Randomized View Reconciliation in Permissionless Distributed Systems

Ruomu Hou
Department of Computer Science
National University of Singapore
Republic of Singapore
houruomu@u.nus.edu

Irvan Jahja
Department of Computer Science
National University of Singapore
Republic of Singapore
irvan@comp.nus.edu.sg

Loi Luu
Department of Computer Science
National University of Singapore
Republic of Singapore
loiluu@comp.nus.edu.sg

Prateek Saxena
Department of Computer Science
National University of Singapore
Republic of Singapore
prateeks@comp.nus.edu.sg

Haifeng Yu
Department of Computer Science
National University of Singapore
Republic of Singapore
haifeng@comp.nus.edu.sg

***Abstract*—In a *sybil attack*, an adversary creates a large number of fake identities/nodes and have them join the system. Computational puzzles have long been investigated as a possible sybil defense: If a node fails to solve the puzzle in a timely fashion, it will no longer be accepted by other nodes. However, it is still possible for a malicious node to behave in such a way that it is accepted by some honest nodes but not other honest nodes. This results in different honest nodes having different *views* on which set of nodes should form the system. Such *view divergence*, unfortunately, breaks the overarching assumption required by many existing security protocols.**

Partly spurred by the growing popularity of Bitcoin, researchers have recently formalized the above *view divergence* problem and proposed interesting solutions (which we call *view reconciliation protocols*). For example, in CRYPTO 2015, Andrychowicz and Dziembowski proposed a view reconciliation protocol with $\Theta(N)$ time complexity, with N being the number of honest nodes in the system. All existing view reconciliation protocols so far have a similar $\Theta(N)$ time complexity. As this paper's main contribution, we propose a novel view reconciliation protocol with a time complexity of only $\Theta(\frac{\ln N}{\ln \ln N})$. To achieve such an exponential improvement, we aggressively exploit randomization.

I. INTRODUCTION

Sybil attack and computational puzzles. The *sybil attack* [1] problem has attracted tremendous amount attention from researchers over the past two decades. In a sybil attack, the adversary creates a large number of fake identities/nodes and have them join a system. The adversary can then leverage them to effectively launch various follow-up attacks. Sybil attacks exploit the fact that modern distributed systems are typically *permissionless* and cannot rely on public key infrastructures, in the sense that they allow nodes to join the system without for example, going through an identity check based on credit card numbers to get permission to join.

The authors of this paper are alphabetically ordered. This work is partly supported by the research grant T1_251RES1616 from Singapore Ministry of Education Academic Research Fund Tier-1, and partly supported by a research grant R-252-000-661-592 from DSO, Singapore.

Computational puzzles [2], [3] have long been investigated as an approach to defend against sybil attacks. With computational puzzles, every node is asked to complete some non-trivial computational task simultaneously. If a node fails to complete such a task within some specific time constraint, then the node's identity will no longer be accepted by other nodes. If the attacker has limited computational power per time unit (e.g. only 10%) as compared to the aggregate computational power per time unit of all the N honest nodes, then the number of malicious nodes it can sustain in a sybil attack will also be limited (e.g., at most $0.1N$ malicious nodes). The developments of Bitcoin [4] in recent years have spurred much renewed interest in this well-known approach.

View divergence. Using computational puzzles, by itself, is yet not sufficient to offer a *security basis* for many permissionless distributed systems, for the following reason. In permissionless distributed systems, there is often no trusted central authority. Thus each honest node u will need to determine itself whether another node v should be accepted as part of the system. A malicious node v may then cooperate with u for the verification of its solution to the computational puzzle, but not with another honest node w. This results in different honest nodes having different *views* on which set of nodes should form the system. A further complication is that v (which is malicious) will also have its own view, and v can include many other sybil nodes in its view. At this point, we end up dealing with a (potentially infinite) set of nodes each with their own view about the membership of the system.

Such *view divergence*, unfortunately, breaks the basis of many security protocols in distributed systems: For example byzantine consensus protocols (e.g., [5]–[11]) and secure multi-party computation protocols (e.g., [12]–[14]) typically assume an agreed-upon set of nodes running the protocol. As another example, protocols for building dynamic and robust overlay networks [15], [16] typically have a bootstrapping stage during which they also assume an agreed-upon set of nodes running the protocol.

State-of-the-art protocols. Partly spurred by the growing popularity of Bitcoin [4], researchers have recently formalized [17], [18] the above view divergence problem, and proposed interesting solutions. Specifically, in CRYPTO 2015, Andrychowicz and Dziembowski [17] proposed a protocol enabling the honest nodes to agree on a final view which is guaranteed to contain all N honest nodes and no more than a certain threshold number ($f \cdot N$) of malicious nodes. Katz et al. [18] proposed another protocol offering similar functionalities. For convenience, we refer to all such protocols as *view reconciliation* protocols – namely protocols that enable all the nodes, starting from divergent views, to obtain the same final reconciled view. While the existing view reconciliation protocols are quite different from each other in terms of algorithmic techniques, all of them [17], [18] require $\Theta(N)$ *rounds* of execution time. Here each *round* needs to be long enough to accommodate message propagation delay in the network as well as some amount of local processing.

Our central result. As our main contribution, this paper proposes and analyzes a novel randomized view reconciliation protocol, called RandomizedViewReconciliation or RVR. We prove that RVR only takes $\Theta(\frac{\ln N}{\ln \ln N})$ rounds to terminate[1]. Our novel result has two major implications. First, relatively speaking, our protocol is an *exponential* improvement over the state-of-the-art $\Theta(N)$ protocols [17], [18]. Second, many security protocols (e.g., [15], [16]) for large-scale distributed systems have sub-linear time complexity, since linear time complexity is considered overly expensive. But since the previous view reconciliation protocols [17], [18] take $\Theta(N)$ rounds, their overhead often ends up being the dominating term in the end-to-end solution. Namely, without alleviating this $\Theta(N)$ bottleneck, additional research will not help to further reduce the asymptotic complexity of many problems. Our protocol with only $\Theta(\frac{\ln N}{\ln \ln N})$ rounds fills up this critical need.

Our RVR protocol has similar communication complexity as previous protocols per round: Each honest node in our protocol sends on expectation $\Theta(N \ln N)$ bits per round[2], as compared to $\Theta(N)$ in previous protocols [17], [18]. But the total number of bits sent by each honest node in our protocol (i.e., $\Theta(N \frac{\ln^2 N}{\ln \ln N})$) is much smaller than in previous protocols (i.e., at least $\Theta(N^2)$).

Finally, our RVR protocol currently assumes that the adversary's computational power is no more than the aggregate computational power of fN honest honest node, with f being any constant smaller than $\frac{1}{3}$. In comparison, [18] can tolerate all $f < 1$, while [17] can even tolerate $f > 1$ (though in most cases, to provide useful functionality later on, f needs to be below 1 anyway). Hence our protocol imposes a stronger restriction on the adversary's power than the existing protocols. It is our future work to investigate how to generalize to $f \geq \frac{1}{3}$.

Our approach. A key commonality among the existing view reconciliation protocols [17], [18] is that they are all deterministic (excluding invocations to crypto primitives). To decrease the time complexity, we aggressively exploit randomization and to allow a positive and tunable error probability δ. On the surface, having a positive error probability seems to be rather unacceptable for a security protocol. But even basic crypto primitives (such as public key crypto used in [17], [18]) have a positive probability of error: For example, the adversary might correctly guess the private key. This means that strictly speaking, even those previous protocols [17], [18] have positive error probability. The crux in our approach thus is to ensure that i) our error probability δ is tunable, and ii) δ decreases *exponentially* with the overhead of the protocol (as in the case of crypto primitives). Specifically in our RVR protocol, if δ were not viewed as a constant, then as δ approached zero, the time complexity of our RVR protocol would be $\Theta(\frac{\ln N}{\ln \ln N} \ln \frac{1}{\delta})$ rounds. Thus increasing the number of rounds decreases δ exponentially.

Allowing randomization opens up a lot of possibilities for better performance. For example, we will use computational puzzles to elect leaders, *probabilistically*. This differs from [17], [18] where the puzzles are used solely to challenge the computational power of the nodes. Our protocol will also rely on *randomized gossiping* and *randomized sampling* to achieve its end goal. As a side benefit of aggressively exploiting randomization, our protocol is rather elegant and conceptually much simpler than the previous solutions [17], [18].

Roadmap. Section II discusses related work, while Section III formalizes the view divergence problem. Section IV gives an overview of our RVR protocol, and Section V through VII elaborate on the key subroutines in RVR. Finally, Section VIII puts everything together.

II. Related Work

View reconciliation protocols. Computational puzzles [2], [3] have been investigated extensively for defending against sybil attacks in various contexts (e.g., [4], [19]–[21]). This work focuses on the view divergence problem that stems from the use of computational puzzles. Section I has already discussed the key differences between our RVR protocol and the two existing view reconciliation protocols [17], [18]. In addition, none of [17], [18] use randomization or the leader-based paradigm as we do, and none use any components similar to our three key subroutines.

Byzantine consensus protocols. Byzantine consensus protocols (e.g., [5]–[11]) are different but quite related to view reconciliation protocols. Both kinds enable the honest nodes to agree on something. But byzantine consensus protocols typically have the implicit assumption that the protocols are run over *some given set of nodes*. For example, a protocol typically assumes that there are $(1 + f)N$ nodes running the protocol, where the identities of these $(1 + f)N$ nodes are

[1] If the error probability δ (see definition later) is not viewed as a constant, then this will be $\Theta(\frac{\ln N}{\ln \ln N} \ln \frac{1}{\delta})$.

[2] If the error probability δ (see definition later) is not viewed as a constant, then this will be $\Theta(N \ln \frac{N}{\delta})$.

agreed upon by every node prior to the protocol. In our context, we do not have any given and agreed-upon set of nodes. In fact, establishing some agreed-upon set of $(1+f)N$ nodes is *precisely the goal of the view reconciliation problem itself.* It is worth noting that [5] explicitly deals with sybil attacks. But the work still assumes an agreed-upon set of nodes, and explicitly leaves the implementation of such an assumption to future work.

