Fast Asynchronous Byzantine Agreement and Leader Election with Full Information

Bruce Kapron
University of Victoria
and
David Kempe
University of Southern California
and
Valerie King
University of Victoria
and
Jared Saia
University of New Mexico
and
Vishal Sanwalani
University of New Mexico

We resolve two long-standing open problems in distributed computation by describing polylogarithmic protocols for Byzantine agreement and leader election in the asynchronous full information model with a non-adaptive malicious adversary. All past protocols for asynchronous Byzantine Agreement had been exponential, and *no* protocol for asynchronous leader election had been known. Our protocols tolerate up to $(\frac{1}{3} - \epsilon) \cdot n$ faulty processors, for any positive constant ϵ. They are Monte Carlo, succeeding with probability $1 - o(1)$ for Byzantine agreement, and constant probability for leader election. A key technical contribution of our paper is a new approach for emulating Feige's lightest bin protocol, even with adversarial message scheduling.

Categories and Subject Descriptors: F.1.2 [**Theory of Computation**]: Computation by Abstract Devices—*Parallelism and Concurrency*; F.2.2 [**Theory of Computation**]: Analysis of

Bruce Kapron Contact Info: Dept. of Computer Science, University of Victoria; email: bmkapron@uvic.ca.
David Kempe Contact Info: Dept. of Computer Science, University of Southern California, Los Angeles, CA 90089-0781; e-mail: dkempe@usc.edu. Work supported in part by NSF CAREER Award 0545855.
Valerie King Contact Info: Dept. of Computer Science, University of Victoria, P.O. Box 3055, Victoria, BC, Canada V8W 3P6; email: val@uvic.ca. This research was partially supported by NSERC and Microsoft Research SVC.
Jared Saia Contact Info: Dept. of Computer Science, University of New Mexico, Albuquerque, NM 87131-1386; email: saia@cs.unm.edu. This research was partially supported by NSF CAREER Award 0644058 and NSF CCR-0313160.
Vishal Sanwalani Contact Info: email: vsanwalani@gmail.com.

Algorithms and Problem Complexity—*Nonnumerical Algorithms and Problems*

General Terms: Theory, Algorithms, Reliability, Security

Additional Key Words and Phrases: Byzantine Agreement, distributed algorithms, asynchronous communication, probabilistic method, Monte Carlo Algorithms

1. INTRODUCTION

Two fundamental problems in distributed computing are Byzantine agreement and leader election. In both, up to a constant fraction of n processors are *bad* (or *faulty*), while the others are *good* (or *non-faulty*). Faulty processors can deviate from the protocol in arbitrary ways, and are thus modeled as controlled by an adversary. In the *Byzantine agreement* problem, each processor is initially given an input bit, and all good processors must come to an agreement on a bit which coincides with at least one of their input bits. In the *leader election* problem, all good processors must come to agreement on some good processor.

We study these problems in a very general model of computing, the asynchronous, full information message passing model. In the *full information* model, the adversary is computationally unbounded and has access to the content of all messages. In the *asynchronous* model, each communication can take an arbitrary and unknown amount of time, and there is no assumption of a joint clock as in the synchronous model. Communication is by passing a message from one processor to another. The advantages of the full information model are its simplicity and avoidance of complexity assumptions.

Asynchrony introduces fundamental difficulties into distributed protocol design; intuitively, protocols are unable to distinguish failed processors from delayed messages. Indeed, Fischer, Lynch and Patterson [Fischer et al. 1983] showed that with just a single faulty processor, no deterministic asynchronous Byzantine agreement is possible. Even with randomization, there has been no significant progress in the asynchronous full information model in the last 22 years since Ben-Or and Bracha [Ben-Or 1983; Bracha 1984] gave a randomized protocol for Byzantine agreement succeeding with probability 1 in expected exponential time. The *resilience* of Bracha's protocol, i.e., the number of faulty processors it can tolerate, is $t < n/3$.

Asynchronous Byzantine agreement in the full information model in expected time $o(n)$ is impossible with an adversary which can corrupt processors adaptively [Aspnes 1998; Attiya and Censor 2007]. Since our goal is to design polylogarithmic time protocols, we consider here a model where the adversary is *non-adaptive*, in that it must choose the corrupt processors at the outset of the protocol.

We give the first sub-exponential protocol for Byzantine agreement in the asynchronous full information model. In fact, our protocol takes worst-case time polylogarithmic (resp. quasi-polynomial) in n and succeeds with probability at least $1 - 1/\log^c n$ (resp. $1 - 1/n^c$) for any constant c, while tolerating a constant fraction of faulty processors. Specifically, we prove the following for Byzantine agreement:

THEOREM 1.1. *In the asynchronous full information model with a non-adaptive adversary, for any constants $\epsilon, c > 0$, there are Byzantine agreement protocols using $\tilde{O}(n^2)$ bits of communication, with*

—*running time* $2^{\Theta(\log^8 n)}$*, resilience* $\frac{1}{3} - \epsilon$*, and success probability* $1 - 1/n^c$*.*
—*running time polylogarithmic in* n*, resilience* $\frac{1}{3} - \epsilon$*, and success probability* $1 - 1/\log^c n$*.*

Our result substantially narrows the gap (to within $\log n$ factors) between synchronous and asynchronous distributed computing in the full information model with a non-adaptive adversary. However, we should point out that the synchronous protocols are (or can be easily made) Las Vegas in the sense that all processors eventually output the same bit.

The *leader election* problem is clearly impossible to solve against an adaptive adversary, and the success probability against a non-adaptive adversary cannot exceed $1 - t/n$, as bad processors can just behave like good processors. Tighter upper bounds on the success probability are given by Feige [Feige 1999]. As our second main result, we give the first asynchronous leader election protocol in the full information model with constant success probability against a constant fraction of bad processors. This immediately implies the first asynchronous coin-flipping protocol as well. Our leader election protocol, like the Byzantine agreement protocol, runs in polylogarithmic time. We show:

THEOREM 1.2. *There is a leader election protocol in the asynchronous full information model against a non-adaptive adversary that uses* $\tilde{O}(n^2)$ *bits of communication, and has running time polylogarithmic in* n*, resilience* $\frac{1}{3} - \epsilon$*, and success probability that is a positive constant.*

Technical Contribution: First, we present a novel table-based approach that enables a set of processors to collaboratively make a random choice, such that this random choice is essentially independent of the order in which the adversary schedules messages (Section 3.2). Second, we describe how to adapt the layered network from [King et al. 2006a; 2006b] to an asynchronous setting (Section 4). In [King et al. 2006a; 2006b], the layered network was used to reduce communication costs in a synchronous setting. Here, we adapt it to reduce the runtime of an algorithm in an asynchronous setting. This adaptation suggests that the layered network approach may be useful for reducing other resource costs for other distributed algorithms running in an adversarial setting.

A core technique for all of our protocols is *asynchronous universe reduction*. Universe reduction consists of reducing the number of processors under consideration from n to $k \ll n$ while ensuring that the fraction of good processors among the selected k is nearly the same as among the initial n (a formal definition is given in Section 2). Feige [Feige 1999] proposed a simple and elegant synchronous universe reduction technique in the broadcast model; we show how to extend this technique to the asynchronous domain.

Other Related Work: Leader election and global coin-tossing have been extensively studied in a synchronous full information model with a non-adaptive adversary and an *atomic broadcast* primitive, where each message sent (even by corrupt processors) is received identically by all processors [Ben-Or and Linial 1989; Ajtai and Linial 1993; Saks 1989; Ostrovsky et al. 1994; Russell and Zuckerman 1998]. These papers culminated in Feige's $O(\log^* n)$ protocol for leader election [Feige 1999], which we adapt and use here.

Without the atomic broadcast primitive, polylogarithmic round randomized protocols against a non-adaptive adversary in the synchronous full information model were developed independently by King, Saia, Sanwalani, and Vee [King et al. 2006a; 2006b] and by Ben-Or, Goldwasser, Pavlov and Vaikuntanathan [Ben-Or et al. 2006; Goldwasser et al. 2006]. [King et al. 2006a] presents protocols for both Byzantine agreement and leader election with resilience $\frac{1}{3+\epsilon}$ which are “scalable” in that each processor sends and processes a number of bits polylogarithmic in n. The protocols obtain “almost everywhere” agreement (agreement among a $1 - O(1/\log n)$ fraction of good processors) with probability $1 - 1/n^c$. With one additional round of sending one bit to every other processor, agreement can be obtained among all good processors, in a worst-case number of rounds which is polylogarithmic in n. [King et al. 2006b] showed that the almost everywhere agreement protocol could be implemented scalably on a sparse network in polylogarithmically many rounds. [Ben-Or et al. 2006] and [Goldwasser et al. 2006] give protocols with resilience $\frac{1}{4+\epsilon}$ and $\frac{1}{3+\epsilon}$, respectively, using $O(\log n)$ rounds in expectation.

Universe reduction protocols are found in several papers on leader election and coin tossing in the synchronous broadcast model. In particular, Gradwohl, Vadhan, and Zuckerman [Gradwohl et al. 2006] use a 2-stage process involving Feige’s leader election protocol, followed by the application of a sampler, to perform a selection with an adversarial majority. This two-stage protocol is similar to the subcommittee election and expansion procedure we present in Section 3.

Several papers for asynchronous Byzantine agreement have appeared since the 1980’s which do not assume a full information model, but instead assume either private channels between each pair of processors [Berman and Garay 1993; Canetti and Rabin 1993] or a computationally bounded adversary and cryptographic primitives [Toueg 1984], culminating in protocols with expected $O(1)$ round time with optimal resilience $\frac{n}{3}$ and $O(n^2)$ messages [Cachin et al. 2005; Nielsen 2002]. The problem of parallel broadcast addressed in Section 3.1 of this paper was posed as the *Interactive Consistency Problem* for a model with private channels by Pease, Shostak and Lamport [Pease et al. 1980] and solved in constant time by Ben-Or and El-Yaniv [Ben-Or and El-Yaniv 2003].

Recently, Attiya and Censor [Attiya and Censor 2008] improved a result of [Chor et al. 1989] to show that any randomized Byzantine agreement protocol has probability at least $\Omega(\frac{1}{\text{polylog}(n)})$ of not terminating (or failing to agree) within $O(\log \log n)$ (asynchronous) rounds, and probability at least $\Omega(\frac{1}{\text{poly}(n)})$ for not terminating within $O(\log n)$ (asynchronous) rounds. This lower bound holds in our model, against a non-adaptive adversary, and even against a much weaker adversary which is oblivious to the content of messages.

2. PRELIMINARIES AND OVERVIEW

We assume a fully connected network of n processors, whose IDs are common knowledge. Each processor has a private coin. Communication channels are authenticated, in the sense that whenever a processor sends a message to another, the identity of the sender is known to the recipient.

We assume an asynchronous full information model with a nonadaptive (sometimes called *static*) malicious adversary. That is, the adversary chooses the set of bad processors at the start of the protocol. Bad processors can engage in any kind of deviations from the protocol, including false messages, collusion, or crash failures.

The adversary can view each message as soon as it is sent, and determines the message's delay as well as the order in which messages are received. The running time of a protocol is described in terms of the maximum delay of any message Δ. Thus, a running time of $f(n)$ means that all good processors reach agreement by time $\Delta f(n)$. Alternatively, $f(n)$ is the maximum length of any chain of messages [Chor and Dwork 1989]. For notational convenience, we will sometimes describe the running time of a protocol by the number of *steps* a protocol takes or has taken. Thus, when we say a protocol has run for $q(n)$ steps, we mean it has run for at most time $\Delta q(n)$.

