RANDOMIZED CONSENSUS IN EXPECTED $O(N \log^2 N)$ OPERATIONS PER PROCESSOR

JAMES ASPNES* AND ORLI WAARTS†

Abstract. This paper presents a new randomized algorithm for achieving consensus among asynchronous processors that communicate by reading and writing shared registers. The fastest previously known algorithm requires a processor to perform an expected $O(n^2 \log n)$ read and write operations in the worst case. In our algorithm, each processor executes at most an expected $O(n \log^2 n)$ read and write operations, which is close to the trivial lower bound of $\Omega(n)$.

All previously known polynomial-time consensus algorithms were structured around a *shared coin* protocol [4] in which each processor repeatedly adds random ± 1 votes to a common pool. Consequently, in all of these protocols, the worst case expected bound on the number of read and write operations done by a single processor is asymptotically no better than the bound on the total number of read and write operations done by all of the processors together. We succeed in breaking this tradition by allowing the processors to cast votes of increasing weights. This grants the adversary greater control since he can choose from up to n different weights (one for each processor) when determining the weight of the next vote to be cast. We prove that our shared coin protocol is correct nevertheless using martingale arguments.[1]

1. Introduction. In the **consensus** problem, each of n asynchronous processors starts with an input value 0 or 1 not known to the others and runs until it chooses a *decision value* and halts. The protocol must be **consistent**: no two processors choose different decision values; **valid**: the decision value is some processor's input value; and **wait-free**: each processor decides after a finite expected number of its *own* steps regardless of other processors' halting failures or relative speeds.

We consider the consensus problem in the standard model of asynchronous shared memory systems. The processors communicate via a set of single-writer, multi-reader atomic registers. Each such register can be written only by one processor, its owner, but all processors can read it. Reads and writes to such a register can be viewed as occurring at a single instant of time.

Consensus is fundamental to synchronization without mutual exclusion and hence lies at the heart of the more general problem of constructing highly concurrent data structures [20]. It can be used to obtain wait-free implementations of arbitrary abstract data types with atomic operations [20, 23]. Consensus is also **complete** for **distributed decision tasks** [11] in the sense that it can be used to solve all such tasks that have a wait-free solution.

Consensus is often viewed as a game played between a set of processors and an adversary scheduler. Using the standard wait-free model of an asynchronous shared-memory system, each processor can execute as an atomic step either a single read or write operation, or a flip of a local fair coin not visible to the other processors. The sequencing of the processors' actions is controlled by a **scheduler**, defined as a function that at each step selects a processor to run based on the entire prior history of the system, including the internal states of the processors. (Concurrency is modeled

* Yale University, Department of Computer Science, 51 Prospect Street / P.O. Box 208285, New Haven, CT 06520-8285. E-mail: aspnes@cs.yale.edu. During the time of this research the first author was at Carnegie-Mellon University, supported in part by an IBM Graduate Fellowship.

† Computer Science Division, University of California, Berkeley, CA 94720. E-mail: waarts@cs.berkeley.edu. During the time of this research the second author was at Stanford, supported in part by an IBM Graduate Fellowship, U.S. Army Research Office Grant DAAL-03-91-G-0102, and NSF grant CCR-8814921.

[1] A preliminary version of this work appeared in the proceedings of the Thirty-Third IEEE Symposium on Foundations of Computer Science.

by interleaving.) Remarkably, it has been shown that the ability of the scheduler to stop even a single processor is sufficient to prevent consensus from being solved by a deterministic algorithm [10, 12, 16, 20, 22]. Nevertheless, it can be solved by *randomized* protocols in which each processor is guaranteed to decide after a finite *expected* number of steps.

Chor, Israeli, and Li [10] provided the first solution to the problem, but their solution deviated from the standard model by assuming that the processor can flip a coin and write the result in a single atomic step. Abrahamson [1] demonstrated that consensus is possible even for the standard model, but his protocol required an exponential expected number of steps. Since then a number of polynomial-work consensus protocols have been proposed. Protocols that use unbounded registers have been proposed by Aspnes and Herlihy [4] (the first polynomial-time algorithm), by Saks, Shavit, and Woll [24] (optimized for the case where processors run in lock step), and by Bracha and Rachman [8] (running time $O(n^2 \log n)$). Protocols that use bounded registers have been proposed by Attiya, Dolev, and Shavit [5] (running time $O(n^3)$), by Aspnes [3] (running time $O(n^2(p^2+n))$, where p is the number of active processors), by Bracha and Rachman [7] (running time $O(n(p^2+n))$), and by Dwork, Herlihy, Plotkin and Waarts [13] (immediate application of what they call time-lapse snapshots and with the same running time as [7]).

The main goal of a wait-free algorithm is usually to minimize the worst case expected bound on the work done by a single processor. Still, for all of the known polynomial-work wait-free consensus protocols, the worst case expected bound on the work done by a single processor is asymptotically no better than the bound on the total work done by all of the processors together.

Therefore, one of the main contributions of this paper is in showing that wait-free consensus can be solved without requiring the fast processors to perform much more than their fair share of the worst case total amount of work executed by all processors together. At the same time, we improve significantly on the complexity of all currently known wait-free consensus protocols, obtaining a protocol in which a processor executes at most an expected $O(n \log^2 n)$ read and write operations, which is close to the trivial lower bound of $\Omega(n)$.[2] To do this we introduce a new *weak shared coin* protocol [4] that is based on a combination of the shared coin protocol described by Bracha and Rachman [8] and a new technique called *weighted voting*, where votes of faster processors carry more weight.[3] We believe that our weighted voting technique will find applications in other wait-free shared memory problems such as approximated consensus and resource allocation.

