}{\alpha_{H'} n}(1+\gamma)^{\ell'} \;>\; \frac{1}{2}\,.$$

This is satisfied for

$$\ell' \;:=\; \left\lceil \log_{1+\gamma}\left(\frac{\alpha_{H'} n}{2c_{\mathsf{min}}}\right) + 1 \right\rceil \;\le\; \left\lceil \log_{1+\gamma}\left(\frac{n}{2c_{\mathsf{min}}}\right) + 1 \right\rceil \;=\; \ell\,.$$

Reaching half of the stake in H' according to SD' translates to reaching

$$\frac{\alpha_{H'}}{2} \;\ge\; \frac{\alpha_H - \beta}{2}$$

of stake according to SD. ◻

Since the security argument in Theorem 4.3 is non-constructive, it is unclear whether set H' can be efficiently determined. Instead, consider the set

$$I_t \;:=\; \{h \in H \mid \text{can reach } \beta + c_{\mathsf{min}}/n \text{ HS in } \ell \text{ steps from } h\}\,,$$

where HS stands for "honest stake." The subscript is dropped from I whenever clear from the context.

Claim 4.5. $\alpha_I \ge \alpha_H - \beta - c_{\mathsf{min}}$.

Proof. First, observe that $(\alpha_H - \beta)/2 \ge (\alpha - \beta)/2 \ge \beta + c_{\mathsf{min}}/n$, where the last inequality is by assumption. By Claim 4.4, all parties in H' with stake at least c_{min}/n are therefore in I. The claim follows by observing that at most c_{min} of the total stake is held by parties with more than c_{min}/n stake. ◻

As the subgraph induced by H', that induced by I also has bounded diameter:

Claim 4.6. *There is a path of length at most 4ℓ between any two parties in I.*

Proof. By definition of I, from any node in I, one can reach at least $\beta + c_{\mathsf{min}}/n$ stake in H in ℓ steps, and, therefore, c_{min}/n stake in H'. The argument in the proof of Claim 4.4 suggests there is a path of length at most 2ℓ between any two nodes with stake at least c_{min}/n in H'. One concludes that there must be a path of length at most 4ℓ between any two nodes in I. ◻

Given the above, the eclipse strategy of $\mathcal{S}$ is to, in each round t, use ($\textsc{eclipse}, P$) to eclipse all parties $P \notin I_t$. However, due to the eclipse-delay property of F_{sync}, $\mathcal{S}$ is forced to set eclipsed status based on I_t at or before time $t - \xi_{\mathsf{ecl}}$. This is, however, possible:

- Since $\xi_{\mathsf{ecl}} \le r$ and the edges added during the latest round $t' \le t$ that was a multiple of r are ignored, the topology of G in round t is known by round $t - \xi_{\mathsf{ecl}}$.
- Since $\xi_{\mathsf{corr}} \ge \xi_{\mathsf{ecl}}$, which parties are honest/corrupted is also known by round $t - \xi_{\mathsf{ecl}}$.

Bounding synchronization time. The next step in the proof consists of arguing that F_{sync} never has to add any chain C to set D' during a $\textsc{fetch}$ call by party P in some round t.

To that end, assume first that there is no multiple of r in $\{t - 4\ell, \ldots, t-1\}$, i.e., the underlying graph G does not change in this period. Consequently, $I_{t-4\ell} \supseteq I_{t-(4\ell-1)} \supseteq \ldots \supseteq I_{t-1}$, where the only reason elements are dropped from I over time is corruption.

Consider now the $\textsc{fetch}$ command by P in round t. Recall that a chain C is added to D' only if

(i) C was held by some honest party P' in round $t - \Delta_{\mathsf{sync}}$,
(ii) neither P nor P' were eclipsed in rounds $t-1, \ldots, t-\Delta_{\mathsf{sync}}$,

(iii) C is preferable to the chain held by P' in round $t-1$ and to all chains in set D (during the FETCH call).

Observe that $\Delta_{\mathsf{sync}} \geq 4\ell$; thus, that the fact that (ii) holds (i.e., $\mathcal{S}$ did not issue, in advance, eclipse commands for P or P' for rounds $t-1,\ldots,t-\Delta_{\mathsf{sync}}$) means that both parties P and P' were in sets I during rounds $t-1,\ldots,t-4\ell$.

Therefore, by the monotonicity of the sets I, there has been a path of length at most 4ℓ between P' and P, which, combined with the facts that $\mathcal{Z}$ is H-improving and (again) $\Delta_{\mathsf{sync}} \geq 4\ell\delta_{\mathsf{sync}}$, means that by time t, either P's local chain will already be set to a chain $C' \nleq C$ or a chain $C' \nleq C$ is in set D at time t. In either case, C is not added to D'.

Finally, for intervals including changes to G, observe that since $\Delta_{\mathsf{sync}} = 8\ell$, there are at least 4ℓ rounds either before or after the change to G.

Amount of non-free stake. The fact that the amount of non-free stake remains below $\lambda = 2(\beta + c_{\mathsf{min}})$ follows from Claim 4.5, noting that the extra factor of 2 is required for Δ_{sync}-sized intervals containing a multiple of r.

PROOF (OF THEOREM 3.5). In order to complete the proof based on the above, note that it remains merely to apply a union bound over all refresh periods (which are of length r). □

5 FULLY SECURE POS CONSENSUS

This section discusses the application of running a PoS consensus protocol on top of the chain synchronization functionality F_{sync}. For concreteness, the protocol considered here is Ouroboros Praos [11] which is a longest chain Nakamoto-style blockchain PoS protocol and is described first. This is followed by a comparison of the "diffusion" functionality F_{diff} used in [11] and F_{sync}. Based on this comparison, the changes to Ouroboros Praos' security proof are presented and discussed; in particular, as a contribution of possibly independent interest, a generalized version of the so-called *forkable-string analysis* is provided.

5.1 Ouroboros Praos

Ouroboros Praos (or, simply, *Praos*) proceeds in so-called slots. In each slot, each party P first checks whether it has "received" valid chains that are *longer* than its current local chain; if so, it switches to a longest one among those. Subsequently, based on the stake distribution used in the current epoch, P checks whether it is the slot leader. This determination is made based on the output y of a VRF evaluated on (in addition to P's key) the slot number and the so-called epoch randomness; if y ends up below a certain threshold, which is a monotonically increasing function of P's stake, P is a slot leader. As such, it creates a new block extending its current local chain C; the block consists of the hash of the last block of C, the slot number, P's identity, the VRF output and VRF proof, a data payload, as well as a signature. The resulting chain C' is then "sent to everyone."

5.2 Diffusion vs. Synchronization

In [11], parties share a diffusion functionality F_{diff}, parametrized by a value Δ, with the simple intuitive property that any chain an honest party diffuses via F_{diff} will "arrive" at all other honest parties with a delay of no more than Δ slots.

Consider now using F_{sync}, in order to "synchronize" rather than "diffuse" chains. Figure 6 contains a description of the (static-stake) Praos protocol using F_{sync}. The protocol additionally uses hybrids F_{vrf} for slot leadership as well as the following two hybrids, which are only described on a high-level sufficient for the context of this section:

- F_{init} is used for master-index and genesis-block generation; more precisely, parties initially send information such as keys, IPs, etc. to F_{init}, which then generates the initial master index and the genesis block.
- F_{kes} is a functionality idealizing key-evolving signatures and, in particular, allows parties to create keys as well as to issue and verify signatures.

Due to the fact that it is implemented by a protocol—rather than merely assumed—the guarantees offered by F_{sync} are weaker in several ways:

- A λ-fraction of the total stake may be eclipsed, and the corresponding parties are excluded from the timely synchronization (TS) guarantees.
- The TS properties of F_{sync} are weaker compared to those offered by F_{diff}: instead of a particular chain C "propagating" through the network within Δ slots, F_{sync} may deliver different chains C' that are not worse (i.e., equal length or longer).
- In order to replace F_{sync} by its implementation, the environment must be H-improving (cf. Section 3.4).

The next section details how to update the security proofs of Praos in order to deal with these weaker guarantees of F_{sync}.

5.3 Updated Proofs for Praos

5.3.1 Characteristic Strings and Forks. At the heart of the Praos security proof (with functionality F_{diff}) [11] are the notions of *characteristic strings*, *forks*, and *margin*, which capture the security-relevant information about events in an execution of Praos. Characteristic strings indicate the relevant information about the sequence of elected leaders in an execution of the protocol. The analysis shows that consistency violations can be controlled by (i) distilling the family of possible blocktrees that can arise for a given sequence of leader elections (as determined by a characteristic string) into a single numeric metric of interest called *margin* and (ii) an analysis of the stochastic process that governs generation of characteristic strings and the resulting behavior of margin. The final result is articulated as a large-deviation bound for margin, which establishes consistency with high probability. In this section, we discuss how that analysis can be adapted to our setting with a finer-grained view of networking and block delivery.

Characteristic strings in Praos have the form $w = w_1 w_2 \ldots$ and record information about slot leadership in an execution. Each character in the string is a symbol $w_i \in \{\mathsf{A}, \mathsf{H}, \perp\}$, where

$$w_i = \begin{cases} \mathsf{A} & \text{if slot } i \text{ has an adversarial or multiple leaders,} \\ \mathsf{H} & \text{if slot } i \text{ has a single honest leader,} \\ \perp & \text{if slot } i \text{ has no leader.} \end{cases}$$

Protocol π_{praos}

Parameters: k: common-prefix parameter; prefer$(\cdot,\cdot)$: strict partial order.

Hybrids: F_{init}: generates initial master index and genesis block; F_{vrf}: used to determine slot leadership; F_{kes}: used for key-evolving signatures; F_{sync}: allows to synchronize chains.

Admin & Initialization

Variables: The protocol is described from the point of view of a party P; it keeps track of the following items, initialized to the values below:

(1) $C_{\text{local}} := \bot$: local chain of P;
(2) $\eta := \bot$: VRF salt value.