Our protocols succeed with a small probability of error. Errors may be either deadlock or finishing with disagreement among the processors.

We use the phrase *with high probability* (*w.h.p.*) to mean that an event happens with probability at least $1 - 1/n^c$ for every constant c and sufficiently large n. For readability, we treat $\log n$ as an integer throughout.

2.1 Overview

Our protocols are based on a novel asynchronous protocol for universe reduction. We define the (θ, k)-universe reduction problem for the asynchronous model as follows: Given n processors with an (unknown) subset of good processors G, each good processor p should output a subset of k processors S_p such that $\frac{|\bigcap_{p \in G} S_p \cap G|}{k} > \frac{|G|}{n} - \theta$. That is, the sets output by every good processor p contain a common subset of good processors that makes up nearly the same fraction of each set as the fraction of good processors in the universe.

To establish this result (and Theorems 1.1 and 1.2), we adapt the synchronous universe reduction protocol of King et al. [King et al. 2006a] to the asynchronous communication model. The processors are divided into committee mulitsets of polylogarithmic size; each processor is assigned to multiple committees. Each committee in parallel elects a small number of processors (called a *subcommittee*) from within itself. The process is repeated on the multiset of elected processors until the number of remaining processors has been sufficiently reduced. At that point, to solve Byzantine agreement, the remaining few processors run Bracha's [Bracha 1984] (exponential-time) randomized asynchronous Byzantine agreement protocol. To solve leader election, the universe reduction protocol is first applied to reduce the number of processors to a single small committee. Then, using a variant of the subcommittee election protocol, the committee is further repeatedly reduced until a constant number of processors remain. With a smaller (constant) probability, these processors select the same leader and verify this selection. For both problems, every good processor may associate a different subset of processors with each committee, but since the intersection of the subsets is large and contains mostly good processors, the good processors in the intersection come to a decision that all good processors then agree to.

Two main issues need to be addressed: (1) Each committee with enough good processors must be able to robustly and efficiently hold an election. (2) A sufficient number of committees must contain enough good processors that are known to all processors. The first issue is addressed in Section 3, where we adapt Feige's "Lightest Bin Protocol" for leader election in synchronous broadcast environments to our asynchronous point-to-point connection model.

The second issue is resolved in Section 4, using a network structure similar to that in [King et al. 2006a] and reasoning about the different "views" of the processors. Specifically, in order to assign processors to committees, we use a layered network of averaging samplers, bipartite graphs with random-like properties.

We discuss in Section 4 how to put the various building blocks together to obtain a quasi-polynomial time protocol for Byzantine agreement, and in Section 5 how to modify the protocol to achieve polylogarithmic running time. Section 6 describes the leader election protocol.

2.2 Samplers

Our protocols rely extensively on the use of averaging (or oblivious) samplers, families of bipartite graphs which define subsets of elements such that all but a small number contain at most a fraction of "bad" elements close to the fraction of bad elements of the entire set. Bipartite graphs with such random-like properties have been used extensively in the design of distributed protocols [Chor and Dwork 1989] and have alternatively been called expanders, dispersers and samplers. An exact correspondence between extractors and averaging samplers is given in [Zuckerman 1997].

DEFINITION 2.1. *Let $[r]$ denote the set of integers $\{1,\ldots,r\}$, and $[s]^d$ the multisets of size d consisting of elements of $[s]$. Let $H : [r] \to [s]^d$ be a function assigning multisets of size d to integers. We define the intersection of a multiset A and a set B to be the number of elements of A which are in B.*

H is a (θ,δ) sampler *if for every set $S \subset [s]$, at most a δ fraction of all inputs x have $\frac{|H(x)\cap S|}{d} > \frac{|S|}{s} + \theta$.*

The following two lemmas establish the existence of samplers with various parameter combinations. For the use of these samplers in our protocols, we assume either a nonuniform model in which each processor has a copy of the required samplers for a given input size, or else that each processor initializes by constructing the required samplers in exponential time. Alternatively, we could use versions of the efficient constructions given in [Gradwohl et al. 2006] at the expense of a polylogarithmic overhead in the overall running time of the protocol.

LEMMA 2.2 [GRADWOHL ET AL. 2006, LEMMA 4.7]. *For every $s, \theta, \delta > 0$ and $r \geq s/\delta$, there exists a constant c such that for all $d \geq c\log(1/\delta)/\theta^2$, there is a (θ,δ) sampler $H : [r] \to [s]^d$.*

LEMMA 2.3. *For every $r, s, d, \theta, \delta > 0$ such that $2\log_2(e) \cdot d\theta^2\delta - (1+\delta) > s/r$, there exists a (θ,δ) sampler $H : [r] \to [s]^d$.*

Proof. We prove the existence of this sampler by the probabilistic method. For each input x of $[r]$, form $H(x)$ by randomly selecting with replacement d elements of $[s]$. Fix any set $S \subseteq [s]$. Then, $\mathrm{E}\left[|H(x)\cap S|\right] = d|S|/s$. By standard Chernoff-

Hoeffding bounds (e.g., [McDiarmid 1998, Theorem 2(a)]),

$$\mathrm{Prob}[||H(x) \cap S| - \mathrm{E}\,[|H(x) \cap S|]\,| \geq \theta d] \;\leq\; 2e^{-2d\theta^2},$$

for any fixed $x \in R$ and S.

Thus, for a fixed set R of size δr, and a fixed subset S of $[s]$, the probability that $|H(x) \cap S|$ exceeds $(\frac{|S|}{s} + \theta) \cdot d$ for *all* inputs $x \in R$ is less than $2^{r\delta} e^{-2d\theta^2 r\delta}$. Taking a union bound over all choices of subsets S and R, the probability that this occurs for any subset of δr inputs and any subset S of $[s]$ is less than $2^{r+s+r\delta} e^{-2d\theta^2 r\delta}$. This is less than 1 when $s/r < 2\log_2(e)d\delta\theta^2 - (1+\delta)$. Hence, whenever the condition is satisfied, the desired sampler exists. ■

3. SUBCOMMITTEE ELECTION

We describe a protocol ELECT-SUBCOMMITTEE for asynchronously electing a subcommittee of processors from a committee of k processors in which at least $k-t$ are good. In Feige's protocol [Feige 1999], each processor randomly selects a "bin number" from $\{1, \ldots, b\}$ and broadcasts it to the other processors. The subcommittee then consists of all processors which chose the bin selected by the *smallest* number of processors. The intuition is that w.h.p., each bin will have a similar number of good processors, and the bad processors cannot gain a disproportionate fraction of a bin without increasing the bin's size to a point where it will not be selected any more.

In adapting this protocol to our model, we note that when t or fewer messages are delayed, the protocol cannot wait for them (otherwise, the adversary could deadlock the protocol by simply not sending any messages). Several difficulties have to be overcome:

(1) *Broadcast simulation:* Each processor needs to communicate its message (e.g., bin choice), but the adversary can prevent some good processors from being heard by any of the others by delaying their messages. Also, processors need to come to consensus about the choices of the bad processors.

(2) *Keeping the bins balanced:* If $t > k/b$, the adversary can delay messages from the good processors which choose a particular bin i and fill it with few enough bad processors to make it the lightest bin, gaining full control over the subcommittee.

(3) *Restoring the fraction of good processors:* Even if the message delays were independent of their content, only a constant fraction of the good processors will be heard from; good processors will thus be underrepresented in the subcommittee.

3.1 Broadcast simulation

A broadcast is a communication of a processor to all the other processors. If all processors accept the message that was sent by the broadcasting processor, then the broadcast is *successful*. We consider the problem of *parallel broadcast* in which all (except possibly bad) processors attempt to broadcast their respective messages at the same time. Assuming that at most $t < (\frac{1}{3} - \epsilon) \cdot k$ processors are bad, our protocol guarantees that when k processors broadcast in parallel, all processors will come to agreement on each message (which may differ from the original message sent), and the broadcasts of at least ϵk good processors will be successful. We assume that all messages are binary strings of length m.

Our protocol PARALLEL-BROADCAST bears a lot of similarity with the protocol SPREAD by Ben-Or and El-Yaniv [Ben-Or and El-Yaniv 2003], and has similar guarantees. Since [Ben-Or and El-Yaniv 2003] do not explicitly state the bit complexity of their protocols, and the correctness guarantees are stated somewhat differently, we present our protocol here for completeness.

The basis for broadcast simulation among the k processors is Bracha's exponential time asynchronous protocol for solving the Byzantine agreement problem [Bracha 1984]. Bracha's protocol is Las Vegas. Here, we treat it as Monte Carlo instead, and characterize the probability that it has succeeded after a given number of steps.

PROPOSITION 3.1 [BRACHA'S BYZANTINE AGREEMENT [BRACHA 1984]]. *Assume that the fraction of good processors is strictly greater than* $2/3$*, and let* q *be any positive constant. Then, Bracha's Byzantine agreement protocol has the following properties:*

(1) *Every processor terminates after* $q \cdot 2^k$ *steps, and is required to send and receive at most* $O((\log q + k) \cdot q2^k)$ *bits.*

(2) *With probability at least* $1 - 1/e^q$*, each processor outputs the same bit. This bit was the input bit of at least one good processor.*

(3) *If, in addition, more than* $2k/3$ *good processors have the same input bit* v'*, then upon termination, after a constant number of deterministic steps, every good processor outputs the value* v'*.*

Bracha's exponential time asynchronous Byzantine agreement protocol [Bracha 1984] can easily be extended to the case where each processor has three possible input values $0, 1$ and $*$, using techniques similar to those of Turpin and Coan [Turpin and Coan 1984]. To do so, one runs Bracha's protocol twice. On the first run, the processors decide whether the outcome is 0 or not. If the outcome is not 0, the processors run the protocol a second time to decide between 1 and $*$. Crucially, if the outcome of the first run is not 0, then every good processor with an input bit of 0 resets its input bit to $*$ for the second run of Bracha's protocol. For the remainder of the paper, we refer to this protocol as HEAVY-BA (heavy weight Byzantine agreement).

PROPOSITION 3.2 [EXTENDED BRACHA'S BYZANTINE AGREEMENT [BRACHA 1984]]. *Assume that the fraction of good processors is strictly greater than* $2/3$*, and let* q *be any positive constant. Let the possible inputs be* $0, 1$ *and* $*$*. Then, the extended Bracha's Byzantine agreement protocol has the following properties:*

(1) *Every processor terminates after* $2(q+1) \cdot 2^k$ *steps, and is required to send and receive at most* $O((\log q + k) \cdot q2^k)$ *bits.*

(2) *With probability at least* $1 - 1/e^q$*, each processor outputs the same bit. This bit was the input bit of at least one good processor.*

(3) *If more than* $2k/3$ *good processors have the same input bit* v'*, then upon termination, after a constant number of deterministic steps, every good processor outputs the value* v'*.*

(*4*) *If the input values of more than $2k/3$ good processors are all contained in $\{0, *\}$ or are all contained in $\{1, *\}$, then upon termination, after a constant number of deterministic steps, the outputs of good processors are all contained in $\{0, *\}$ or all contained in $\{1, *\}$, respectively.*

The parallel broadcast protocol has two parts. The first part uses as a subroutine Bracha's reliable broadcast protocol (which we denote by Reliable-Broadcast). Reliable-Broadcast is a part of Bracha's Byzantine agreement protocol [Bracha 1984]. We describe its relevant properties here:

Proposition 3.3 Bracha's reliable broadcast [Bracha 1984]. Reliable-Broadcast *begins with the broadcasting processor p sending a message to every other processor. Assume that the fraction of good processors is greater than $2/3$. Then,* Reliable-Broadcast *has the following properties:*

(*1*) *If a good processor p' accepts a message from processor p, then every other good processor will accept the same message within $O(1)$ time steps.*

(*2*) *If the broadcasting processor p is good, then within $O(1)$ time steps, its message is accepted by every other good processor.*

Using Reliable-Broadcast, we describe our parallel broadcast protocol Parallel-Broadcast as Protocol 1. In this protocol, c is any positive constant; the choice of c affects the success probability.