The rest of the paper is organized as follows. The next section describes the intuition behind our solution while emphasizing the main difference between our solution and that in [3, 4, 5, 7, 8, 13, 24]. Section 3 describes our shared coin protocol. Section 4 reviews martingales and derives some of their properties. Section 5 contains the proof of correctness of our shared coin protocol. A discussion of the results appears in Section 6.

2. Intuition and relation to previous results. All of the known polynomial-work consensus protocols are based on the same primitive, the **weak shared coin**. A weak shared coin returns a single bit to each processor; for each possible value

[2] As discussed in Section 6, this gain in per-processor performance involves a slight increase in the total work performed by all processors when compared with the Bracha-Rachman protocol.

[3] The consensus protocol can be constructed around our shared coin protocol using the established techniques of Aspnes and Herlihy [4].

$b \in \{0, 1\}$ the probability that all processors see b must be at least a constant δ (the **agreement parameter** of the coin), regardless of scheduler behavior.[4] Aspnes and Herlihy [4] showed that given a weak shared coin with constant agreement parameter it is possible to construct a consensus protocol by executing the coin repeatedly within a rounds-based framework which detects agreement. The number of operations executed by each processor in this construction is $O((n + T(n))/\delta)$, where $T(n)$ is the expected work per processor for the weak shared coin protocol. For constant δ, and under the reasonable assumption that $T(n)$ dominates n, the work per processor to achieve consensus becomes simply $O(T(n))$.

So to construct a fast consensus protocol one need only construct a fast weak shared coin. The underlying technique for building a weak shared coin has not changed substantially since the protocol described in [4]; each processor repeatedly adds random ± 1 votes to a common pool until either the total vote is far from the origin [3, 4, 5, 7, 13] or until a predetermined number of votes have been cast [8, 24]. Any processor that sees a nonnegative total vote decides 1, and those that see a negative total vote decide 0. (The differences between the protocols are largely in how termination is detected and how the counter for the vote is implemented.)

There are many advantages to this approach. The processors effectively act as anonymous conduits of a stream of unpredictable random increments. If the scheduler stops a particular processor, at worst all it does is keep one vote from being written out to the common pool— the next local coin flip executed by some other processor is no more or less likely to give the value the scheduler wants than the next one executed by the processor it has just stopped. Intuitively, the scheduler's power over the outcome of the shared coin is limited to filtering out up to $n - 1$ local coin flips from this stream of independent random variables. But the effect of this filtering is at worst equivalent to adjusting the final tally of votes by up to $n - 1$. If a constant multiple of n^2 votes are cast, the total variance will be $\Omega(n^2)$, and using a normal approximation the protocol can guarantee that with constant probability the total vote is more than n away from the origin, rendering the scheduler's adjustment ineffective.

Alas, the very anonymity of the processors that is the strength of the voting technique is also its greatest weakness. To overcome the scheduler's power to withhold votes, it is necessary that a total of $\Omega(n^2)$ votes are cast— but the scheduler might also choose to stop all but one of the processors, leaving that lone processor to generate all $\Omega(n^2)$ votes by itself. Consequently, for all of the polynomial-work wait-free consensus protocols currently known, the worst-case expected bound on the work done by a single processor is asymptotically no better than the bound on the total work done by all of the processors together.

We overcome this problem by modifying the $O(n^2 \log n)$ protocol of Bracha and Rachman [8] to allow the processor to cast votes of increasing weight. Thus a fast processor or a processor running in isolation can quickly generate votes of sufficient total variance to finish the protocol, at the cost of giving the scheduler greater control by allowing it both to withhold votes with larger impact and to choose among up to n different weights (one for each processor) when determining the weight of the next vote to be cast.

There are two main difficulties that this approach entails; the first is that careful

[4] The term *agreement parameter* was first used by Saks et al [24] in place of the more melodramatic but less descriptive term *defiance probability* of Aspnes and Herlihy [4]. Aspnes [3] used a *bias* parameter, equal to 1/2 minus the agreement parameter; however, this quantity is not as useful as the agreement parameter in the context of a multi-round consensus protocol.

```
1  procedure shared_coin()
2  begin
3       my_reg(variance, vote) ← (0, 0)
4       t ← 1
5       repeat
6            for i = 1 to c do
7                 vote ← local_flip() × w(t)
8                 my_reg ← (my_reg.variance + w(t)^2, my_reg.vote + vote)
9                 t ← t + 1
10           end
11           read all the registers, summing the variance fields into the local
             variable total_variance
12      until total_variance > K
13      read all the registers, summing the vote fields into the local variable
        total_vote
14      if total_vote > 0
15      then output 1
16      else output 0
17 end
```

FIG. 1. *Shared coin protocol.*

adjustment of the weight function and other parameters of the protocol is necessary to make sure that it performs correctly. More importantly, correctness proofs for previous shared coins based on random walks or voting [3, 4, 5, 7, 8, 13, 24] considered only equally weighted votes, and have therefore been able to treat the sequence of votes as a sequence of independent random variables using a substitution argument. Because our protocol allows the weight of the i-th vote to depend on which processor the scheduler chooses to run, which may depend on the outcomes of previous votes, we cannot assume independence.

