Fast Asynchronous Byzantine Agreement and Leader Election with Full Information

Bruce Kapron * David Kempe † Valerie King ‡ Jared Saia §
Vishal Sanwalani ¶

Abstract

We resolve two long-standing open problems in distributed computation by showing that both Byzantine agreement and Leader Election can be solved in sub-exponential time in the asynchronous full information model. Surprisingly, our protocols for both problems run in only polylogarithmic time. We thus achieve a better than exponential speedup over previous results for asynchronous Byzantine agreement. In addition, to the best of our knowledge, ours is the *first* protocol for asynchronous full-information leader election. Our protocols work in the full information model with a non-adaptive adversary: the adversary is assumed to control up to a constant fraction of the processors, have unlimited computational power as well as access to all communications, but no access to processors' private random bits. The adversary is non-adaptive only in the sense that the corrupted processors must be chosen at the outset. Our protocols run in time that is polylogarithmic in the number of processors, n, and tolerate $t < \frac{n}{6+\epsilon}$ faulty processors for any positive constant ϵ. Our protocols are Monte Carlo, succeeding with probability $1 - o(1)$ for Byzantine agreement, and constant probability for leader election.

*Department of Computer Science, Stanford University and Department of Computer Science, University of Victoria; email: bmkapron@uvic.ca

†Department 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.

‡Microsoft Research SVC, Mountain View, CA and Department of Computer Science, University of Victoria, P.O. Box 3055, Victoria, BC, Canada V8W 3P6; email: val@uvic.ca

§Department 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.

¶Department of Computer Science, University of Victoria; email: vsanwalani@gmail.com.

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.

Our results hold for a very general model of computation, 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 an 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. Also, protocols which rely on cryptographic primitives may require sources of perfect randomness [13], whereas in the full information model, lower entropy sources of randomness have been shown to suffice. [17].

Asynchrony introduces fundamental difficulties into distributed protocol design; intuitively, protocols are unable to distinguish failed processors from delayed messages. Indeed, Fischer, Lynch and Patterson [15] showed that even 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 [4, 8] gave a randomized protocol (with private coins) 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 $o(\sqrt{n/\log n})$ time is impossible [3] with an adversary which can corrupt processors adaptively, even if the model is synchronous and the faults are fail-stop rather than malicious. (See also [2]) for improved lower bounds). Hence we look at a model where the adverary is *non-adaptive* in that it must choose the corrupt processors at the outset of the protocol.

In this paper, we give the first sub-exponential protocols for Byzantine agreement in the asynchronous full information model, with an adaptive or non-adaptive adversary with constant factor resiliance. In fact, our protocols take worst case time only poly-logarithmic (resp. quasi-poly-logarithmic) in n and succeed with $1 - 1/\ln^c n$ (resp. $1 - 1/n^c$) probability for any constant c.

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 $2^{\Theta(\log^7 n)}$, resilience $\frac{n}{6+\epsilon}$, and success probability $1 - 1/n^c$.*
- *running time polylogarithmic in n resilience $\frac{n}{6+\epsilon}$, and success probability $1 - 1/\ln^c n$.*

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 cannot exceed $1 - t/n$, as bad processors can just behave like good processors. Tighter upper bounds are given in [14]. 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 equal to $\frac{n}{6+\epsilon}$, and success probability that is a positive constant.*

The core technique for all our protocols is *asynchronous universe reduction*. Universe reduction consists in 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 in the initial n (a formal definition is given in Section 2).

Other related work: Leader election and global coin-tossing was considered in a series of papers [5, 1, 24, 22, 23] in a synchronous full information model with a non-adaptive adversary and an *atomic broadcast* primitive: each message sent (even by corrupt processors) is received identically by all processors. These papers culminated in Feige's $O(\log^* n)$ protocol for leader election [14], which we adapt and use here.

Without the atomic broadcast primitive, a polylogarithmic round randomized protocols against a non-adaptive adversary in the synchronous full information model were developed independently by King, Saia, Sanwalani, and Vee [19, 20] and by Ben-Or, Goldwasser, Pavlov and Vaikuntanathan [6, 16]. [19] presents protocol for both Byzantine agreement and leader election with resilience $\frac{n}{3+\epsilon}$ which is ""scalable"" in that each processor sends and processes a number of bits polylogarithmic in n. The protocol obtains "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 sending one bit to every other processor needed to obtain agreement among all good processors, in a worst case number of rounds which is polylogarithmic in n. [20] showed that the almost everywhere agreement protocol could be implemented with a scalable protocol on a sparse network in a worst case polylogarithmically many rounds. [6] and [16] give expected $O(\log n)$ round protocols with resilience $\frac{n}{4+\epsilon}$ and $\frac{n}{3+\epsilon}$, respectively.

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 [18] use a 2-stage process involving Feige's leader election protocol, followed by the application of a sampler which is similar to our subcommittee election and expansion procedure, to perform a selection with an adversarial majority.

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 [7, 10] or a computationally bounded adversary and cryptographic primitives [25], culminating in protocols with expected $O(1)$ round time with optimal resilience $\frac{n}{3}$ and $O(n^2)$ messages [9, 21]. But the tails of the running times can be high (see below).

The only known lower bound which applies to our work (and the protocols assuming private channels and cryptographic primitives) is obtained by slightly modifying the proof in [12]. That result shows that any randomized protocol which runs in r synchronous rounds in the fail-stop model with t processor failures has probability at least $1/2(2\lceil n/\lfloor t/r \rfloor)\rceil^{-r}$ of failing to terminate against an

adversary which is oblivious to the content of messages. The same proof goes through for protocols which fail to agree on the same value or fail to terminate. This is $1/n$ when $r = O(\log n/\log\log n)$ and $1/\log n$ when $r = O(\log\log n/\log\log\log n)$.

2 Preliminaries and Overview

We assume a fully connected network of n processors, among whom both n and all processors' IDs are common knowledge. Throughout, n will denote the total number of processors. 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.

Out of the n processors, up to t can be faulty. The model is an asynchronous full information model with a nonadaptive (sometimes called *static*) adversary. That is, the adversary chooses a 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 among bad processors, or crash failures.