Initialization: Initially, P proceeds as follows:

(1) Use F_{sync} to obtain an IP address and a network (VRF) key.
(2) Use F_{kes} to obtain a signature public key.
(3) Use F_{vrf} to obtain a VRF key (for slot leadership).
(4) Pass the above values to F_{init} and obtain the initial master index MI, genesis chain C_{genesis}, and epoch randomness η'; set $C_{\text{local}} := C_{\text{genesis}}$ and $\eta := \eta'$.
(5) Input (SETUP, MI, C_{genesis}) to F_{sync}.

Main Operation

In each slot $s \geq 1$, execute the following steps:

Fetch chains: Input (FETCH) to F_{sync} and obtain a set D of chains. Pick a maximal (w.r.t. prefer) chain among the *valid* chains $C' \in D$ with prefer(C', C_{local}) and set $C_{\text{local}} := C'$; if no such chain C' exists, leave C_{local} unchanged. A chain is valid if (1) it has the same genesis block as C_{local}, (2) all hashes are correct, (3) slot numbers of blocks are strictly increasing, (4) all VRF values y and proofs π are valid and y is below the threshold for the issuing party, (5) all signatures are valid.

Chain extension: Send (EVAL, v, $\eta \| s$) to F_{vrf}, where v is P's VRF key (stored in the genesis block), and obtain a VRF value y as well as a proof π. If y is below P's leadership threshold, create a block $B = (h, s, P, y, \pi, d, \sigma)$, where h is a hash of the last block in C_{local}, d is a data payload, and σ is a signature, obtained via F_{kes}, on (h, s, P, y, π, d). Set $C_{\text{local}} := C_{\text{local}} \| B$.

Chain synchronization: Input (SETC, C_{local}) to F_{sync}.

Ledger: Output as the ledger the concatenation of all data d contained in blocks of depth at least k in C_{local}.

Figure 6: Protocol π_{praos}.

A Δ-fork F for a characteristic string w is a directed, rooted tree, intended to represent the topology of all chains observed during an execution of Praos: Each vertex v of F corresponds to a block in a particular chain and has a label $\ell(v) \in \mathbb{N}$, which records the block's slot number. The genesis block is represented by the root of the tree. The edges of a fork are directed "away from" the root so that there is a unique directed path from the root to any vertex.

Based on the description of Praos, it is easy to see that a fork satisfies the following properties:

(i) The root $r \in V$ has label $\ell(r) = 0$ and is considered honest by fiat.
(ii) The sequence of labels $\ell(\cdot)$ along any directed path is strictly increasing. Reason: this is enforced by the protocol.
(iii) If $w_i = \mathsf{H}$, there is a unique vertex v for which $\ell(v) = i$. Reason: honest parties do not create multiple blocks.
(iv) For any pair of honest vertices v, v' (i.e., $w_{\ell(v)} = w_{\ell(v')} = \mathsf{H}$) with $\ell(v) + \Delta \leq \ell(v')$, their lengths (i.e., distance from root) len(v) and len(v') satisfy len$(v) <$ len(v'). Reason: As per the guarantees of F_{diff}, during the creation of the block corresponding to vertex v' in slot $\ell(v')$, the block corresponding to v was available to build on since it was created in slot $\ell(v)$, which was at least Δ slots before v'; hence, len(v') must be strictly larger than len(v).

The analysis proceeds by defining a notion of *margin* for a fork, which reflects the presence of pairs of paths (chains) in the fork that diverge prior to a particular slot and exceed the length of the deepest honest block. Intuitively, such pairs of paths correspond to a consistency failure and, indeed, the precise definition of margin ensures that the quantity is positive exactly when such a consistency failure exists. This notion is extended to characteristic strings by maximizing over all forks consistent with the string. The analysis then shows that margin satisfies a recurrence relation in terms of the characteristic string (corresponding to an execution of the protocol) and, finally, that the probability of observing a positive margin is small.

5.3.2 *Accounting for Eclipsed Parties.* Clearly, the Δ-fork formalism above does not consider cases where (a) the leader is out of sync and potentially fails to build on some longest chain C whose last block is more than Δ slots old or (b) the leader's block takes longer than Δ to reach the next slot leader; naturally, an eclipse event can cause both (a) and (b).[12] In the following, an honest leader *not* suffering from (a) is called *current*, and an honest leader *not* suffering from (b) is called *relayed*. An honest leader that is both current and relayed is called *synchronized*.

In order to update the forkable-string analysis to account for non-relayed parties, the following changes are introduced to it. First, the characteristic string is now over an alphabet $w_i \in \{\mathsf{A}, \mathsf{C\!\!R}, \mathsf{C}, \mathsf{R}, \bot\}$, where

$$w_i = \begin{cases} \mathsf{A} & \text{if slot } i \text{ has an adversarial or multiple leaders,} \\ \mathsf{C\!\!R} & \text{if slot } i \text{ has a single synchronized leader,} \\ \mathsf{C} & \text{if slot } i \text{ has a single current leader,} \\ \mathsf{R} & \text{if slot } i \text{ has a single relayed leader,} \\ \bot & \text{if slot } i \text{ has no leader.} \end{cases}$$

Second, condition (iv) for forks is amended as follows:

[12] Note that there are implementations of the network layer in which the connections to peers are unidirectional. In such cases, it is possible that due to an eclipse event (a) occurs but (b) does not (or vice versa).

(iv) For any pair v, v' with $w_{\ell(v)} \in \{\mathsf{R}, \mathsf{C_R}\}$ and $w_{\ell(v')} \in \{\mathsf{C}, \mathsf{C_R}\}$ and where $\ell(v) + \Delta \leq \ell(v')$, their lengths (i.e., distance from root) $\mathrm{len}(v)$ and $\mathrm{len}(v')$ satisfy $\mathrm{len}(v) < \mathrm{len}(v')$.

The definition of margin is essentially unchanged with this new definition of fork. However, the recursive behavior depends on the new semantics of these richer characteristic string symbols and exhibits some rather interesting properties: in particular, the effect of the C and R symbols depends on whether the worst-case margin is currently negative or positive. We discuss this in detail in Appendix B.

5.3.3 Alternative chains and H-improving. It is easy to see that (1) condition (iv) of the forks is not violated if instead of a chain C non-worse chains C' are delivered after Δ_{sync} slots and (2) Praos is an H-improving environment for F_{sync}.

5.4 Multi-Epoch Praos

There are two approaches to running Praos with F_{sync} in a multi-epoch setting with an evolving stake distribution: (a) use a different instance of F_{sync} in every epoch or (b) define a version of F_{sync} that allows for changing stake distributions.

The conceptually simpler approach is, of course, (a). The idea is to, Δ_{init} before the end of each epoch, begin a new instance of F_{sync}. At that point, the master index (including the network directory, the stake distribution, and the randomness) to be used in the new epoch is determined.[13] Taking approach (a) would mean that the random overlay is completely re-sampled for each new epoch. In practice, this may constitute an unacceptable overhead. Using approach (b) avoids this issue, but makes the definition, realization, and proofs of F_{sync} somewhat more cumbersome.

Observe that in either case it is important to properly deal with honest parties that have been eclipsed for more than an epoch. The issue with these parties is that once they have been eclipsed long enough for their master index to be out of date, they will face difficulties ever reconnecting to the network since the choice of neighbors via the VRF is based on an up-to-date stake distribution.

While in practice such a scenario is unlikely to occur due to the evolving nature of the random-graph overlay, in theory the adaptive attacker can keep corrupting a party's neighbors until it exhausts its corruption budget, thereby keeping it eclipsed for a long period of time. This issue can be mitigated by assuming a checkpointing functionality (realized, e.g., via some light-client infrastructure) that supplies parties, sufficiently in advance of an epoch's beginning, with the relevant master index.

[13] Of course, parameters such as epoch length, common prefix, etc. need to be suitably chosen for this to be the case.

REFERENCES

[1] Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 317–326. ACM, 2019.

[2] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 375–392. IEEE Computer Society, 2017.

[3] Erica Blum, Aggelos Kiayias, Cristopher Moore, Saad Quader, and Alexander Russell. The combinatorics of the longest-chain rule: Linear consistency for proof-of-stake blockchains. In Shuchi Chawla, editor, *31st SODA*, pages 1135–1154. ACM-SIAM, January 2020.

[4] Lars Brünjes, Aggelos Kiayias, Elias Koutsoupias, and Aikaterini-Panagiota Stouka. Reward sharing schemes for stake pools. In *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020*, pages 256–275. IEEE, 2020.

[5] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[6] Nishanth Chandran, Wutichai Chongchitmate, Juan A. Garay, Shafi Goldwasser, Rafail Ostrovsky, and Vassilis Zikas. The hidden graph model: Communication locality and optimal resiliency with adaptive faults. In Tim Roughgarden, editor, *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 153–162. ACM, 2015.

[7] Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.*, 777:155–183, 2019.

[8] D. Coutts, N. Davies, K. Knutsson, M. Fontaine, A. Santos, M. Szamotulski, and A. Vieth. The shelley networking protocol. https://hydra.iohk.io/build/13272760/download/2/network-spec.pdf, March 2022.

[9] D. Coutts, N. Davies, M. Szamotulski, and P. Thompson. Introduction to the design of the data diffusion and networking for cardano shelley. https://hydra.iohk.io/job/Cardano/ouroboros-network/native.network-docs.x86_64-linux/latest/download/1, August 2020.

[10] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In Ian Goldberg and Tyler Moore, editors, *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, volume 11598 of *Lecture Notes in Computer Science*, pages 23–41. Springer, 2019.

[11] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98. Springer, Heidelberg, April / May 2018.

[12] Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. Epidemic algorithms for replicated database maintenance. In Fred B. Schneider, editor, *6th ACM PODC*, pages 1–12. ACM, August 1987.