However, the *sign* of each vote is determined by a fair coin flip that the scheduler cannot predict in advance, and so despite all the scheduler's powers, the expected value of each vote before it is cast is always 0. This is the primary requirement of a **martingale process** [6, 15, 21]. Under the right conditions, martingales have many similarities to sequences of sums of independent random variables. In particular, martingale analogues of the Central Limit Theorem and Chernoff bounds will be used in the proof of correctness.

3. The Shared Coin Protocol. Figure 1 gives pseudocode for each processor's behavior during the shared coin protocol. Each processor repeatedly flips a local coin that returns the values $+1$ and -1 with equal probability. The weighted value of each flip is $w(t)$ or $-w(t)$ respectively, where t is the number of coins flipped by the processor up to and including its current flip. Each weighted flip represents a vote for either the output value 1 (if positive) or 0 (if non-positive). After each flip, the processor updates its register to hold the sum of the weighted flips it has performed, and the sum of the squares of their values. After every c flips, the processor reads the registers of all the other processors, and computes the sum of all the weighted flips (the total vote) and the sum of the squares of their values (the total variance). If the total variance is greater than the quorum K, it stops, and outputs 1 if the total vote

is positive, and 0 otherwise. Alternatively, if the total variance has not yet reached the quorum K, it continues to flip its local coin.

The function *local_flip* returns the values 1 and -1 randomly with equal probability. The values K and c are parameters of the protocol which will be set depending on the number of processors n to give the desired bounds on the agreement parameter and running time. The weight function $w(t)$ is used to make later local coin flips have more effect than earlier ones, so that a processor running in isolation will be able to achieve the quorum K quickly. The weight function will be assumed to be of the form $w(t) = t^a$ where a is a nonnegative parameter depending on n; though other weight functions are possible, this choice simplifies the analysis.

We will demonstrate that for suitable choice of K, c and a all processors return 1 with constant probability; the case of all processors returning 0 will follow by symmetry. The structure of the argument follows the proof of correctness of the less sophisticated protocol of Bracha and Rachman [8], which corresponds to Figure 1 when $w(t)$ is the constant 1, $K = \Theta(n^2)$, and $c = \Theta(n/\log n)$. Votes cast before the quorum K is reached will form a pool of **common votes** that all processors see.[5] We will show that with constant probability (i) the total of the common votes is far from the origin and (ii) the sum of the **extra votes** cast between the time the quorum is reached and the time some processor does its final read in line 13 is small, so that the total vote read by each processor will have the same sign as the total common vote.

This simple overview of the proof hides many tricky details. To simplify the analysis we will concentrate not on the votes actually written to the registers but on the votes whose values have been **decided** by the processors' execution of the local coin flip in line 7; conversion back to the values actually in the registers will be done by showing a bound on the difference between the total decided vote and the total of the register values. In effect, we are treating a vote as having been "cast" the moment that its value is determined, instead of when it becomes visible to the other processors.

Some care is also needed to correctly model the sequence of votes. Most importantly, as pointed out above, allowing the weight of the i-th vote to depend on which processor the scheduler chooses to run means the votes are not independent. So the straightforward proof techniques used for protocols based on a stream of identically-distributed random votes no longer apply, and it is necessary to bring in the theory of martingales to describe the execution of the protocol.

4. Martingales. A **martingale** is a sequence of random variables $S_1, S_2, \ldots$, which informally may be thought of as representing the changes in the fortune of a gambler playing in a fair casino. Because the gambler can choose how much to bet or which game to play at each instant, each random variable S_i may depend on all previous events. But because the casino is fair and the gambler cannot predict the future, the expected change in the gambler's fortune at any play is always 0.

We will need to use a very general definition of a martingale [6, 15, 21]. The simplest definition of a martingale says that the expected value of S_{i+1} given $S_1, S_2, \ldots, S_i$ is just S_i. To use a gambling analogy, this definition says that a gambler who knows only the previous values of her fortune cannot predict its expected future value any better than by simply using its current value. But what if the gambler knows more information than just the changing size of her bankroll? For example, imagine that

[5] The definitions of the common and extra votes we will use differ slightly from those used in [8]; the formal definitions appear in Section 5.

she is placing bets on a fair version of roulette, and always bets on either red or black. Knowing that her fortune increased after betting red will tell her only that one of eighteen red numbers came up; but a real gambler will see precisely *which* of the eighteen numbers it was. Still, we would like to claim that this additional knowledge does not affect her ability to predict the future. To do so, the definition of a martingale must be extended to allow additional information to be represented explicitly.

The tool used to represent the information known at any point in time will be a concept from measure theory, a **σ-algebra**.[6] The description given here is informal; more complete definitions can be found in [15, Sections IV.3, IV.4, and V.11] or [6].

4.1. Knowledge, σ-algebras, and measurability. Recall that any probabilistic statement is always made in the context of some (possibly implicit) **sample space**. The elements of the sample space (called **sample points**) represent all possible results of some set of experiments, such as flipping a sequence of coins or choosing a point at random from the unit interval. Intuitively, all randomness is reduced to selecting a single point from the sample space. An **event**, such as a particular coin-flip coming up heads or a random variable taking on the value 0, is simply a subset of the sample space that "occurs" if one of the sample points it contains is selected.

If we are omniscient, we can see which sample point is chosen and thus can tell for each event whether it occurs or not. However, if we have only partial information, we will not be able to determine whether some events occurred or not. We can represent the extent of our knowledge by making a list of all events we do know about. This list will have to satisfy certain closure properties; for example, if we know whether or not A occurred, and whether or not B occurred, then we should know whether or not the event "A or B" occurred.

