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 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 $\frac{n}{6+\epsilon}$ 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.

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 [14] 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 [4, 8] 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 $o(\sqrt{n/\log n})$ 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. Hence we consider 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 poly-logarithmic (resp. quasi-poly-logarithmic) in n and succeed with probability at least $1 - 1/\ln^c n$ (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*

*Dept. of Computer Science, University of Victoria; email: bmkapron@uvic.ca

†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.

‡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

§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.

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

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 [13]. 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* $\frac{n}{6+\epsilon}$, *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). Second, we describe how to adapt the layered network from [17, 18] to an asynchronous setting (Section 4). In [17, 18], the layered network was used to reduce communication costs in a synchronous setting. Here, we adapt it to reduce the run time 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 [13] 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 were considered in a series of papers [5, 1, 22, 20, 21] 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 [13], 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 [17, 18] and by Ben-Or, Goldwasser, Pavlov and Vaikuntanathan [6, 15]. [17] presents protocols for both Byzantine agreement and leader election with resilience $\frac{n}{3+\epsilon}$ 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 needed to obtain agreement among all good processors, in a worst case number of rounds which is polylogarithmic in n. [18] showed that the almost everywhere agreement protocol could be implemented scalably on a sparse network in polylogarithmically many rounds. [6] and [15] give protocols with resilience $\frac{n}{4+\epsilon}$ and $\frac{n}{3+\epsilon}$, 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 [16] 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 [23], culminating in protocols with expected $O(1)$ round time with optimal resilience $\frac{n}{3}$ and $O(n^2)$ messages [9, 19].

The only known lower bounds which apply to our work (and the protocols assuming private channels and cryptographic primitives) is obtained by slightly modifying the proof in [12]. They show that any randomized protocol which runs in r synchronous rounds in the fail-stop model with t processor failures has probability at most $1 - (1/2)\lceil n/\lfloor t/r \rfloor \rceil^{-r}$ of terminating against an adversary oblivious to the content of messages. The same proof applies to protocols which fail to agree on the same value or fail to terminate. The bound is $1/n$ when $r = O(\log n / \log \log n)$. Recently this bound was improved for the asynchronous model [2].

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: 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 determine 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 $\Delta f(n)$. 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 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 making up nearly the same fraction of each set as the fraction of good processors in the universe.

To establish this result (and the main Theorems 1.1 and 1.2), we adapt the synchronous universe reduction protocol of King et al. [17] 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) 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, the committee is repeatedly reduced using a variant of the subcommittee election protocol, until a constant number of processors remain 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.

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 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.

The second issue is resolved in Section 4, using a network structure similar to that in [17] 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. In Section 5, we sketch the modifications required for the protocol to achieve polylogarithmic running time. Section 6 briefly describes the leader election protocol. Due to space constraints, a detailed version of the last two sections is deferred to the full version of this paper.

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 [24].

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 $|S| > \beta s$, at most an ϵ fraction of all inputs x have $\frac{|H(x) \cap S|}{d} > \frac{|S|}{s} + \theta$.*

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 [16] and [25], 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 in which at least $k - t$ are good. In Feige's protocol [13], 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 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 bin choice, but the adversary can prevent some processors from being heard any of the others by delaying their messages.

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 We allow processors to agree that a null message $*$ has been received from a processor rather than a bin number. We first note that Bracha's exponential time asynchronous Byzantine agreement protocol [8] can easily be extended to the case where each processor has three possible input values $0, 1$ and $*$, by running it 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. (Sketch). The proof of items (1) and (2) by induction is straightforward. To prove part 3, we note that each good processor receives $k - 2t$ messages from good processors. An averaging argument over the total number of good messages received by good processors shows that a sufficient number of good processors have their messages received by at least $2k/3$ good processors. By Property (3) of Proposition 3.1, their messages are accepted. ■

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 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: 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 $S_k \subseteq S_{k-1} \subseteq S_{k-2} \cdots \subseteq S_1$. By the Pigeonhole Principle, 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 that 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 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 $\gamma \cdot c' \ln n$, 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 $i-1$ 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 $Pr(X_j = 1) = 1/b$, and $E[X] = |A_i|/b \geq \frac{\epsilon}{1+\epsilon}(k-t)/b$ by Lemma 3.1 (3). 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 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 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, 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 $a = 1 - c'\gamma/2$ for some constant c', 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 subsequences of blocks, and the different bad inputs, the probability of the lightest bin generating a bad input is at most $2^{x\gamma k/2b}(kb)(1/2^{x\gamma k/b}) < n^{-c''}$ 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

---

for $i = 1$ to k **do**
    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.
    Send $((B_1^{(p)}, X_1^{(p)}), \ldots, (B_i^{(p)}, X_i^{(p)}))$ to every processor in C.
    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.
    Update all $(B_{i'}^{(p')}, X_{i'}^{(p')})$ for $i' \leq i$.
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 $S_r = S_{r-1}$.
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.
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* $v_i : I \rightarrow P$ 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 of size $|R| > k(5/6 + \epsilon)$ for some fixed $\epsilon > 0$, every $p \in R$ will come to agreement on the identifiers of the processors in the elected subcommittee, and consequently on the actual identities of those processors in the elected subcommittee 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 $s_\ell = n/\ln^\ell n$ and $r_\ell = n/\ln^{\ell+4} n$. The layers of the network are numbered $0(a), 0(b), 1(a), 1(b), \ldots, \ell^*(a), \ell^*(b)$. Layers $\ell(a)$ and $\ell(b)$ contain the set of s_ℓ *processor nodes* $\mathcal{P}_\ell$ and r_ℓ *committee nodes* $\mathcal{C}_\ell$, respectively. 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 $r = r_\ell$, $s = s_\ell$ 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 node in $\mathcal{P}_{\ell+1}$ is incident to exactly one such edge. 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. 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 includes 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. 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 $v_p(C)$ of C to all other processors in $v_p(C)$. Once p receives views of C from at least $2k/3$ 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 $v_p^r(C)$.) If p is in $v_p(C)$, then p runs subcommittee election; else p waits to hear from other processors about the results. $v_p(C)$ is considered `fixed` after the election is run.