In addition to controlling the behavior of bad processors, the adversary can view each message as soon as it is sent, and determine the message's delay as well as the order in which messages are received. Each message sent from a good processor to another good processor is received within some maximum finite time Δ, but the exact time is determined by the adversary. It is unknown during the execution, but the running time of a protocol is described in terms of this basic unit. Thus, a running time of $f(n)$ means that all good processors reach agreement by time $\Delta f(n)$. Alternately, $f(n)$ is the maximum length of any chain of messages [11].

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 $\ln n$ as an integer throughout.

2.1 Overview

As mentioned above, our protocols are based on a novel asynchronous protocol for universe reduction. The (θ, k)-universe reduction problem is defined 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 the universe. Implicit in our work are, to our knowledge, the first protocols for asynchronous universe reduction in the full information model. (See Appendix). To establish this result (and the main Theorems 1.1 and 1.2), we adapt the synchronous universe reduction protocol of King et al. [19] to the asynchronous communication model. The processors are divided into committees (sets) 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 [8] (exponential-time) probabilistic 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 the committee is repeatedly reduced until a constant number of processors remain using a variant of the subcommittee election protocol. 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 which all good processors then agree to.

The asynchronous nature of communication requires addressing several nontrivial technical issues to make this approach work: (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 which 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. This is the technically most challenging part of our paper.

The second issue is resolved in Section 4, using the network structure similar to that in [19] 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 randomlike properties.

Committees with too many bad processors or not enough good processors which are known to all processors may cause inconsistent views about which processors they elect. We show in Section 3.5 that the election protocol still works if enough processors in the committee are known to all the processors. We show in Section 4.1 that a sufficiently large number of processors are good and known to all processors on each layer and finally in the proof of Theorem **??** that there are at least 5/6 fraction of processors in the top committee which are good and known to all processors . We discuss in Section 4 how to put 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 more efficiency. 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 [11] and have alternatively been called expanders, dispersers and samplers. An exact correspondence between extractors and averaging samplers is given in [26].

DEFINITION 1. *Let $[r]$ denote the set of integers $\{1, \ldots, r\}$, and $[s]^d$ the subsets of $[s]$ of size d. Let $H : [r] \to [s]^d$ be a function assigning sets of size d to elements.*

1. *H is a $(\theta, \epsilon, \beta)$ sampler if for every set $S \subset [s]$ with $|S| > \beta s$,*

$$\left|\left\{x \mid \frac{|H(x) \cap S|}{d} > \frac{|S|}{s} + \theta\right\}\right| \leq \epsilon r$$

2. *H is a (θ, ϵ) sampler iff it is a $(\theta, \epsilon, 0)$ sampler.*

The following two lemmas, establishing the existence of samplers, can be shown using the probabilistic method. 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 of [18] and [27], at the expense of a polylogarithmic overhead in the overall running time of the protocol.

LEMMA 2.1. *For every $s, \theta, \epsilon > 0$ and $r \geq s/\epsilon$, there exists a constant c such that for all $d \geq c \log(1/\epsilon)/\theta^2)$, there is a (θ, ϵ) sampler $H : [r] \to [s]^d$.*

LEMMA 2.2. *For every $r, s, d, \theta, \epsilon, \beta > 0$ such that $(\log_2 e)(d\theta^2\beta r)/3 > s/\epsilon$, there exists a $(\theta, \epsilon, \beta)$ sampler $H : [r] \to [s]^d$.*

3 Subcommittee Election

We describe a protocol ELECT-SUBCOMMITTEE for asynchronously electing a subcommittee of processors from a committee of k processors where $\geq k - t$ are good. In Feige's protocol, each processor randomly selects a "bin number" from $\{1, ..., b\}$ and broadcasts it to the other processors. The subcommittee then consists of all processors which choose 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 the asynchronous communication model without broadcast, we note that when fewer than t 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) We need to simulate broadcast in order for the good processors to agree on the lightest bin. (2) 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) 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. We show below how to address these difficulties.

3.1 Broadcast simulation

To deal with the first problem, we note that if all k processors broadcast their bin numbers, the adversary can delay some good processors and prevent them from being heard by any processors. Hence we allow processors to agree on a null message $*$. We first note that Bracha's exponential time asynchronous Byzantine agreement protocol can easily be extended to the case where each processor has three possible input values $0, 1$ and $*$, by running Bracha's protocol [8] twice. We refer to the resulting protocol as HEAVY-BA (heavyweight Byzantine agreement.) The following properties follow directly from the corresponding properties of Bracha's protocol:

PROPOSITION 3.1. *The* HEAVY-BA *protocol has the following properties.*

1. HEAVY-BA *tolerates* $t < k/3$ *corrupt processors.*
2. *W.h.p., every processor terminates after* $\ln^2 n \cdot 2^k$ *rounds and is required to send and receive at most* $\ln^2 n \cdot 2^k$ *bits.*
3. *If more than* $2k/3$ *of the processors are both uncorrupted and have the same value* v'*, then upon termination, w.h.p., every good processor sets its value* $v = v'$*.*

HEAVY-BA can be composed to simulate multiple rounds of synchronous broadcast without significant loss of resilience. The high-level idea is to use HEAVY-BA on each bit of each message independently. If all processors agree on the value v of a message by a good processor, the message is *accepted*. If even one bit of the message becomes a $*$, the entire message is replaced by a string of $*$, and called *rejected*. Let m be (an upper bound on) the length of the longest message sent as part of the protocol. To perform a simulated broadcast, each processor p does the following:

1. p sends its message to all processors including itself, and then waits until it receives messages from $k - t$ processors.
2. p sets the values of all unreceived expected messages to $*^m$ and then participates in km parallel versions of HEAVY-BA to determine each bit of every message sent by every processor.

3. If any bit of an agreed-upon message is set to a $*$, p records the message as $*^m$.

4. p enters the next round only after committing to every message sent by every processor.