[13] Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. Epidemic algorithms for replicated database maintenance. In Fred B. Schneider, editor, *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, Vancouver, British Columbia, Canada, August 10-12, 1987*, pages 1–12. ACM, 1987.

[14] Cynthia Dwork, David Peleg, Nicholas Pippenger, and Eli Upfal. Fault tolerance in networks of bounded degree. *SIAM J. Comput.*, 17(5):975–988, 1988.

[15] Peter Ga?i, Aggelos Kiayias, and Alexander Russell. Tight consistency bounds for bitcoin. Cryptology ePrint Archive, Report 2020/661, 2020. https://eprint.iacr.org/2020/661.

[16] Peter Gazi, Aggelos Kiayias, and Alexander Russell. Tight consistency bounds for bitcoin. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 819–838. ACM Press, November 2020.

[17] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In Jaeyeon Jung and Thorsten Holz, editors, *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, pages 129–144. USENIX Association, 2015.

[18] Richard M. Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vöcking. Randomized rumor spreading. In *41st FOCS*, pages 565–574. IEEE Computer Society Press, November 2000.

[19] Richard M. Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vöcking. Randomized rumor spreading. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 565–574. IEEE Computer Society, 2000.

[20] Anne-Marie Kermarrec and Maarten van Steen. Gossiping in distributed systems. *ACM SIGOPS Oper. Syst. Rev.*, 41(5):2–7, 2007.

[21] Aggelos Kiayias, Saad Quader, and Alexander Russell. Consistency of proof-of-stake blockchains with concurrent honest slot leaders. In *40th IEEE International Conference on Distributed Computing Systems, ICDCS 2020, Singapore, November 29 - December 1, 2020*, pages 776–786. IEEE, 2020.

[22] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 357–388. Springer, Heidelberg, August 2017.

[23] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[24] Christian Matt, Jesper Buus Nielsen, and Søren Eller Thomsen. Formalizing delayed adaptive corruptions and the security of flooding networks. *IACR Cryptol. ePrint Arch.*, page 10, 2022.

[25] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf, 2008.

[26] Joachim Neu, Srivatsan Sridhar, Lei Yang, David Tse, and Mohammad Alizadeh. Securing proof-of-stake nakamoto consensus under bandwidth constraint. *CoRR*, abs/2111.12332, 2021.

[27] Elias Rohrer and Florian Tschorsch. Kadcast: A structured approach to broadcast in blockchain networks. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies, AFT 2019, Zurich, Switzerland, October 21-23, 2019*, pages 199–213. ACM, 2019.

[28] Eli Upfal. Tolerating linear number of faults in networks of bounded degree. In *Proceedings of the Eleventh Annual ACM Symposium on Principles of Distributed Computing*, PODC '92, pages 83–89, New York, NY, USA, 1992. Association for Computing Machinery.

[29] Vassilis Zikas, Sarah Hauser, and Ueli M. Maurer. Realistic failures in secure multi-party computation. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 274–293. Springer, 2009.

A EXPANDERS RESILIENT AGAINST VERTEX DELETION

We consider the following family of distributions of random directed graphs.

Definition A.1 (Restatement of Definition 4.1.). Let $\overline{d}$ and n be positive integers and let $\mathcal{D}$ be a probability distribution on $[n]$ with the property that $d_v \triangleq n\overline{d}\mathcal{D}(v)$ is an integer for every vertex v. We let $\mathcal{G}_{n,\overline{d};\mathcal{D}}$ denote the probability law on directed multigraphs with $V = [n]$ obtained by selecting, for each v, d_v outgoing neighbors $w_1, \ldots, w_{d_v}$ independently according to $\mathcal{D}$ and defining the multiset of directed edges to be $E = \bigcup_v \bigcup_{i=1}^{d_v}(v, w_i)$. In our main application, we are actually interested in the properties of the *undirected* version of such graphs; however, for analytic purposes it is convenient to distinguish the source and sinks for each edge.

In the definition above, we let $\mathcal{G}_{n,\overline{d}}$ denote the special case when $\mathcal{D}$ is the uniform distribution, in which case $d_v = \overline{d}$ for all v.

Notation warning. Note that $\mathcal{G}_{n,\overline{d}}$ does not denote the random $\overline{d}$-regular graph as it often does in the literature. Rather, it is a directed graph where every vertex has out-degree $\overline{d}$ and in-degree distributed as $\mathrm{Bin}(\overline{d}n, 1/n)$.

We wish to show that $\mathcal{G}_{n,\overline{d};\mathcal{D}}$ is typically a "stake expander": that is, that sets $S \subset V$ have a number of neighbors outside them, or edges leaving them, proportional to their total stake.

Definition A.2 (Restatement of Definition 4.2). Let $G = (V, E)$ be a directed graph. For a subset $S \subset V$, we define the (outer) boundary of S as

$$\partial(S) = \{w \notin S \mid \exists s \in S : (s, w) \in E \text{ or } (w, s) \in E\} .$$

Let $\mathcal{D}$ be a distribution on the set V and $\gamma > 0$. We say that G is a $(\mathcal{D}, \gamma)$-expander if for every subset of vertices $S \subset V$ for which $\mathcal{D}(S) \leq 1/2$,

$$\mathcal{D}(\partial(S)) \geq \gamma\mathcal{D}(S) ,$$

where we use the notation $\mathcal{D}(S)$ to denote $\sum_{s\in S}\mathcal{D}(s)$. When $\mathcal{D}$ is the uniform distribution, we say that G is an γ-expander.

Proposition A.3 (Diameter bound for stake expanders.). *Let $G = (V, E)$ be a γ-stake expander with (stake) distribution $\mathcal{D}$. Let $m = \min_{v\in V}\mathcal{D}(v)$ be the minimum stake of any vertex in G. Let $\overline{G}$ be the undirected counterpart of G, obtained by replacing each directed edge with an undirected edge. Writing $m = c/n$, where $n = |V|$, the diameter of $\overline{G}$ is no more than $2\log_{1+\gamma}(2n/c)$.*

Proof. Consider two vertices u and v of G. Note that the total stake reachable from u by paths of length k is at least $(1+\gamma)^k\mathcal{D}(v) \geq (1+\gamma)^k m$. If $k \geq \log_{1+\gamma}(1/(2m))$ the stake of this set exceeds $1/2$. The same can be said of the vertex v. Composing these two paths (and reversing the directions of arrows on the second) yields a path of length $2\log_{1+\gamma}(1/2m)$, as desired. □

Theorem A.4 (Restatement of Theorem 4.3.). *Let $\alpha > \beta > 0$ and $\delta > 0$ be positive constants, and let c and C be positive constants satisfying $c < 1 < C$. For sufficiently large $\overline{d}$ there is a constant $\gamma > 0$ for which the following holds. Let $\mathcal{D}$ be a distribution on $V = [n]$ for which $c/n \leq \mathcal{D}(v) \leq C/n$ and $d_v \triangleq n\overline{d}\mathcal{D}(v)$ is an integer for each*

$v \in V$. Consider $G = (V, E)$ drawn according to $\mathcal{G}_{n,\overline{d};\mathcal{D}}$. Then, except with probability $p_{\text{fail}} \le e^{-\delta n}$, for every subset $H \subseteq V$ for which $\mathcal{D}(H) \ge \alpha$, there is a subset $H' \subset H$ for which $\mathcal{D}(H') \ge \alpha - \beta$ and the subgraph induced by H' is a $(\mathcal{D}', \gamma)$-expander, where $\mathcal{D}'$ is the distribution $\mathcal{D}$ scaled by $1/\mathcal{D}(H')$.

Remark. The condition in Definition 4.1 and Theorem 4.3 that $n\overline{d}\mathcal{D}(v)$ is an integer for all v is a mere convienience so that the out-degree of each vertex will be a fixed integer d_v. We can also define $d_v = \lceil n\overline{d}\mathcal{D}(v)\rceil$. This changes the effective stake of each player by a factor of $1 \pm O(1/\overline{d})$.

In fact we only need an upper bound on the stake each vertex has, if we are willing to hand over all the vertices with very low stake to the adversary:

Lemma A.5. *Let $c < 1 < C$ and let $\mathcal{D}$ be a distribution on $[n]$ such that $\mathcal{D}(v) \le C/n$ for all v. Then the fraction q of vertices v with $c/n \le \mathcal{D}(v) \le C/n$, is bounded below by*

$$q \ge \frac{1-c}{C-c}\,.$$

It follows that the total probability mass associated with vertices v for which $\mathcal{D}(v) \le c/n$ is no more than $c(1-q) \le c(C-1)/(C-c) \le c$.

Proof. Otherwise, the total stake would be at most $qC+(1-q)c < 1$. □

A.1 Mixing and expansion

The Expander Mixing Lemma is a classic result asserting that if $G = (V, E)$ is a d-regular expander and $S, T \subset V$ are sufficiently large disjoint sets, then the number of edges between S and T is close to $(d/|V|)|S| \cdot |T|$ (which can be interpreted as the expected value if the graph had been drawn randomly). This property is useful to us since it is inherited by any induced subgraph of G, e.g., the subgraph H induced by the good players, albeit with modified parameters. As a partial converse, this mixing property implies expansion for sufficiently large sets, since we can take T to be the complement of S.

Here we prove a mixing lemma for graphs in our model. The proof is a simple Chernoff bound with a union bound over all S and T, and is essentially the same as for random d-regular graphs or sparse Erdős-Rényi graphs $G_{n,p=d/n}$. For simplicity, we focus here on the uniform case $\mathcal{G}_{n,\overline{d}}$ and then generalize to the stake-weighted case. We then use it to establish edge expansion and vertex expansion for sufficiently large sets.