While byzantine consensus protocols cannot solve the view divergence problem, one can nevertheless build upon ideas from them. For example, the two existing view reconciliation protocols [17], [18] both build upon ideas from [8]. Our RVR protocol also builds upon a well-known leader-based paradigm [6], [11] widely used in byzantine consensus protocols. While this general paradigm has been well-known for decades, the key novel aspects in our protocol are the specifics of how we implement such a paradigm i) using only total $O(\frac{\ln N}{\ln \ln N})$ rounds, ii) with each honest node sending on expectation only $O(N \ln N)$ bits per round, and iii) under a setting where there is no agreed-upon set of nodes to start with. Note that *even with an agreed-upon set of nodes*, a direct application of the protocol in [6], [11] to our setting would result in a deterministic algorithm with $O(N)$ rounds and requiring each node to send $O(N^2)$ bits per round.

Bitcoin. While Bitcoin [4] is an excellent example of permissionless large-scale distributed system, Bitcoin itself does not solve the view divergence problem. In fact, the recently discovered network-level *eclipse attack* [22] on Bitcoin is one ultimate consequence of view divergence. Using a view reconciliation protocol to bootstrap, and then dynamically maintaining a robust overlay [15], [16] will offer a provable defense against such eclipse attacks in Bitcoin.

Recently, researchers have used the Bitcoin protocol (or its variants) as a building block to solve other problems in the permissionless settings, such as for achieving consensus and electing a committee (e.g., [23]). Solving these problems in the permissionless settings, in turn, would enable one to solve the view divergence problem. However, such an approach necessarily inherits all the assumptions needed by the Bitcoin protocol. In particular, the Bitcoin protocol assumes the existence of a *genesis block*, which is unpredictable, agreed upon by all honest nodes, and released to all nodes simultaneously. As a direct contrast, our RVR protocol, as well as the previous view reconciliation protocols [17], [18], avoids such a critical assumption, by solving the view divergence problem directly rather than indirectly via Bitcoin.

III. Problem formulation

Since the view divergence problem has already been previous formulated in [17], [18], this section will follow the existing formulation in [17], to the extent possible.

System model and attack model. The system has N *honest nodes* that always follows the protocol. The honest nodes do not know N, and the lack of such knowledge will be one technical difficulty that we overcome. Each honest node holds a (locally and randomly generated) public/private key pair, and the public key is the unique identity of the node. To simplify discussion, we will often simply use "node" to refer the public key of that node. Whenever a node sends a message, it includes its public key (as the sender's "identity") and then append a digital signature. Each node also has an IP address and a port number. Each honest node has one unit of computational power.

The adversary can have an unlimited number of *malicious nodes*, public/private key pairs, IP addresses and port numbers. Malicious nodes are byzantine and colluding. Independent of the number of malicious nodes, the adversary has total fN units of computational power, for some constant f. The adversary may spoof the IP addresses of the honest nodes, and can inject arbitrary messages into the network. It also sees all messages in the network. But the adversary cannot corrupt or remove a message from the network. The adversary cannot break crypto primitives such as digital signatures.

To allow a direct comparison, we follow [17], [18] and assume that the honest nodes are synchronized. Time is divided into rounds, where each round is sufficiently long for every honest node to do some local processing and for all messages sent at the beginning of the round to reach the corresponding receivers by the end of that round. Our protocol actually works even if the honest nodes are "out-of-synch" with each other when starting the protocol (see Section IV).

Hash functions and security parameter. We will use hash functions for various purposes in our protocol. When proving the formal guarantees, we follow [17] and model all our hash functions as random oracles [24]. The outputs of the hash functions will be viewed as integers in the range of $[0, \texttt{max\_hash})$. We use m to denote the number of hash operations that one unit of computational power can perform in one round. For simplicity, all asymptotic complexities in this paper treat the length of security keys, hashes, nonces, and so on, as $O(1)$.[3]

View reconciliation. In the *view reconciliation* problem, each node u has an initial view $\texttt{initview}_u$, which is a set of nodes (i.e., the public keys of the nodes). The initial views are established using computational puzzles, via some existing approach (which we will review later). For all honest node u, $\texttt{initview}_u$ contains all the N honest nodes (since each honest node can properly solve one puzzle). Let `union_honest_initview` be the union of the initial views of all the honest nodes. Then `union_honest_initview` contains at most fN malicious nodes. This constraints come from the adversary being able to solve only fN puzzles. Note that each puzzle solution corresponds to a single malicious node v, since the puzzle is tied to the public key of v. For a given malicious node v, the adversary can cause v to only be included in $\texttt{initview}_u$ and not in $\texttt{initview}_w$, thus causing view divergence.

[3]If needed, one could use a security parameter λ, in which case the asymptotic time complexity of RVR and the previous protocols [17], [18] will not be affected, while the communication complexities in all cases will increase by a factor of $\Theta(\lambda)$.

The goal of *view reconciliation* is to give each honest node u a final view $\texttt{final\_view}_u$ such that i) the final views of all honest nodes are identical, and ii) the final view contains all the N honest nodes and at most fN malicious nodes. One may observe that the set `union_honest_initview` already satisfies these properties. But it is difficult to determine `union_honest_initview`, since we do not know which nodes are honest.

Establishing the initial views. The initial views on the nodes can be established using similar approaches in existing works [5], [17], [18]. This is not our contribution, not part of the view reconciliation problem, and not part of RVR. Hence, we only provide a concise review below.

First, there needs to be some mechanism for the nodes to initially learn about each other. This is addressed in [17], [18] by assuming the existence of some (insecure) *public channel*. Every node may post messages to the public channel. A message posted by an honest node will be received by all honest nodes, while a message posted by a malicious node may reach any subset (as chosen by the adversary) of the honest nodes. This public channel can be implemented by flooding.

Now if every node posts to the public channel its public key, IP address/port, together with a digital signature, then every node u will learn about all the honest nodes, together with some arbitrary number of malicious nodes. Let this set be $\texttt{pre\_initview}_u$. Next every u in the system will simultaneously send a random challenge to all nodes in $\texttt{pre\_initview}_u$ A receiving node v will combine all the challenges it receives (e.g., using a Merkle tree) into a single challenge. Node v will then solve a computation puzzle instantiated by the challenge, and then send back the puzzle solution (as well as the Merkle proofs) to each node u from which v received a challenge. Once u verifies the solution and the proof, u will include v in $\texttt{initview}_u$. For simplicity, we will assume that an honest node v can always successfully solve the puzzle, while the adversary can solve total no more than fN puzzles here.

IV. Our Protocol: Randomized View Reconciliation

This section provides an overview of our novel RandomizedViewReconcile (or RVR in short) protocol.

Messages with signatures. In RVR, each node u maintains a view ($\texttt{view}_u$) that can change over time. Initially, $\texttt{view}_u = \texttt{initview}_u$, and at the end of the protocol, $\texttt{final\_view}_u$ will be set to $\texttt{view}_u$. Throughout the protocol, u only sends messages to and receives messages from node in its initial view $\texttt{initview}_u$. A message always contains the sender, the receiver, a sequence number, and a digital signature, to enable authentication as well as to prevent replay attacks. A node u will discard all message received from nodes not in $\texttt{initview}_u$, and also all messages whose sequence numbers are not expected. RVR has a certain number of iterations (see Algorithm 1), and the iteration number is also included in a message. In a particular iteration, a node will only use messages with the corresponding iteration number. To be concise, we will not explicitly mention these mechanisms in the pseudo-code or in the discussion.

Algorithm 1 RandomizedViewReconcile or RVR in short
input: f, δ, `offset`, u, $\texttt{initview}_u$, m;
output: $\texttt{final\_view}_u$;

$\texttt{view}_u = \texttt{initview}_u$;
repeat $6 \ln \frac{2}{\delta}$ times **do**
  $\texttt{leader} = \mathsf{ProbLeaderElect}(f, \texttt{offset}, u, \texttt{initview}_u, m)$;
  $\texttt{score}_u[\cdot] = \mathsf{TwoStageSample}(f, \delta, \texttt{offset}, u, \texttt{initview}_u, \texttt{view}_u)$;
  foreach v **do**
    if $\texttt{score}_u[v] \geq 0.50$ **then** add v to $\texttt{proposal}_u$;
    if $\texttt{score}_u[v] \geq 0.75$ **then** add v to $\texttt{over}_u$;
    if $\texttt{score}_u[v] \leq 0.25$ **then** add v to $\texttt{under}_u$;
  $\texttt{proposal} = \mathsf{CoordinatedGossip}(f, \delta, \texttt{offset}, u, \texttt{view}_u, \texttt{initview}_u, \texttt{leader}, \texttt{proposal}_u)$;
  $\texttt{view}_u = \texttt{over}_u \cup (\texttt{proposal} \setminus \texttt{under}_u)$;
  $\texttt{offset} = 2$; //`offset` is 2 once gossiping is done
end
$\texttt{final\_view}_u = \texttt{view}_u$; **return** $\texttt{final\_view}_u$;

Non-simultaneous start. Different honest nodes do not have to start running RVR at the same time. When we say that all honest nodes starts executing a certain protocol *within* `offset` *rounds of each other*, we mean that the time between the first honest node starting and the last honest node starting is at most $(\texttt{offset} - 1)$ rounds. Hence $\texttt{offset} = 1$ means simultaneous start. The value of `offset` will be fed into our protocol as a parameter.

Allowing $\texttt{offset} \geq 2$ enables our protocol to work even if the nodes are not perfectly in synch. Furthermore, even if all the nodes start simultaneously, due to the lack of knowledge of N, the nodes may get out-of-synch in the CoordinatedGossip subroutine. Hence we will need to allow offset in all other parts of our protocol anyway.

Overview of design. RVR has a certain number of iterations (see Algorithm 1). At a high level, we build upon a well-known leader-based paradigm [6], [11] (see Section II for discussion on related work): In each iteration some leader *proposes* a certain view to all the nodes. The protocol ensures that if the leader is honest, then all nodes will properly adopt the proposal. This adopted proposal (i.e., view) will already contain N honest nodes and at most fN malicious nodes, which means that it has all the desired properties. The protocol further ensures that in later iterations, even if the leader is malicious, the previously adopted proposal will not be disrupted — namely, the (bad) proposal made by a malicious leader will not be adopted (see proof of Theorem 8).

Three key subroutines. Our overarching approach in RVR to achieving small time complexity is to aggressively rely on randomization, and then carefully ensure that the error δ resulted from randomization can be decreased *exponentially*. Specifically, RVR uses three key subroutines, all of which are randomized. The first ProbLeaderElect subroutine aims to elect an honest leader in the system. Doing so perfectly will be challenging — we have unlimited number of malicious nodes in the system. But fortunately, the leader election here does not need to be always correct. Hence we elect the leader by simply having the nodes solve computational puzzles. Whoever solves the puzzle will become the leader. It is certainly possible to have multiple nodes solving the puzzles at roughly the same time, which would result in multiple leaders. This gets further complicated when the nodes do not start executing ProbLeaderElect simultaneously. Section V will show that with a proper puzzle difficulty, we can nevertheless ensure a positive constant probability of having a unique and honest leader.