Protocol 1 The Parallel-Broadcast protocol at good processor p

```
Send own message to all other processors using Reliable-Broadcast.
Participate in Reliable-Broadcast for the messages of all other processors in parallel.
Maintain a partial list $L_p$ of accepted messages.
if $|L_p| \geq k - t$ then
    Send $L_p$ to all other processors (without using Reliable-Broadcast).
    (Continue participating in Reliable-Broadcast for other messages.)
For each received list $L_{p'}$, mark it as verified if $p$ has accepted all messages in $L_{p'}$ (before or after sending $L_p$).
if $p$ has verified at least $k - t$ lists $L_{p'}$ then
    Create a full list $F_p$, as follows:
    Each message accepted by $p$ is in $F_p$.
    For all messages not yet accepted by $p$, $F_p$ contains $*^m$.
    (Continue participating in Reliable-Broadcast for other messages. However, messages accepted later will not affect $F_p$.)
if $p$ has created $F_p$ then
    Participate in $km$ parallel versions of Heavy-BA with the parameter $q = \log(kmn^c)$, to determine each bit of every message sent by every processor.
```

Proposition 3.4. *The protocol* Parallel-Broadcast *does not deadlock.*

Proof. At least $k-t$ good processors participate in Reliable-Broadcast. By part (2) of Proposition 3.3, every broadcast from a good processor is accepted. Therefore, each partial list L_p will contain at least $k-t$ messages within a constant number of steps.

Since p will receive at least $k-t$ partial lists from good processors, it will succeed in verifying $k-t$ partial lists. This uses part (1) of Proposition 3.3, which guarantees that any message accepted by a good processor p' will be accepted by p as well, within a constant number of steps.

Thus, all good processors eventually enter the Heavy-BA part of the protocol, and by the termination property (1) of Proposition 3.2, every processor terminates. ■

The relevant properties of the protocol Parallel-Broadcast are summarized by the following lemma:

Lemma 3.5. *Assume that* $t \leq (\frac{1}{3} - \epsilon) \cdot k$ *processors are bad. If* Parallel-Broadcast *executes on* k *processors, then for any positive* c *and* n*:*

(1) *With probability at least* $1 - 1/n^c$*, all good processors agree on a value of the message sent by each processor.*

(2) *Every processor enters* Heavy-BA *after no more than* $O(1)$ *time steps and sending* $O(k^2 m)$ *bits.*

(3) *Every processor terminates after* $2(q+1)2^k + O(1)$ *steps, and is required to send and receive* $O((\log q + k)q2^k)$ *bits.*

(4) *The broadcasts of at least* $3\epsilon k$ *good processors are successful, i.e., the messages agreed on for these good processors are the messages which were sent by these processors.*

(5) *The broadcast of a good processor is either successful, or the agreed-upon message for the good processor is* $*^m$.

Proof. (1) By Proposition 3.4, there is no deadlock. Therefore, all good processors will complete the protocol, running Heavy-BA on all bits of all messages. By part (2) of Proposition 3.2 with $q = \log(kmn^c)$, we obtain that with probability at least $1 - 1/n^c$, all good processors will agree on a message for each processor after running Heavy-BA.

(2) By definition of Parallel-Broadcast, the number of time steps before the execution of Heavy-BA is a constant. The number of bits sent is dominated by the distribution of the list of accepted messages to other processors.

(3) The total number of time steps is dominated by the running time of Heavy-BA which is given in part (1) of Proposition 3.2.

(4) We first show, using an averaging argument, that there are at least $3\epsilon k$ good processors whose messages are on the partial lists L_p of at least $t+1$ good processors p. The sum of the sizes of the partial lists held by good processors is at least $(k-t)^2$. Let A be the set of processors which are good and whose messages (as sent) are on at least $t+1$ partial lists of good processors. The partial lists of good processors are partitioned into messages from (1) processors in A, (2) good processors not in A, and (3) bad processors. The respective numbers of messages in the lists are at most (1) $|A| \cdot (k-t)$, (2) $(k-t-|A|) \cdot t$, and

(3) $t \cdot (k-t)$. Hence, we have that $|A| \cdot (k-t) + (k-t-|A|) \cdot t + t \cdot (k-t) \geq (k-t)^2$, and rearranging yields that

$$|A| \geq \frac{(k-3t)(k-t)}{k-2t} \geq 3\epsilon k \cdot \frac{2/3+\epsilon}{1/3+2\epsilon} \geq 3\epsilon k.$$

Next, we show that any message by a good processor p appearing on the partial lists of at least $t+1$ good processors will be on the full lists of all good processors. This follows because in order to proceed, a good processor p' is required to verify at least $k-t$ partial lists (i.e., accept all messages in them). Of these, at least one must contain p's message; hence, p's message will be on the full list of p'. Thus, all good processors enter HEAVY-BA agreeing on these $3\epsilon k$ messages. By part (3) of Proposition 3.2, all good processors will agree with these processors' messages after running HEAVY-BA.

(5) Let p be a good processor. By Proposition 3.3 and Line 11 of PARALLEL-BROADCAST, for each good processor p', either p's correct message is accepted by p', or p' sets p's message to $*^m$ in its full list.
By the definition of Byzantine Agreement, the agreed-upon message has to match the message value held by at least one good processor p', which can only be the correct message or $*^m$ by the above discussion. ■

PARALLEL-BROADCAST can be composed to simulate multiple rounds of synchronous broadcast of messages. To perform r rounds of simulated broadcast by each processor, each processor p does the following:

(1) p runs PARALLEL-BROADCAST with the parameter in each execution of HEAVY-BA set to $q = \log(kmn^c r)$.

(2) p enters the next round only after committing to every message sent by every processor.

LEMMA 3.6. *Let ϵ and c be any positive constants. Let $t \leq (\frac{1}{3}-\epsilon) \cdot k$ be the number of bad processors.* PARALLEL-BROADCAST *with the parameter $q = \log(kmn^c r)$ can be used to simulate a sequence of r rounds of synchronous broadcasts of messages of length m, so that with probability at least $1 - 1/n^c$:*

(1) For each round, all good processors agree on a message for each processor within $2(q+1)2^k + O(1)$ time per round, for a total of $2r(q+1)2^k$ time. The total number of bits sent is $O(kmrq2^k) \cdot \log(kmrq2^k)$.

(2) For any round number i, once any processor p enters round $i+1$, the agreed-upon value of every message sent by every processor in round i is known to p.

(3) After each broadcast round, the messages sent by more than $3\epsilon k$ good processors have been accepted.

Proof. (1) Both the time and communication complexity are dominated by the invocation of HEAVY-BA at the end of each round. Because k processors' messages need to be agreed upon, and each message contains m bits, there will be a total of km parallel invocations of HEAVY-BA per round, with a probability of failure of at most $1/e^q$ for a single broadcast. By a union bound over all kmr invocations of HEAVY-BA, the probability of failure of any invocation of

HEAVY-BA is no greater than $kmr(1/e^q) = O(1/n^c)$. That is, with probability at least $1 - 1/n^c$, all good processors agree on every bit of every message throughout.

Each bit of each message (of length m) sent by each of the k processors in each of the r rounds is agreed upon using a separate version of HEAVY-BA. Each such invocation takes $2(q+1)2^k$ steps; these steps also need to be labeled by the identity of the sending processor, the round, and the number of the bit. Therefore, each transmission is accompanied by a label of length $O(\log(q2^k) + \log(mkr))$.

(2) This follows directly from the definition of multi-step PARALLEL-BROADCAST (Step 2) and part (1) of Lemma 3.5.

(3) This follows from part (4) of Lemma 3.5. ■

3.2 Keeping bins balanced

To address problem (2), our election protocol is designed so that the set of good processors whose choices are used to select the subcommittee is in a sense determined *before* they make their choices. The simulation of Feige's "Lightest Bin" protocol is run for $k+1$ rounds. Each good processor p maintains a $(k+1) \times (k+1)$ table whose (i, j) entry is the agreed-upon value of processor j's bin choice in round i. Each message sent by a processor p in round i contains *all of its previous bin choices* $B_{i'}^{(p)}$ from rounds $i' < i$, and is broadcast using simulated broadcast. The message length is no greater than k times the length of the bin choice ($k \lg k$), and $q = \log(k \cdot k \lg k \cdot n^c \cdot (k+1)) = O(\log n)$. The table is (retroactively) updated if necessary: if a message from processor j is accepted in round i, then none of the previous entries regarding processor j will be $*$.

Let S_i be the set of processors whose messages were accepted in some round $i' \geq i$, and let $A_i \subseteq S_i$ be the set of good processors whose messages for round i were accepted by the end of round i. Then $S_{k+1} \subseteq S_k \subseteq S_{k-1} \cdots \subseteq S_1$. If $|S_1| > 0$, then by the Pigeonhole Principle, there is an r such that $S_r = S_{r-1}$; we fix the smallest such r. The accepted bin choices in round r are then used to determine the lightest bin in Feige's protocol. We say that *processor p is in bin B* in round i if $B_i^{(p)} = B$, and p's message selecting B is (eventually) accepted.

LEMMA 3.7. *Let c, ϵ be any positive constants, and suppose that there are k processors of which $t \leq (\frac{1}{3} - \epsilon) \cdot k$ are bad. There is a constant c' such that if the number of bins is $b = k/(c' \log n)$, then*

(*1*) *All good processors agree on the set of processors in A_r.*

(*2*) *With probability at least $1 - 1/n^c$, the number of good processors in the lightest bin in round r (from A_{r-1}) is at least $\gamma \cdot c' \log n$, where $\gamma = \frac{3\epsilon}{2}$.*

(*3*) *The total time is $(k+1)(q+1)2^k = O(k2^k \log n)$.*

(*4*) *The total number of bits sent is $O(k^4 2^k \log n \log k)$.*

Proof. (1) This follows directly from Lemma 3.6 (2).

(2) Lemma 3.6 implies that with probability $1 - 1/n^{c+1}$, all rounds of broadcast have been successful, i.e., all good processors' tables are identical. From Lemma 3.6 (3), it follows that $|A_i| \geq 3\epsilon k$ for all i.

Fix any bin B and round i. By Lemma 3.6 (2), when $p \in A_{i-1}$ chooses its bin in round i, the messages for all processors in A_{i-1} for round $i-1$ are already accepted. Let the random variable $X_j = 1$ if $p \in A_{i-1}$ chooses B in round i and 0 otherwise. Then $X = \sum_j X_j$ is the number of good processors in A_{i-1} who choose B in round i. The X_j are independent coin tosses with $\mathrm{Prob}[X_j = 1] = 1/b$ and $\mathrm{E}[X] = |A_i|/b$.