Lemma A.6. *Fix disjoint sets $S, T \in V$ of size $|S| = \sigma n$ and $|T| = \tau n$. Let $G \in \mathcal{G}_{n,\overline{d}}$ and let $E(S, T)$ denote the number of edges in either direction between S and T. For any $\kappa \le 1$,*

$$\Pr_G\left[\left|\frac{E(S,T)}{2\overline{d}\sigma\tau n} - 1\right| > \kappa\right] \le \exp\left(-\frac{\kappa^2\overline{d}\sigma\tau n}{3}\right). \tag{1}$$

Proof. Recall that in $\mathcal{G}_{n,\overline{d}}$ each vertex in S chooses $\overline{d}$ outgoing neighbors uniformly and independently from the graph. Since each of these neighbors falls in T with probability $|T|/n$, the number of directed edges from S to T is binomially distributed as $\text{Bin}(\overline{d}|S|, |T|/n)$. Similarly, the number of edges from T to S is distributed as $\text{Bin}(\overline{d}|T|, |S|/n)$. Since $E(S, T)$ is the sum of these two random variables, both of which have expectation $\overline{d}|S||T|/n = \overline{d}\sigma\tau n$, the event $\left|E(S, T) - 2\overline{d}\sigma\tau n\right| > 2t$ implies that at least one of them deviates from its mean by t. A standard two-sided Chernoff bound for binomial variables X with mean μ states that

$$\Pr[|X-\mu| > t] \le e^{-\frac{t^2}{3\mu}} \tag{2}$$

if $t \le \mu$. Setting $t = \kappa\overline{d}\sigma\tau n$ and $\mu = \overline{d}\sigma\tau n$ completes the proof. □

Lemma A.7. *Let $0 < \varepsilon < 1/2$, $\kappa > 0$, and $\delta > 0$. Then for sufficiently large $\overline{d}$, with probability $1 - e^{-\delta n}$ in $G \in \mathcal{G}_{n,\overline{d}}$, for all pairs of disjoint sets $S, T \subset V$, if $|S| = \sigma n$ and $|T| = \tau n$ where $\varepsilon \le \sigma, \tau \le 1 - \varepsilon$, then*

$$1 - \kappa \le \frac{E(S,T)}{2\overline{d}\sigma\tau n} \le 1 + \kappa\,. \tag{3}$$

Proof. We use Lemma A.6 and take the union bound over S and T. The number of pairs of sets of size σn and τn, disjoint or not, is

$$\binom{n}{\sigma n}\binom{n}{\tau n} \le \exp\big((h(\sigma) + h(\tau))n\big)\,,$$

where $h(x) = -x\ln x - (1-x)\ln(1-x)$ is the entropy function. Comparing (1), we need to set $\overline{d}$ such that

$$\frac{\kappa^2\overline{d}\sigma\tau}{3} > \delta + h(\sigma) + h(\tau)$$

for all $\varepsilon < \sigma, \tau < 1 - \varepsilon$. Since $h(x) \le \ln 2$, setting

$$\overline{d} > \frac{3(\delta + 2\ln 2)}{\kappa^2\varepsilon^2}$$

suffices to beat the union bound over all pairs of sets S, T for any fixed σ, τ. Since the failure probability is exponentially small, it also easily covers the union bound easily covers all $O(n^2)$ possible values of σ and τ. □

A.2 Inferring expansion of induced subgraphs

Lemma A.7 implies that, with high probability in $G \in \mathcal{G}_{n,\overline{d}}$, any sufficiently large set is "edge expanding", with a large number of edges connecting it to its complement. The same holds in any induced subgraph of G. Consider the following definition.

Definition A.8. For $\rho > 0$ and a multigraph $G = (V, E)$, we say a subset $S \subset V$ is *ρ-edge-expanding* if $E(S, V \setminus S) \ge \rho|S|$. We say G is an *ρ-edge-expander* if all subsets $S \subset V$ with $|S| \le |V|/2$ are ρ-edge-expanding.

Then Lemma A.7 states that in an induced subgraph of size n', with high probability any set of size greater than εn is ρ-edge-expanding where

$$\rho = \overline{d}(1-\kappa)n'/n \tag{4}$$

where we took $|T| \ge n'/2$ and thus $\tau \ge n'/(2n)$ in (3).

Intuitively, a graph where the only non-expanding subsets are small is close to an expander. Here we show that we can obtain an expander simply by removing the largest non-expanding subset.

LEMMA A.9. *Let $G = (V, E)$ be an undirected multi-graph of maximum degree $\overline{d}$. Let $\rho > 0$, and let $S \subset V$ be a subset of maximum cardinality such that $|S| \leq |V|/2$ and S is non-ρ-edge-expanding. Assume that $|S| \leq \mu|V|$ where $\mu < 1/4$. Then the subgraph G' induced by $V' = V \setminus S$ is an ρ'-edge-expander, where*

$$\rho' = \rho - 4\overline{d}\mu\,.$$

PROOF. To the contrary, assume there is some subset $T \subset V'$ with $|T| \leq |V'|/2$ which is not ρ'-edge-expanding: that is, $E(T, V' \setminus T) < \rho'|T|$.

There are two cases. If $|T| \leq (1/2 - \mu)|V|$ then $|S \cup T| \leq |V|/2$. However, this gives

$$E\big(S\cup T, V\setminus(S\cup T)\big) \leq E(S, V\setminus S)+E(T, V'\setminus T) < \rho|S|+\rho'|T| \leq \rho(|S\cup T|)\,.$$

Thus $|S \cup T|$ is non-ρ-edge-expanding in V, contradicting the maximality of $|S|$.

In the other case, $|T| > (1/2 - \mu)|V| \geq |V|/4$. But this implies

$$\begin{aligned} E(T, V \setminus T) &= E(T, V' \setminus T) + E(S, T) \\ &\leq E(T, V' \setminus T) + \overline{d}|S| \qquad (5) \\ &< \rho'|T| + \overline{d}\mu|V| \leq (\rho' + 4\overline{d}\mu)|T| \leq \rho|T|\,. \end{aligned}$$

Thus T is non-ρ-expanding in V. But since $|S| \leq \mu|V| < |T| \leq |V'|/2 \leq |V|/2$, this again contradicts the maximality of $|S|$. ◻

Remark. We have chosen to write Lemma A.9 as a purely graph-theoretic result with as few assumptions as possible. In our setting, however, we can be considerably less pessimistic. First, we assumed in (5) that $E(S, T)$ could be as large as $\overline{d}|S|$, but the upper bound of Lemma A.7 shows that this is almost certainly not the case. Second, just as Lemma A.7 implies that $|S|/|V|$ is bounded by a small constant μ, the same holds for $|T|/|V'|$. Thus we can in fact exclude the second case where $|S\cup T| > |V|/2$ with high probability.

A.3 From edge expansion to vertex expansion

In order to bound the diameter of the subgraph induced by H', we care more about vertex expansion—as in Definition 4.2—than about edge expansion. Even if a set S is edge-expanding, if many vertices outside S have large in-degree, then many of S's outgoing edges could be incident to just a few vertices.

The next lemma shows that if $\mathcal{D}(v) \leq C/n$ for all v as in the statement of Theorem 4.3, then the set of vertices v with in-degree more than $2C\overline{d}$, and thus total degree more than $d_v + 2C\overline{d} \leq 3C\overline{d}$, is almost always a small fraction of the graph. By condemning these vertices to spend eternity with the adversary, we are left with a subgraph where no vertex has degree greater than $3C\overline{d}$. If S has m edges connecting it to its complement in that subgraph, we have $|\partial(S)| \geq m/(3C\overline{d})$. Thus if that subgraph is an α-edge-expander, it is also a γ-expander where $\gamma = \alpha/(3C\overline{d})$.

LEMMA A.10. *Let $\delta > 0$ and $\varepsilon > 0$ and assume that $\mathcal{D}(v) \leq C/n$ for all v. Then for sufficiently large $\overline{d}$, with probability $1 - e^{-\delta n}$ in $G \in \mathcal{G}_{n,\overline{d};\mathcal{D}}$ at most εn vertices have in-degree greater than $2C\overline{d}$.*

PROOF. Since the outgoing neighbors of each vertex in $\mathcal{G}_{n,\overline{d};\mathcal{D}}$ are chosen independently and the total out-degree is $\overline{d}n$, the in-degree of any given vertex v is distributed as $\mathrm{Bin}(\overline{d}n, \mathcal{D}(v))$, which is stochastically dominated by $\mathrm{Bin}(\overline{d}n, C/n)$. Using the Chernoff bound (2) with $t = \mu \leq C\overline{d}$, the probability this exceeds $2C\overline{d}$ is at most $e^{-C\overline{d}/3}$. Since the total in-degree is fixed, the number of such vertices is stochastically dominated by $\mathrm{Bin}(n, e^{-C\overline{d}/3})$. Using a crude union bound, the probability that there are more than εn such vertices is at most

$$\binom{n}{\varepsilon n} e^{-(C\overline{d}/3)\varepsilon n} \leq 2^n e^{-(C\overline{d}/3)\varepsilon n}\,,$$

which is less than $e^{-\delta n}$ whenever $\overline{d} > (3/(\varepsilon C))(\delta + \ln 2)$. ◻

A.4 Proof of Theorem 4.3

Now we put these pieces together.

PROOF OF THEOREM 4.3. We begin wth the uniform case. Consider $G \in \mathcal{G}_{n,\overline{d}}$ where $c = C = 1$. Fix a subset $H \subseteq V$ such that $|H| \geq \alpha n$. Our task is to find a subset $H' \subseteq H$ of size at least $(\alpha - \beta)n$ whose induced subgraph is a γ-expander for some $\gamma > 0$.

First, we remove from H the vertices whose in-degree exceeds $2\overline{d}$. Using Lemma A.10, with high probability there are at most εn of these, and we set $\overline{d}$ large enough so that $\varepsilon = \beta/2$. Denote the remaining set of vertices H'' and note that $|H''| \geq (\alpha - \beta/2)n$.