Our CoordinatedGossip subroutine allows the elected leader to disseminate its proposal to all honest nodes, via randomized gossiping. A difficulty here is that because the honest nodes do not know N, they may disagree on how many rounds they should gossip for. As a result, they may get out-of-synch with each other when they finish CoordinatedGossip. Making them fully synchronous would require solving the classic *byzantine firing squad* problem [25]. Unfortunately, byzantine firing squad protocols typically assume that the protocols are run over *some given set of nodes* (which we do not have until after view reconciliation is complete). To overcome this difficulty, we will design a simple yet elegant mechanism so that the nodes will return from CoordinatedGossip within 2 rounds (instead of 1 round, which would imply perfect synchrony) of each other.

Finally, in the leader-based paradigm, when the leader forms its proposal and when a node decides whether to adopt the leader's proposal, they need to collect the current views of all the honest nodes. Naturally in our setting, a node u will conceptually collect the views of all those nodes in $\texttt{initview}_u$. Doing so directly entails large communication complexity, hence we instead randomly sample some of the nodes for their views. A tricky part is that the malicious nodes either may request for too many samples (hence blowing up communication complexity) or may push samples aggressively to other nodes (hence causing their views to be over-represented). We will use *two-stage sampling* in our TwoStageSample subroutine to overcome this issue.

Provable guarantees. Algorithm 1 gives the pseudo-code for the RVR. We defer a more detailed discussion of the pseudo-code and the formal analysis to Section VIII, after we elaborate the three key subroutines in Section V through VII.

V. Probabilistic Leader Election

Conceptual design. Recall that RVR allows a δ probability of error. We say that ProbLeaderElect *succeeds* if it elects a unique leader and the elected leader is an honest node. As long as ProbLeaderElect succeeds with some constant probability (e.g., 0.16 as in our proof later), repeating it for $O(\ln \frac{1}{\delta})$ times can already guarantee at least one success with $1-\delta$ probability. By our design of RVR, we only need one successful iteration anyway, to achieve our final goal.

Algorithm 2 ProbLeaderElect
input: f, $\texttt{offset}$, u, $\texttt{initview}_u$, m; **output:** $\texttt{leader}$;

1: generate fresh $\texttt{challenge}_u$ and send $\langle u, \texttt{challenge}_u \rangle$ to all nodes in $\texttt{initview}_u$;
2: spend $\texttt{offset}$ rounds receiving messages in the form of $\langle w, \texttt{challenge}_w \rangle$;
3: form a Merkle tree with all the $\texttt{challenge}_w$'s received as leaves, and let $\texttt{root}_u$ be the root of the Merkle tree;
4: spend $6 \cdot \texttt{offset}$ rounds trying to find x such that $\frac{\mathsf{hash}(x,u,\texttt{root}_u)}{\texttt{max\_hash}} \le \frac{1}{6m(1+f)|\texttt{initview}_u|\cdot\texttt{offset}}$;
5: if find such x, send $\langle x, u, \texttt{root}_u, \texttt{off\_path\_hashes} \rangle$ to all nodes in $\texttt{initview}_u$;
6: spend $\texttt{offset}$ rounds receiving messages in the form of $\langle y, w, \texttt{root}_w, \texttt{off\_path\_hashes} \rangle$;
7: $\texttt{leader} = \texttt{null}$;
8: **for each** $\langle y, w, \texttt{root}_w, \texttt{off\_path\_hashes} \rangle$ received
9: **if** $\texttt{off\_path\_hashes}$ validates properly and **if** $\frac{\mathsf{hash}(y,w,\texttt{root}_w)}{\texttt{max\_hash}} \le \frac{1}{6m|\texttt{initview}_u|\cdot\texttt{offset}}$ **then** $\texttt{leader} = w$;
10: **return** $\texttt{leader}$;

This leads to the following design: We will exploit the limit on the adversary's computation power. We let all the nodes in the system solve computational puzzles. Whoever solves the puzzle will claim itself as the leader, by directly notifying all other nodes. ProbLeaderElect fails if i) a malicious node solves the puzzle, or ii) none of the honest nodes solve the puzzle, or iii) more than one honest nodes solve the puzzle. By choosing a proper puzzle difficulty, we will be able to properly bound the probability of these three events away from 1.

Details of the design. Algorithm 2 gives the pseudo-code for ProbLeaderElect. To instantiate the computational puzzles, each node w generates a fresh random nonce as $\texttt{challenge}_w$ at Line 1 and sends to every node in $\texttt{view}_w$. Each node u will collect challenges from all nodes in $\texttt{initview}_u$, and then construct a standard Merkle tree with root $\texttt{root}_u$. Next u will spend $6 \cdot \texttt{offset}$ rounds (at Line 4) trying to find a solution x such that $\frac{\mathsf{hash}(x,u,\texttt{root}_u)}{\texttt{max\_hash}} \le \frac{1}{6m(1+f)|\texttt{initview}_u|\cdot\texttt{offset}}$. (We will explain this threshold later.) If u finds such x, it will notify every node w in $\texttt{initview}_u$, while including a standard Merkle proof showing that $\texttt{challenge}_w$ has been included in the Merkle tree, if any. Node w will validate (at Line 9) that the solution x is for a sufficiently hard puzzle, by checking that $\frac{\mathsf{hash}(x,u,\texttt{root}_u)}{\texttt{max\_hash}} \le \frac{1}{6m|\texttt{initview}_w|\cdot\texttt{offset}}$. The threshold $\frac{1}{6m|\texttt{initview}_w|\cdot\texttt{offset}}$ here is such that regardless of the values of $|\texttt{initview}_u|$ and $|\texttt{initview}_w|$, a puzzle solution obtained at Line 4 on an honest node u can always be validated at Line 9 on an honest node w. Furthermore, the threshold $\frac{1}{6m|\texttt{initview}_w|\cdot\texttt{offset}}$ is such that on expectation in the system, there will roughly be one honest node finding a puzzle solution. However, this threshold may allow the malicious nodes to solve a somewhat easier puzzle, which will be addressed when we prove the guarantees of the subroutine.

Provable guarantees. Theorem 1 proves the complexity and success probability of ProbLeaderElect. Our proof will properly take care of the fact that the malicious nodes may i) solve easier puzzles, and ii) have more time to solve the puzzles compared to the honest nodes.

Theorem 1. *Assume that* $m \geq 1$*,* $N \geq 1000$*, and* $f < \frac{1}{3}$*. If all honest nodes start running* ProbLeaderElect *within* `offset` *rounds of each other, then:*

- *Each honest node returns after exactly* $8 \cdot \texttt{offset}$ *rounds, while sending* $\Theta(N \ln N)$ *bits in each round.*
- *With probability at least* 0.16*,* ProbLeaderElect *returns the same* `leader` *on all honest nodes* and *the* `leader` *returned is an honest node.*

Proof. We only prove the second claim since the first claim is trivial. First consider all the honest nodes. Let h denote the total number of hash operations performed at Line 4 across all N honest nodes. Hence we have $h = 6mN \cdot \texttt{offset}$. The threshold at Line 4 may range from $\frac{1}{6mN(1+f)^2 \cdot \texttt{offset}}$ to $\frac{1}{6mN(1+f) \cdot \texttt{offset}}$ on different u's. Let p be the probability of a hash operation done by any honest node finding a solution x at Line 4. We thus have $\frac{1}{6mN(1+f)^2 \cdot \texttt{offset}} \leq p \leq \frac{1}{6mN(1+f) \cdot \texttt{offset}}$. Note that a solution x found by an honest node at Line 4 is guaranteed to pass the checking at Line 9 on every honest node w, since the threshold at Line 9 is $\frac{1}{6m|\texttt{initview}_w| \cdot \texttt{offset}} \geq \frac{1}{6mN(1+f) \cdot \texttt{offset}}$.

Next consider the malicious nodes. Let h' denote the total number of hash operations done by the fN malicious nodes, in their attempts to find a solution to the puzzle. A malicious node will have at most $8 \cdot \texttt{offset}$ rounds to solve the puzzle. [4] Thus $h' \leq 8fmN \cdot \texttt{offset}$. For a puzzle solution y from some malicious node w to be validated properly by honest node u at Line 9 of u's execution, we need to have $\frac{\mathsf{hash}(y,u,\mathtt{root}_w)}{\texttt{max\_hash}} \leq \frac{1}{6m|\texttt{initview}_u| \cdot \texttt{offset}}$. Hence $\frac{\mathsf{hash}(y,u,\mathtt{root}_w)}{\texttt{max\_hash}}$ needs to be no larger than $\frac{1}{6mN \cdot \texttt{offset}}$, in order for y to be possibly validated on *any* honest node. Let p' be the probability of a hash operation done by any malicious node finding a solution y such that $\frac{\mathsf{hash}(y,\mathtt{root}_w)}{\texttt{max\_hash}} \leq \frac{1}{6mN \cdot \texttt{offset}}$. We have $p' \leq \frac{1}{6mN \cdot \texttt{offset}}$.

Let $\mathcal{E}_1$ denote the event that ProbLeaderElect returns the same `leader` on all honest nodes with `leader` being an honest node, and let $\mathcal{E}_2$ denote the event that exactly one hash operation done by some honest node finds a solution at Line 4 and no other hash operations (done by honest or malicious nodes) find a solution. Based on the ranges of p and p', we have: $Pr[\mathcal{E}_1] \geq \Pr[\mathcal{E}_2] \geq h \cdot \left(\frac{1}{6mN(1+f)^2 \cdot \texttt{offset}}\right)^1 \cdot \left(1 - \frac{1}{6mN(1+f) \cdot \texttt{offset}}\right)^{h-1} \cdot \left(1 - \frac{1}{6mN \cdot \texttt{offset}}\right)^{h'}$. Plugging in h and h', and with some manipulation, one can eventually show that for $m \geq 1$, $N \geq 1000$, and $f < \frac{1}{3}$, the above probability is at least: $\frac{1}{(1+f)^2} \cdot 0.36^{\frac{1}{1+f}} \cdot 0.36^{\frac{4f}{3}} > 0.16$. ☐

VI. Coordinated Gossiping

Conceptual design. CoordinatedGossip aims to disseminate a message (i.e., $\texttt{proposal}_{\texttt{leader}}$) from the `leader` to all honest nodes, after ProbLeaderElect has been invoked and succeeded. If ProbLeaderElect did not succeed, we will not care about the correctness of CoordinatedGossip (though we still care about its performance overhead).