The formal statement of the Quasi-Poly-BA protocol appears on the next page as Protocol 2. Note (*): if a majority of views have $v_p(P) = \perp$ or there is no majority view, then the majority view is set to $\perp$.

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 shows that there are committees which have large enough *cores*, i.e., subsets

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 $5k/6$ 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.

of good processors 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 in 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.

Lemma 4.1. *Let P and C be any good processor and good committee node in $\mathcal{P}_\ell$ and $\mathcal{C}_\ell$, respectively. 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 $S = \bigcap_{p \in S} v_p(C)$ 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.

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

Assume that it is true for $\mathcal{C}_{\ell-1}$ and $\mathcal{P}_\ell$. We will show that it is true for $\mathcal{C}_\ell$ and $\mathcal{P}_{\ell+1}$. We first show Statement (2). Processor p starts an election protocol for a good node C when it has determined the processors in a $1 - 3/\log n$ 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 an $\alpha_\ell - 4/\log n > 5/6$ fraction of good predecessors of C. Each processor may know identities for a different $1 - 3/\log n$ fraction of 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 less than $k/6$ 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 more than $k/6$ 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 more than $k/6$ 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 $(\alpha_\ell - 1/\ln n)k$ good predecessors of C.

We next prove Statement (1). It follows from Observation 1 in Section 3.5 that the subcommittee election will be run in time $ck \log^2 k 2^k$ with 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 $ck \log^2 k 2^k \ell$, every processor has identified each good predecessor in $\mathcal{P}_\ell$ so that by this time, it has identified the more than $5k/6$ 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 $ck \log^2 k 2^k(\ell + 1)$. 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 $ck \log^2 k 2^k(\ell + 1)$, the identities of all good processor nodes in $\mathcal{P}_\ell$ are known to every processor.

Finally, 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 $\ell > 0$. 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. ■

The next lemma shows that that the fraction of good processors and committees remain sufficiently high for all levels.

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. The proof is by induction.

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 at least a $1-3/\ln n$ fraction of predecessors which come from the $1-2/\ln n$ fraction of good committees in the previous layer, 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. (after taking a union bound), by time $O(k\log^2 k 2^k \ell)$, every good committee on level ℓ elects a subcommittee such that its core contains at least an $s = \alpha_\ell - 19/\ln n$ fraction of good processors. By Lemma 3.4, each subcommittee contains a fraction of the core of size $s - 1/\log n$. Since $\alpha_{\ell+1} = \alpha_\ell - 20/\ln n$, the fraction of good successor nodes in $\mathcal{P}_{\ell+1}$ is at least $(1-2\ln n)\cdot(\alpha_\ell - 20/\ln n) \geq \alpha_\ell - 22/\ln n = \alpha_{\ell+1}$. ■

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

Proof of Theorem 4.1. As $\ell^* = o(\ln n)$, Lemma 4.2 implies that w.h.p., the committee C^* constituting $\mathcal{C}_{\ell^*}$ is good. Thus, by Lemma 4.1, C^* has a core of size at least $(\alpha_{\ell^*} - 19/\ln n)k$. Since the core consists of good processor nodes, these are good processors of which *all* processors in the network have the same view, by Lemma 4.1 (1). But $(\alpha_{\ell^*} - 19/\ln n) > 5/6$, 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 protocol Polylog-BA by modifying Quasi-Poly-BA so that there are two levels of recursion. 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 lower degree) for selecting this last subcommittee. Finally, on the next level of recursion, where committees are of size $O((\log\log\log)^7 n)$ (and on layer ℓ^*, where they are of size $O(\log\log n)$), the call to Heavy-BA is actually executed by calling Heavy-BA.

There are two observations which allow this amount of recursion (and no more). First, 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. The second observation is that while we cannot use such reasoning for the top level committee's election (as there is only one), we can instead reduce the size of the subcommittees output on layer $\ell^* - 1$, by using a sampler of smaller degree $O(\log\log\log n)$, for a total of $O(\log\log n/\log\log\log n) \cdot \log\log n/\log\log\log n = O(\log\log n)$ processors on layer ℓ^*. This in turn reduces by a small constant the fraction of good processors produced in each such election, but it happens only on one layer, so there are still sufficient good processors. Due to space constraints, details of the construction and analysis are deferred to the full version of this paper.

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 a $5/6+\epsilon$ fraction of good processors is 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. Subsequently, every processor in the committee sends messages with the identity of the leader to all other processors. Again, the details are deferred to the full version of this paper.

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 lower bound from [12] 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. Personal communication, 2007.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

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

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

[25] David Zuckerman. Personal communication, 2006.