LEMMA 3.1. *Let ϵ be a positive constant and $t < \frac{k}{3(2+\epsilon)}$ the number of bad processors. Using the above protocol to simulate a sequence of $q(k) = \text{poly}(k)$ rounds of synchronous broadcast, w.h.p.:*

1. *For each round $i \leq q(k)$, all good processors agree on each message sent by each processor, or they agree that the message is rejected, within $O(i \ln^2 n \cdot 2^k)$ time.*

2. *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 at least an $\alpha = \epsilon/(1+\epsilon)$ fraction of the good processors have been accepted.*

Proof. The proof of items (1) and (2) by induction is straightforward. To prove part 3 we note that each good processor received $k - 2t$ messages from good processors and do an averaging argument over the total number of good messages received by good processors. Details are given in the appendix. ■

3.2 Making message delay (mostly) independent of bin choice.

To address problem (2), our election protocol is designed so that there is a large subset of good processors determined in some sense before their choices are made, whose bin choices cannot be delayed by the adversary. The lightest bin protocol is run for k rounds. Each good processor p maintains a $k \times k$ 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 table is (retroactively) updated if necessary, so 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 at some round $i'' \geq i$, and let A_i be the set of good processors whose messages for round i were accepted at the end of round i. Then $S_k \subseteq S_{k-1} \subseteq S_{k-2} \cdots \subseteq S_1$. By Lemma 3.1 (3), for all i, $|A_i| \geq \frac{\epsilon}{1+\epsilon} \cdot (k - t)$. Thus, by the Pigeonhole Principle and the fact that $|S_i| \geq |A_i|$, there is an r such that $S_r = S_{r-1}$; we fix the smallest such r. The 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 3.2. *Let c, ϵ be any positive constants and suppose there are k processors of which $t \leq \frac{k}{6+\epsilon}$ are bad. There is a constant c' such that if the number of bins is $b = k/(c' \ln n)$, then with probability $1 - 1/n^c$, the number of good processors in the lightest bin in round r (from A_{r-1}) is at least $\gamma \cdot c' \ln n$, where $\gamma = \frac{1}{2} \cdot \frac{\epsilon}{1+\epsilon} \cdot \frac{k-t}{k}$.*

Proof. Fix any bin B and round i. By Lemma 3.1 (2), when $p \in A_{i-1}$ chooses its bin in round i, the messages for all processors in A_{i-1} for round $i - 1$ were 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 $Pr(x_j = 1) = 1/b$, and $E[X] = |A_i|/b \geq \frac{\epsilon}{1+\epsilon}(k - t)/b$. Applying Chernoff bounds gives $\Pr(X < E[X]/2) \leq n^{-(c+2)}$, for $c' \geq \frac{48(c+2)(1+\epsilon)}{5\epsilon}$. By a union bound over all rounds $i = 1, \ldots, k, k \leq n$ and all $b \leq n$ bins, the probability that any bin B is chosen in round i by fewer than $\frac{1}{2}\frac{\epsilon}{1+\epsilon}\frac{k-t}{b}$ processors in A_{i-1} is at most n^{-c}.

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 $\frac{1}{2}\frac{\epsilon}{1+\epsilon}\frac{k-t}{b}$ processors. ■

3.3 Enlarging the fraction of good processors.

To address problem (3), the set of processors in the lightest bin are used to select a better subcommittee w.h.p. 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 x number 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 and $r = |R| < (xk/b)2^{xk/b}$, where γ is defined in Lemma 3.2. By Lemma 2.1, there is a $(1/\ln n, r^{-a})$ 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 $O(\log^3 n)$.

LEMMA 3.3. *Assume that $k = \ln^c n$ for a constant $c > 3$. W.h.p., the subcommittee produced by H contains less than a $t/k + 1/\ln n$ fraction of bad processors.*

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 $t/k + 1/\ln n$ fraction of bad processors. Fix a bad input ρ. By Lemma 3.2, 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 $1/2^{x\gamma k/b}$. By the definition of a sampler, there are at most r^{1-a} bad inputs. Thus, by choosing $a = 1 - c'\gamma/2$, we have $r^{1-a} = 2^{x\gamma k/2b}$. Taking the union over all possible bins and round numbers for the lightest bin and all possible subsequence of blocks, and the different bad inputs, the probability of the lightest bin generating a bad input is given by $2^{x\gamma k/2b}(kb)(1/2^{-x\gamma k/b}) < n^{-c''}$ for every positive constant c'', and a sufficiently large x. ■

3.4 Protocol and Proof.

We now obtain the following protocol ELECT-SUBCOMMITTEE, run at each processor p in committee $C = \{p_1, \ldots, p_k\}$ of processors, with $k > \ln^3 n$.

Protocol 1 The ELECT-SUBCOMMITTEE protocol on good processor p

1: **for** $i = 1$ to k **do**
2: Randomly select a bin number $B_i^{(p)} \in \{1, 2, \ldots, b\}$, where $b = k/(c' \ln n)$ and a random bit string $X_i^{(p)}$ of length x.
3: Send $((B_1^{(p)}, X_1^{(p)}), \ldots, (B_i^{(p)}, X_i^{(p)}))$ to every processor in C.
4: Wait to receive messages for round i from $k - t$ processors. Set the value of all unreceived expected messages to $*^m$, where m is the length of the expected message. Then run in parallel HEAVY-BA to agree on each bit of the message of every processor.
5: Update all $(B_{i'}^{(p')}, X_{i'}^{(p')})$ for $i' \leq i$.
6: Let S_i be the set of processors p' for whom $B_i^{(p')}$ (and $X_i^{(p')}$) are known. Let r be smallest such that $S_r = S_{r-1}$.
7: Let B be the lightest bin in round r, and ρ be the concatenation, ordered by p', of the blocks $\{X_r^{(p')} \mid B_r^{(p')} = B\}$ chosen by the set of processors in bin B in round r.
8: Return $H(\rho)$ as the elected subcommittee.