The basic idea in CoordinatedGossip is simple: `leader` will first generate a signature on $\texttt{proposal}_{\texttt{leader}}$. Next in each round, each node u will relay $\texttt{proposal}_{\texttt{leader}}$, together with the signature, to $O(\log \frac{N}{\delta})$ nodes in u's current view. Before relaying, a node needs to check `proposal`'s size to avoid sending too many bits when the leader is malicious. A receiving node will verify `leader`'s signature before accepting it. We will prove that $\frac{3 \ln N}{2 \ln \ln N}$ rounds is sufficient for all honest nodes to receive $\texttt{proposal}_{\texttt{leader}}$ with probability close to 1. During such gossiping process, each node only sends $O(N \ln \frac{N}{\delta})$ bits per round. In comparison, directly having `leader` send $\texttt{proposal}_{\texttt{leader}}$ to all nodes in its view would require `leader` to send $\Omega(N^2)$ bits.

The only difficulty in CoordinatedGossip is that the nodes do not know N, and hence do not know for how many rounds they should gossip. Each node u does have $\texttt{initview}_u$, and can use $|\texttt{initview}_u|$ in place of N for calculating $\frac{3 \ln N}{2 \ln \ln N}$. However, $|\texttt{initview}_u|$ is different on different u's. This means that two nodes u and w may spend different number of rounds in running CoordinatedGossip. A further problem is that CoordinatedGossip will be invoked multiple times in RVR, making u and w more and more out-of-synch each time the subroutine is invoked. Making all the nodes return from CoordinatedGossip simultaneously would correspond to the classic *byzantine firing squad* problem [25]. Unfortunately, same as typical byzantine consensus protocols, byzantine firing squad protocols [25] also have the implicit assumption that the

Algorithm 3 CoordinatedGossip
input: f, δ, `offset`, u, $\texttt{initview}_u$, `leader`, $\texttt{proposal}_u$;
output: `proposal`;

1: **if** $u \neq \texttt{leader}$ **then** $\texttt{proposal} = \texttt{null}$ **else** $\texttt{proposal} = \texttt{proposal}_u$ (attached with a signature from u);
2: $\texttt{count} = 0$;
3: **repeat** $\frac{3 \ln |\texttt{initview}_u|}{2 \ln \ln |\texttt{initview}_u|} + \texttt{offset}$ rounds **do**
4: // each iteration here takes one round
5: **if** $\texttt{proposal} \neq \texttt{null}$ **and** $|\texttt{proposal}| \leq (1+f)|\texttt{initview}_u|$ **then** send `proposal` (together with the attached signature) to $8 \ln \frac{|\texttt{initview}_u|}{\delta}$ uniformly random node chosen from $\texttt{initview}_u$;
6: **for each** message `msg` received **do**
7: **if** $\texttt{msg} = \texttt{msg\_fin}$ **then** increment `count`
8: **else if** `msg` has `leader`'s signature attached **then** $\texttt{proposal} = \texttt{msg}$;
9: **if** $\texttt{count} > \frac{f}{1+f} \cdot |\texttt{initview}_u|$ **then** break;
10: **end**
11: send `msg_fin` to all nodes in $\texttt{initview}_u$;
12: wait until `msg_fin` is received (including `msg_fin` received at Line 6) from at least $\frac{|\texttt{initview}_u|}{1+f}$ nodes;
13: **return** `proposal`;

[4]The malicious node may successfully obtain challenges from all the honest nodes at the very beginning, spend $8 \cdot \texttt{offset}$ rounds, and then finally send the puzzle solution right before the honest nodes stops receiving solutions.

protocols are run over *some given set of nodes*.

In our design of CoordinatedGossip, we do not make all the honest nodes return in the same round. Rather, we ensure that there exists some r, such that each honest node either returns in round r or in round $r+1$. Achieving this will be much easier than solving the byzantine firing squad problem under our setting. Furthermore, our guarantee continues to hold even if the nodes do not start executing CoordinatedGossip simultaneously. This ensure that the "out-of-synch" will not accumulate over multiple invocations of the subroutine.

Details of the design. Algorithm 3 gives the pseudo-code for CoordinatedGossip. Our central goal here is to ensure that all honest nodes return within two rounds of each other. To achieve this, when a node u has finished its gossiping, it will send out a special `msg_fin` message to all nodes in $\mathtt{initview}_u$. Node u will then wait until it has received `msg_fin` from $\frac{|\mathtt{initview}_u|}{1+f}$ nodes in $\mathtt{initview}_u$ (at Line 12), before u finally returns from the subroutine. Note that $\frac{|\mathtt{initview}_u|}{1+f} \leq N$ and $\mathtt{initview}_u$ always contains N honest nodes. Hence as long as all N honest nodes send out `msg_fin`, u is guaranteed to eventually receive `msg_fin` from a sufficient number of nodes and return.

Let r be the round during which u returns. Consider any other honest node v. We want v to return no later than round $r+1$. To achieve this, we let v break from the gossiping process and move on (at Line 9), as long as v has received `msg_fin` from more than $\frac{f}{1+f} \cdot |\mathtt{initview}_v|$ nodes in its view. We will be able to prove later that at least one node out of all these $\frac{f}{1+f} \cdot |\mathtt{initview}_v|$ nodes must be honest. Hence the malicious nodes by themselves will not be able to trick v into breaking prematurely. At the same time, note that u's returning in round r (at Line 12) implies that u has received `msg_fin` from $\frac{|\mathtt{initview}_u|}{1+f}$ nodes. We will be able to prove later that, out of these $\frac{|\mathtt{initview}_u|}{1+f}$ nodes, at least $(1-f)N$ nodes are honest. Hence v will also receive `msg_fin` from at least $(1-f)N$ nodes in its view in round r. Since $(1-f)N > fN \geq \frac{f}{1+f} \cdot |\mathtt{view}_v|$ for all constants $f < \frac{1}{3}$, these `msg_fin` messages will be sufficient for v to break from the gossiping process (at Line 9). We will later prove that v will then return in the next round.

Provable guarantees. The following theorem presents the complexity of CoordinatedGossip, and further proves that i) all honest nodes will return within two rounds of each other from CoordinatedGossip, and ii) there will be a sufficient number of rounds during which all honest nodes are gossiping.

Theorem 2. *Assume $f < \frac{1}{3}$. If all honest nodes start executing* CoordinatedGossip *within* `offset` *rounds of each other, then:*

1) *Each honest node sends $\Theta(N \ln \frac{N}{\delta})$ bits in each round of the execution;*
2) *Each honest node spends at most $\frac{3\ln((1+f)N)}{2\ln\ln((1+f)N)} + (2 \cdot \mathtt{offset}) + 1$ rounds before returning from the subroutine;*
3) *All honest nodes will return within 2 round of each other;*
4) *There are at least $\frac{3\ln N}{2\ln\ln N}$ rounds during which all honest nodes are executing the loop at Line 3.*

Proof.

1) Obvious from the pseudo-code.
2) Every honest node u will eventually reach Line 11 and send `msg_fin` to all nodes, no later than $\frac{3\ln((1+f)N)}{2\ln\ln((1+f)N)} + \mathtt{offset} + 1$ rounds after u starts executing CoordinatedGossip. For any other given honest node v, u will send its `msg_fin` message no later than $\frac{3\ln((1+f)N)}{2\ln\ln((1+f)N)} + (2 \cdot \mathtt{offset}) + 1$ rounds *after v starts executing* CoordinatedGossip. There will be total N such u's. Receiving finished messages from all of them will enable v to return.
3) Let round r be the first round during which one or more honest nodes return. Let u be one such honest node, where $\mathtt{initview}_u$ contains N honest nodes and aN malicious nodes, for some a where $0 \leq a \leq f$. Consider any other honest node v. We claim that if v does not return in round r, then it must return in round $r+1$. To see why, note that u must have received `msg_fin` (at Line 12) from $\frac{|\mathtt{initview}_u|}{1+f} = \frac{1+a}{1+f}N$ nodes by round r. At least $\frac{1+a}{1+f}N - aN$ such messages must be from honest nodes. For all $f < \frac{1}{3}$, we have $\frac{1+a}{1+f}N - aN = \frac{1-af}{1+f}N \geq (1-f)N > fN \geq |\mathtt{view}_v| \frac{f}{1+f}$. This means that v must satisfy the condition at Line 9 by round r, and thus break from the loop. Once v breaks, in round $r+1$, it will send out `msg_fin`. In fact, such argument applies to all honest nodes, and they will all have sent out `msg_fin` by round $r+1$. All these $N \geq \frac{|\mathtt{view}_v|}{1+f}$ `msg_fin` messages will be received by v and enable v to return in round $r+1$.
4) For all honest node u, we have $N \leq |\mathtt{initview}_u| \leq (1+f)N$, and hence the number of iterations in the loop at Line 3 will be from $\frac{3\ln N}{2\ln\ln N} + \mathtt{offset}$ to $\frac{3\ln((1+f)N)}{2\ln\ln((1+f)N)} + \mathtt{offset}$. Since all honest nodes starts executing CoordinatedGossip within `offset` rounds of each other, there will be at least $\frac{3\ln N}{2\ln\ln N}$ rounds during which all honest nodes are executing the loop, unless the condition at Line 9 is satisfied during these rounds. To prove that the condition is not satisfied, we need to show that those `msg_fin` messages from the malicious nodes will never be sufficient to satisfy the condition. Consider any honest node u where $\mathtt{initview}_u$ contains N honest nodes and aN malicious nodes, for some a where $0 \leq a \leq f$. For the condition at Line 9, it is easy to verify that $\frac{f}{1+f} \cdot |\mathtt{initview}_u| = \frac{f}{1+f} \cdot (1+a)N \geq aN$. This means that the condition for u will never be satisfied by those `msg_fin` messages from those aN malicious nodes. □

The next theorem proves the success probability of CoordinatedGossip:

Theorem 3. *Assume that $N \geq 1000$, $\delta \leq 0.1$, and $f < \frac{1}{3}$. If i) all honest nodes start executing* CoordinatedGossip *within* `offset` *rounds of each other, ii) all honest nodes invoke* CoordinatedGossip *using the same* `leader` *parameter, iii)* `leader` *is an honest node, and iv) $|\mathtt{proposal}_{\mathtt{leader}}| \leq (1+f)N$, then with probability at least $1 - \frac{1}{4000}$, all honest nodes return* $\mathtt{proposal}_{\mathtt{leader}}$.