We now want to find a subset $H' \subseteq H''$ whose induced subgraph is an edge expander and where $|H'| \geq (\alpha - \beta)n$. Lemma A.9 tells us that we can do that by removing the largest subset S of H'' which is non-ρ-edge-expanding, as long as this set is of size at most μn where $\mu \leq \beta/2$. In fact we set $\mu = \beta/16$.

To bound the size of S we use the fact that H'', like any induced subgraph of G, inherits the mixing property of Lemma A.7. We set $\overline{d}$ large enough so that $\varepsilon = \mu = \beta/16$ and $\kappa = 1/2$ in that lemma. Then with high probability every set of size between εn and $|H''|/2$ is ρ-edge-expanding in H'' where (from (4))

$$\rho = (1 - \kappa)\overline{d}|H''|/n \geq \overline{d}(\alpha - \beta/2)/2\,.$$

Lemma A.9 then tells us that $H' = H'' \setminus S$ is a ρ'-edge-expander where

$$\rho' = \rho - 4\overline{d}\mu \geq \overline{d}(\alpha - \beta/2)/2 - \overline{d}\beta/4 = \overline{d}(\alpha - \beta)/2\,.$$

Since the maximum total degree of any vertex in H'' is at most $3\overline{d}$, H' is a γ-expander where

$$\gamma = \rho'/(3\overline{d}) = (\alpha - \beta)/6\,.$$

Finally, we set δ in both Lemmas A.7 and A.10 to be strictly larger than $\delta + \ln 2$ where δ is the desired parameter for p_{fail} in Theorem 4.3. Then the probability either of these lemmas fails is less than $2^{-n}e^{-\delta n}$. Finally, we take a union bound over all subsets $H \subset V$ with $|H| \geq \alpha n$, of which there are at most 2^n.

The nonuniform case of the theorem follows with minor alterations. Specifically, the mixing lemma follows because the number of independent random variables appearing in the Chernoff bounds there depend only on stake. The subgraph removal lemma translates immediately to a stake-weighted version and the remainder of the argument treats individual vertices. ◻

Remark. Note that the expansion parameter γ stays constant as $\overline{d}$ increases, since the edge expansion parameter α is proportional to $\overline{d}$.

B AN ANALYSIS OF NAKAMOTO CONSENSUS WITH ADVERSARIALLY SUPPRESSED MESSAGES.

Existing work on the analysis of Nakamoto consensus has focused on the setting with Δ-synchrony, where all message broadcasts are delivered with adversarially determined delays of no more than Δ time steps. In this section, we show how to analyze a setting where the adversary may entirely eliminate messages of his choice from the protocol. We remark that this axis of adversarial activity does not have the same hard threshold at 1/2: indeed, there are natural settings where Nakamoto consensus can survive even if a majority of honestly sent messages never arrive.

B.1 The synchronous setting with message suppression.

We adopt the framework of characteristic strings and margin, developed in [22], to study the behavior of the longest chain rule under message suppression. Specifically, the analysis divides time into equal length time slots. While these time slots can be chosen to be small enough so that the possibility of multiple proof-of-work successes in a single slot can be neglected, it is convenient for our analysis to permit multiple proof-of-work discoveries in each time slot. We refer to the set of parties that discover proofs-of-work in a particular time slot as the "leaders" of that slot.

In this setting where we wish to explore the effects of message suppression, we identify two properties of interest: Δ-*current* leaders and Δ-*relayed* leaders. In the context of a particular execution of the blockchain protocol these properties—which are only applied to honest parties—have the following meaning: If a party is Δ-current at a particular time slot t then she holds a chain at least as long as any chain broadcast by a Δ-relayed party at time slot $t - \Delta$ or earlier. If a party is both Δ-current and Δ-relayed, we say that it is Δ-synchronized. When the network delay Δ can be inferred from context, we simply say *current*, *relayed*, and *synchronized*. Intuitively, any block produced by a "relayed" leader is successfully transmitted to the network layer; likewise, any "current" leader is aware of all blocks recently transmitted to the network layer.

With any fixed execution, we may associate a *characteristic string*

$$w_1 w_2 \ldots$$

indicating certain properties of the sequence of parties elected to be slot leaders (and hence block producers). Specifically, each w_i is a symbol $w_i \in \Sigma = \{\mathsf{A}, \mathsf{C}, \mathsf{R}, \mathsf{C}_{\mathsf{R}}, \perp\}$. These symbols have the following interpretation:

- A indicates a slot with an adversarial leader, or with more than one leader;
- C_{R} indicates a slot with a synchronized leader;
- C indicates a slot with a current leader;
- R indicates a slot with a relayed leader;
- $\perp$ indicates a slot with no leader.

When $w_i \in \{\mathsf{C}_{\mathsf{R}}, \mathsf{C}, \mathsf{R}\}$, we say that the index (or, alternatively, "slot") i is *honest*; we further distinguish honest indices according to which of these symbols actually arises: if $w_i \neq \mathsf{C}$, we say that the index is *relayed*; if $w_i \neq \mathsf{R}$, we say that the index is *current*; if an index is both current and relayed, we say that it is *synchronized.* If $w_i = \mathsf{A}$, we say that the index is *adversarial.*

We remark that this framework is "first order" modeling, in the sense that it simply treats any slot with multiple leaders as adversarial. Generalizing the model to handle more sophisticated leadership patterns in a single slot is straightforward, but complicates the case analyses of the proofs below. Furthermore, for typical settings where slots are short with respect to network delays the major improvements in analysis arises from careful treatment of the leadership pattern *across* slots within Δ of each other (rather than careful treatment of the pattern in a single slot).

B.2 Fork notation, closure.

We adopt the modeling infrastructure developed in [11, 22] for reasoning about the consistency and liveness of Nakamoto consensus algorithms. Specifically, the relevant state of a blockchain algorithm is reflected with the following graph-theoretic object, the Δ-*fork*.

Definition B.1 (Δ-fork). Let Δ be a positive integer and $L \in \mathbb{N}$. A Δ-*fork* for the string $w \in \Sigma^L$ is a directed, rooted tree $F = (V, E)$ with a labeling

$$\ell : V \to \mathbb{N}$$

satisfying the axioms below. Edges are directed "away from" the root so that there is a unique directed path from the root to any vertex. The value $\ell(v)$ is referred to as the *label* of v, and takes values in the set $\{i \mid w_i \neq \perp\} \cup \{0\}$. We apply the same terminology to vertices as to the indices that label them; that is, the vertex v is respectively *adversarial*, *honest*, *relayed*, *current*, or *synchronized* if this is true of $\ell(v)$.

(i) the root $r \in V$ has label $\ell(r) = 0$ and is considered honest and relayed by fiat;
(ii) the sequence of labels $\ell()$ along any directed path is strictly increasing;
(iii) if $w_i \in \{\mathsf{C}_{\mathsf{R}}, \mathsf{C}, \mathsf{R}\}$, there is a unique vertex v for which $\ell(v) = i$;
(iv) for any pair of vertices v, w for which v is relayed, w is current, and $\ell(v) + \Delta \leq \ell(w)$, their depths $\text{len}(v)$ and $\text{len}(w)$ satisfy $\text{len}(v) < \text{len}(w)$.

We write $F \vdash_\Delta w$ to indicate that F is a Δ-fork for the string w. When $\Delta = 1$, corresponding to the synchronous case, we may just write $F \vdash w$. If $F' \vdash_\Delta w'$ for a prefix w' of w, we say that F' is a *subfork* of $F \vdash w$, denoted $F' \sqsubset F$, if F contains F' as a consistently-labeled subgraph. We remark that with any prefix w' of w there is a unique maximal subfork $F' \vdash w'$ of a given fork $F \vdash w$ (given by all vertices labeled with the indices of w').

Tines. A path in a fork F originating at the root is called a *tine*. (Note that tines do not necessarily terminate at a leaf.) For a vertex v in F, $F(v)$ denotes the tine in F terminating in v. Given this one-to-one correspondence between vertices and tines of a fork, we routinely overload notation so that it applies to both tines and vertices. For example, we let $\text{len}(T)$ denote the *length* of the tine T, equal to the number of edges on the path; recall that $\text{len}(v)$ also indicates the depth of the vertex v. If we must identify the fork from which v is drawn, we sometimes write $\text{len}_F(v)$. We further overload len() to apply to forks: $\text{len}(F)$ denotes the length of the longest tine in a fork F. A tine is called *honest* if it terminates in an honest vertex; we likewise apply the terms *adversarial*, *current*, *relayed*, and *synchronized*.

For two tines T, T' of a fork F, we write $T \sim_\ell T'$ if the two tines share a vertex with a label greater or equal to ℓ. Intuitively, $T \sim_\ell T'$ guarantees that the respective blockchains agree on the state of the ledger up to time ℓ. Observe that $\sim_\ell$ is a "partial equivalence relation": it is symmetric and transitive, but not necessarily reflexive (for example, $T \nsim_\ell T$ if this tine has no blocks appearing in slots ℓ or larger). Looking ahead, the adversary can only make two parties disagree on the state of the ledger up to time ℓ if she makes them hold two chains corresponding to tines for which $T \nsim_\ell T'$.

Fork trimming; dominance. For a characteristic string $w = w_1 \dots w_n \in \Sigma^n$ and a positive integer k, we let $w_{\lceil k} = w_1 \dots w_{n-k+1}$ denote the string obtained by removing the last $k-1$ symbols. When $|w| < k$, $w_{\lceil k} = \epsilon$. For a fork $F \vdash_\Delta w_1 \dots w_n$ we let $F_{\lceil k} \vdash_\Delta w_{\lceil k}$ denote the fork obtained by retaining only those vertices labeled from the set $\{0\} \cup \{1, \dots, n-k+1\}$. Observe that relayed tines appearing in $F_{\lceil \Delta}$ are those that are necessarily visible to current players at a round just beyond the last one described by the characteristic string. We say that a tine T in F is Δ-*dominant* if $\mathrm{len}(T) \geq \mathrm{len}(\overline{F_{\lceil \Delta}})$ and simply call it *dominant* if Δ is clear from the context.