LEMMA 3.4. *For $\epsilon > 0$, if a committee of k processors has $t < \frac{k}{6+\epsilon}$ bad processors, then w.h.p.*

1. ELECT-SUBCOMMITTEE *runs in time* $O(k \log^2 k 2^k)$.

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

3. *The elected subcommittee contains* $O(\log^3 n)$ *processors, of which no more than a* $t/k + 1/\ln n$ *fraction are bad.*

3.5 Extension: When processors have differing views.

In later sections, we will want to run the subcommittee election procedure even when the committee members have differing views of the membership of the committee. The members of the committee are represented by the (ordered) set of identifiers $I = \{1, 2, ..., k\}$, while the set of all processors is $P = \{1, 2, ..., n\}$. Each processor p_i maintains a *view* $v_i : I \to P$ of the committee. The input to H is then a string whose blocks are ordered by identifiers in I and the output is a subset of size $\Theta(\ln^3 n)$ of I. A *core* of a committee is a set R of good processors such that every $p \in R$ believes it is in the committee and all other processors in R agree on p's identifier. If a processor p_i receives a message from a processor p_j such that $p_j \notin v_i(I)$, it is disregarded. The following observation then follows from Lemma 3.4:

OBSERVATION 1. *With high probability, in any committee with a core set* R*, such that* $|R| > k(5/6 + \epsilon)$ *for some fixed* $\epsilon > 0$*, every* $p \in R$ *will come to agreement on the identifiers of the processors in the lightest bin* B*, and consequently on the actual identities of those processors in the lightest bin which come from* R*.*

4 A Quasi-Polynomial Asynchronous Byzantine Agreement Protocol

We describe the protocol in terms of an ordered layered network. Each processor has a copy of the same network which is either specified for a given size of the network n at the start of the protocol or generated in polynomial time.

Structure of the network: Let ℓ^* be the minimum integer ℓ such that $n/\ln^\ell n \leq \ln^7 n$; note that $\ell^* = O(\ln n / \ln \ln n)$. Let $r_\ell = n/\ln^{\ell+4} n$ and $s_\ell = n/\ln^\ell n$. The layers of the network are numbered $0(a), 0(b), 1(a), 1(b), ..., \ell^*(a)\ell^*(b)$. Layers $\ell(a)$ and $\ell(b)$ contain the set of s_ℓ *processor nodes* $\mathcal{P}_\ell$ and r_ℓ *committee nodes* $\mathcal{C}_\ell$ resp. Layer $\ell^*(b)$ has a single node C^*.

For $\ell < \ell^*$, the edges between $\ell(a)$ and $\ell(b)$ are determined by the $(1/\ln n, 1/(2\ln n), 5/6)$ sampler H_ℓ where $r = r_\ell$ and $s = s_\ell$ and $d = O(\log^7 n)$. There is an edge from the i^{th} node in $\mathcal{P}_\ell$ to the j^{th} node in $\mathcal{C}_\ell$ iff $i \in H_\ell(j)$. There is an ordered set of $O(\log^3 n)$ edges from each node in $\mathcal{C}_\ell$ to nodes in $\mathcal{P}_{\ell+1}$ so that each edge is incident to exactly one such node in $\mathcal{P}_{\ell+1}$. There is an edge to C^* from every node in $\mathcal{P}_{\ell^*}$.

The interpretation of the network: The layered network describes a protocol in which initially every processor is assigned to a node in $\mathcal{P}_0$ and where each committee node represents a subcommittee election subroutine involving processors associated with its predecessor (processor) nodes. The elected subcommittee is then associated with the committee node's successor (processor) nodes. Due to message loss and the adversary's choices, processors might not agree on which processor is associated with a given processor node node. 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 the ordered list of C's predecessors. Each processor p will update its views throughout the protocol.

Each message sent between processors is sent with the name of an associated node C. In all stages but the last, these messages will be for the subcommittee election; in the last stage, they

will be for the HEAVY-BA protocol. 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 a committee node C, a processor p requires that there be "enough" processors in its view of the committee node. In a slight abuse of notation, we write $|v_p(C)|$ for the number of actual processors (i.e., non-$\perp$ entries) in p's view of C.

When $|v_p(C)|$ is at least a fraction $1-2/\ln n$ of the total number of edges into C and $p \in v_p(C)$, p sends out its view of C, $v_p(C)$, to all other processors in its view of C. Once p receives views of C from at least $2/3k$ processors in $v_p(C)$, p revises its view to agree with the majority and sets $v_p(C)$ to `ready` (we denote this view by $v_p^r(C)$), p runs subcommittee election if p is in $v_p(C)$ or else waits to hear from other processors about the results. $v_p(C)$ is considered `fixed` afterward the election is run. A formal description of the protocol is given in the Appendix.

4.1 Proof of Correctness

We will prove the following in this section:

THEOREM 4.1. *W.h.p., the protocol* QUASI-POLY-BA *terminates in quasi-polynomial time and achieves Byzantine agreement.*

First, we prove a lemma which show that there are committees which have large enough *cores*, subsets of good processors which whose identities are known to each other and which all processors eventually agree on. The second lemma shows that there are enough of these committees on each layer.

Let k be the number of predecessor nodes of any committee node, i.e., the size of a committee. Let $\alpha_\ell = 5/6 + \epsilon - 22\ell/\log n$. A processor node $P \in \mathcal{P}_0$ is *good* if there is a good processor assigned to this node. A committee node $C \in \mathcal{C}_\ell$ for $\ell \geq 0$ is *good* if at least a $(1 - 3/\ln n)$ fraction of its predecessor nodes are successors of good committee nodes and at least an $(\alpha_\ell - 1/\ln n)$ fraction of its predecessors are good. A processor node $P \in \mathcal{P}_\ell$ for $\ell > 0$ is good if all its predecessor (committee and processor) nodes are good.

Proofs of the following lemmas appear in the Appendix.

LEMMA 4.1. *Let P and C be any good processor and good committee node in $\mathcal{P}_\ell$ and $\mathcal{C}_\ell$, resp. Then w.h.p. the following are true:*
(1) By time $O(k \log^2 k 2^k \ell)$, there is a good processor q such that for all processors p, $v_p(P) = q$.
(2) By time $O(k \log^2 k 2^k \ell)$ there is a subset (or core*) of good processors $S = \bigcap_{p \in S} v_p^r(C)$ and $|S| \geq (\alpha_\ell - 19/\ln n)k$, which come from good processor nodes (i.e., for all $q \in S$, there is a good processor node P' which is a predecessor of C such that $v_q(P') = q$.)*
(3) No other view of good processor nodes besides $\perp$ is ever held by any processor.