Conditioned on the event that all rounds of broadcast have been successful, we apply Chernoff Bounds to obtain that $\mathrm{Prob}[X < \mathrm{E}[X]/2] \leq e^{-\mathrm{E}[X]/12}$ or $\mathrm{Prob}[X < \mathrm{E}[X]/2] \leq 1/n^{(c+3)}$, for $c' \geq \frac{12(c+3)}{\epsilon}$. By a union bound over all rounds $i = 1, \ldots, k$ and all $b \leq n$ bins, the probability that any bin B is chosen in round i by fewer than $\frac{3k\epsilon}{2b}$ processors in A_{i-1} is at most $1/n^{c+1}$.

Hence with probability at least $1 - 2/n^{c+1} > 1 - 1/n^c$ (for sufficiently large n), all rounds of the broadcast are successful, $|A_i| \geq \frac{3k\epsilon}{b}$ for all i, and at least $\frac{3k\epsilon}{2b}$ processors are in every bin in every round.

In particular, at time step r when $S_r = S_{r-1}$, $A_{r-1} \subseteq S_{r-1} = S_r$, so the lightest bin in round r contains at least $\frac{3k\epsilon}{2b}$ processors.

(3) This follows directly from Lemma 3.6 (1), by substituting $r = k+1$, $m = k \lg k$, and $q = \log(k \cdot k \lg k \cdot n^c \cdot (k+1)) = O(\log n)$.

(4) Also follows from Lemma 3.6 (1) by substitution. ■

3.3 Enlarging the fraction of good processors

To address problem (3), the processors in the lightest bin are used to select a better subcommittee with probability at least $1 - 1/n^c$ for any positive constant c. Assume that the processors in the committee are numbered 1 to k. The processors in the subcommittee each randomly pick a *block* of a constant number x of bits, which are then concatenated in order of processor number to form an input to a sampler. These bits are sent in the same message as the bin choice.

Let R be the collection of all binary strings of length between $x\gamma k/b$ and xk/b (where γ is defined in Lemma 3.7), and let $r = |R|$. Then $2^{xk/b} < r < (xk/b) \cdot 2^{xk/b}$. By Lemma 2.2, there is a $(1/\log n, r^{-a})$ sampler $H : [r] \to [k]^d$ for any fixed $a < 1$, with $d = O(\log^3 n)$. The output of H is the subcommittee elected by the committee, of size $O(\log^3 n)$. Notice that this subcommittee is a multiset, i.e., may contain multiple copies of the same processor.

Lemma 3.8. *Assume that $k = \log^c n$ for a constant $c > 3$. W.h.p., the subcommittee produced by H contains less than a $(t/k + 1/\log n)$ fraction of bad processors, and all processors in the committee agree on this subcommittee.*

Proof. We call an input $\rho \in R$ to the sampler *bad* if it maps to an output $H(\rho)$ corresponding to a subcommittee with more than a $(t/k + 1/\log n)$ fraction of bad processors.

Fix a bad input ρ. By Lemma 3.7, if ρ is generated by the lightest bin, then w.h.p., at least $\gamma k/b$ blocks chosen uniformly at random by processors in A_{r-1} match a subsequence of blocks of ρ. The probability of matching a particular subsequence is less than $2^{-x\gamma k/b}$. By the definition of a sampler, there are at most r^{1-a} bad inputs. There is a constant c' such that we may set $a = 1 - c'\gamma/2$ and $r^{1-a} = 2^{x\gamma k/2b}$. Taking the union bound over all possible bins and round numbers for the lightest bin and all possible subsequences of blocks, and the different bad inputs, the probability of the lightest bin generating a bad input is at most $b(k+1) \cdot 2^{x\gamma k/2b} \cdot 2^{-x\gamma k/b} < n^{-c''}$ for every positive constant c'', and a sufficiently large constant x which depends only on c''.

Thus, w.h.p., the input to the sampler is good, and a subcommittee with no more than a $(t/k + 1/\log n)$ fraction of bad processors is output. ■

3.4 The Subcommittee Election Protocol

We obtain the protocol Elect-Subcommittee, run at each processor p in a committee $C = \{p_1, \ldots, p_k\}$ of processors, with $k > \log^3 n$.

Protocol 2 The Elect-Subcommittee protocol on good processor p

```
for $i = 1$ to $k+1$ do
    Randomly select a bin number $B_i^{(p)} \in \{1, 2, \ldots, b\}$ (where $b = k/(c' \log n)$),
    and a random bit string $X_i^{(p)}$ of length $x$.
    Use Parallel-Broadcast to send $((B_1^{(p)}, X_1^{(p)}), \ldots, (B_i^{(p)}, X_i^{(p)}))$ to every processor in $C$.
    Update all $(B_{i'}^{(p')}, X_{i'}^{(p')})$ for $i' \leq i$.
Let $S_i$ be the multiset of processors $p'$ for whom $B_i^{(p')}$ (and $X_i^{(p')}$) are
known. Let $r$ be smallest such that $S_r = S_{r-1}$.
Let $B$ be the lightest bin in round $r$, and $\rho$ be the concatenation, ordered
by $p'$, of the blocks $\{X_r^{(p')} \mid B_r^{(p')} = B\}$ chosen by the multiset of processors
in bin $B$ in round $r$.
Return $H(\rho)$ as the elected subcommittee.
```

The properties of the multiset elected as a subcommittee are summarized below. The proof follows directly from Lemmas 3.6, 3.7 and 3.8.

Lemma 3.9. *For $\epsilon > 0$, if a committee of k processors has $t < (\frac{1}{3} - \epsilon) \cdot k$ bad processors, then with high probability:*

(1) Elect-Subcommittee *runs in time $O(k2^k \cdot \log n)$.*

(2) *Every good processor outputs the same elected subcommittee.*

(3) *The elected subcommittee contains $O(\log^3 n)$ processors, of which at most a $(t/k + 1/\log n)$ fraction are bad.*

We call a subcommittee election *successful* if it satisfies the conditions of the lemma above.

4. A QUASI-POLYNOMIAL ASYNCHRONOUS BYZANTINE AGREEMENT PROTOCOL

The high-level idea of our construction is the following. We want to reduce the number of processors under consideration to a polylogarithmic number, in order to run the Heavy-BA protocol on the remaining processors, and have it take sub-exponential time. The Elect-Subcommittee protocol provides us with a way of electing such a small multiset of processors. However, since the Elect-Subcommittee protocol itself also invokes the Heavy-BA protocol, it cannot be run on any multiset of processors of more than polylogarithmic size.

Hence, instead of immediately reducing the number of processors, we reduce the number repeatedly ($O(\log n/\log\log n)$ times) by a factor of $(1-1/\log n)$. This is done by assigning, in each iteration, the remaining processors to *committees* of size $k=\Theta(\log^8 n)$, such that each processor participates in $\log^3 n$ committees. Each committee runs the Elect-Subcommittee protocol to elect a subcommittee of $\Theta(\log^3 n)$ processors. All processors elected to a subcommittee proceed to the next round. Notice that this does indeed accomplish the desired reduction in the number of processors.

In order to avoid basing decisions on the *actual* identities of processors (which could give an adversary more power by choosing different identities), we instead determine the assignments to committees by a *template network*. The template network consists of processor nodes $\mathcal{P}_\ell$ and committee nodes $\mathcal{C}_\ell$ in each layer ℓ. Edges from processor nodes to committee nodes indicate membership in the committee. Edges from committee nodes to processor nodes in the next layer indicate that the committee's election will determine which *actual processors* play the role of the corresponding processor nodes. As the protocol proceeds, processor nodes are thus gradually "assigned" processors who will serve in the role prescribed by that node.

To define the template network more formally, let ℓ^* be the minimum integer ℓ such that $n/\log^\ell n \le \log^8 n$; note that $\ell^* = O(\log n/\log\log n)$. The layers of the network alternate between *processor nodes* and *committee nodes*. The ℓ^{th} layer of processor nodes is a set $\mathcal{P}_\ell$ of $s_\ell = n/\log^\ell n$ nodes. The ℓ^{th} layer of committee nodes is a set $\mathcal{C}_\ell$ of $r_\ell = n/\log^{\ell+4} n$ nodes. The $\ell^{*\text{th}}$ layer of committee nodes has a single node C^*.

Each committee node $C \in \mathcal{C}_\ell$ (for $\ell < \ell^*$) has $\log^3 n$ outgoing edges to nodes in $\mathcal{P}_{\ell+1}$, such that each node in $\mathcal{P}_{\ell+1}$ has exactly one incoming edge. For each node C, these edges are labeled from $1,\ldots,\log^3 n$: they determine which of the $\log^3 n$ processors elected to the subcommittee will fill the role of which processor nodes in layer $\ell+1$.

C also has $k=\Theta(\log^8 n)$ ordered *slots*. These slots will eventually be filled with identities of processors who participate in the committee. The processors assigned to slots are determined by the edges into C from the previous layer. Specifically, the edges between $\mathcal{P}_\ell$ and $\mathcal{C}_\ell$ are determined by a $(1/\log n, 1/\log n)$ sampler H_ℓ where $r=r_\ell$, $s=s_\ell$ and $d=\Theta(\log^8 n)$. (The existence of such a sampler follows from Lemma 2.3.) There is an edge from the i^{th} node in $\mathcal{P}_\ell$ to the j^{th} node in $\mathcal{C}_\ell$ iff $i \in H_\ell(j)$. (In layer ℓ^*, there is an edge to C^* from every node in $\mathcal{P}_{\ell^*}$.)

The reason for the use of a sampler is as follows: During the execution of the protocol, up to a $(\frac{1}{3}-\epsilon+o(1))$ fraction of the processor nodes at each layer will be assigned to bad nodes. If bad nodes are disproportionately represented in a committee, they can too strongly influence the outcome of the subcommittee election. By definition, the sampler H_ℓ ensures that whatever set of processor nodes are assigned to bad nodes, at most a $1-1/(2\log n)$ fraction of the committees will have more than a $(\frac{1}{3}-\epsilon+o(1)+1/\log n)$ fraction of bad members. This in turn ensures that the committee's execution of Elect-Subcommittee succeeds with high probability. This intuition lies at the core of our formal proof below.

Remark 4.1. *Notice that we assume that each processor has access to a copy of the same template network. This can be accomplished either by pre-specifying it for the given size of the network n at the start of the protocol, or by generating it in polynomial time. (See the discussion about samplers in Section 2.2.)*

4.1 Views and Cores

In the description above, we casually referred to a processor being "assigned" to a node of the template graph. Given that these assignments are the result of a distributed protocol with an adversary, this is not a well-defined notion. Instead, each processor p maintains *views* $v_p(P)$ and $v_p(C)$ of each processor node P or committee node C. A view $v_p(P)$ of a processor node is either the name of a processor (which p believes to be assigned to P) or $\perp$. The view $v_p(C)$ of a committee node is an ordered list of the processors which p associates with C's slots. These are the same as the processors p associates with the predecessors of C. Each processor p updates its views throughout the protocol.

Due to delayed messages and adversarial behavior, the views of different processors p, p' about a processor node P might differ. Thus, we have to be careful about how we describe the execution of the protocol Elect-Subcommittee for a committee C. Denote the slots in C by $I = \{1, 2, \ldots, k\}$.

definition 4.2 [Core]. *A* core *of a committee is a set R of good processors p such that*

(1) every p believes itself to be in the committee, and

(2) all processors in R agree on p's identifier.