We will require that the set of known events be a **σ-algebra**. A σ-algebra $\mathcal{F}$ is a family of subsets of a sample space Ω that (i) contains the empty set; (ii) is closed under complement: if $\mathcal{F}$ contains A, it contains $\Omega \setminus A$ (the **complement** of A); and (iii) is closed under countable union: if $\mathcal{F}$ contains all of $A_1, A_2, \ldots$, it contains $\bigcup_{i=1}^{\infty} A_i$.[7] An event A is said to be **$\mathcal{F}$-measurable** if it is contained in $\mathcal{F}$. In our context, the term "measurable," which comes from the original measure-theoretic use of σ-algebras to represent families of sets on which a probability distribution is well-defined, simply means "known."

We "know" about an event if we can determine whether or not it occurred. What about random variables? A random variable X is defined to be **$\mathcal{F}$-measurable** if every event of the form $X \leq c$ is $\mathcal{F}$-measurable. (The closure properties of $\mathcal{F}$ then imply that such events as $a \leq X < b$, $X = d$, and so forth are also $\mathcal{F}$-measurable.) Looking at the situation in reverse, given random variables $X_1, X_2, \ldots$ we can consider the minimum σ-algebra $\mathcal{F}$ for which each of the random variables is $\mathcal{F}$-measurable; this σ-algebra, written $\langle X_i \rangle$, is called the σ-algebra **generated** by $X_1, X_2, \ldots$, and represents all information that can be inferred from knowing the values of the generators.

A σ-algebra gives us a rigorous way to define "knowledge" in a probabilistic context. Measurability and generated σ-algebras give us a way to move back and forth between the abstract concept of a σ-algebra and concrete statements about which random variables are completely known. To analyze random variables that are only *partially* known, we need one more definition. We need to extend conditional

[6] Sometimes called a **σ-field**.

[7] Additional properties, such as being closed under finite union or intersection, follow immediately from this definition.

expectations so that the condition can be a σ-algebra rather than just a collection of random variables.

For each event A let I_A be the indicator variable that is 1 if A occurs and 0 otherwise. Let $U = \mathrm{E}[X \mid \mathcal{F}]$ be a random variable such that (i) U is $\mathcal{F}$-measurable and (ii) $\mathrm{E}[UI_A] = \mathrm{E}[XI_A]$ for all A in $\mathcal{F}$. The random variable $\mathrm{E}[X \mid \mathcal{F}]$ is called the **conditional expectation** of X with respect to $\mathcal{F}$ [15, Section V.11]. Intuitively, the first condition on $\mathrm{E}[X \mid \mathcal{F}]$ says that it reveals no information not already found in $\mathcal{F}$. The second condition says that just knowing that some event in $\mathcal{F}$ occurred does not allow one to distinguish between X and $\mathrm{E}[X \mid \mathcal{F}]$; this fact ultimately implies that $\mathrm{E}[X \mid \mathcal{F}]$ uses all information that is found in $\mathcal{F}$ and is relevant to X.

If $\mathcal{F}$ is generated by random variables $X_1, X_2, \ldots$, the conditional expectation $\mathrm{E}[X \mid \mathcal{F}]$ reduces to the simpler version $\mathrm{E}[X \mid X_1, X_2, \ldots]$. Some other facts about conditional expectation that we will use (but not prove): if X is $\mathcal{F}$-measurable, then $\mathrm{E}[XY \mid \mathcal{F}] = X\,\mathrm{E}[Y \mid \mathcal{F}]$ (which implies $\mathrm{E}[X \mid \mathcal{F}] = X$); and if $\mathcal{F}' \subseteq \mathcal{F}$, then $\mathrm{E}[\mathrm{E}[X \mid \mathcal{F}] \mid \mathcal{F}'] = \mathrm{E}[X \mid \mathcal{F}']$. See [15, Section V.11].

4.2. Definition of a martingale. We now have the tools to define a martingale when the information available at each point in time is not limited to just the values of earlier random variables. A **martingale** $\{S_i, \mathcal{F}_i\}, 1 \le i \le n$, is a stochastic process where each S_i is a random variable representing the state of the process at time i and $\mathcal{F}_i$ is a σ-algebra representing the knowledge of the underlying probability distribution available at time i. Martingales are required to satisfy three axioms, for all i:

1. $\mathcal{F}_i \subseteq \mathcal{F}_{i+1}$. (The past is never forgotten.)
2. S_i is $\mathcal{F}_i$-measurable. (The present is always known.)
3. $\mathrm{E}[S_{i+1} \mid \mathcal{F}_i] = S_i$. (The future cannot be foreseen.)

Often $\mathcal{F}_i$ will simply be the σ-algebra $\langle S_1, \ldots S_i \rangle$ generated by the variables S_1 through S_i; in this case axioms 1 and 2 will hold automatically.

To avoid special cases let $\mathcal{F}_0$ denote the trivial σ-algebra consisting of the empty set and the entire probability space. The **difference sequence** of a martingale is the sequence $X_1, X_2, \ldots X_n$ where $X_1 = S_1$ and $X_i = S_i - S_{i-1}$ for $i > 1$. A **zero-mean martingale** is a martingale for which $\mathrm{E}[S_i] = 0$.