Lemma 4.1 is shown by induction on ℓ. If there is a sufficiently large core of good processors who have each other in their view of a committee node C then by Observation 1, the election can be run so that a sufficiently large fraction of good nodes are elected from this core. We use an averaging argument to show that for the next layer the processor in the core may each be missing a different $3/\log n$ fraction from their view of good predecessor nodes, but hearing from enough processors in their views of C, a core sufficiently large is mutually agreed upon.

The next lemma is proved by induction using the properties of samplers of the network.

LEMMA 4.2. *With high probability, for all $\ell \leq \ell^*$, by time $O(k \log^2 k 2^k \ell)$, the fraction of good processor nodes in level $\mathcal{P}_\ell$ is at least α_ℓ, and the fraction of good committee nodes in level $\mathcal{C}_\ell$ is at least $1 - 2/\ln n$.*

Proof of Theorem: Because $\ell^* = o(\ln n)$, Lemma 4.2 implies that w.h.p., the committee C^* constituting $\mathcal{C}_{\ell^*}$ is good. Thus, by Lemma 4.1 has a core of size at least $(\alpha_{\ell^*} - 19/\ln n)k$. Since

the core comes from good processor nodes, these are good processors of which *all* processors in the network have the same view. But $(\alpha_{\ell^*} - 2/\ln n) > 5/6$ which more than suffices to run Heavy-BA. As each processor waits to hear from at least a 5/6 fraction of processors in C^*, w.h.p., all good processors will reach Byzantine agreement in time $O(\ell^* k \log^2 k 2^k)$. Thus, w.h.p., Quasi-Poly-BA terminates correctly in quasi-polynomial time.

5 The Polylogarithmic Time Protocol

We create a new algorithm Polylog-BA by modifying Quasi-Poly-BA so that there are two levels of recursion. That is, each call to Heavy-BA on committees of size $O(\log^7 n)$ is replaced by a call to Quasi-Poly-BA. In the second level of recursion, each call to Heavy-BA on committees of size $O((\log\log)^7 n)$ is replaced by a call to M-Quasi-Poly-BA. M-Quasi-Poly-BA is simply Quasi-Poly-BA, except that the number of layers ℓ^* is reduced so that on layer $\ell^* - 1$, there are $O(\log\log n/\log\log\log n)$ committee nodes, and the committee nodes in layer $\ell^* - 1$ use a less accurate sampler (with smaller degree). Finally, on the next level of recursion, the call to Heavy-BA where committees are of size $O((\log\log\log)^7 n)$ (and on layer ℓ^* of size $O(\log\log n)$) is actually executed by calling Heavy-BA.

There are two observations which allows this amount of recursion (and no more). We can afford to allow the calls to Quasi-Poly-BA to fail with higher probability. Since the probability of this failure is independent, we use Chernoff bounds to show that a sufficient number of calls on each layer do not fail and this suffices. The second observation is that while we can't use such reasoning for the top level committee's election (as there's only one), we can afford to reduce by a constant the fraction of good processors produced by that committee's election as this happens only once. A more detailed version is in the appendix.

6 Leader election

The *leader election* problem requires that all good processors output a common good processor p. We achieve leader election by first reducing the number of processors to the number of processors (k_0) in the top node C^* using Polylog-BA. The processors in that committee then undergo a series of rounds in which the number of candidate leaders is halved at each step (using the subcommittee election protocol with different parameter settings) while reducing the fraction of good processors by no more than $1/\ln^2 k_i$, where k_i is the current number of participants. Each such round requires use of Polylog-BA, and successfully completes with probability at least $1 - 1/k_i - 1/\ln^c k_i$. After $\log k_0 - c$ rounds for some constant c, a committee of constant size 2^c with $5/6 + \epsilon$ fraction of good processors are left. At this point, each of these processors randomly picks a leader, and with probability greater than $1/2^{2^c}$, the leader is good. The committee runs Polylog-BA to agree on the identity of the leader, and then every processor in the committee sends messages with the identity of the leader to all other processors. More details are in the Appendix.

7 Conclusions and Open Problems

We have demonstrated that the assumption of an asynchronous 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 resilience of the Byzantine agreement protocol be improved to the known lower bound

of $n/3$, or the success probability to 1? Can the running time be brought down to $O(\log n)$ or lower? Can the [12] lower bound be increased for the asynchronous model? It would be desirable to bring the probability of successful leader election 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.

References

[1] Miklós Ajtai and Nathan Linial. The influence of large coalitions. *Combinatorica*, 13(2):129–145, 1993.

[2] Hagit Attiya and Keren Censor. Tight bounds for asynchronous randomized consensus. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 155–164, New York, NY, USA, 2007. ACM Press.

[3] Ziv Bar-Joseph and Michael Ben-Or. A tight lower bound for randomized synchronous consensus. In *Proceedings of the 17th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 193–199, 1998.

[4] Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement prototols. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 27–30, 1983.

[5] Michael Ben-Or and Nathan Linial. Collective coin flipping. In Silvio Micali, editor, *Advances in Computing Research 5: Randomness and Computation*, pages 91–115. JAI Press, 1989.

[6] Michael Ben-Or, Elan Pavlov, and Vinod Vaikuntanathan. Byzantine agreement in the full-information model in O(log n) rounds. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computin (STOC)*, pages 179–186, 2006.

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

[8] Gabriel Bracha. An asynchronous $[(n-1)/3]$-resilient consensus protocol. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 154–162, 1984.

[9] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. *J. Cryptology*, 18(3):219–246, 2005.

[10] Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 42–51, 1993.

[11] Benny Chor and Cynthia Dwork. Randomization in Byzantine agreement. In Silvio Micali, editor, *Advances in Computing Research 5: Randomness and Computation*, pages 443–497. JAI Press, 1989.