B.3 Advantage, reach, and margin

We generally focus our analysis on forks that represent the view(s) of honest parties during the protocol. This places special emphasis on two properties defined next.

Definition B.2 (Honest and relayed forks and subforks).

- We say that a fork F if *honest* if every leaf is honest. In general, the *honest subfork* of a fork F is the maximal honest subfork of F.
- We say that a fork F is *relayed* if every leaf is relayed. The *relayed subfork* of a fork F is the maximal relayed subfork of F. We let $\overline{F}$ denote the relayed subfork of the fork F.

For a Δ-fork $F \vdash_\Delta w$, we define the *advantage* of a tine $T \in F$ as

$$\alpha_F(T) = \mathrm{len}(T) - \mathrm{len}(\overline{F_{\lceil \Delta}}) .$$

When F can be inferred from context, we remove the subscripts. Observe that $\alpha_F^\Delta(T) \geq 0$ if and only if T is Δ-dominant in F.

We pause to remark that our treatment differs somewhat from that developed in [22]. They consider a notion of "reserve" for a tine, which intuitively adds to that tine's advantage any yet unused adversarial slots; with this addition, the analysis can focus entirely on *honest subforks* forks (which are called closed forks in [22]). In our current setting this mechanism is less convenient due to the more complicated role played by the various classes of honest parties. We thus opt for a treatment that considers arbitrary forks, rather than honest subforks—this somewhat complicates the analysis, but permits us to work with a simpler notion of advantage.

For $\ell \geq 1$, we define two quantities of interest

$$\rho(F) = \max_T \alpha_F^\Delta(T) , \tag{6}$$

$$\mu_\ell^\Delta(F) = \max_{T \nsim_\ell T^*} \min(\alpha_F^\Delta(T), \alpha_F^\Delta(T^*)) , \tag{7}$$

this maximum extended over all pairs of tines (T, T^*) for which $T \nsim_\ell T^*$. Note that there might exist multiple such pairs in F, but under the condition $\ell \geq 1$ there will always exist at least one such pair, as the trivial tine T_0 containing only the root vertex satisfies $T_0 \nsim_\ell T$ for any T and $\ell \geq 1$; in particular $T_0 \nsim_\ell T_0$. For this reason, we will always consider μ_ℓ^Δ only for $\ell \geq 1$. Intuitively, $\alpha_F^\Delta(T)$ captures the length advantage (or deficit) of the tine T against the longest relayed tine created at least Δ slots before the upcoming slot, and hence now known to all current parties. Consequently, $\mu_\ell^\Delta(F)$ records the maximal advantage of any pair of tines F that agree only prior to ℓ, where "advantage" in this setting is treated as the minimum of the two tines.

The quantity $\rho(F)$ is the *reach* of F; the quantity $\mu_\ell(F)$ is the *margin* of F. We say that a tine T^* is a *witness* for the reach $\rho(F)$ when it achieves the maximum of definition (6); we use the same terminology for margin: a pair of tines T and T' *witness* the margin $\mu_\ell(F)$ if they achieve the maximum of the definition (7).

We remark that one may always chose a pair of tines (T, T^*) that witness $\mu_\ell(F)$ for which one is, additionally, a witness for reach. To see this, consider any witness T^* of $\rho(F)$ and a pair (T, T') that witness $\mu_\ell(F)$; as $T \nsim_\ell T'$, we must have either $T^* \nsim_\ell T$ or $T^* \nsim_\ell T'$ and hence T^* can replace one of these without reducing the margin quantity.

We overload the notation and let

$$\mu_\ell^\Delta(w) = \max_{F \vdash_\Delta w} \mu_\ell^\Delta(F) .$$

We again emphasize the difference between this approach and that of [22], which maximizes only over honest forks.

The crucial property motivating these definitions is that $\mu_\ell^\Delta()$ provides explicit control over consistency failure events. This is reflected in the lemma below, which follows directly from the analogous formulation and proof in [3, 16].

LEMMA B.3. *Consider the sequence of forks $F_1 \vdash w_1, F_2 \vdash w_1 w_2, \dots$ associated with each step of an execution corresponding to a characteristic string $w = w_1 w_2 \dots$. Consider a tine T held by a current honest party in round r, which is hence Δ-dominant in F_r; let ℓ be a round associated with a vertex (block) B that appears on T. If $\mu_\ell(w_1 \dots w_s) < 0$ for every $r \leq s \leq t$ then any Δ-dominant tine T^* of F_t contains the vertex B.*

Thus, in order to rule out consistency violations it suffices to establish that $\mu_\ell^\Delta(w) < 0$ for appropriate ℓ and $w_1 \dots w_{\ell+t}$ (note that this bounds above $\mu_\ell^\Delta(F)$ for any relevant fork). Similar connections were originally established for PoS forks [22] with guaranteed network delivery.

B.4 An exact analysis in the serialized, synchronous setting.

We begin with an analysis of the quantity $\mu_\ell^\Delta(w)$ in the synchronous setting, corresponding to the case when $\Delta = 1$ and block creation is strictly serialized. In this synchronous case we abbreviate $\alpha_F^1()$ by $\alpha_F()$, and note that $\alpha_F(T) = \mathrm{len}(T) - \mathrm{len}(\overline{F})$. We similarly abbreviate μ_ℓ^1 by $\mu_\ell()$. In preparation for the statement of the theorem below, one final definition: we say that a characteristic string is *silent* if it contains no relayed symbols, and *relayed* otherwise.

Theorem B.4. *Fix $\ell \geq 1$. By definition $\rho(\epsilon) = \mu_\ell(\epsilon) = 0$. In general,*

$$
\begin{aligned}
&\rho(w\mathsf{A}) = \rho(w) + 1\,,\\
&\rho(w\mathsf{R}) \begin{cases} = \max(\rho(w) - 1, 0) & \textit{if } w \textit{ is silent,}\\ = \rho(w) & \textit{otherwise,}\end{cases}\\
&\rho(w\mathsf{C}) \leq \rho(w) + 1\,,\\
&\rho(w\mathsf{C}\!\!_{\mathsf{R}}) = \max(0, \rho(w) - 1)\,,
\end{aligned}
\tag{8}
$$

and, turning to margin: $\mu_\ell(w) = \rho(w)$ for $|w| < \ell$, and, for $|w| \geq \ell$,

$$
\begin{aligned}
&\mu_\ell(w\mathsf{A}) = \mu_\ell(w) + 1\,,\\
&\mu_\ell(w\mathsf{R}) \begin{cases} = \mu_\ell(w) + 1 & \textit{if } \mu_\ell(w) < 0,\\ = \mu_\ell(w) & \textit{if } \mu_\ell(w) \geq 0 \textit{ and } w \textit{ is relayed,}\\ = \mu_\ell(w) - 1 & \textit{if } \mu_\ell(w) \geq 0 \textit{ and } w \textit{ is silent,}\end{cases}\\
&\mu_\ell(w\mathsf{C}) \begin{cases} = \mu_\ell(w) & \textit{if } \mu_\ell(w) < 0,\\ \leq \mu_\ell(w) + 1 & \textit{if } \mu_\ell(w) \geq 0,\end{cases}\\
&\mu_\ell(w\mathsf{C}\!\!_{\mathsf{R}}) = \begin{cases} 0, & \textit{if } \rho(w) > 0 \textit{ and } \mu_\ell(w) = 0,\\ \mu_\ell(w) - 1, & \textit{otherwise.}\end{cases}
\end{aligned}
\tag{9}
$$

The proof is presented as a sequence of lemmas, the first establishes the relationship between ρ and μ_ℓ when $|x| < \ell$. The remaining lemmas handle each input symbol separately.

Lemma B.5. *$\rho(\varepsilon) = 0$ and, for any $x \in \Sigma^*$ and $|x| < \ell$, $\mu_\ell(x) = \rho(x)$.*

Proof. The only possible fork for the empty string ε contains a single honest vertex; the only tine in this fork has advantage zero; hence $\rho(\varepsilon) = 0$.

Let $F \vdash x$ be a fork for which $\rho(F) = \rho(x)$; let T^* be a tine witnessing $\rho(F)$. Observe that $T^* \not\sim_\ell T^*$ because $\ell > |x|$. Therefore, $\mu_\ell(F) \geq \rho(x)$ by considering the pair (T^*, T^*) as a witness for margin. As $\mu_\ell(F)$ can clearly not exceed $\rho(x)$, it follows that $\mu_\ell(x) = \rho(x)$. ◻

Lemma B.6 (A). *For any $w \in \Sigma^*$, $\rho(w\mathsf{A}) = \rho(w) + 1$. Additionally, for any $w \in \Sigma^*$, $\mu_\ell(w\mathsf{A}) = \mu_\ell(w\mathsf{A}) + 1$.*

Proof. We first establish the bounds $\rho(w\mathsf{A}) \geq \rho(w) + 1$ and $\mu_\ell(w\mathsf{A}) \geq \mu_\ell(w) + 1$. Consider a fork $F \vdash w$ and a pair of tines, T and T^*, for which $\alpha(T^*) = \rho(F)$, $\alpha(T) = \mu_\ell(F)$, and $T \not\sim_\ell T^*$. We may construct a new fork $\operatorname{ext} F \vdash w\mathsf{A}$ from F by adding two new adversarial vertices, v and v^*, both labeled with the index of the additional A symbol. Each of the two tines T and T^* of F are then extended by adding a directed edge to one of these vertices: T is extended by the vertex v; T^* is extended by v^*. Note that $\overline{F} = \overline{\operatorname{ext} F}$. Then $\alpha_{\operatorname{ext} F}(\operatorname{ext} F(v^*)) = \alpha_F(T^*) + 1$ and $\alpha_{\operatorname{ext} F}(\operatorname{ext} F(v)) = \alpha_F(T) + 1$. Applying this transformation to a fork F for which $\rho(F) = \rho(w)$ shows that $\rho(w\mathsf{A}) \geq \rho(w) + 1$. Similarly, applying this transformation to a fork F for which $\mu_\ell(F) = \mu_\ell(w)$ shows that $\mu_\ell(w\mathsf{A}) \geq \mu_\ell(w) + 1$.