4.3. Gambling systems. A remarkably useful theorem, which has its origins in the study of gambling systems, is due to Halmos [18]. We restate his theorem below in modern notation:

THEOREM 4.1. *Let $\{S_i, \mathcal{F}_i\}, 1 \le i \le n$ be a martingale with difference sequence $\{X_i\}$. Let $\{\zeta_i\}, 1 \le i \le n$ be random variables taking on the values 0 and 1 such that each ζ_i is $\mathcal{F}_{i-1}$-measurable. Then the sequence of random variables $S'_i = \sum_{j=1}^{i} \zeta_j X_j$ is a martingale relative to $\mathcal{F}_i$. (That is, $\{S'_i, \mathcal{F}_i\}$ is a martingale.)*

Proof. The first two properties are easily verified. Because ζ_i is $\mathcal{F}_{i-1}$-measurable, $\mathrm{E}[\zeta_i X_i \mid \mathcal{F}_{i-1}] = \zeta_i\,\mathrm{E}[X_i \mid \mathcal{F}_{i-1}] = 0$, and the third property also follows. □

4.4. Limit theorems. Many results that hold for sums of independent random variables carry over in modified form to martingales. For example, the following theorem of Hall and Heyde [17, Theorem 3.9] is a martingale version of the classical Central Limit Theorem:

THEOREM 4.2 ([17]). *Let $\{S_i, \mathcal{F}_i\}$ be a zero-mean martingale. Let $V_n^2 = \sum_{i=1}^{n} \mathrm{E}\left[X_i^2 \mid \mathcal{F}_{i-1}\right]$ and let $0 < \delta \le 1$. Define $L_n = \sum_{i=1}^{n} \mathrm{E}\left[|X_i|^{2+2\delta}\right] + \mathrm{E}\left[|V_n^2 - 1|^{1+\delta}\right]$. Then there exists a constant C depending only on δ such that whenever $L_n \le 1$,*

$$|\Pr[S_n \le x] - \Phi(x)| \le C L_n^{1/(3+2\delta)} \left[\frac{1}{1 + |x|^{4(1+\delta)^2/(3+2\delta)}}\right], \tag{1}$$

where Φ is the standard unit normal distribution with mean 0 and variance 1.

For our purposes we will need only the case where x and δ are both set to 1. This allows the statement of the theorem to be simplified considerably. Furthermore, the rather complicated fraction containing x is never more than 1 and so can disappear into the constant. The result is:

THEOREM 4.3. *Let $\{S_i, \mathcal{F}_i\}$ be a zero-mean martingale. Let $V_n^2 = \sum_{i=1}^n \mathrm{E}\left[X_i^2 \mid \mathcal{F}_{i-1}\right]$. Define $L_n = \sum_{i=1}^n \mathrm{E}\left[|X_i|^4\right] + \mathrm{E}\left[|V_n^2 - 1|^2\right]$. Then there exists a constant C such that whenever $L_n \leq 1$,*

$$|\Pr\left[S_n \leq 1\right] - \Phi(1)| \leq C L_n^{1/5}, \tag{2}$$

where Φ is the standard unit normal distribution with mean 0 and variance 1.

If we are interested only in the tails of the distribution of S_n, we can get a tighter bound using Azuma's inequality, a martingale analogue of the standard Chernoff bound [9] for sums of independent random variables. The usual form of this bound (see [2, 25]) assumes that the difference variables X_i satisfy $|X_i| \leq 1$. This restriction is too severe for our purposes, so below we prove a generalization of the inequality. In order to do so we will need the following technical lemma.

LEMMA 4.4. *Let $\{S_i, \mathcal{F}_i\}, 1 \leq i \leq n$ be a zero-mean martingale with difference sequence $\{X_i\}$. Let $\mathcal{F}_0 \subseteq \mathcal{F}_1$ be a (not necessarily trivial) σ-algebra such that $\mathrm{E}\left[S_1 \mid \mathcal{F}_0\right] = 0$. If there exists a sequence of random variables $w_1, w_2, \ldots w_n$, and a random variable W, such that*

1. *W is $\mathcal{F}_0$-measurable,*
2. *Each w_i is $\mathcal{F}_{i-1}$-measurable,*
3. *For all i, $|X_i| \leq w_i$ with probability 1, and*
4. *$\sum_{i=1}^n w_i^2 \leq W$ with probability 1,*

then for any $\alpha > 0$,

$$\mathrm{E}\left[e^{\alpha S_n} \mid \mathcal{F}_0\right] \leq e^{\alpha^2 W/2} \tag{3}$$

Proof. The proof is by induction on n. First, notice that since $e^{\alpha X_1}$ is convex we have

$$e^{\alpha X_1} \leq \left(\frac{w_1 - X_1}{2w_1}\right) e^{-\alpha w_1} + \left(1 - \frac{w_1 - X_1}{2w_1}\right) e^{\alpha w_1},$$

and thus

$$\begin{aligned}
\mathrm{E}\left[e^{\alpha X_1} \mid \mathcal{F}_0\right] &\leq \mathrm{E}\left[\left(\frac{w_1 - X_1}{2w_1}\right) e^{-\alpha w_1} + \left(1 - \frac{w_1 - X_1}{2w_1}\right) e^{\alpha w_1} \mid \mathcal{F}_0\right] \\
&= \frac{1}{2}e^{-\alpha w_1} + \frac{1}{2}e^{\alpha w_1} - \left(\frac{e^{-\alpha w_1} - e^{\alpha w_1}}{2w_1}\right) \mathrm{E}\left[X_1 \mid \mathcal{F}_0\right] \\
&= \frac{1}{2}e^{-\alpha w_1} + \frac{1}{2}e^{\alpha w_1}
\end{aligned}$$

since $\mathrm{E}\left[X_1 \mid \mathcal{F}_0\right]$ is zero.