That is,

$$R = \{p \text{ is good} \mid v_p^{-1}(p) \cap I \neq \emptyset \text{ and for all } p' \in R : v_{p'}^{-1}(p) = v_p^{-1}(p)\}.$$

By inspecting the proof of Lemma 3.9, it follows that a large core is sufficient for a successful execution of Elect-Subcommittee. This is captured in the following

Observation 4.3. *With high probability, in any committee with a core set R such that $|R| > (\frac{2}{3} + \epsilon) \cdot k$ for some fixed $\epsilon > 0$, every $p \in R$ will come to agreement on the identifiers of the processors in the elected subcommittee, and consequently on the actual identities of those processors in the elected subcommittee which come from R.*

Notice that the binary strings $X_i^{(p)}$ of length x in the protocol Elect-Subcommittee are now concatenated in order of increasing slot number from I.

4.2 The Protocol

The protocol consists of executions of the Elect-Subcommittee protocol in each committee node C. Recall that each committee has size $k = \Theta(\log^8 n)$. Since processors participate in many such committees (both per layer, and across layers), messages sent between processors are annotated with the name of the committee node C whose election procedure the message contributes to. If a processor p

receives a message from a processor p', annotated with the committee node C, yet $p' \notin v_p(C)$, then p simply disregards the message.

Initially, all processors have identical views of processor nodes in layer $\mathcal{P}_0$, and views $\perp$ for all other processor nodes. In order to proceed with the Elect-Subcommittee protocol for a committee node C, a processor p requires that there be "enough" processors in its view of the committee node. We write $|v_p(C)|$ for the number of actual processors (i.e., non-$\perp$ entries) in p's view of C. If a processor is in multiple slots of C, it is counted separately for each slot.

If $p \in v_p(C)$, then p intends to participate in the Elect-Subcommittee protocol at C. It waits until $|v_p(C)| \geq (1 - 3/\log n) \cdot k$, i.e., it knows the identities of at least a $(1 - 3/\log n)$ fraction of the processors to participate, and then sends its view $v_p(C)$ of C to all processors. Once p receives views of C from at least $2k/3$ processors in $v_p(C)$, p revises its view for each slot to agree with the majority and sets its status $\sigma_p(C)$ to `ready`. It is possible that there is no majority view for one or more slots, or that the majority of views have $v_{p'}(P) = \perp$. In either of these cases, p sets its revised view of that slot to $\perp$.

If p is still in the revised $v_p(C)$, then p participates in the subcommittee election at C; otherwise, p waits to hear from other processors about the results. After the election is run, the status $\sigma_p(C)$ is set to `fixed`. The formal statement of the Quasi-Poly-BA protocol appears as Protocol 2.

4.3 Proof of Correctness

We will prove the following in this section:

Theorem 4.4. *W.h.p., the protocol* Quasi-Poly-BA *terminates in quasi-polynomial time and achieves Byzantine agreement.*

The idea of the proof is to show that in each layer ℓ, at most a $\frac{1}{3} - \epsilon + o(1)$ fraction of processor nodes $P \in \mathcal{P}_\ell$ is "bad", in the sense that they do not contribute accurately to the execution of Elect-Subcommittee protocols. This can be either because P is assigned a bad processor, or because of a previous execution of Elect-Subcommittee which failed to provide consistent views of P to other nodes. We show that the latter only happens to at most an $O(1/\log n)$ fraction of committees in each layer, because committees are formed according to a $(1/\log n, 1/\log n)$ sampler. Even the accumulation of all such failures across all layers is not enough to bring the fraction of bad processor nodes to more than $\frac{1}{3} - \epsilon + o(1)$ in any layer. Formally, we define the following:

definition 4.5 [Good, Bad, and Unreliable Nodes]. *(1) A processor node $P \in \mathcal{P}_0$ is* good *if there is a good processor assigned to this node. All processor nodes $P \in \mathcal{P}_0$ are* reliable.

(2) A slot i of a committee node C is good *iff its predecessor processor node is good. It is* reliable *iff its predecessor processor node is reliable.*

(3) $\alpha_\ell = \frac{1}{3} - \epsilon + \frac{10\ell}{\epsilon \log n}$ denotes the badness tolerance *in layer ℓ.*

(4) A committee node $C \in \mathcal{C}_\ell$ for $\ell \geq 0$ is good *if the following are true:*
—At least a $(1 - \alpha_\ell - 1/\log n)$ fraction of its slots are good.
—At least a $(1 - 3/\log n)$ fraction of its slots are reliable.

Protocol 3 The Quasi-Poly-BA protocol on good processor p

```
while there is a committee node $C$ such that $\sigma_p(C)$ is not fixed do
  for all such nodes $C$ in parallel do
    if $\sigma_p(C)$ is not ready then {find processors in this committee}
      if $|v_p(C)| \geq (1 - 3/\log n) \cdot k$ then
        if $p \in v_p(C)$ then
          $p$ sends $v_p(C)$ to all processors.
        $p$ waits until it receives $v_{p'}(C)$ from at least a $2k/3$ processors $p' \in v_p(C)$.
        For each position of $v_p(C)$, $p$ revises its view to agree with the majority of
          the $v_{p'}(C)$ received.
        $\sigma_p(C)$ is set to be ready.
    else if $C$ is in layer $\ell$ for some $\ell < \ell^*$ then {subcommittee election node}
      if $p \in v_p(C)$ then
        $p$ runs Elect-Subcommittee with the other processors in $v_p(C)$ to elect
          a subcommittee.
        When $p$ has decided on election results, it sends the set of winning identi-
          fiers to all processors.
      $p$ waits to receive election result messages from at least $2k/3$ processors in
      $v_p(C)$.
      if the majority of received messages agree on the elected subcommittee
      then
        $p$ determines the elected subcommittee by taking the (agreeing) majority.
        $p$ uses $v_p(C)$ and the elected subcommittee to determine $v_p(P)$ for layer-
          $(\ell + 1)$ neighbors $P$ of $C$, and slots of layer-$(\ell + 1)$ committee nodes that
          these neighbors are assigned to.
      else
        $p$ sets $v_p(P)$ arbitrarily for layer-$(\ell + 1)$ neighbors $P$ of $C$, and for slots of
          layer-$(\ell + 1)$ committee nodes that these neighbors are assigned to.
      $\sigma_p(C)$ is set to be fixed.
    else {$C = C^*$, actual Byzantine agreement performed}
      if $p \in v_p(C)$ then
        $p$ runs Heavy-BA with the processors in $v_p(C)$.
        $p$ sends the agreed-upon bit value to all processors.
      else
        $p$ waits to receive a bit from at least a $2k/3$ processors in $v_p(C)$.
        $p$ takes the majority of received bit values as agreed upon.
      $\sigma_p(C)$ is set to fixed, and $p$ terminates.
```

(5) *A processor node $P \in \mathcal{P}_\ell$ for $\ell > 0$ is* unreliable *if its predecessor committee node is bad. Otherwise, it is* reliable.

(6) *A processor node $P \in \mathcal{P}_\ell$ for $\ell > 0$ is* good *if it is reliable, and the predecessor node of C which is sent to P according to the subcommittee election outcome is good.*

(7) *A node that is not good is* bad.

Figure 1 illustrates these definitions and their consequences.

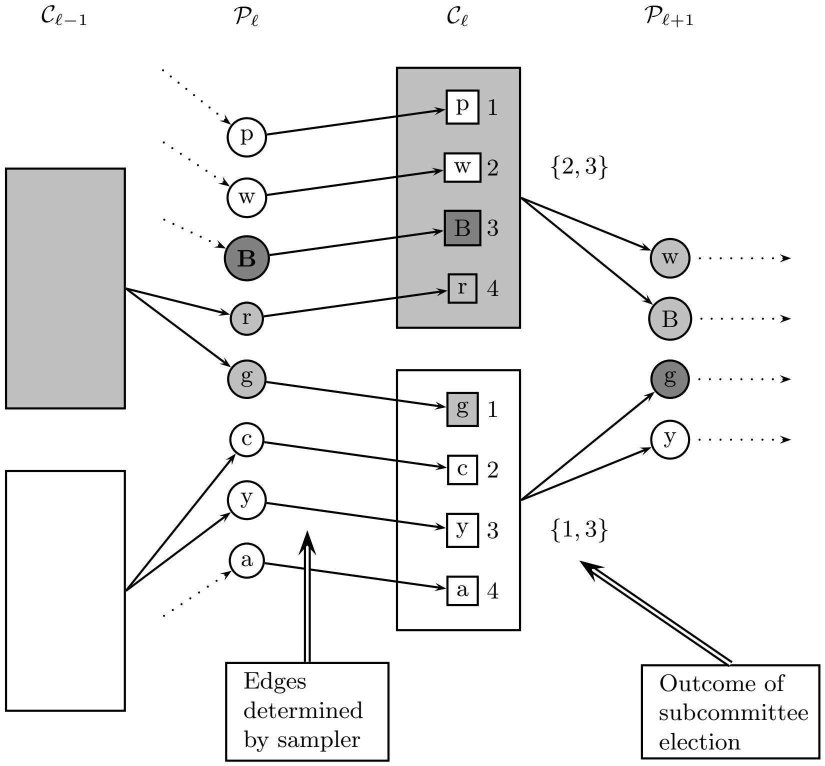

Fig. 1. A depiction of part of the template network. Processor nodes are circles; committee nodes large rectangles, and their slots are small squares, with their identifiers given. Processors are identified by letters; the processors assigned to processor nodes and slots are drawn inside the nodes. A bad processor 'B' is identified by its capitalized name. Bad processor or committee nodes are gray, as are bad slots within committee nodes. Unreliable processor nodes or slots are light gray. The processor nodes assigned to processors 'r' and 'g' in layer ℓ are the result of a bad committee, and hence unreliable. The processor node containing 'B' is bad because its processor is bad. The upper committee node is bad because too many of its slots come from bad processor nodes. Hence, both processor nodes it assigns are unreliable. The outcome of the elections in the committee nodes are the slot identifiers whose assigned processors are assigned to the processor nodes in the next layer. Because the slot assigned to 'g' was bad, so is the processor node assigned to processor 'g' in layer $\ell + 1$.

The key lemma in establishing Theorem 4.4 is the following, capturing precisely the intuition we gave above.

Lemma 4.6. *Let $k = \Theta(\log^8 n)$ be the size of committees. With high probability, the following hold for all $\ell \leq \ell^*$ by time $O(\ell \cdot k2^k \cdot \log^2 k)$:*

(1) *The fraction of good processor nodes in layer $\mathcal{P}_\ell$ is at least $1 - \alpha_\ell$.*

(2) *For each good processor node P, there exists a processor p such that all views of P are either p or $\perp$. (We call p the* identity *of P.)*

(3) *The fraction of good committee nodes in layer $\mathcal{C}_\ell$ is at least $1 - 2/\log n$.*

Proof. The proof is by induction on ℓ. For the base case, consider $\mathcal{P}_0$. All nodes are reliable, so only nodes assigned to bad processors are bad. The fraction of good processor nodes is therefore at least $2/3 + \epsilon = \alpha_0$, establishing (1). Property (2) follows because the initial assignment is common knowledge.

For the inductive step, assume that properties (1) and (2) hold up to layer ℓ, and property (3) up to layer $\ell - 1$, within time $O(\ell \cdot k2^k \cdot \log^2 k)$.