Proof. First, because of the signature verification at Line 8, only the `leader`'s proposal will ever be adopted at Line 8. Furthermore, because $|\mathtt{proposal}_{\mathtt{leader}}| \leq (1+f)N \leq (1+$

$f)|\texttt{initview}_u|$, a node that has adopted the leader's proposal will always send the proposal in Line 5. In any given round, we call an honest node as a *black node* if it has adopted the leader's proposal at Line 8. Otherwise the honest node is a *white node*. A malicious node is neither black nor white. Theorem 2 has proved that there are at least $\frac{3\ln N}{2\ln\ln N}$ rounds during which all honest nodes are executing the loop at Line 3. For convenience, in this proof, number these rounds as round 1 through $\frac{3\ln N}{2\ln\ln N}$. For any given round r between 1 and $\frac{3\ln N}{2\ln\ln N}$, define b_r to be the number of black nodes at the end of that round. We will separately prove the following two key equations:

$$\Pr[b_{\frac{\ln N}{\ln\ln N}} \geq 0.2N] \quad \geq \quad 1 - \frac{1}{8000} \quad (1)$$

$$\Pr[b_{\frac{3\ln N}{2\ln\ln N}} = N \mid b_{\frac{\ln N}{\ln\ln N}} \geq 0.2N] \quad \geq \quad 1 - \frac{1}{8000} \quad (2)$$

Combining these two via a union bound will directly lead to the theorem.

We first prove Equation 2, which is easier. Since $b_{\frac{\ln N}{\ln\ln N}} \geq 0.2N$, starting from the beginning of round $\frac{\ln N}{\ln\ln N}+1$, in each round we have at least $0.2N$ black nodes. We call a black node's sending $\texttt{view}'$ at Line 5 as a *push*. Each black node u does $8\ln\frac{|\texttt{initview}_u|}{\delta} \geq 8\ln\frac{N}{\delta}$ pushes per round. Hence from round $\frac{\ln N}{\ln\ln N}$ to round $\frac{3\ln N}{2\ln\ln N}$, there will be at least total $0.2N \cdot 8\ln\frac{N}{\delta} \cdot \frac{\ln N}{2\ln\ln N} = 0.8 \cdot N\ln N \cdot \frac{\ln\frac{N}{\delta}}{\ln\ln N} > 3.6N\ln N$ pushes (since $N \geq 1000$ and $\delta \leq 0.1$). There are at most $0.8N$ white nodes at the end of round $\frac{\ln N}{\ln\ln N}$. For any such white node v and any given push, v is chosen as the target of the push with probability at least $\frac{1}{(1+f)N} > \frac{2}{3N}$. Hence the probability that v is never chosen as the target of any of the $3.6N\ln N$ pushes is at most:

$$\begin{aligned}(1-\frac{2}{3N})^{3.6N\ln N} &= ((1-\frac{2}{3N})^{\frac{3N}{2}})^{2.4\ln N}\\ &< e^{-2.4\ln N} = \frac{1}{N^{2.4}}\end{aligned}$$

Taking a union bound across all $0.8N$ possible v's, we know that with probability at least $1-\frac{0.8}{N^{1.4}}$, all white nodes have been chosen as the target of some push at least once by the end of round $\frac{3\ln N}{2\ln\ln N}$. Hence $\Pr[b_{\frac{3\ln N}{2\ln\ln N}} = N \mid b_{\frac{\ln N}{\ln\ln N}} \geq 0.2N] \geq 1-\frac{0.8}{N^{1.4}} > 1-\frac{1}{8000}$ (since $N \geq 1000$).

We next move on to Equation 1. We will later prove the following key equation for all r:

$$\Pr[b_{r+1} \geq \min(0.2N, b_r\ln\frac{N}{\delta})] \quad \geq \quad 1-\frac{\delta^{1.125}}{N^{1.125}} \quad (3)$$

It is easy to prove Equation 1 via Equation 3. Consider rounds 1 through $\frac{\ln N}{\ln\ln N}$. With probability at least $1-\frac{\delta^{1.125}}{N^{1.125}} \times \frac{\ln N}{\ln\ln N}$, we have $b_{r+1} \geq \min(0.2N, b_r\ln\frac{N}{\delta})$ for all $1 \leq r \leq \frac{\ln N}{\ln\ln N}$. Since $(\ln\frac{N}{\delta})^{\frac{\ln N}{\ln\ln N}} > 0.2N$, this will immediately imply $b_{\frac{\ln N}{\ln\ln N}} \geq 0.2N$. Hence we have $\Pr[b_{\frac{\ln N}{\ln\ln N}} \geq 0.2N] \geq 1-\frac{\delta^{1.125}}{N^{1.125}} \times \frac{\ln N}{\ln\ln N} > 1-\frac{1}{8000}$ (since $N \geq 1000$ and $\delta \leq 0.1$).

Now the only missing part is the proof for Equation 3. If $b_r \geq 0.2N$, then b_{r+1} must be at least $0.2N$ and Equation 3 holds trivially. Now consider the case where $b_r < 0.2N$. At the beginning of round $r+1$, there are total b_r black nodes, doing total at least $8b_r\ln\frac{N}{\delta}$ pushes in round $r+1$. Order all these pushes into an arbitrary sequence, and it will be convenient to imagine that these pushes are done one by one sequentially. For each push, we say it is *effective* if the target of the push is a white node at the time of the push. Before the number of black nodes reaches $\min(0.2N, b_r\ln\frac{N}{\delta})$, the probability of a push being effective is at least $\frac{N-0.2N}{(1+f)N} \geq \frac{0.8}{1+f} > \frac{0.8}{1.5} > 0.5$. Having $b_r\ln\frac{N}{\delta}$ effective pushes is sufficient for the number of black nodes (and hence b_{r+1}) to reach $\min(0.2N, b_r\ln\frac{N}{\delta})$. Let random variable x denote the number of pushes needed (starting from the beginning of the above sequence) for the number of black nodes to reach $\min(0.2N, b_r\ln\frac{N}{\delta})$. If the number of black nodes never reaches this quantity after all the pushes in the sequence, x is defined to be ∞.

In the next, we will reason about x via a simple coupling argument. Consider a separate experiment where we flip a fair coin for $8b_r\ln\frac{N}{\delta}$ times. Let random variable y denote the number of flips done, in order to encounter the first $b_r\ln\frac{N}{\delta}$ heads. (We define $y=\infty$ if we do not encounter so many heads even after all the flips.) We can easily couple x and y in such a way that for all a, we have $\Pr[x \leq a] \geq \Pr[y \leq a]$. Putting everything so far together, we have $\Pr[b_{r+1} \geq \min(0.2N, b_r\ln\frac{N}{\delta})] = \Pr[x \leq 8b_r\ln\frac{N}{\delta}] \geq \Pr[y \leq 8b_r\ln\frac{N}{\delta}]$.

Now define random variable z to be the total number of heads observed when flipping a fair coin for $8b_r\ln\frac{N}{\delta}$ times. We draw a trivial connection between y and z, and then invoke a standard Chernoff bound on z:

$$\begin{aligned}&\Pr[y > 8b_r\ln\frac{N}{\delta}] = \Pr[z < b_r\ln\frac{N}{\delta}]\\ =\; &\Pr[z < (1-0.75)(4b_r\ln\frac{N}{\delta})]\\ \leq\; &\exp(-\frac{1}{2}(0.75)^2(4b_r\ln\frac{N}{\delta})) = \frac{\delta^{1.125}}{N^{1.125}}\end{aligned}$$

Hence $\Pr[b_{r+1} \geq \min(0.3N, b_r\ln\frac{N}{\delta})] \geq \Pr[y \leq 8b_r\ln\frac{N}{\delta}] \geq 1-\frac{\delta^{1.125}}{N^{1.125}}$. ☐

VII. Two-stage Sampling

Conceptual design. Consider any given honest node u. For each node v, define $\texttt{frac}_u[v] = \frac{|\{w \mid w \in \texttt{initview}_u \text{ and } v \in \texttt{view}_w\}|}{|\texttt{initview}_u|}$ — namely, $\texttt{frac}_u[v]$ is the fraction of the nodes in $\texttt{initview}_u$ that have v in their current views. Our TwoStageSample() subroutine enables u to estimate $\texttt{frac}_u[v]$ for all node v. We only obtain an estimate since otherwise u will need to retrieve $O(N)$ views where each view is of $O(N)$ size, which would result in $\Omega(N^2)$ communication complexity. Later we will prove that obtaining a (good enough) estimate is nevertheless still sufficient for everything to work.

The simplest way to estimate $\texttt{frac}_u[v]$ is perhaps for u to choose t random nodes from $\texttt{initview}_u$. For each w chosen, node u will then *pull* from w, by requesting w to send $\texttt{view}_w$ (which is considered one sample) to u. Node u can then estimate $\texttt{frac}_u[v]$ (for all v), based on the $\texttt{view}_w$'s received. While such a way of estimating does work, all the malicious nodes may pull from w and cause w to send $\Omega(N^2)$

Algorithm 4 TwoStageSample()
input: f, δ, $\texttt{offset}$, u, $\texttt{initview}_u$, $\texttt{view}_u$;
output: $\texttt{score}_u[\cdot]$;