[12] Benny Chor, Michael Merritt, and David B. Shmoys. Simple constant-time consensus protocols in realistic failure models. *J. ACM*, 36(3):591–614, 1989.

[13] Yevgeniy Dodis, Shien Jin Ong, Manoj Prabhakaran, and Amit Sahai. On the (im)possibility of cryptography with imperfect randomness. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 196–205, 2004.

[14] Uriel Feige. Noncryptographic selection protocols. In *Proceedings of 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 142–153, 1999.

[15] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. In *Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database System (PODS)*, pages 1–7, 1983.

[16] Shafi Goldwasser, Elan Pavlov, and Vinod Vaikuntanathan. Fault-tolerant distributed computing in full-information networks. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 15–26, 2006.

[17] Shafi Goldwasser, Madhu Sudan, and Vinod Vaikuntanathan. Distributed computing with imperfect randomness. In *Distributed Computing, 19th International Conference, DISC 2005, Cracow, Poland, September 26-29, 2005, Proceedings*, volume 3724 of *Lecture Notes in Computer Science*, pages 288–302. Springer, 2005.

[18] Ronen Gradwohl, Salil P. Vadhan, and David Zuckerman. Random selection with an adversarial

majority. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference*, volume 4117 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

[19] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *Proceedings of 27th ACM-SIAM Symposium on Discrete Algorithms(SODA)*, pages 990–999, 2006.

[20] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Towards secure and scalable computation in peer-to-peer networks. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 87–98, 2006.

[21] Jesper Buus Nielsen. A threshold pseudorandom function construction and its applications. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 401–416. Springer, 2002.

[22] R. Ostrovsky, S. Rajagoplan, and U. Vazirani. Simple and efficient leader election in the full information model. In *Proceedings of the Twenty-Sixth Annual ACM Syposium on Theory of Computing (STOC)*, pages 234–242, 1994.

[23] A. Russell and D. Zuckerman. Perfect information leader election in $\log^* n + O(1)$ rounds. In *Proceedings of 39th Annual Symposium on Foundations of Computer Science(FOCS)*, 1998.

[24] Michael Saks. A robust noncryptographic protocol for collective coin flipping. *SIAM Journal of Discrete Mathematics*, pages 240–244, 1989.

[25] Sam Toueg. Randomized byzantine agreements. In *Proceedings of the Third Annual ACM Symposium on Princiles of Distributed Computing (PODC)*, pages 163–178, 1984.

[26] David Zuckerman. Randomness-optimal oblivious sampling. *Random Struct. Algorithms*, 11(4):345–367, 1997.

[27] David Zuckerman. Personal communication, 2006.

Appendix

Proof of Lemma 3.1 (3)

Consider one round, and the number of messages from good processors received by good processors. At least $k-2t$ good processors must have had their messages received by all $k-t$ good processors, for a total of $(k-2t)(k-t)$. On the other hand, each message was (trivially) received by at most $k-t$ good processors. Furthermore, if a message was rejected, then by Property (3) of Proposition 3.1, it was received by strictly fewer than $2k/3$ good processors. By the bound on t, $2k/3 \leq k-(2+\epsilon)t$. Hence, if a total of A messages were accepted, the total number of messages received was at most $A(k-t)+(k-t-A)(k-(2+\epsilon)t) = (k-t)(k-(2+\epsilon)t)+A(1+\epsilon)t$. Since the total number of messages received is also at least $(k-2t)(k-t)$, we obtain that $(k-t)(k-(2+\epsilon)t)+A(1+\epsilon)t \geq (k-2t)(k-t)$, or $A \geq \epsilon/(1+\epsilon) \cdot (k-t)$.

Proof of Lemma 4.1

The statement is clearly true for $\ell = 0$.

Assume it is true for $\mathcal{C}_{\ell-1}$ and $\mathcal{P}_\ell$. We will show it is true for $\mathcal{C}_\ell$ and $\mathcal{P}_{\ell+1}$. We first show statement (2). As a processor p does not know the time, it starts an election protocol for a good node C when it has determined the processors in $1 - 3/\log n$ fraction of C's predecessors. By assumption, for good predecessors, there is only one possible value that is not $\perp$. We call this the *identity* of the processor node. Hence p knows the identity of $\alpha_l - 4/\log n > 5/6$ fraction of good predecessors of C. Now, each processor may know identities for a different fraction of $1 - 3/\log n$ predecessors. In lines 7-9 of the protocol, each processor p waits to receive views from $2k/3$ processors in $v_p(C)$ and takes the majority. Since $< k/6$ predecessors of C are not good, the $2k/3$ processors include at least $k/2$ good processors from good predecessors which send either the identity of a node or $\perp$. Thus a good predecessor will appear as $\perp$ in a revised view only if $> k/6$ good processors have not determined its identity. Since each processor has $\perp$ values for only $3k/\log n$ predecessors, a simple averaging argument shows that there are at most $18k/\log n$ good predecessors such that $> k/6$ good processors all have $\perp$ values for them. Hence the core can only exclude this many good predecessors of C and statement (2) follows from the fact that there are $\alpha_\ell - 1/\ln n$ good predecessors of C.

We prove Statement (1): It follows from Observation 1, Section 3.5, that the subcommittee election will be run in time $ck\log^2 k2^k$, c a constant and with the results specified in Lemma 3.4. W.h.p., $(|S| - 1/\log n)k$ processors from the core will be elected by the committee for C and their selection will be sent to all processors. We assume by induction that by time $ck\log^2 k2^k\ell$, every processor has identified each good predecessor in $\mathcal{P}_\ell$ so that by this time it has identified the $> 5/k/6$ processors in the core of C which have sent it messages upon completion of the election. Thus when a processor hears from $5k/6$ processors, at least $2k/3$ messages are from processors in the core. Hence every processor's view of the election results is the view of the core which by virtue of having executed Byzantine agreement is in accord and occurs within time $ck\log^2 k2^k(\ell+1)$. Since the identities of the good predecessor nodes are known, the election results determine the identities of the subset of good predecessor nodes which are in C's elected subcommittee (See Section 3.5). Consequently, all processors learn the identities of C's good successor nodes $\in \mathcal{P}_{\ell+1}$. Thus we have shown that by time $ck\log^2 k2^k(\ell+1)$ identities of all good processor nodes in $\mathcal{P}_\ell$ are known to every processor.