We first prove property (3) for layer ℓ. Because each processor node is assigned by exactly one committee node, the fraction of unreliable processor nodes in layer ℓ is at most $2/\log n$.

Recall that the edges between $\mathcal{P}_\ell$ and $\mathcal{C}_\ell$ are defined by a $(1/\log n, 1/\log n)$ sampler. We apply its expansion properties twice: once for the set of all *unreliable* processor nodes, and once for the set of all *bad* processor nodes. We obtain that:

(1) At most a $1/\log n$ fraction of committee nodes have more than a $2/\log n + 1/\log n = 3/\log n$ fraction of unreliable slots.

(2) At most a $1/\log n$ fraction of committee nodes have more than an $\alpha_\ell + 1/\log n$ fraction of bad slots.

Thus, by definition of good committee nodes, at least a $1 - 2/\log n$ fraction of the committee nodes in layer ℓ are good by the time their slots are assigned, establishing (3).

We now focus on any one good committee node C, and a processor $p \in C$. Processor p starts an election protocol for C when it has non-$\perp$ views for at least a $1 - 3/\log n$ fraction of C's predecessors. (Notice that this happens eventually for good committee nodes, as at most a $1 - 3/\log n$ fraction of slots are unreliable.) Thus, it has non-$\perp$ views for at least a $1 - 3/\log n - (\alpha_\ell + 1/\log n) > 2/3$ fraction of *good* predecessors of C, i.e., knows their identities. In lines 7–9 of the protocol, p waits to receive views from $2k/3$ processors in $v_p(C)$ and takes the majority view for each slot. Since fewer than $(\frac{1}{3} - \epsilon) \cdot k$ slots of C are bad, and at most $3k/\log n$ are unreliable, the $2k/3$ processors include at least $(k/3 + \epsilon k/2)$ good processors from good reliable slots. By property (2), for good predecessors, all views are either p' (for some unique p') or $\perp$. Thus, the view of p for any slot can become $\perp$ only if — in addition to the at most $k/3$ bad processors — at least $\epsilon k/2$ good processors in the committee have not yet determined the identity of that slot.

We use a counting argument to show that the number of such slots must be small. Each good processor initially has at most $3k/\log n$ $\perp$-entries in its view, for a total of at most $3k^2/\log n$ such entries. For any slot to be $\perp$ after the majority vote, $\epsilon k/2$ good processors have to have $\perp$ entries for that particular slot. Thus, at most $\frac{3k^2/\log n}{\epsilon k/2} = \frac{6k}{\epsilon} \log n$ entries can be $\perp$ after the majority vote. Thus, the core can exclude at most $\frac{6k}{\epsilon} \log n$ good slots of C in addition to at most $(\alpha_\ell + 1/\log n) \cdot k$ bad slots. The core therefore contains at least a $(1 - \alpha_\ell - \frac{7}{\epsilon} \log n)$ fraction of the nodes in the committee.

We can therefore apply Lemma 3.9 (or, to be precise, Observation 4.3) with $\epsilon' = \epsilon/2$. They imply that within time $O(k2^k \cdot \log n)$, with high probability, the subcommittee election in committee C succeeds in the following sense: Every good processor in the core outputs the same elected subcommittee of size $\log n$, of which at most an $(\alpha_\ell + \frac{7}{\epsilon} \log n + 1/\log n)$ fraction are not in the core. In particular, the fraction of bad successor nodes of C is at most $(\alpha_\ell + \frac{8}{\epsilon} \log n)$.

By taking a union bound over all good committee nodes C, with high probability, at most an $(\alpha_\ell + \frac{8}{\epsilon}\log n)$ fraction of their successors are bad. In addition, at most a $2/\log n$ fraction of the processor nodes in layer $\ell+1$ are successors of bad committee nodes, making them unreliable and thus also bad. In total, the fraction of bad processor nodes in layer $\ell+1$ is thus at most $(\alpha_\ell + \frac{10}{\epsilon}\log n) = \alpha_{\ell+1}$. This proves property (1).

Finally, to prove property (2), focus on one good processor node $P \in \mathcal{P}_{\ell+1}$. A view $v_p(P)$ is only determined by p when it receives the result of an election, or by the later majority vote. We show that all good processors set their views $v_p(P)$ to the same processor p' when they set it for the first time. Then, the subsequent majority vote can only result in outcomes p' or $\perp$. Let $C \in \mathcal{C}_\ell$ be the unique (good) predecessor node of P. Then, under the above high-probability event of successful subcommittee elections, all members of the core of C agreed on the processor p' assigned to P. Because all members of the core are from good processor nodes by definition, p by induction hypothesis either knows their identities or has their identities as $\perp$. If p sets its view of P to anything except $\perp$, it must have received messages from at least $2k/3$ processors from $v_p(C)$. Of these, at least $k/3 + \epsilon k/2$ must come from the core; in particular, the majority is the selection of the core.

■

In particular, the last layer ℓ^* has the desired properties from which the theorem follows:

Proof of Theorem 4.4. Because $\ell^* = o(\log n)$, Lemma 4.6 implies that w.h.p., the committee C^* constituting $\mathcal{C}_{\ell^*}$ is good. Similar to the proof of Lemma 4.6, C^* thus has a core R of at least $(1 - \alpha_\ell - \frac{10}{\epsilon}\log n) \cdot k$ good processors. Because $(1 - \alpha_\ell - \frac{10}{\epsilon}\log n) > 2/3$, Heavy-BA guarantees that w.h.p., the processors of C^* will reach Byzantine agreement in quasi-polynomial time.

By property (2), all good processors will have the same view of each processor node participating in C^*. Since they wait to hear from at least $2k/3$ processors in C^* and then take the majority, w.h.p., all good processors will reach Byzantine agreement in time $O(\ell^* \cdot k2^k \cdot \log^2 k)$, completing the proof. ■

5. THE POLYLOGARITHMIC TIME PROTOCOL

Our goal in this section is to improve the running time from quasi-polynomial to polylogarithmic. The bottleneck for the running time is the exponential Heavy-BA protocol used for Byzantine Agreement on the committees of size $k_1 = k = \Theta(\log^8 n)$. By replacing these invocations of Heavy-BA instead with recursive calls to essentially Quasi-Poly-BA, we will obtain the desired speedup. At the first level of recursion, the size of the committees is $k_2 = \Theta(\log^8 k_1) = \Theta((\log\log n)^8)$. At that point, Heavy-BA would take quasi-polylogarithmic time. To get it all the way to polylogarithmic, we add one more level of recursion. At the third level, the committee size is $k_3 = \Theta(\log^8 k_2) = \Theta((\log\log\log n)^8)$. (For notational convenience later, we define $k_0 = n$ to be the number of all processors.)

More specifically, we define a protocol Consensus(ρ, C, Σ), where $\rho \in \{1, 2, 3\}$ is the level of the recursion, C is the set of nodes participating in the consensus protocol, and Σ describes the "history" leading to this call. The history of an invocation at level ρ contains the following for all levels $\rho' \leq \rho$ in the recursion tree: the identifiers for the committee nodes, the round of computation in Elect-Subcommittee, the identifier of the message, and the identifier of the bit in the message. It thus uniquely determines where in the recursive history of the computation this particular invocation is located. Intuitively, we can view it as a "stack" of the recursive protocol.

We describe Consensus(ρ, C, Σ) in terms of the changes compared to Quasi-Poly-BA.

—At levels $\rho \in \{1, 2\}$, the protocol starts with n resp. k_1 processors. The template network structure describing the protocol is exactly the same as for Quasi-Poly-BA with the appropriate number of processors. The protocol is changed by replacing each call to Heavy-BA inside a committee C' instead with a call to Consensus$(\rho + 1, C', \Sigma')$, where Σ' is obtained from Σ by appending C' as well as all relevant information about the round of the protocol, bit of the message, etc.

—At level $\rho = 3$, we can safely invoke Heavy-BA for consensus computations in committee nodes. Also, since a committee size of $\Theta(\log\log n)$ is enough to safely run Heavy-BA, we reduce the number of layers ℓ^* of the network. This is necessary because we will apply tail bounds in each layer, which require "enough" committee nodes in each layer.

When starting with $k_2 = \Theta((\log\log n)^8)$ nodes, we set ℓ^* such that the number of committee nodes in layer $\ell^* - 2$ is $c_1 \log\log n$ for some constant c_1 that will be implicitly determined below. Notice that this makes ℓ^* smaller than the value prescribed in Quasi-Poly-BA for k_2 processors. Also notice that the number of committees in layer $\ell^* - 1$ is $\Theta(\log\log n / \log\log\log n)$, while the number in layers $\ell \le \ell^* - 3$ is $\Omega(\log\log n \cdot \log\log\log n)$.

One final change is necessary: for the committees in layer $\ell^* - 1$, the sampler used in Elect-Subcommittee for the final selection of the subcommittee (see Section 3.3, also for the definitions of r, a) is now an $(\epsilon/6, r^{-a})$ sampler $H' : [r] \to [k_2]^d$ where $d = O(\log\log\log n)$. (Recall that the sampler in Section 3.3 would instead be a $(1/\log n, r^{-a})$ sampler of degree $d = O((\log\log\log n)^3)$.) The existence of this sampler follows from Lemma 2.2.

Remark 5.1. *We remark here that the problem solved in the committee C^* performing Byzantine Agreement is slightly easier than the Byzantine Agreements required for agreeing on message bits, since there are no '$*$' entries. While we could keep track of which recursive invocations could potentially involve '$*$' entries, we are instead using the same protocol (*Consensus *or* Heavy-BA*) in all instances for clarity.*

definition 5.2 [Successful Execution]. *We say that a call to a subprotocol* Consensus(ρ, C, Σ) *(with $k = k_\rho$ processors in C) for some $\epsilon > 0$ is* successful *if it satisfies the following:*

(1) If at least a $\frac{2}{3} + \epsilon$ fraction of processors are good, then all good processors $i \in C$ set their values v_i to the same value in $\{0, 1, *\}$.

(2) If at least $(\frac{2}{3} + \epsilon) \cdot k$ good processors have the same value $v \in \{0, 1, *\}$ initially, then all good processors i set v_i to v.

(*3*) *If the input values of at least* $(\frac{2}{3}+\epsilon)\cdot k$ *good processors are all contained in* $\{0,*\}$ *or are all contained in* $\{1,*\}$*, then upon termination, after a constant number of deterministic steps, the outputs of good processors are all contained in* $\{0,*\}$ *or all contained in* $\{1,*\}$*, respectively.*

Our main theorem in this section guarantees successful execution of CONSENSUS. Its proof will occupy all of Section 5.1.

THEOREM 5.3. *For any* C *and* Σ*, and any* $\rho \in \{1,2,3\}$*, the execution of the protocol* CONSENSUS(ρ, C, Σ) *is successful with probability at least* $1-1/\log^c n$ *for every constant* c*.*

(Notice that the success probability is indeed polylogarithmic in n, not in k_ρ, even for much smaller sets of processors.) By setting $\rho = 1$, we thus obtain that CONSENSUS$(1,\{1,\ldots,n\},\emptyset)$ succeeds with probability at least $1-\log^c n$ for every constant c.