1: generate fresh $\texttt{nonce}_u$ and send $\langle u, \mathsf{hash}(\texttt{nonce}_u)\rangle$ to all nodes in $\texttt{initview}_u$;
2: spend $\texttt{offset}$ rounds receiving messages in the form of $\langle w, \mathsf{hash}(\texttt{nonce}_w)\rangle$;
3: send $\langle u, \texttt{nonce}_u\rangle$ to all nodes in $\texttt{initview}_u$;
4: spend $\texttt{offset}$ rounds receiving messages in the form of $\langle w, \texttt{nonce}_w\rangle$;
5: **validate** each $\texttt{nonce}_w$ received **if** $\texttt{nonce}_w$ matches the previously received hash;
6: **for each** validated $\texttt{nonce}_w$, send $\langle u, \texttt{view}_u\rangle$ to w **if** $\frac{\mathsf{hash}(u,\texttt{nonce}_u,w,\texttt{nonce}_w)}{\texttt{max\_hash}} \le \frac{1+f}{|\texttt{initview}_u|}(\frac{30}{1-3f})^2 \ln \frac{3(1+f)|\texttt{initview}_u|}{\delta}$ and $|\texttt{view}_u| \le (1+f)|\texttt{initview}_u|$;
7: spend $\texttt{offset}$ round receiving messages in the form of $\langle w, \texttt{view}_w\rangle$;
8: **validate** each $\texttt{view}_w$ received **if** $\texttt{nonce}_w$ was validated and **if** $\frac{\mathsf{hash}(w,\texttt{nonce}_w,u,\texttt{nonce}_u)}{\texttt{max\_hash}} \le \frac{1}{|\texttt{initview}_u|}(\frac{30}{1-3f})^2 \ln \frac{3|\texttt{initview}_u|}{\delta}$;
9: **for each** v **do**
10: $\quad\texttt{votes}_u[v] = |\{w \mid v \in \texttt{view}_w \text{ and } \texttt{view}_w \text{ was validated}\}|$;
11: $\quad\texttt{score}_u[v] = \texttt{votes}_u[v]/((\frac{30}{1-3f})^2 \ln \frac{3|\texttt{initview}_u|}{\delta})$;
12: **return** $\texttt{score}_u[\cdot]$;

bits, breaking the desirable guarantee on communication complexity. Instead of u *pulling* from w, an alternative approach is for w to choose t random nodes from $\texttt{initview}_w$. For each u chosen, w will *push* to u, by sending $\texttt{view}_w$ to u (without u requesting for it). This gives w control over the communication complexity incurred. But now a malicious node w will aggressively push $\texttt{view}_w$ to all nodes, causing $\texttt{view}_w$ to be over-represented.

Our TwoStageSample() overcomes the above problem via two stages. The first stage is the same as the push-based design above, where w pushes $\texttt{view}_w$ to t other nodes. In the second stage, each receiving node u will decide whether to validate and use the received $\texttt{view}_w$, and use it for the estimation later. Node u validates $\texttt{view}_w$ if it feels that w should indeed have pushed $\texttt{view}_w$ to u. Two things are still needed to enable such design. First, w and u need to determine, in a consistent way, whether $\texttt{view}_w$ should be a sampled (i.e., pushed) to u. We achieve this by letting w and u generate a shared random number. Second, the probability of $\texttt{view}_w$ being a sample for u depends on N. Neither w nor u knows N, and they can only use $|\texttt{initview}_u|$ and $|\texttt{initview}_w|$ in place of N. But $|\texttt{initview}_u|$ and $|\texttt{initview}_w|$ may be different. To resolve this issue, w will need to slightly over-push, by using a somewhat larger probability.

Details of the design. Algorithm 4 gives the pseudo-code for TwoStageSample(). For Line 1 to Line 5, each node u sends a fresh nonce $\texttt{nonce}_u$ to all other nodes. To prevent a malicious node from carefully constructing its nonce after seeing other nodes' nonces first, the protocol requires each node to publish a commitment of its nonce first. Next, any given ordered pair $w \to u$ corresponds to two nonces, $\texttt{nonce}_w$ and $\texttt{nonce}_u$. Node w will push to u iff i) w knows both nonces, and ii) $\frac{u,\mathsf{hash}(\texttt{nonce}_u,w,\texttt{nonce}_w)}{\texttt{max\_hash}} \le \frac{1+f}{|\texttt{initview}_u|}(\frac{30}{1-3f})^2 \ln \frac{3(1+f)|\texttt{initview}_u|}{\delta}$. Similarly, node v validates w's push iff i) u knows both nonces, and $\frac{\mathsf{hash}(\texttt{nonce}_w,w,\texttt{nonce}_u,u)}{\texttt{max\_hash}} \le \frac{1}{|\texttt{initview}_u|}(\frac{30}{1-3f})^2 \ln \frac{3|\texttt{initview}_u|}{\delta}$. The two conditions on w and u controls how many samples are taken. Furthermore, they are designed in such a way that w always tries to over-push: Namely, the condition on w is always no harder to satisfy that the condition on u.

Provable guarantees. The following theorem summarizes the complexity of TwoStageSampling():

Theorem 4. *Assume $f < \frac{1}{3}$. If all honest nodes start executing* TwoStageSampling() *within* $\texttt{offset}$ *rounds of each other, then each honest node returns after exactly* $3 \cdot \texttt{offset}$ *rounds, while sending on expectation* $\Theta(N \ln \frac{N}{\delta})$ *bits per round.*

Proof. The time complexity is obvious. For the communication complexity, note that for sending the nonce and the hashes of the nonce, it takes $O(N)$ bits per round. For sending the samples at Line 6, each sample (i.e., $\texttt{view}_u$) has size of $O(N)$ bits if they are sent. On expectation, each honest node sends $|\texttt{initview}_u| \cdot \frac{1+f}{|\texttt{initview}_u|}(\frac{30}{1-3f})^2 \ln \frac{3(1+f)|\texttt{initview}_u|}{\delta} = O(\ln \frac{N}{\delta})$ samples. Hence the total expected number of bits is $O(N \ln \frac{N}{\delta})$. □

The next theorem proves that for any pair $w \to u$, u controls the probability of $\texttt{view}_w$ being sampled. In particular, malicious nodes cannot force a higher probability on u.

Theorem 5. *Assume $f < \frac{1}{3}$. Consider any given node w and any given honest node u. Define $p_u = \frac{1}{|\texttt{initview}_u|}(\frac{30}{1-3f})^2 \ln \frac{3|\texttt{initview}_u|}{\delta}$. Define $p(w \to u)$ to be the probability that $\texttt{view}_w$ is received and then validated by u at Line 8. If all honest nodes start* TwoStageSampling() *within* $\texttt{offset}$ *rounds of each other, then $p(w \to u) = p_u$ for honest w and $p(w \to u) \le p_u$ for malicious w.*

Proof. First, a nonce sent by an honest node will always be validated by all other honest nodes. Let event $\mathcal{E}_1$ denote $\frac{\mathsf{hash}(\texttt{nonce}_w,w,\texttt{nonce}_u,u)}{\texttt{max\_hash}} \le \frac{1+f}{|\texttt{initview}_w|}(\frac{30}{1-3f})^2 \ln \frac{3(1+f)|\texttt{initview}_w|}{\delta}$, which corresponds to Line 6 in w's invocation of the subroutine. Let event $\mathcal{E}_2$ denote $\frac{\mathsf{hash}(\texttt{nonce}_w,w,\texttt{nonce}_u,u)}{\texttt{max\_hash}} \le \frac{1}{|\texttt{initview}_u|}(\frac{30}{1-3f})^2 \ln \frac{3|\texttt{initview}_u|}{\delta}$, which corresponds to Line 8 in u's invocation of the subroutine. Since $\frac{1}{|\texttt{initview}_u|} \le \frac{1+f}{|\texttt{initview}_w|}$ and $\frac{3|\texttt{initview}_u|}{\delta} \le \frac{3(1+f)|\texttt{initview}_w|}{\delta}$ always hold, we know that if $\mathcal{E}_2$ happens, $\mathcal{E}_1$ must happen.

If w is honest, then u will receive and validate $\texttt{view}_w$ if and only if $\mathcal{E}_2$ happens. We thus have $p(w,u) = \Pr[\mathcal{E}_2] = \frac{1}{|\texttt{initview}_u|}(\frac{30}{1-3f})^2 \ln \frac{3|\texttt{initview}_u|}{\delta}$. If w is malicious, for u to receive and validate $\texttt{view}_w$, $\mathcal{E}_2$ is a necessary (but not sufficient) condition. Hence $p(w,u) \le \Pr[\mathcal{E}_2] = \frac{1}{|\texttt{initview}_u|}(\frac{30}{1-3f})^2 \ln \frac{3|\texttt{initview}_u|}{\delta}$. □

The next theorem proves that the estimates ($\texttt{score}_u[\cdot]$) returned by TwoStageSampling() satisfy some critical properties.

Theorem 6. *Assume $f < \frac{1}{3}$. For any honest or malicious node v, define $\texttt{count}[v] = |\{u | v \in \texttt{initview}_u$ and u is honest$\}|$. If for all honest nodes u, $|\texttt{view}_u| \le (1+f)N$ and they all start executing* $\mathsf{TwoStageSampling}()$ *within* $\texttt{offset}$ *rounds of each other, then with probability at least $1 - \frac{16\delta^3}{81}$, none of the following (bad) events will happen:*

1) *There exists some node v and some honest node u such that $\texttt{count}[v] = N$ and $\texttt{score}_u[v] \le 0.75$.*
2) *There exists some node v and some honest node u such that $\texttt{count}[v] = 0$ and $\texttt{score}_u[v] \ge 0.25$.*
3) *There exists some node v, some honest nodes u and w, such that $\texttt{score}_u[v] \ge 0.5$ and $\texttt{score}_w[v] \le 0.25$.*
4) *There exists some node v, some honest nodes u and w, such that $\texttt{score}_u[v] \le 0.5$ and $\texttt{score}_w[v] \ge 0.75$.*

Proof. Define $s_u = (\frac{30}{1-3f})^2 \ln \frac{3|\texttt{initview}_u|}{\delta}$ and $s_w = (\frac{30}{1-3f})^2 \ln \frac{3|\texttt{initview}_w|}{\delta}$. Let $a_u \cdot N$ and $a_w \cdot N$ be the number of malicious nodes in $\texttt{initview}_u$ and $\texttt{initview}_w$, respectively. Here a_u and a_w are both (unknown) values between 0 and f. Since for all honest nodes u, $|\texttt{view}_u| \le (1+f)N \le (1+f)|\texttt{initview}_u|$, the second condition of Line 6 will always be satisfied.