To complete the proof we establish the bounds $\rho(w\mathsf{A}) \leq \rho(w) + 1$ and $\mu_\ell(w) \leq \mu_\ell(w) + 1$. Consider a fork $F \vdash w\mathsf{A}$ and a pair of tines, T and T^*, for which $\alpha(T^*) = \rho(F)$, $\alpha(T) = \mu_\ell(F)$, and $T \not\sim_\ell T^*$. Let $\operatorname{res} F \vdash w$ be the maximal subfork of F for the prefix w and let $\operatorname{res}(T)$ and $\operatorname{res}(T^*)$ the restrictions of T and T^* to $\operatorname{res} F$. Observe that $\overline{\operatorname{res} F} = \overline{F}$. Furthermore, the tines T and T^*, when restricted to $\operatorname{res} F$, lose at most one vertex: thus $\operatorname{len}(\operatorname{res}(T)) \geq \operatorname{len}(T) - 1$ and $\operatorname{len}(\operatorname{res}(T^*)) \geq \operatorname{len}(T^*) - 1$. It follows that $\rho(\operatorname{res} F) \geq \rho(F) - 1$ and $\mu_\ell(\operatorname{res} F) \geq \mu_\ell(F) - 1$. Applying this transformation to a fork F for which $\rho(F) = \rho(w\mathsf{A})$ proves that $\rho(w) \geq \rho(w\mathsf{A}) - 1$; similarly, applying this transformation to a fork F for which $\mu_\ell(F) = \mu_\ell(w\mathsf{A})$ proves that $\mu_\ell(w) \geq \mu_\ell(w\mathsf{A}) - 1$, as desired. ◻

Lemma B.7 (R). *For any $w \in \Sigma^*$,*

$$
\rho(w\mathsf{R}) = \begin{cases} \max(\rho(w) - 1, 0) & \textit{if } w \textit{ is silent,}\\ \rho(w) & \textit{otherwise.}\end{cases}
$$

Additionally, for any $w \in \Sigma^*$ with $|w| \geq \ell$,

$$
\mu_\ell(w\mathsf{R}) = \begin{cases} \mu_\ell(w) + 1\,, & \textit{if } \mu_\ell(w) < 0,\\ \mu_\ell(w)\,, & \textit{if } \mu_\ell(w) \geq 0 \textit{ and } w \textit{ relayed,}\\ \mu_\ell(w) - 1\,, & \textit{if } \mu_\ell(w) \geq 0 \textit{ and } w \textit{ silent.}\end{cases}
$$

Proof. We consider the quantity $\rho()$ first and begin with the setting when w is silent. It is easy to confirm that (for silent w), $\rho(w) = |w|$ while $\rho(w\mathsf{R}) = \max(|w| - 1, 0)$. When w is not silent, we wish to show that $\rho(w) = \rho(w\mathsf{R})$ and we first establish that $\rho(w) \geq \rho(w\mathsf{R})$. If $\rho(w\mathsf{R}) = 0$, this follows from the fact that $\rho()$ is nonnegative. Otherwise, $\rho(w\mathsf{R}) > 0$ and we let $F \vdash w\mathsf{R}$ be a fork for which $\rho(F) = \rho(w\mathsf{R})$, T a tine witnessing $\rho(F)$, and v the vertex in F associated with the last symbol (R). Let $\operatorname{res} F$ denote the fork restricted to the string w. As $\rho(F) > 0$, the relayed vertex v cannot appear on T; thus T appears unchanged in $\operatorname{res} F$, where its advantage can only increase by the removal of the relayed vertex v. It follows that $\rho(w) \geq \rho(w\mathsf{R})$, as desired. Conversely, we establish the inequality $\rho(w\mathsf{R}) \geq \rho(w)$. Let $F \vdash w$ be a fork for which $\rho(F) = \rho(w)$ and let T be a tine witnessing $\rho(w)$. Let $\operatorname{ext} F \vdash w\mathsf{R}$ denote the fork formed from F by adding a single vertex v labeled by the last symbol (R) and attached by an edge to the root. As w is not silent, $\operatorname{len}(\overline{F}) > 0$ and we conclude that $\operatorname{len}(\overline{F}) = \operatorname{len}(\overline{\operatorname{ext} F})$: then $\rho(F) = \rho(\operatorname{ext} F)$ and hence $\rho(w) \geq \rho(w\mathsf{R})$.

Turning now to $\mu_\ell(w)$ (with $|w| > \ell$), we begin with the two bounds

$$
\mu_\ell(w) < 0 \Rightarrow \mu_\ell(w\mathsf{R}) \geq \mu_\ell(w) + 1\,, \tag{10}
$$

$$
\mu_\ell(w) \geq 0 \Rightarrow \mu_\ell(w\mathsf{R}) \geq \begin{cases} \mu_\ell(w) & \text{if } w \text{ is not silent,}\\ \mu_\ell(w) - 1 & \text{if } w \text{ is silent.}\end{cases} \tag{11}
$$

Let $F \vdash w$ be a fork for which $\mu_\ell(w) = \mu_\ell(F)$ and let T^* and T be two tines witnessing $\mu_\ell(w)$; as usual we assume that $\rho(F) = \alpha(T^*) \geq \alpha(T) = \mu_\ell(w)$. If $\mu_\ell(w) < 0$, define $\operatorname{ext} F \vdash w\mathsf{R}$ to be the fork obtained from F by adding a single vertex v to the end of the tine T labeled with the index of the last symbol (R). This is a legal fork and, moreover, as $\alpha_F(T) < 0$ the new relayed vertex appears with depth no more than $\operatorname{len}(\overline{F})$. Thus $\operatorname{len}(\overline{\operatorname{ext} F}) = \operatorname{len}(\overline{F})$ and $\alpha_{\operatorname{ext} F}(\operatorname{ext} F(v)) = \alpha_F(T) + 1$. The two tines T^* and $\operatorname{ext} F(v)$ of $\operatorname{ext} F$ establish that $\mu(\operatorname{ext} F) = \mu_\ell(w) + 1$ and hence (10) as desired. As for the case $\mu_\ell(w) \leq 0$, we again consider a fork $F \vdash w$ and construct a fork $\operatorname{ext} F \vdash w\mathsf{R}$ by addition of a single new vertex v attached directly to the root vertex and labeled by the index of the last symbol (R). Observe that $\operatorname{ext} F$ is a legal fork and that $\operatorname{len}(\overline{\operatorname{ext} F}) = \max(1, \operatorname{len}(\overline{F}))$. Unless w is silent, then, $\operatorname{len}(\overline{\operatorname{ext} F}) = \operatorname{len}(\overline{F})$ and the tines T and T^* in $\operatorname{ext} F$ witness $\mu_\ell(\operatorname{ext} F) \geq \mu_\ell(F)$, which establishes the non-silent case of (11). If w is silent, the same construction establishes $\mu_\ell(w\mathrm{R}) \geq \mu_\ell(\operatorname{ext} F) \geq \mu_\ell(F) - 1$ as $\operatorname{len}(\overline{\operatorname{ext} F}) = 1$ (and $\operatorname{len}(\overline{F})) = 0$), concluding the argument for (11).

To complete the argument, we establish the complementary bounds:

$$\mu_\ell(w\mathrm{R}) > 0 \Rightarrow \mu_\ell(w) \geq \begin{cases} \mu_\ell(w\mathrm{R}) & \text{if } w \text{ is relayed,} \\ \mu_\ell(w\mathrm{R}) + 1 & \text{if } w \text{ is silent,} \end{cases} \tag{12}$$

$$\mu_\ell(w\mathrm{R}) \leq 0 \Rightarrow \mu_\ell(w) \geq \mu_\ell(w\mathrm{R}) - 1\,. \tag{13}$$

Consider a fork $F \vdash w\mathrm{R}$ for which $\mu_\ell(F) = \mu_\ell(w\mathrm{R})$; let T and T^* be a pair of tines for which $\alpha(T^*) = \rho(F)$, and $\alpha(T^*) = \mu_\ell(F) = \mu_\ell(w\mathrm{R})$. Let v be the unique vertex labeled by the last symbol (R) and let $\operatorname{res} F \vdash w$ be the fork obtained by removal of the vertex v. Observe that if $\mu_\ell(w\mathrm{R}) > 0$ the vertex v cannot appear on either of the tines T or T^*, as $\operatorname{len}(\overline{F}) < \operatorname{len}(T) \leq \operatorname{len}(T^*)$ (and v is a leaf in the fork F). Noting that the removal of v can only reduce the length of $\overline{\operatorname{res} F}$, and that the two tines T and T^* appear in $\operatorname{res} F$, we conclude that $\mu_\ell(w) \geq \mu_\ell(\operatorname{res} F) \geq \alpha_{\operatorname{res} F}(T) \geq \alpha_F(T)$. Observe, furthermore, that if w is silent then $\operatorname{len}(\overline{\operatorname{res} F}) = 0$ and $\operatorname{len}(\overline{F}) > 0$; thus $\alpha_{\operatorname{res} F}(T) \geq \alpha_F(T) + 1$. This establishes (12). We proceed to the last case (13), in which we again consider the construction $\operatorname{res} F \vdash w$ above with the same notation. In this case, however, there is no guarantee that the leaf node v (labeled with the index of the last symbol (R)) does not appear on one of the two tines of interest, T and T^*. We let $\operatorname{res} T$ denote the result of removing the vertex v from T, so that $\operatorname{res} T = T$ if v does not appear on T, and $\operatorname{res} T$ is the resulting prefix of T otherwise; we adopt the same notation $\operatorname{res} T^*$ for the tine T^*. Thus, $\operatorname{res} T$ and $\operatorname{res} T^*$ are tines of $\operatorname{res} F$. Clearly, the lengths of T and T^* can differ from their restricted counterparts $\operatorname{res} T$ and $\operatorname{res} T^*$ by no more than one and

$$\min(\operatorname{len}_{\operatorname{res} F}(\operatorname{res} T^*), \operatorname{len}_{\operatorname{res} F}(\operatorname{res} T)) \geq \min(\operatorname{len}_F(T^*), \operatorname{len}_F(T)) - 1\,.$$