We prove Statement(3). Assume the contrary. Let ℓ be the minimum index such that $\mathcal{P}_\ell$ contains a good processor node such that there have existed two distinct non-$\perp$ views of P. Clearly $\ell > 0$. Then the view of P was determined by some processor p when it received enough messages

from processors in $v_p(C)$ about an election in a good committee node $C \in \mathcal{C}_{\ell-1}$. Since most of these processors are p's views of good processor nodes in $\mathcal{P}_{\ell-1}$, their identities are fixed, by the induction assumption. As each election is run only once, their messages do not change. Hence only one identity for P is fixed by p.

This concludes the induction.

Proof of Lemma 4.2

The proof is by induction. The goal is to show that at each level, there is a sufficiently high fraction of good committee nodes whose elections run to completion and output the right fraction of good processors. We require that the random bits used for each subcommittee election be independent; therefore, the successes of subcommittee elections in different nodes are independent. At each step in the protocol, a processor waiting to hear from a set of k' processors of which at most t' are bad need only hear from any subset of $k' - t'$ processors for the protocol to work correctly. These two observations allow the protocol to work correctly under arbitrary timing delays as we will now show.

For the base case $\ell = 0$, we notice that all nodes in $\mathcal{P}_0$ are consistent, and at least a $5/6 + \epsilon$ fraction are good, establishing the first property. For the inductive step, assume (by inductive hypothesis) that the first property holds up to level ℓ, and the second up to level $\ell - 1$, within time $O(k \log^2 k 2^k \ell)$.

By the definition of a $(1/\ln n, 1/(2\ln n), 5/6)$ sampler, all but a $2/\ln n$ fraction of committee nodes in $\mathcal{C}_\ell$ will have a fraction of at least $1 - 3/\ln n$ predecessors which come from the $1 - 2/\ln n$ fraction of good committees on the layer below, as well as an $\alpha_\ell - 1/\ln n$ fraction of good predecessors. Hence, the fraction of good committee nodes in $\mathcal{C}_\ell$ is at least $1 - 2/\ln n$.

Thus, by Lemma 4.1, w.h.p., by time $O(k \log^2 k 2^k \ell)$, each of these good committee nodes elects a subcommittee. Since the core contains a fraction of processors of size least $s = \alpha_\ell - 19/\ln n$ then by Lemma 3.4, the subcommittee contains a fraction of the core of size $s - 1/\log n$ and $\alpha_{\ell+1} = \alpha_\ell - 20/\ln n$. Thus the fraction of good successor nodes in $\mathcal{P}_{\ell+1}$ is $(1-2\ln n)\cdot(\alpha_\ell - 20/\ln n) \geq \alpha_\ell - 22/\ln n = \alpha_{\ell+1}$.

More Details on the Polylogarithmic Protocol

We sketch the reasons why Polylog-BA is correct. First, we note that if each call to Quasi-Poly-BA in Polylog-BA has independent probability $1/\ln^c n$ of failing for any constant c, Polylog-BA will fail with probability $O(1/\ln^{c'} n)$ for any constant c'. The reason is that since each layer less than ℓ^* has $\Omega(\log^7 n)$ committee nodes, we can show that the probability of more than a $1/\ln n$ fraction of the committee nodes failing because of failed BA protocols at these nodes is bounded above using Chernoff bounds and is less than $1/n^c$. Adding another fraction of $1/\ln n$ failed committee nodes to each layer does not substantially affect the old proof of correctness. The call to Heavy-BA on layer ℓ^* would have probability of $1/\ln^c n$ of failure, and this would dominate the probability of failure for the protocol.

We now show how Quasi-Poly-BA when called on $n' = O(\log^7 n)$ processors can run in polylogarithmic time and have failure probability $O(1/\log^c n)$ for any constant c. As described above, at the first level of recursion Quasi-Poly-BA calls M-Quasi-Poly-BA instead of Heavy-BA. In Quasi-Poly-BA, when $n' = O(\log^7 n)$, every layer except ℓ^* has $\Omega((\log\log)^7 n)$ committee nodes. Note that on these layers, it again suffices if every committee node fails with probability less than $1/(\ln\ln)^c n$ for any c; using Chernoff bounds, we can then again bound the probability that no more than a $1/\ln n'$ fraction of committee elections fail. On ℓ^*, we need a smaller probability (of $1/\ln^c n$) of failure for the single node.

It remains to show that M-Quasi-Poly-BA, when called on $n'' = O((\log\log)^7 n)$ processors, fails with probability less than $1/(\ln\ln)^c n$ for any constant c. On layers $\ell < \ell^* - 1$ in M-Quasi-Poly-BA, when called on n'' processors, the number of committee nodes is $\Omega(\ln\ln\ln n)$. As in the analysis of Quasi-Poly-BA on n'' nodes, the probability that an election fails is less than $1/(n'')^{c'} = O(1/(\log\log)^{7c'} n)$. Again using Chernoff bounds, we can show that the probability that more than a $1/\ln n''$ fraction of election nodes fail on any layer less than $\ell^* - 1$ is less than $1/(\ln)^c n$. On layer $\ell^* - 1$, we have a problem as there are only $\Omega(\log\log n/\log\log\log n)$ committee nodes there. Now, we require that $< \epsilon/5$ fraction of committee nodes fail. Using Chernoff bounds, we can again show that this happens with probability $< 1/(\ln)^{c'} n$.

We have one more difficulty with M-Quasi-Poly-BA. If we are using the unmodified subcommittee election protocol, the sampler H would output a set of size $O(\ln^3 n'') = O((\ln\ln\ln)^3 n)$. This would give us too many processors left at layer ℓ^*. So we use a modified sampler with $\theta = \epsilon'/6$ and degree $O(\ln\ln\ln n)$ where the fraction of good processors on layer $\ell^* - 1$ is $5/6 + \epsilon'$. Now, the fraction of bad processors is increased by $\leq \epsilon'/6$ and the fraction of good processors in layer $\mathcal{P}_{\ell^*}$ is $\geq (1 - \epsilon'/5)(5/6 + 5\epsilon'/6) \geq 5/6 + \epsilon'/2$. There are then enough good processors so that Heavy-BA will succeed. Thus, Polylog-BA succeeds with probability $1/(\ln)^c n$ for any constant c. The total running time of Polylog-BA is dominated by the polylogarithmic number of calls to Heavy-BA on $O(\log\log n)$ processors.