But then

$$\mathrm{E}\left[e^{\alpha X_1} \mid \mathcal{F}_0\right] \leq \frac{1}{2}\left(e^{-\alpha w_1} + e^{\alpha w_1}\right) = \cosh \alpha w_1 \leq e^{\alpha^2 w_1^2/2}.$$

If $n = 1$ we are done, since $w_1^2 \leq W$. If n is greater than 1, for each $i \leq n-1$ let $S'_i = S_{i+1} - X_1$ and $\mathcal{F}'_i = \mathcal{F}_{i+1}$. Then $\{S'_i, \mathcal{F}'_i\}, 1 \leq i \leq n-1$ satisfies the conditions of the lemma with $\mathcal{F}'_0 = \mathcal{F}_1$, $w'_i = w_{i+1}$ and $W' = W - w_1^2$, so by the induction hypothesis $\mathrm{E}\left[e^{\alpha S'_{n-1}} \mid \mathcal{F}'_0\right] \leq e^{\alpha^2(W-w_1^2)/2}$. But then, using the fact that $\mathrm{E}[X \mid \mathcal{F}] = \mathrm{E}[\mathrm{E}[X \mid \mathcal{F}'] \mid \mathcal{F}]$ when $\mathcal{F} \subseteq \mathcal{F}'$, we can compute:

$$\begin{aligned}
\mathrm{E}\left[e^{\alpha S_n} \mid \mathcal{F}_0\right] &= \mathrm{E}\left[\mathrm{E}\left[e^{\alpha X_1} e^{\alpha(S_n - X_1)} \mid \mathcal{F}_1\right] \mid \mathcal{F}_0\right] \\
&= \mathrm{E}\left[e^{\alpha X_1}\, \mathrm{E}\left[e^{\alpha S'_{n-1}} \mid \mathcal{F}'_0\right] \mid \mathcal{F}_0\right] \\
&\leq \mathrm{E}\left[e^{\alpha X_1} e^{\alpha^2(W-w_1^2)/2} \mid \mathcal{F}_0\right] \\
&= e^{\alpha^2(W-w_1^2)/2}\, \mathrm{E}\left[e^{\alpha X_1} \mid \mathcal{F}_0\right] \\
&\leq e^{\alpha^2(W-w_1^2)/2} e^{\alpha^2 w_1^2/2} \\
&= e^{\alpha^2 W/2}.
\end{aligned}$$

□

THEOREM 4.5. *Let $\{S_i, \mathcal{F}_i\}, 1 \leq i \leq n$ be a zero-mean martingale with difference sequence $\{X_i\}$. If there exists a sequence of random variables $w_1, w_2, \ldots w_n$, and a constant W, such that*

1. *Each w_i is $\mathcal{F}_{i-1}$-measurable.*
2. *For all i, $|X_i| \leq w_i$ with probability 1, and*
3. *$\sum_{i=1}^n w_i^2 \leq W$ with probability 1,*

then for any $\lambda > 0$,

$$\Pr[S_n \geq \lambda] \leq e^{-\lambda^2/2W}. \tag{4}$$

Proof. By Lemma 4.4, for any $\alpha > 0$, $\mathrm{E}\left[e^{\alpha S_n}\right] \leq e^{\alpha^2 W/2}$. Thus by Markov's inequality

$$\Pr[S_n \geq \lambda] = \Pr\left[e^{\alpha S_n} \geq e^{\alpha\lambda}\right] \leq e^{\alpha^2 W/2} e^{-\alpha\lambda}.$$

Setting $\alpha = \lambda/W$ gives (4). □

Symmetry immediately gives us:

COROLLARY 4.6. *For any martingale $\{S_i, \mathcal{F}_i\}$ satisfying the premises of Theorem 4.5, and any $\lambda > 0$*

$$\Pr[S_n \leq -\lambda] \leq e^{-\lambda^2/2W}. \tag{5}$$

Proof. Replace each S_i by $-S_i$ and apply Theorem 4.5. □

5. Proof of correctness. For this section we will fix a particular scheduler. We may assume without loss of generality that the scheduler is deterministic, because any random inputs the scheduler might use cannot depend on the history of the execution and therefore may also be fixed in advance.

Consider the sequence of random variables $X_1, X_2, \ldots$ where X_i represents the i-th vote that is decided by some processor executing line 7, or 0 if fewer than i local coin flips occur. Note that the notion of the i-th vote is well-defined since we model concurrency by interleaving; it is assumed that the scheduler advances processors one

at a time. For each i let $\mathcal{F}_i$ be $\langle X_1 \ldots X_i \rangle$, the σ-algebra generated by X_1 through X_i. Because the scheduler is deterministic, all of the random events in the system preceding the i-th vote are captured in the variables X_1 through X_{i-1}, and the σ-algebra $\mathcal{F}_{i-1}$ thus determines the entire history of the system up to but not including the i-th vote. Furthermore, since the scheduler's behavior depends only on the history of the system, $\mathcal{F}_{i-1}$ in fact determines the scheduler's choice of which processor will cast the i-th vote. Thus conditioned on $\mathcal{F}_{i-1}$, X_i is just a random variable which takes on the values $\pm w$ with equal probability for some weight w determined by the scheduler's choice of which processor to run. Hence $\mathrm{E}[X_i \mid \mathcal{F}_{i-1}] = 0$, and the sequence of partial sums $S_i = \sum_{j=1}^{i} X_i$ is a martingale relative to $\{\mathcal{F}_i\}$.