5.1 Correctness of the protocol

The proof of correctness will be bottom up, establishing Theorem 5.3 for $\rho = 3,2,1$. Note that Definition 5.2 is essentially the same as the standard definition of the correctness of Byzantine agreement, albeit with a slightly less stringent requirement due to the ϵ term. We can combine the analysis of both properties of Definition 5.2 as follows: If at least $(\frac{2}{3}+\epsilon)\cdot k$ good processors have the same values, then we will simply consider those as “good” processors, and treat all other processors as bad. By showing that enough “good” processors remain, we will also have shown that the fraction of processors with the same value v remains large. On the other hand, if there is no set of at least $(\frac{2}{3}+\epsilon)\cdot k$ good processors with the same value, then there is nothing to prove for Condition (2), and we simply focus on the good processors.

We will be using the following variants of Chernoff Bounds:

PROPOSITION 5.4. *Let* $X=\sum X_i$ *be a sum of independent 0-1 random variables* X_i*, with* $\mu = \mathrm{E}[X]$*.*

(*1*) *If* $y \geq 2e\cdot\mu$*, then* $\mathrm{Prob}[X>y] < 2^{\mu-y}$*.*

(*2*) *If* $y \geq e^2\cdot\mu$*, then* $\mathrm{Prob}[X>y] < e^{-\frac{1}{2}y\log(y/\mu)}$*.*

Proof. Both forms can be readily derived from the standard Chernoff Bound $\mathrm{Prob}[X>(1+\delta)\mu] < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$. In both cases, substitute $\delta = y/\mu - 1$. For the first bound, use that $(1+\delta)^{1+\delta} \geq (2e)^\delta$ by assumption. For the second bound, substitute to obtain $e^{-y\log(y/\mu)+y-\mu}$, and then use that $y \leq \frac{1}{2}y\log(y/\mu)$ by assumption on y. ■

Proof of Theorem 5.3. (1) We first prove the theorem for the recursion level $\rho = 3$. Even though the number of nodes in the committee is $k_2 = \Theta((\log\log n)^8)$, we want to guarantee a success probability of $1-1/\log^c n$ for any c. This does not allow us to take a union bound over the success of each subcommittee election. Instead, we use Chernoff Bounds to show that in each layer, the number of successful subcommittee elections is close to the expected number, which in turn is enough to ensure success of the protocol.

First, focus on a layer $\ell \leq \ell^* - 3$ of the network. By definition of ℓ^*, this means that there are $r_\ell = \Omega(\log\log n \cdot \log\log\log n)$ committee nodes in layer ℓ. For any subcommittee election in such a committee node (which involves k_3 processors), Lemma 3.9 (with $n = k_2, k = k_3$) guarantees that with probability at least $1 - 1/k_2^c$, the subcommittee election succeeds, in the sense that the fraction of bad processors is increased by at most $1/\log k_2$. Thus, the expected number of unsuccessful subcommittee elections is at most r_ℓ/k_2^c. Furthermore, the random choices in all of these subcommittee elections are independent. Therefore, by Proposition 5.4 (1) with $y = c' r_\ell / \log k_2 = \Omega(\log\log n)$ and $\mu \leq r_\ell/k_2^c$, with probability at least $1 - 2^{\mu - y} \geq 1 - 2^{-y/2} = 1 - 1/\log^{c''} n$, at least a $1 - O(\frac{1}{\log k_2})$ fraction of the subcommittee elections are successful. The exponent c'' can be increased arbitrarily by increasing c'.

For layer $\ell = \ell^* - 2$, the number of committee nodes is $r_\ell \geq c_1 \log\log n$. In order to have at most a ϕ (a suitably small constant to be defined below) fraction of subcommittee elections failing, we apply Proposition 5.4 (1) with $y = \phi r_\ell \geq \phi c_1 \log\log n$, and $\mu \leq r_\ell/k_2^c$. Then, with probability at least $1 - 2^{-y/2} = 1/\log^{c''} n$, at least a $1 - \phi$ fraction of the subcommittee elections are successful. By increasing c_1, we can make c'' as large as desired.

For layer $\ell = \ell^* - 1$, the number of committee nodes is $r_\ell \geq c_1 \frac{\log\log n}{\log\log\log n}$. Again, our goal is to have at most a ϕ fraction of subcommittee elections failing. We apply Proposition 5.4 (2) with $y = \phi r_\ell$ and $\mu \leq r_\ell/k_2^c$. Then, the upper bound on the failure probability is $e^{-\frac{1}{2}\phi r_\ell \log(\phi k_2^c)} = 1/\log^{c''} n$ for some c'' that can be made arbitrarily large by increasing c_1. (Notice that we used here that $\log(\phi k_2^c) = \Theta(\log\log\log n)$.)

By taking a union bound over all of the $O(\log n)$ layers, we obtain that with probability at least $1 - 1/\log^{c''} n$, the failure rate in all layers $\ell \leq \ell^* - 3$ is at most $1/\log k_2$, and in layers $\ell = \ell^* - 2, \ell^* - 1$, it is at most ϕ.

By redefining α_ℓ (in Section 4.3) appropriately, accounting for the $1/\log k_2$ fraction of failed subcommittee elections in each layer, we can now essentially redo the proof of Lemma 4.6. This proves that the fraction of bad processor nodes in layer $\ell^* - 3$ is at most $(\frac{1}{3} - \epsilon) + o(1)$.

For the next two layers, we set $\phi = \frac{\epsilon}{6}$. For each committee node C in layer $\ell^* - 1$, the elected subcommittee is determined by the modified $(\epsilon/6, r^{-a})$-sampler $H' : [r] \to [k_2]^d$ (where r and a are defined in Section 3.3). Each of the two layers $\ell^* - 1, \ell^* - 2$ contains at most an $\epsilon/6$ fraction of bad committee nodes (causing a corresponding increase in bad processor nodes), and the sampler increases the fraction of bad processor nodes by at most another $\epsilon/6$. Thus, the fraction of bad processor nodes in layer ℓ^* is at most $\frac{1}{3} - \epsilon + \epsilon/2 + o(1)$.

Recall that the sampler H' has degree $d = O(\log\log\log n)$, i.e., it outputs fewer processors than the sampler H used in other layers. Because there were $\Theta(\log\log n / \log\log\log n)$ committees on level $\ell^* - 1$, the total number of processor nodes on level ℓ^* is $O(\log\log n)$. Thus, we obtain a set of $O(\log\log n)$ processor nodes in layer ℓ^*, of which at most a fraction $\frac{1}{3} - \epsilon/2 + o(1)$ are bad.

By Proposition 3.2, with probability at least $1 - 1/\log^{c''} n$ (for any constant c''), the output of the final invocation of Heavy-BA on the $O(\log\log n)$ processors in the lone committee of layer ℓ^* satisfies all requirements. A union bound over all layers now completes the proof.

(2) At the level $\rho = 2$ of the recursion, there are $k_1 = \Theta(\log^8 n)$ nodes, and subcommittees have size $k_2 = \Theta((\log\log n)^8)$. Each invocation of ELECT-SUBCOMMITTEE runs for $O(k_2)$ rounds, requiring Byzantine agreement for each of $O(k_2)$ bits of each of $O(k_2)$ messages sent in that round in order to simulate broadcast. Thus, each ELECT-SUBCOMMITTEE invocation requires $O(k_2^3)$ executions of Byzantine agreement. The number of committee nodes in each layer is $O(k_1/\log k_1)$, and the number of layers is $O(\log k_1)$. Thus, the total number of invocations of CONSENSUS$(3, C, \Sigma)$ is $O(k_1 k_2^3) = O(\log^8 n(\log\log n)^{24}) = O(\log^9 n)$. Since each of these fails with probability at most $1/\log^{c'} n$ for any c' by the previous case, a union bound over all of the calls (setting $c' = c + 10$) gives us that with probability at least $1 - 1/\log^{c+1} n$, none of the calls to CONSENSUS$(3, C, \Sigma)$ fails.
Under the condition that none of the Byzantine agreement calls fail, we can now mimic the analysis in the proof of Lemma 4.6 and Theorem 4.4. This analysis guarantees that with high probability (at least $1 - 1/k_1^{c''} = 1 - 1/\log^{8c''} n$ for any c''), the Byzantine agreement at level $\rho = 2$ succeeds. A union bound over the success of all calls to CONSENSUS$(3, C, \Sigma)$ and the remainder of the execution now completes the proof for $\rho = 2$.

(3) At the level $\rho = 1$, subcommittees have size $k_1 = \Theta(\log^8 n)$, and similar to the case of $\rho = 2$, each execution of ELECT-SUBCOMMITTEE requires $O(k_1^3) = O(\log^{24} n)$ correct executions of CONSENSUS$(2, C, \Sigma)$. Because each such execution is correct with probability at least $1 - \log^{c''} n$ for any c'' (by the analysis for $\rho = 2$), each subcommittee election succeeds with probability at least $1 - \log^{c'} n$ for any c' (setting $c'' = c' + 24$ and taking a union bound).
However, since there are now $\Theta(n)$ such subcommittee elections, a simple union bound over all of them (as in the case $\rho = 2$) does not work any more. Instead, we use Chernoff Bounds much like in the analysis for $\rho = 3$. Consider any layer ℓ of the network, with $r_\ell = \Omega(\log^4 n)$ committee nodes. The expected number of unsuccessful subcommittee elections is at most $r_\ell/\log^{c'} n$ for any c'. By Proposition 5.4 (1) with $y = r_\ell/\log n = \Omega(\log^3 n)$ and $\mu \leq r_\ell/\log^2 n$, with probability at least $1 - 2^{\mu - y} \geq 1 - 2^{-y/2} \geq 1 - 1/n^c$ (for any c), at least a $1 - 1/\log n$ fraction of the subcommittee elections are successful. Because the number of levels is $O(\log n)$, a union bound guarantees that with probability at least $1 - 1/n^c$, the fraction of failing subcommittee elections in each layer is at most $1/\log n$.
Under the event that these fractions of failing subcommittee elections are sufficiently small, we can again (like in the proof for $\rho = 3$) redefine the α_ℓ appropriately to account for the $1/\log n$ fraction of failed subcommittee elections in each layer. Then, we can essentially repeat the proof of Lemma 4.6, to prove that the fraction of bad processor nodes in layer ℓ^* is at most $(\frac{1}{3} - \epsilon) + o(1)$.
Finally, by the analysis for the case $\rho = 2$, the final execution of the protocol CONSENSUS$(2, C^*, \Sigma)$ for the actual Byzantine agreement succeeds with probability at least $1 - \log^c n$ for any c, and a union bound over all possible failures now concludes the proof of the theorem.

■

5.2 Time Analysis

By Lemma 3.9, at the recursion level $\rho = 3$, each (successful) subcommittee election takes time $O(k_3 2^{k_3} \log k_2)$. There are $O(\log k_2)$ layers of such subcommittee elections, giving us that layers $1, \ldots, \ell^* - 1$ together take time at most $O(k_3 2^{k_3} \log^2 k_2) = O(\log n)$. The final Byzantine Agreement in the committee node C^* takes time $O(2^{\log \log n}) = O(\log n)$. Therefore, Consensus$(3, C, \Sigma)$ overall takes time $O(\log n)$.

At the recursion level $\rho = 2$, the subcommittee election in any committee node C involves at most $O(k_2^3) = O((\log \log n)^{24})$ calls to Consensus$(3, C, \Sigma)$, each taking time $O(\log n)$. In the node C^*, we are now using another call to Consensus$(3, C^*, \Sigma)$, taking at most $O(\log n)$. The number of layers is $O(\log k_1) = O(\log \log n)$, so the total time is $O(\log n (\log \log n)^{25}) = O(\log^2 n)$.