Note: On levels 2 and 3 of the recursion, we require an additional property: if $2/3 + \epsilon$ fraction agree on a bit at the start of the Byzantine agreement protocol, that bit must be output. We note that the samplers will be representative of these processors, as well; i.e., for some constant ϵ', every $\mathcal{P}_\ell$ contains $2/3 + \epsilon'$ fraction of processors which are good and know this bit (as well as containing $5/6 + \epsilon'$ fraction of processors which are good), with a probabilty of failure increased by only a constant factor.

More Details on Leader Election

We modify our existing subcommittee election protocol as follows: Initially, $k_0 = O(\log^7)$ and $\beta_0 = 5/6 + \epsilon$. For $i = 0 \cdots \lg k_0 - c$, we begin a round i with a committee of $k_i = k_0/2^i$ processors, a $\beta_i = \beta_0 + \sum_j = 0^i 1/\ln^2 k_i$ fraction of which are bad or whose identity is not known to all processors. We execute the subcommittee election protocol with a number of bins $b_i = k_i/48 \ln k_i$. Instead of Heavy-BA, we use Polylog-BA. As before, we use the bin to select a subcommittee of size $k_i/2$, while maintaining the fraction of good processors in the result. This subcommittee becomes the committee for round $i + 1$. A version of Lemma 3.2 still goes through for this number of bins and we have:

Lemma 7.1. *Assuming the calls to* Polylog-BA *are successful, then with probability* $> 1 - 1/k_i$ *the fraction of good processors in the lightest bin is* $> \frac{\alpha}{2}(1 - \beta_i)$

Say that a subcommittee or committee is *i-bad* if the fraction of processors which are bad or such that all good processors do not agree on their identities is less than $1 - \beta + \sum_j^i \frac{1}{\lg^2 k_j}$), and *i-good* otherwise. By Lemma 2.2, we obtain a family of $(1/\ln^2 k_i, \ln^4 k_i/k_i, \beta_i)$ samplers H_i^L, where $r = O(k)$, $s = k_i$ and $d = k_i/2$ and $\beta_0 < \beta_i < \beta_0 + \epsilon$.

Using an argument similar to the proof of Lemma 3.3 and including the probability that the protoc we obtain

Lemma 7.2. *Assuming the calls to* Polylog-BA *are successful, the probability* $> 1 - 1/k_i$*, when* Elect-Subcommittee *is run with samplers* H_i^L *the output is a i-good subcommittee of size* $k_i/2$.

We have shown in the previous section that all good processors agree on a fraction of $\beta_0 = 5/6 + \epsilon$ good processors in the top node's committee with probability $1 - 1/n^{c_1}$, for some constant c_1.

Note that if all good processors agree on the identities of a set of processors in the top node's committee, they will continue to agree on the views of the identities of those processors in the subcommittee (which go on to become the committee in the next iteration). After c successful rounds, the fraction of bad processors is less than $\beta_0 + \sum_{j=c}^{\lg k_0} 1/j^2$ which is less than $\beta_0 + \epsilon$ for some constant c. The probability of success of all the rounds is $P_1 = 1 - \sum_0^c 1/k_j > 1 - 1/2^{c-1}$.

Given that all rounds are successful, the subcommittee returned in the last iteration contains at least $5/6 + \epsilon/2$ good processors known to all the good processors. Then the probability that each processor outputs the same leader is at least the probability that 2/3 of the processors choose the same good leader. There are a constant 2^c processors in the last committee. The probability that 2/3 of these agree on a good leader is greater than $P_2 = 2^{-2^c}(5/6)$ which is also a constant. With constant probability $P_1 * P_2$ the rounds are all successful and the 2/3 of the processors agree on a good leader. Hence Heavy-BA results in all good processors in the committee coming to agreement.

The running time of this protocol is dominated by Heavy-BA and is $O(2^{k_0})$. If we use Polylog-BA instead of Heavy-BA , we reduce the running time to polylogarithmic in n. The probability of failure of each round i is dominated by the probability of failure of Polylog-BA which is $O(1/\ln^{c'} k_i)$, for any constant c'. Summing over all the rounds again gives a constant probability of failure.

The Quasi-Poly-BA protocol

Note (*) below: if a majority of views have $v(P) = \perp$ or there is no majority view, then the majority view is set to $\perp$.

Protocol 2 The QUASI-POLY-BA protocol on good processor p

while there is a committee node C such that $v_p(C)$ is not `fixed` **do**
  for all such nodes C in parallel **do**
    if $v_p(C)$ is not `ready` **then** {find processors in this committee}
      Let $P_1, \ldots, P_k$ be the nodes with edges to C.
      if $v_p(P_i)$ is fixed for at least a $(1 - 2/\ln n)$ fraction of these P_i's **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 2/3 fraction of the processors $p' \in v_p(C)$
        For each position of $v_p(C)$, p revises its view to agree with the majority of $v_{p'}(C)$ received*.
        $v_p(C)$ is set to be `ready`.
    else if C is in layer ℓ 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 identifiers to all processors.
      p waits to receive election result messages from at least $5/6k$ 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 neighbors P of C at level $\ell + 1$.
      else
        p sets $v_p(C)$ arbitrarily.
      $v_p(C)$ is set to be `fixed`, as are $v_p(P)$ for neighbors P of C at level $\ell + 1$.
    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. $v_p(C)$ is set to be `fixed`, and p terminates.
      else
        p waits to receive a bit from at least a α_{ℓ^*} fraction of processors in $v_p(C)$.
        p takes the majority of received bit values as agreed upon. $v_p(C)$ is set to be `fixed`, and p terminates.