We are not going to analyze $\{S_i, \mathcal{F}_i\}$ directly. Instead, it will be used as a base on which other martingales will be built using Theorem 4.1.

Let $\kappa_i = 1$ if $\sum_{j=1}^{i} X_j^2 \leq K$ and 0 otherwise. Votes for which $\kappa_i = 1$ will be called **common votes**. For each processor P let $\zeta_i^P = 1$ if the vote X_i occurs before P reads, during its final read in line 13, the register of the processor that determines the value of X_i, and let $\zeta_i^P = 0$ otherwise. In effect, ζ_i^P is the indicator variable for whether P would see X_i if it were written out immediately. Observe that for a fixed scheduler the values of both κ_i and ζ_i^P can be determined by examining the history of the system up to but not including the time when the vote X_i is cast, and thus both κ_i and ζ_i^P are $\mathcal{F}_{i-1}$-measurable. Consequently the sequences $\left\{\sum_{j=1}^{i} \kappa_j X_j\right\}$ and $\left\{\sum_{j=1}^{i} \zeta_j^P X_j\right\}$ are martingales relative to $\{\mathcal{F}_i\}$ by Theorem 4.1. Votes for which $\zeta_i^P = 1$ but $\kappa_i = 0$ will be referred to as the **extra votes** for processor P. (Observe that $\zeta_i^P \geq \kappa_i$ since P could not have started its final read until the total variance was at least K.) The sequence $\left\{\sum_{j=1}^{i} (\zeta_i^P - \kappa_i) X_i\right\}$ of the partial sums of these extra votes is a difference of martingales and is thus also a martingale relative to $\{\mathcal{F}_i\}$.

The structure of the proof of correctness is as follows. First, we observe that the distribution of the total common vote, $\sum \kappa_i X_i$, is close to a normal distribution with mean 0 and variance K for suitable choices of a and K; in particular, we show that for n sufficiently large, the probability that $\sum \kappa_i X_i > \sqrt{K}$ will be at least a constant. Next, we complete the proof by showing that if the total common vote is far from the origin the chances that any processor will read a total vote whose sign differs from the common vote is small. This fact is itself shown in two steps. First, it is shown that, for suitable choice of c, the total of the extra votes for a processor P, $\sum (\zeta_i^P - \kappa_i) X_i$, will be small with high probability. Second, a bound Δ is derived on the difference between $\sum \zeta_i^P X_i$ and the total vote actually read by P.

It will be necessary to select values for a, K, and c that give the correct bounds on the probabilities. However, we will be in a better position to justify our choice for these parameters after we have developed more of the analysis, so the choice of parameters will be deferred until Section 5.5.

5.1. Phases of the protocol. We begin by defining the phases of the protocol more carefully. Let t_i be the value of the i-th processor's internal variable t at any given step of the protocol. Let U_i be the random variable representing the maximum value of t_i during the entire execution of the protocol. Let T_i be the random variable representing the maximum value of t_i during the part of the execution of the protocol where $\kappa = 1$.

In the proof of correctness we will encounter many quantities of the form $\sum_{i=1}^{n} \chi(T_i)$ or $\sum_{i=1}^{n} \chi(U_i)$ for various functions χ. We will want to get bounds on these quanti-

ties without having to look too closely at the particular values of each T_i or U_i. This section proves several very general inequalities about quantities of this form, all of which are ultimately based on the following constraint:

$$K \geq \sum_i \sum_{j=1}^{T_i} j^{2a} \geq \sum_i \int_0^{T_i} j^{2a}\, dj = \sum_i \frac{{T_i}^{2a+1}}{2a+1}. \tag{6}$$

The constant $2a+1$ will reappear often; for convenience we will write it as A. As noted above, $a \geq 0$, and hence $A \geq 1$.

Define $T_K = \left(\frac{AK}{n}\right)^{1/A}$, so that $K = \frac{nT_K^A}{A}$. The constant T_K represents the maximum value of each T_i if they are set to be equal while satisfying inequality (6). Note that T_K need not be an integer. Now we can show:

LEMMA 5.1. *Let $\psi(x) = x^A/A$ and let χ be any strictly increasing function such that $\chi\psi^{-1}$ is concave. Then for any non-negative $\{x_i\}$, if $\sum_{i=1}^n \psi(x_i) \leq K$, then $\sum_{i=1}^n \chi(x_i) \leq n\chi(T_K)$.*

Proof. Since $\chi\psi^{-1}$ is concave, we have

$$\chi^{-1}\left(\sum \frac{\chi(x_i)}{n}\right) \leq \psi^{-1}\left(\sum \frac{\psi(x_i)}{n}\right)$$

[19, Theorem 92]. Simple algebraic manipulation yields

$$\sum \chi(x_i) \leq n\chi\left(\psi^{-1}\left(\sum \frac{\psi(x_i)}{n}\right)\right)$$

But

$$\psi^{-1}\left(\sum \frac{\psi(x_i)}{n}\right) = \psi^{-1}\left(\frac{1}{n}\sum \frac{{x_i}^A}{A}\right) \leq \psi^{-1}\left(\frac{K}{n}\right) = T_K.$$