Finally, at recursion level $\rho = 1$, we have at most $O(k_1^3) = O(\log^{24} n)$ calls to Consensus$(3, C, \Sigma)$, each taking time $O(\log^2 n)$. Since there are $O(\log n)$ layers, the total time is at most $O(\log^{27} n)$, i.e., polylogarithmic. Thus, we have proved:

Theorem 5.5. *The protocol* Consensus *takes polylogarithmic time in the number of processors n.*

6. UNIVERSE REDUCTION AND LEADER ELECTION PROTOCOLS

We now show how to leverage the protocols from Sections 4 and 5 to give a protocol for *leader election* with constant success probability and polylogarithmic time. Recall that in the leader election problem, all good processors must output a common good processor p.

6.1 Universe Reduction

First, observe that the Quasi-Poly-BA and Consensus protocols implicitly provide a protocol for *Universe Reduction*: significantly reducing the number of processors under consideration while hardly increasing the fraction of bad processors. Once the members of C_1^* are determined at level $\rho = 1$, instead of actually performing Byzantine agreement in C_1^*, we do the same at level $\rho = 2$. That is, we determine the members of C_2^* at that level, and instead of performing Byzantine agreement with them, also modify Consensus at level $\rho = 3$ to simply identify the members of C_3^*. The result is a committee of size $O(\log \log n)$. Using the proof in Section 4, we obtain the following properties for the resulting protocol Polylog-Universe-Reduction:

Theorem 6.1. *Let ϵ' be any positive constant. Then, there is a positive constant ϵ such that the following holds when* Polylog-Universe-Reduction *is given as input a set of n processors of which at most a $\frac{1}{3} - \epsilon'$ fraction is bad:*

(*1*) Polylog-Universe-Reduction *runs in time polylogarithmic in n.*

(*2*) *With high probability, all good processors p will output the same set S of $k = O(\log \log n)$ processors such that the fraction of bad processors in S is at most $\frac{1}{3} - \epsilon$.*

6.2 Leader Election

The idea of our leader election protocol Polylog-Leader is to first run Polylog-Universe-Reduction to reduce the universe size to $O(\log\log n)$. Then, using $O(\log\log\log\log n)$ invocations of Elect-Subcommittee, the universe size is further reduced to a suitable constant K, while still maintaining a fraction of at most $\frac{1}{3}-\epsilon$ bad processors, with constant probability. Each of the remaining K processors picks a random leader. Using the Heavy-BA protocol, the committee then agrees on the identity of the leader, and all good processors in the committee send a message with the identity of the leader to all other processors.

We start out with $k_0 = O(\log\log n)$ processors. For each i, we define $n_i = e^{k_i^{1/6}}$, and run Elect-Subcommittee on the committee of size k_i to output a committee of size $k_{i+1} = \Theta(\log^3 n_i) = \Theta(\sqrt{k_i})$. Because the committees all have size $O(\log\log n)$, we can use Heavy-BA within Elect-Subcommittee, and Lemma 3.9 guarantees that with high probability $1-1/n_i^c$, at most a $1/3-\epsilon+1/\log n_i$ fraction of the processors in the subcommittee are bad. In this way, after $i^* = O(\log\log\log\log n)$ iterations, $k_{i^*} = K = O(1)$, and we can choose i^* to make K as large as we wish. Clearly, the total running time is polylogarithmic in n.

We bound the probability of failure over all iterations. The probability of failure in each iteration is at most $1/n_i$. Taking the union bound gives an overall probability of failure of no more than $\sum_{i=0}^{i^*} 1/n_i$. For sufficiently large n, we have $k_i \leq n_i$, so the sum is bounded by

$$\sum_{i=0}^{i^*} 1/k_i \;\leq\; \sum_{i=0}^{i^*} 1/K^{2^{i^*-i}} \;=\; \sum_{i=0}^{i^*} K^{-2^i} \;\leq\; \sum_{i=0}^{\infty} K^{-2^i} \;\leq\; \sum_{i=0}^{\infty} K^{-i} \;=\; \frac{1}{K-1},$$

giving a constant nonzero probability of success.

If all the subcommittee elections are successful, then the increase in the fraction of bad nodes for iteration i is at most $1/\log n_i = 1/k_i^{1/6}$. Thus, the total increase is at most

$$\sum_{i=0}^{i^*} 1/k_i^{1/6} \;\leq\; \sum_{i=0}^{i^*} K^{-2^i/6} \;\leq\; \sum_{i=0}^{\infty} K^{-i/6} \;=\; \frac{1}{K^{1/6}-1}.$$

By choosing K sufficiently large, we can ensure that this is at most $\epsilon/2$, so the fraction of bad nodes among the final K is at most $\frac{1}{3}-\epsilon/2$.

Finally, given that all the iterations are successful, we have a committee of constant size K of which at least a $(2/3+\epsilon/2)$ fraction are good and known to each other. The probability that each good processor outputs the same good processor after running Heavy-BA is at least the probability that a $(\frac{1}{2}+\epsilon/2)$ fraction of the processors randomly chose the same good processor, which is clearly a nonzero constant. The random choice of a leader and the running of Heavy-BA are independent of any random choices made in the previous phase. Overall, Polylog-Leader has a probability of success which is a product of nonzero constants and we have:

Theorem 6.2. *For any positive constant ϵ, if at most a fraction $\frac{1}{3}-\epsilon$ of processors are bad,* Polylog-Leader *ensures that for some good processor p, with constant nonzero probability, all good processors will output p in time polylogarithmic in n.*

7. CONCLUSIONS AND OPEN PROBLEMS

We have demonstrated that the assumption of an asynchronous full information model does not substantially affect the ability to perform distributed computation, if one can assume a non-adaptive adversary and can tolerate a small probability of failure.

Numerous problems remain. We think that our protocols may be made scalable if almost-everywhere agreement is sought. Even for the synchronous model with private channels, it is not known whether a scalable protocol is possible if everywhere agreement is required.

Can the Byzantine agreement protocol be made Las Vegas? Can the running time-probability of failure tradeoff be improved to match the lower bounds? Can the probability of successful leader election be brought closer to the known upper bound? Finally, it is still open if a subexponential time protocol for Byzantine agreement is possible in the asynchronous full information model with an adaptive adversary.

ACKNOWLEDGMENTS

We would like to thank two anonymous reviewers for helpful and constructive comments, and for bringing to our attention several references.

REFERENCES

Ajtai, M. and Linial, N. 1993. The influence of large coalitions. *Combinatorica 13,* 2, 129–145.

Aspnes, J. 1998. Lower bounds for distributed coin-flipping and randomized consensus. *Journal of the ACM 45,* 3, 415–450.

Attiya, H. and Censor, K. 2007. Tight bounds for asynchronous randomized consensus. In *Proc. 38th ACM Symp. on Theory of Computing.* ACM Press, New York, NY, USA, 155–164.

Attiya, H. and Censor, K. 2008. Lower bounds for randomized consensus under a weak adversary. In *Proc. 27th ACM Symp. on Principles of Distributed Computing.* 315–324.

Ben-Or, M. 1983. Another advantage of free choice: Completely asynchronous agreement protocols. In *Proc. 2nd ACM Symp. on Principles of Distributed Computing.* 27–30.

Ben-Or, M. and El-Yaniv, R. 2003. Resilient-optimal interactive consistency in constant time. *Distributed Computing 16,* 4, 249–262.

Ben-Or, M. and Linial, N. 1989. Collective coin flipping. In *Advances in Computing Research 5: Randomness and Computation*, S. Micali, Ed. JAI Press, 91–115.

Ben-Or, M., Pavlov, E., and Vaikuntanathan, V. 2006. Byzantine agreement in the full-information model in $o(\log n)$ rounds. In *Proc. 37th ACM Symp. on Theory of Computing.* 179–186.

Berman, P. and Garay, J. A. 1993. Randomized distributed agreement revisited. In *Digest of Papers: FTCS-23, The 23rd Annual International Symposium on Fault-Tolerant Computing.* 412–419.

Bracha, G. 1984. An asynchronous $[(n-1)/3]$-resilient consensus protocol. In *Proc. 3rd ACM Symp. on Principles of Distributed Computing.* 154–162.

Cachin, C., Kursawe, K., and Shoup, V. 2005. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. *J. Cryptology 18,* 3, 219–246.

Canetti, R. and Rabin, T. 1993. Fast asynchronous byzantine agreement with optimal resilience. In *Proc. 25th ACM Symp. on Theory of Computing.* 42–51.

Chor, B. and Dwork, C. 1989. Randomization in Byzantine agreement. In *Advances in Computing Research 5: Randomness and Computation*, S. Micali, Ed. JAI Press, 443–497.

Chor, B., Merritt, M., and Shmoys, D. B. 1989. Simple constant-time consensus protocols in realistic failure models. *Journal of the ACM 36,* 3, 591–614.

Feige, U. 1999. Noncryptographic selection protocols. In *Proc. 40th IEEE Symp. on Foundations of Computer Science*. 142–153.

Fischer, M. J., Lynch, N. A., and Paterson, M. 1983. Impossibility of distributed consensus with one faulty process. In *Proc. 2nd ACM Symp. on Principles of Database Systems*. 1–7.

Goldwasser, S., Pavlov, E., and Vaikuntanathan, V. 2006. Fault-tolerant distributed computing in full-information networks. In *Proc. 47th IEEE Symp. on Foundations of Computer Science*. 15–26.

Gradwohl, R., Vadhan, S. P., and Zuckerman, D. 2006. Random selection with an adversarial majority. In *Proc. 26th Intl. Cryptology Conference (CRYPTO)*. 409–426.

King, V., Saia, J., Sanwalani, V., and Vee, E. 2006a. Scalable leader election. In *Proc. 17th ACM Symp. on Discrete Algorithms*. 990–999.

King, V., Saia, J., Sanwalani, V., and Vee, E. 2006b. Towards secure and scalable computation in peer-to-peer networks. In *Proc. 47th IEEE Symp. on Foundations of Computer Science*. 87–98.

McDiarmid, C. 1998. Concentration. In *Probabilistic Methods for Algorithmic Discrete Mathematics*, M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, Eds. Springer, 195–247.

Nielsen, J. B. 2002. A threshold pseudorandom function construction and its applications. In *Proc. 22nd Intl. Cryptology Conference (CRYPTO)*. 401–416.

Ostrovsky, R., Rajagoplan, S., and Vazirani, U. 1994. Simple and efficient leader election in the full information model. In *Proc. 26th ACM Symp. on Theory of Computing*. 234–242.

Pease, M. C., Shostak, R. E., and Lamport, L. 1980. Reaching agreement in the presence of faults. *Journal of the ACM 27,* 2, 228–234.

Russell, A. and Zuckerman, D. 1998. Perfect information leader election in $\log^* n + o(1)$ rounds. In *Proc. 39th IEEE Symp. on Foundations of Computer Science*. 576–583.

Saks, M. 1989. A robust noncryptographic protocol for collective coin flipping. *SIAM J. Disc. Math. 2,* 2, 240–244.

Toueg, S. 1984. Randomized byzantine agreements. In *Proc. 3rd ACM Symp. on Principles of Distributed Computing*. 163–178.

Turpin, R. and Coan, B. A. 1984. Extending binary byzantine agreement to multivalued byzantine agreement. *Inf. Process. Lett. 18,* 2, 73–76.

Zuckerman, D. 1997. Randomness-optimal oblivious sampling. *Random Structures and Algorithms 11,* 4, 345–367.