1) Consider any given node v where $\texttt{count}[v] = N$ and any given honest node u. By Theorem 5, each of the N honest nodes in $\texttt{initview}_u$ will be sampled by u with probability $\frac{s_u}{|\texttt{initview}_u|}$, while each of the $a_u N$ malicious nodes will be sampled by u with probability at most $\frac{s_u}{|\texttt{initview}_u|}$. We thus have $E[\texttt{votes}_u[v]] \ge \frac{s_u}{|\texttt{initview}_u|} \cdot \texttt{count}[v] = \frac{s_u}{|\texttt{initview}_u|} N \ge \frac{1}{1+f} \cdot s_u$. Hence:
$$\begin{aligned} & \Pr[\texttt{score}_u[v] \le 0.75] \\ \le\ & \Pr[\texttt{score}_u[v] \cdot s_u \le (1 - \frac{1-3f}{4}) \cdot (\frac{1}{1+f} \cdot s_u)] \\ =\ & \Pr[\texttt{votes}_u[v] \le (1 - \frac{1-3f}{4}) \cdot (\frac{1}{1+f} \cdot s_u)] \\ \le\ & \exp(-(\frac{1-3f}{4})^2 \cdot (\frac{1}{1+f} \cdot s_u) \cdot \frac{1}{2}) \\ <\ & \exp(-3 \ln \frac{3N}{\delta}) = \frac{\delta^3}{27N^3} \end{aligned}$$
One the steps above invoked a Chernoff bound on $\texttt{votes}_u[v]$, which is the sum of independent Poisson trials. Finally, there are at most $(1+f)N^2 \le \frac{4}{3}N^2$ combinations of different v and u. Taking a union bound shows that the first bad event in the theorem happens with probability at most $\frac{4\delta^3}{81}$.
2) Consider any given node v where $\texttt{count}[v] = 0$ and any given honest node u. By Theorem 5, we have $E[\texttt{votes}_u[v]] \le \frac{s_u}{|\texttt{initview}_u|} \cdot \texttt{count}[v] + \frac{s_u}{|\texttt{initview}_u|} \cdot a_u N = \frac{s_u}{|\texttt{initview}_u|} \cdot \frac{a_u}{1+a_u} \cdot |\texttt{initview}_u| \le \frac{f}{1+f} \cdot s_u$. Hence:
$$\begin{aligned} & \Pr[\texttt{score}_u[v] \ge 0.25] \\ \le\ & \Pr[\texttt{score}_u[v] \cdot s_u \ge (1 + \frac{1-3f}{4f}) \cdot (\frac{f}{1+f} \cdot s_u)] \\ =\ & \Pr[\texttt{votes}_u[v] \ge (1 + \frac{1-3f}{4f}) \cdot (\frac{f}{1+f} \cdot s_u)] \\ \le\ & \exp(-(\frac{1-3f}{4f})^2 \cdot (\frac{f}{1+f} \cdot s_u) \cdot \frac{1}{3}) \\ <\ & \exp(-3 \ln \frac{3N}{\delta}) = \frac{\delta^3}{27N^3} \end{aligned}$$
One the steps above invoked a Chernoff bound on $\texttt{votes}_u[v]$. Finally, there are at most $(1+f)N^2 \le \frac{4}{3}N^2$ combinations of different v and u. Taking a union bound shows that the second bad event in the theorem happens with probability at most $\frac{4\delta^3}{81}$.
3) Consider any given node v and any given honest nodes u and w. We consider two separate regions for $\texttt{count}[v]$. First, for $\texttt{count}[v] \le \frac{3N - fN}{8}$, by Theorem 5, we have $E[\texttt{votes}_u[v]] \le \frac{s_u}{|\texttt{initview}_u|} \cdot \texttt{count}[v] + \frac{s_u}{|\texttt{initview}_u|} \cdot a_u N \le (\frac{3-f}{8} + a_u) \frac{s_u N}{|\texttt{initview}_u|} = \frac{\frac{3-f}{8} + a_u}{1+a_u} \cdot s_u \le \frac{3+7f}{8+8f} \cdot s_u$. Hence:
$$\begin{aligned} & \Pr[\texttt{score}_u[v] \ge 0.5] \\ \le\ & \Pr[\texttt{score}_u[v] \cdot s_u \ge (1 + \frac{1-3f}{3+7f}) \cdot (\frac{3+7f}{8+8f} \cdot s_u)] \\ =\ & \Pr[\texttt{votes}_u[v] \ge (1 + \frac{1-3f}{3+7f}) \cdot (\frac{3+7f}{8+8f} \cdot s_u)] \\ \le\ & \exp(-(\frac{1-3f}{3+7f})^2 \cdot (\frac{3+7f}{8+8f} \cdot s_u) \cdot \frac{1}{3}) \\ <\ & \exp(-3 \ln \frac{3N}{\delta}) = \frac{\delta^3}{27N^3} \end{aligned}$$
One the steps above invoked a Chernoff bound on $\texttt{count}_u[v]$. The second case is for $\texttt{count}[v] > \frac{3N - fN}{8}$. By Theorem 5, we have $E[\texttt{votes}_w[v]] \ge \frac{s_w}{|\texttt{initview}_w|} \cdot \texttt{count}[v] > \frac{s_w}{|\texttt{initview}_w|} \frac{3N - fN}{8} \ge \frac{3-f}{8+8f} \cdot s_u$. Hence:
$$\begin{aligned} & \Pr[\texttt{score}_w[v] \le 0.25] \\ \le\ & \Pr[\texttt{score}_w[v] \cdot s_w \le (1 - \frac{1-3f}{3-f}) \cdot (\frac{3-f}{8+8f} \cdot s_w)] \\ =\ & \Pr[\texttt{votes}_w[v] \le (1 - \frac{1-3f}{3-f}) \cdot (\frac{3-f}{8+8f} \cdot s_w)] \\ \le\ & \exp(-(\frac{1-3f}{3-f})^2 \cdot (\frac{3-f}{8+8f} \cdot s_w) \cdot \frac{1}{2}) \\ <\ & \exp(-3 \ln \frac{3N}{\delta}) = \frac{\delta^3}{27N^3} \end{aligned}$$
One the steps above invoked a Chernoff bound on $\texttt{votes}_w[v]$. Finally, there are at most $(1+f)N^3 \le \frac{4}{3}N^3$ combinations of different v, u, and w. Taking a union bound shows that the third bad event in the theorem happens with probability at most $\frac{4\delta^3}{81}$.
4) Consider any given node v and any given honest nodes u and w. We consider two separate regions for $\texttt{count}[v]$. First, for $\texttt{count}[v] \le \frac{5N + fN}{8}$, by Theorem 5, we have

$E[\texttt{votes}_w[v]] \le \frac{s_w}{|\texttt{initview}_w|} \cdot \texttt{count}[v] + \frac{s_w}{|\texttt{initview}_w|} \cdot a_w N \le (\frac{5+f}{8} + a_w)\frac{s_w N}{|\texttt{initview}_w|} = \frac{\frac{5+f}{8}+a_w}{1+a_w} \cdot s_w \le \frac{5+9f}{8+8f} \cdot s_w$. Hence:

$$\begin{aligned}
& \Pr[\texttt{score}_w[v] \ge 0.75] \\
\le\ & \Pr[\texttt{score}_w[v] \cdot s_w \ge (1 + \frac{1-3f}{5+9f}) \cdot (\frac{5+9f}{8+8f} \cdot s_w)] \\
=\ & \Pr[\texttt{votes}_w[v] \ge (1 + \frac{1-3f}{5+9f}) \cdot (\frac{5+9f}{8+8f} \cdot s_w)] \\
\le\ & \exp(-(\frac{1-3f}{5+9f})^2 \cdot (\frac{5+9f}{8+8f} \cdot s_w) \cdot \frac{1}{3}) \\
<\ & \exp(-3 \ln \frac{3N}{\delta}) = \frac{\delta^3}{27N^3}
\end{aligned}$$

One the steps above invoked a Chernoff bound on $\texttt{votes}_w[v]$. The second case is for $\texttt{count}[v] > \frac{5N+fN}{8}$. By Theorem 5, we have $E[\texttt{votes}_u[v]] \ge \frac{s_u}{|\texttt{initview}_u|} \cdot \texttt{count}[v] > \frac{5+f}{8}\frac{s_u N}{|\texttt{initview}_u|} = \frac{5+f}{8+8a_u}s_u \ge \frac{5+f}{8+8f}s_u$. Hence:

$$\begin{aligned}
& \Pr[\texttt{score}_u[v] \le 0.5] \\
\le\ & \Pr[\texttt{score}_u[v] \cdot s \le (1 - \frac{1-3f}{5+f}) \cdot (\frac{5+f}{8+8f} \cdot s_u)] \\
=\ & \Pr[\texttt{votes}_u[v] \cdot s_u \le (1 - \frac{1-3f}{5+f}) \cdot (\frac{5+f}{8+8f} \cdot s_u)] \\
\le\ & \exp(-(\frac{1-3f}{5+f})^2 \cdot (\frac{5+f}{8+8f} \cdot s_u) \cdot \frac{1}{2}) \\
<\ & \exp(-3 \ln \frac{3N}{\delta}) = \frac{\delta^3}{27N^3}
\end{aligned}$$

One the steps above invoked a Chernoff bound on $\texttt{votes}_u[v]$. Finally, there are at most $(1+f)N^3 \le \frac{4}{3}N^3$ combinations of different v, u, and w. Taking a union bound across all these combinations shows that the fourth bad event in the theorem happens with probability at most $\frac{4\delta^3}{81}$.

Finally, combining the probabilities of the four bad events via a union bound directly leads to the theorem. ☐

VIII. Put Everything Together

We are now ready to put everything together and prove the formal guarantees of RVR (Algorithm 1). All line numbers in this section refers to Algorithm 1. We first provide some additional comments on Algorithm 1. On the `leader`, the $\texttt{proposal}_{\texttt{leader}}$ contains a node v if the score for v is at least 0.5 (at Line 6). A node u will overrule the leader's proposal with respect to v, if u's score for v is either overwhelming (at Line 7) or underwhelming (at Line 8). Line 11 sets $\texttt{offset} = 2$ since `offset` must be 2 after the invocation of CoordinatedGossip. The following theorem proves the complexity of our protocol:

Theorem 7. *Assume that $m \ge 1$, $N \ge 1000$, $\delta \le 0.1$, and $f < \frac{1}{3}$. If all honest nodes start executing* RVR *within* `offset` *rounds of each other, then each honest node will return within $\Theta(\frac{\ln N}{\ln \ln N} \ln \frac{1}{\delta})$ rounds, while sending on expectation $\Theta(N \ln \frac{N}{\delta})$ bits per round.*

Proof. The communication complexity of $O(N \ln \frac{N}{\delta})$ bits per round follows directly from Theorem 1, 2, and 4. For the total number of rounds, by Theorem 1, 2, and 4, each iteration in RVR takes at most $8 \cdot \texttt{offset} + 3 \cdot \texttt{offset} + \frac{3\ln((1+f)N)}{2\ln\ln((1+f)N)} + 2 \cdot \texttt{offset} + 1 = \frac{3\ln((1+f)N)}{2\ln\ln((1+f)N)} + 1 + 13 \cdot \texttt{offset}$ rounds. Note that starting from the second iteration in RandomizedViewMerging, `offset` will always be 2. Hence the total number of rounds is at most $(\frac{3\ln((1+f)N)}{2\ln\ln((1+f)N)} \ln \frac{2}{\delta} + 28) \cdot (6 \ln \frac{2}{\delta}) + 13 \cdot (\texttt{offset} - 1) = O(\frac{\ln N}{\ln \ln N} \ln \frac{1}{\delta})$. ☐

The next theorem proves that with probability of at least $1 - \delta$, our protocol achieves the intended goal.