Noting, as above, that $\operatorname{len}(\overline{\operatorname{res} F}) \leq \operatorname{len}(\overline{F})$, we conclude that $\mu_\ell(w) \geq \mu_\ell(\operatorname{res} F) \geq \mu_\ell(F) - 1$, as desired. □

Lemma B.8 (C). *For any* $w \in \Sigma^*$, $\rho(w\mathrm{C}) = \rho(w) + 1$. *Additionally, for any* $w \in \Sigma^*$, *with* $\ell \leq |w|$,

$$\begin{aligned} \mu_\ell(w\mathrm{C}) &\leq \mu_\ell(w) + 1\,, && \textit{if } \mu_\ell(w) \geq 0, \\ \mu_\ell(w\mathrm{C}) &= \mu_\ell(w)\,, && \textit{if } \mu_\ell(w) < 0. \end{aligned}$$

Proof. Note that if $F \vdash w\mathrm{C}$ then $F \vdash w\mathrm{A}$ and it follows that $\rho(w\mathrm{C}) \leq \rho(w\mathrm{A}) \leq \rho(w) + 1$ by Lemma B.6. To establish $\rho(w\mathrm{C}) \geq \rho(w) + 1$, consider a fork $F \vdash w$ and a tine T that witnesses $\rho(w)$. Construct the fork $\operatorname{ext} F \vdash w\mathrm{C}$ by adding a new vertex v to the end of the tine T labeled with the additional symbol (C). This is a legal fork for which $\operatorname{len}(\overline{\operatorname{ext} F}) = \operatorname{len}(\overline{F})$ as the new vertex in $\operatorname{ext} F$ is not relayed. As $\operatorname{len}(v) = \operatorname{len}(T) + 1$, we conclude that $\rho(w\mathrm{C}) \geq \rho(\operatorname{ext} F) = \rho(w) + 1$.

Turning to margin, we first observe that the statement of the lemma is a consequence of the following three inequalities, which we treat individually below.

$$\mu_\ell(w\mathrm{C}) \geq \mu_\ell(w)\,, \tag{14}$$

$$\mu_\ell(w\mathrm{C}) \leq \mu_\ell(w) + 1\,, \tag{15}$$

$$\mu_\ell(w\mathrm{C}) \leq \mu_\ell(w) \qquad \text{if } \mu_\ell(w\mathrm{C}) \leq 0. \tag{16}$$

To establish inequality (14) consider a fork $F \vdash w$ for which $\mu_\ell(F) = \mu_\ell(w)$ and define $\operatorname{ext} F \vdash w\mathrm{C}$ to be the fork obtained from F by adding a single vertex v, labeled by the index of the new symbol (C), so that it extends the longest tine of F. As the new vertex is not relayed, $\operatorname{len}(\overline{\operatorname{ext} F}) = \operatorname{len}(\overline{F})$ and it follows that $\mu_\ell(\operatorname{ext} F) \geq \mu_\ell(F)$, considering that any pair of tines in F exist in $\operatorname{ext} F$. Thus $\mu_\ell(w\mathrm{C}) \geq \mu_\ell(w)$, as desired.

The inequality (15) follows from the straightforward fact mentioned above: any fork $F \vdash w\mathrm{C}$ is also a legal fork for $w\mathrm{A}$. In particular, $\mu_\ell(w\mathrm{C}) \leq \mu_\ell(w\mathrm{A}) \leq \mu_\ell(w) + 1$ by Lemma B.6.

It remains to show (16)—the case when $\mu_\ell(w\mathrm{C}) \leq 0$. Let $F \vdash w\mathrm{C}$ be a fork for which $\mu_\ell(F) = \mu_\ell(w\mathrm{C})$ and let T and T^* be two tines in F that witness $\mu_\ell(F)$ so that $\alpha(T^*) = \rho(F)$ and $\alpha(T) = \mu_\ell(F) = \mu_\ell(w\mathrm{C}) \leq 0$. We construct a fork $\operatorname{res} F \vdash w$ by deleting the vertex v associated with the final symbol (C). Considering that v is not relayed $\operatorname{len}(\overline{\operatorname{ext} F}) = \operatorname{len}(\overline{F})$. As in the previous arguments, we let $\operatorname{res} T^*$ denote the result of removing the vertex v from T^*, so that $\operatorname{res} T^* = T$ when v does not appear on T^*. We adopt the same notation for T. As the vertex v is current, $\operatorname{len}(v) > \operatorname{len}(\overline{\operatorname{res} F}) = \operatorname{len}(\overline{F})$ and it follows that v cannot appear on the tine T (as $\operatorname{len}(T) \leq \operatorname{len}(\overline{F})$ by assumption); thus $\operatorname{res} T = T$. If $\operatorname{len}(T^*) = \operatorname{len}(\overline{F})$, the same argument implies: v cannot appear on T^* and hence $T^* = \operatorname{res} T^*$. In this case, the pair of tines $\operatorname{res} T^*$ and $\operatorname{res} T$ witness $\mu_\ell(\operatorname{res} F) \geq \mu_\ell(F)$, as desired. If $\operatorname{len}(T^*) > \operatorname{len}(\overline{F})$, the restriction $\operatorname{res} T^*$ may have length one less than T but in any case $\operatorname{len}_{\operatorname{res} F}(\operatorname{res} T^*) \geq \operatorname{len}_{\operatorname{res} F}(\operatorname{res} T)$. Again, the two tines $\operatorname{res} T$ and $\operatorname{res} T^*$ of $\operatorname{res} F$ witness $\mu_\ell(\operatorname{res} F) \geq \alpha_{\operatorname{res} F}(\operatorname{res} T) \geq \mu_\ell(w)$, as desired. □

B.5 The case when $\Delta > 1$ and the serialization mapping

We observe that the case when $\Delta > 1$ can be handled via a direct reduction to the synchronous case ($\Delta = 1$) using a standard technique developed in [11]. We remark that one could alternatively carry out more precise accounting, such as that developed in [15], at the cost of significantly more complex analysis.

One simple analysis adopts the following transformation. Call a symbol "left isolated" if it is preceded by $\perp^\Delta$; similarly define the notion of "right isolated" if it is followed by $\perp^\Delta$ and "doubly isolated" if it is both left and right isolated. Then consider the transformation T, operating on characteristic strings, so that

$$\perp \longrightarrow \perp$$

$$\mathrm{A} \longrightarrow \mathrm{A}$$

$$\mathrm{C} \longrightarrow \begin{cases} \mathrm{C} & \text{if left isolated,} \\ \mathrm{A} & \text{otherwise,} \end{cases}$$

$$\mathrm{R} \longrightarrow \begin{cases} \mathrm{R} & \text{if right isolated,} \\ \mathrm{A} & \text{otherwise,} \end{cases}$$

$$\mathrm{C\!\!R} \longrightarrow \begin{cases} \mathrm{C\!\!R} & \text{if doubly isolated,} \\ \mathrm{R} & \text{if right isolated,} \\ \mathrm{C} & \text{if left isolated,} \\ \mathrm{A} & \text{if not isolated on either side.} \end{cases}$$

It is straightforward to check that any Δ-fork for a characteristic string w is also a 1-fork for the characteristic string $T(w)$. This reduces the study of Δ-forks to the synchronous case as in [11].

B.6 The asymptotics

Finally, we consider the distribution placed on characteristic strings corresponding to the executions of interest: each symbol is i.i.d. with distribution of the form

$$
\begin{aligned}
p_{\mathsf{R}} &= \Pr[w_i = \mathsf{R}] = p_h^{!} \cdot p_{\mathrm{send}}(1 - p_{\mathrm{receive}}),\\
p_{\mathsf{C}} &= \Pr[w_i = \mathsf{C}] = p_h^{!} \cdot (1 - p_{\mathrm{send}})p_{\mathrm{receive}},\\
p_{\mathsf{CR}} &= \Pr[w_i = \mathsf{CR}] = p_h^{!} \cdot p_{\mathrm{send}}p_{\mathrm{receive}},\\
p_{\perp} &= \Pr[w_i = \perp] = (1 - p_a)(1 - p_h),\\
p_{\mathsf{A}} &= \Pr[w_i = \mathsf{A}] = 1 - p_{\perp} - p_{\mathsf{R}} - p_{\mathsf{C}} - p_{\mathsf{CR}},
\end{aligned}
$$

where p_h is the probability that a slot has an honest slot leader, $p_h^{!}$ is the probability that a slot has a unique honest slot leader, p_a is the probability that a slot has an adversarial slot leader, p_{send} is the probability that a block sent from a stake-weighted random honest party is received by the backbone in Δ time, and p_{receive} is the probability that a block known to a member of the backbone is received by a stake-weighted random honest party in Δ time.

Then the final probabilistic bounds on margin with this distribution follow directly from the analysis of [21], yielding the following consistency result.

LEMMA B.9. *Let $w = w_1, \ldots, w_n$ be chosen according to the distribution above with the added assumption that $p_a < p_{\mathsf{CR}}$. Then for any ℓ and k for which $\ell + k \leq n$,* $\Pr[\mu_\ell(w) \geq 0] = \exp(-\Theta(k))$.