Theorem 8. *Assume that $m \ge 1$, $N \ge 1000$, $\delta \le 0.1$, and $f < \frac{1}{3}$. If all honest nodes start executing* RVR *within* `offset` *rounds of each other, then with probability at least $1-\delta$, both of the following hold:*

1) *For all honest nodes u, $\texttt{final\_view}_u$ is the same.*
2) *For all honest nodes u, $\texttt{final\_view}_u$ contains all the N honest nodes and at most fN malicious nodes.*

Proof. RVR invokes TwoStageSample exactly $6 \ln \frac{2}{\delta}$ times. Let $\mathcal{E}_1^r$ be the event that none of the four bad events in Theorem 6 happens immediately after Line 4 in the r-th iteration of RVR. We claim that $\Pr[\cap_{1 \le r \le 6 \ln \frac{2}{\delta}} \mathcal{E}_1^r] \ge 1 - 0.04\delta$. In the first iteration immediately before Line 4 we have $|\texttt{view}_u| = |\texttt{initview}_u| \le (1+f)N$ for all honest nodes u, which satisfies the condition required to invoke Theorem 6. Conditioned on $\mathcal{E}_1^r$, immediately before Line 9 in the r-th iteration of RVR, $|\texttt{proposal}_u| \le (1+f)N$ for any honest node u since $v \in \texttt{proposal}_u$ implies that $\texttt{count}[v] > 0$ and thus is must be included in $\texttt{initview}_w$ for some honest node w. Recall the definition of `union_honest_initview` from Section III. Thus $|\texttt{proposal}_u| \le |\texttt{union\_honest\_initview}| \le (1+f)N$. Conditioned on $\mathcal{E}_1^r$, it can be similarly shown that immediately before Line 10 in the r-th iteration of RVR, $|\texttt{view}_u| \le (1+f)N$. The claim then follows trivially via induction on r by repeatedly invoking Theorem 6. Let $\mathcal{E}_1 = \cap_{1 \le r \le 6 \ln \frac{2}{\delta}} \mathcal{E}_1^r$.

Next, we say that an iteration in RVR is *good* if during that iteration, i) ProbLeaderElect returns the same `leader` on all honest nodes and the `leader` returned is an honest node, and ii) CoordinatedGossip returns $\texttt{proposal}_{\texttt{leader}}$ on all honest nodes. By Theorem 1 and 3 and a union bound, an iteration is good with probability at least $0.16 - \frac{1}{4000} = 0.15975$. Hence with probability at least $1 - (1 - 0.15975)^{6 \ln \frac{2}{\delta}} > 1 - \delta^{1.03} > 1 - 0.94\delta$, there exists some good iteration. Let such event be $\mathcal{E}_2$. A union bound immediately tells us that with probability at least $1 - \delta$, both $\mathcal{E}_1$ and $\mathcal{E}_2$ occur. It thus suffices to prove the two claims in the theorem while condition upon that $\mathcal{E}_1$ and $\mathcal{E}_2$ occur. The following proves them one by one.

First claim. We first prove that at the end of the good iteration, $\texttt{view}_u$ is the same on all honest nodes u. The good iteration must have a unique and honest `leader`. Consider any node v. We will show that for all honest nodes u, at the end of that iteration, $v \in \texttt{view}_u$ iff $v \in \texttt{proposal}_{\texttt{leader}}$. To prove this, note that if $v \in \texttt{proposal}_{\texttt{leader}}$, then $\texttt{score}_{\texttt{leader}}[v] \ge 0.5$. By event $\mathcal{E}_1$, $\texttt{score}_u[v] \not\leq 0.25$ and hence $v \notin \texttt{underwhelm}_u$. By Line 10, v will be in $\texttt{view}_u$. The case for $v \notin \texttt{proposal}_{\texttt{leader}}$ is similar. Now because $\texttt{proposal}_{\texttt{leader}}$ is the same on all honest nodes u, $\texttt{view}_u$ is the same on all honest nodes u.

Next we prove that in any iteration after the good iteration, on all honest nodes u, $\texttt{view}_u$ will no longer change. Immediately after the good iteration, for all nodes v, we have either $\texttt{count}[v] = N$ or $\texttt{count}[v] = 0$. If $\texttt{count}[v] = N$, then event $\mathcal{E}_1$ ensures that $v \in \texttt{over}_u$ in all future iterations. If $\texttt{count}[v] = 0$, then $\mathcal{E}_1$ ensures that $v \in \texttt{under}_u$ in all future iterations. Together with the condition at Line 10, we know that $\texttt{view}_u$ will no longer change in future iterations.

Finally, since at the end of the good iteration, $\texttt{view}_u$ is the same on all honest nodes u, and since $\texttt{view}_u$ does not change in all iterations after the good iteration, we know that $\texttt{final\_view}_u$ is the same for all honest nodes u.

Second claim. Consider any given honest node v. We know that $\texttt{count}[v] = N$ at the beginning of the first iteration. By the same argument as earlier, v will continue to be in $\texttt{view}_u$ (and thus $v \in \texttt{final\_view}_u$) for all honest node u in all iterations. Next, union_honest_views contains at most fN malicious nodes. For all malicious node $v \notin$ union_honest_views, we know that $\texttt{count}[v] = 0$ at the beginning of the first iteration. By the same argument as earlier, we know that $v \notin \texttt{view}_u$ (and thus $v \notin \texttt{final\_view}_u$) for all honest nodes u in all iterations. □

IX. Conclusions

This paper proposes a novel randomized RVR protocol for solving the view divergence problem. Compared with the state-of-the-art protocols [17], [18] with time complexity of $\Theta(N)$ rounds, our protocol only takes $\Theta(\frac{\ln N}{\ln \ln N} \ln \frac{1}{\delta})$ rounds to terminate. Our protocol also incur similar amount of communication complexity per round as the previous protocols. Currently our protocol can tolerate all constant $f < \frac{1}{3}$, and it is our future work to investigate how to generalize to $f \geq \frac{1}{3}$.

References

[1] J. Douceur, “The Sybil attack,” in *IPTPS*, 2002.

[2] C. Dwork and M. Naor, “Pricing via processing or combatting junk mail,” in *CRYPTO*, 1992.

[3] C. Dwork, M. Naor, and H. Wee, “Pebbling and proofs of work,” in *CRYPTO*, 2005.

[4] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” See http://bitcoin.org/bitcoin.pdf, 2008.

[5] J. Aspnes, C. Jackson, and A. Krishnamurthy, “Exposing computationally-challenged Byzantine impostors,” Yale University Department of Computer Science, Tech. Rep. YALEU/DCS/TR-1332, Jul. 2005.

[6] P. Berman and J. Garay, “Cloture votes: $n/4$-resilient distributed consensus in $t+1$ rounds,” *Mathematical Systems Theory*, vol. 26, no. 1, pp. 3–19, Jan. 1993.

[7] M. Castro and B. Liskov, “Practical byzantine fault tolerance and proactive recovery,” *ACM Transactions on Computer Systems*, 2002.

[8] D. Dolev and H. Strong, “Authenticated algorithms for byzantine agreement,” *SIAM Journal on Computing*, vol. 12, no. 4, pp. 656–666, 1983.

[9] L. Lamport, R. Shostak, and M. Pease, “The byzantine generals problem,” *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, p. 382C401, 1982.

[10] M. Pease, R. Shostak, and L. Lamport, “Reaching agreement in the presence of faults,” *J. ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980.

[11] M. Raynal, “Consensus in synchronous systems: A concise guided tour,” in *Proceedings of the Pacific Rim International Symposium on Dependable Computing*, 2002.

[12] M. Ben-Or, S. Goldwasser, and A. Wigderson, “Completeness theorems for non-cryptographic fault-tolerant distributed computation,” in *STOC*, 1988, pp. 1–10.

[13] O. Goldreich, S. Micali, and A. Wigderson, “How to play any mental game,” in *STOC*, 1987.

[14] A. C. Yao, “Protocols for secure computations,” in *FOCS*, 1982, pp. 160–164.

[15] B. Awerbuch and C. Scheideler, “Towards a scalable and robust DHT,” *Theory of Computing Systems*, 2009.

[16] R. Guerraoui, F. Huc, and A. Kermarrec, “Highly dynamic distributed computing with byzantine failures,” in *PODC*, 2013.

[17] M. Andrychowicz and S. Dziembowski, “PoW-Based Distributed Cryptography with No Trusted Setup,” in *International Cryptology Conference (CRYPTO)*, Aug. 2015.

[18] J. Katz, A. Miller, and E. Shi, “Pseudonymous secure computation from time-lock puzzles,” Cryptology ePrint Archive, Tech. Rep. 2014/857, 2014.

[19] N. Borisov, “Computational puzzles as sybil defenses,” in *IEEE International Conference on Peer-to-Peer Computing*, 2006.

[20] F. Li, P. Mittal, M. Caesar, and N. Borisov, “Sybilcontrol: Practical sybil defense with computational puzzles,” in *Workshop Scalable Trusted Computing*, 2012.

[21] H. Rowaihy, W. Enck, P. Mcdaniel, and T. La Porta, “Limiting sybil attacks in structured peer-to-peer networks,” in *INFOCOM*, 2007.

[22] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, “Eclipse attacks on bitcoins peer-to-peer network,” in *USENIX Security Symposium*, 2015.

[23] R. Pass and E. Shi, “Feasibilities and infeasibilities for achieving responsiveness in permissionless consensus,” in *DISC*, Oct. 2017.

[24] M. Bellare and P. Rogaway, “Random oracles are practical: A paradigm for designing efficient protocols,” in *ACM Conference on Computer and Communications Security*, Nov. 1993.

[25] J. Burns and N. Lynch, “The byzantine firing squad problem,” *Advances in Computing Research: Parallel and Distributed Computing*, vol. 4, pp. 147–161, 1987.