Hence $\sum \chi(x_i) \leq n\chi(T_K)$. □

Letting χ be the identity function we have $\chi\psi^{-1}(x) = (Ax)^{1/A}$, which is concave for $A \geq 1$. Hence:

COROLLARY 5.2.

$$\sum_{i=1}^n T_i \leq nT_K. \tag{7}$$

In the case where $\chi\psi^{-1}$ is convex, the following lemma applies instead:

LEMMA 5.3. *Let $\psi(x) = x^A/A$ and let χ be any strictly increasing function such that $\chi\psi^{-1}$ is convex. Then for any non-negative $\{x_i\}$, if $\sum_{i=1}^n \psi(x_i) \leq K$, then $\sum_{i=1}^n \chi(x_i) \leq (n-1)\chi(0) + \chi(n^{1/A}T_K)$.*

Proof. Let $Y = \sum \psi(x_i)$. Now $\chi(x_i) = \chi\psi^{-1}\psi(x_i)$ or

$$\chi\psi^{-1}\left(\left(1 - \frac{\psi(x_i)}{Y}\right)0 + \frac{\psi(x_i)}{Y}Y\right)$$

which is at most

$$\left(1 - \frac{\psi(x_i)}{Y}\right)\chi\psi^{-1}(0) + \frac{\psi(x_i)}{Y}\chi\psi^{-1}(Y)$$

given the convexity of $\chi\psi^{-1}$. Hence

$$\begin{aligned}\sum_{i=1}^{n}\chi(x_i) &\leq n\chi\psi^{-1}(0) - \left(\sum_{i=1}^{n}\frac{\psi(x_i)}{Y}\right)\chi\psi^{-1}(0) + \left(\sum_{i=1}^{n}\frac{\psi(x_i)}{Y}\right)\chi\psi^{-1}(Y)\\ &= (n-1)\chi\psi^{-1}(0) + \chi\psi^{-1}\left(\sum_{i=1}^{n}\psi(x_i)\right)\\ &\leq (n-1)\chi\psi^{-1}(0) + \chi\psi^{-1}(K)\end{aligned}$$

which is just $(n-1)\chi(0) + \chi\left(n^{1/A}T_K\right)$. □

The quantity $n^{1/A}T_K$ is the maximum value that any x_i can take on without violating the constraint on $\sum x_i$. So what Lemma 5.3 says is that if $\chi\psi^{-1}$ is convex, $\sum\chi(x_i)$ is maximized by maximizing one of the x_i while setting the rest to zero.

For the variables U_i we can show:

LEMMA 5.4. *Let $\psi(x) = x^A/A$ and let χ be any strictly increasing function such that $\chi(\psi^{-1}(x)+c+1)$ is concave in x. Then,*

$$\sum_{i=1}^{n}\chi(U_i) \leq n\chi(T_K + c + 1) \tag{8}$$

Proof. Let W_i be the number of votes *written* to the registers during the part of the execution where the total of the register variance fields is less than or equal to K. The set of variables $\{W_i\}$ satisfies the inequality $\sum W_i^A/A \leq K$ using the same argument as gives (6). Furthermore $U_i \leq W_i + 1 + c$, because after the i-th processor's next vote the total variance in the registers must exceed K and it can cast at most c more votes before noticing this fact. Define $\chi'(x) = \chi(x+c+1)$. Then $\chi(U_i) \leq \chi(W_i + c + 1) = \chi'(W_i)$. But ψ, χ', W_i satisfy the premises of Lemma 5.1 and thus $\sum_{i=1}^{n}\chi(U_i) \leq \sum_{i=1}^{n}\chi'(W_i) \leq n\chi'(T_K) = n\chi(T_K + c + 1)$. □

Setting χ to be the identity function gives

COROLLARY 5.5.

$$\sum_{i=1}^{n} U_i \leq n(T_K + c + 1) \tag{9}$$

Proof. $\chi(\psi^{-1}(x) + c + 1) = Ax^{1/A} + c + 1$, which is concave since $A \geq 1$. □

Define $g = 1 + \frac{c+3}{T_K}$; then $gT_K = T_K + c + 3$ will be an upper bound for $T_K + c + 1$ as well as a number of closely related constants involving c that will appear later.

5.2. Common votes. The purpose of this section is to show that for n sufficiently large, the total common vote is far from the origin with constant probability. We do so by showing that under the right conditions the total common vote will be nearly normally distributed.

Let $S_i^K = \sum_{j=1}^{i}\kappa_j X_j$. As pointed out above, $\left\{S_i^K = \sum_{j=1}^{i}\kappa_j X_j, \mathcal{F}_i\right\}$ is a martingale. Let $N = \lceil nT_K\rceil$. It follows from Corollary 5.2 that $\kappa_i = 0$ for $i > N$ and thus $S_N^K = \lim_{i\to\infty} S_i^K$ is the sum of all the common votes. The distribution of S_N^K is characterized in the following lemma.

LEMMA 5.6. *If*

$$\frac{4A^2}{n^{1/A}T_K} \leq 1, \tag{10}$$

then

$$\left|\Pr\left[S_N^K \le \sqrt{K}\right] - \Phi(1)\right| \le C_1 \left(\frac{A^2}{n^{1/A}T_K}\right)^{1/5} \tag{11}$$

where C_1 is an absolute constant.

Proof. The proof uses Theorem 4.3, which requires that the martingale be normalized so that the total conditional variance V_N^2 is close to 1. So let $Y_i = \frac{\kappa_i X_i}{\sqrt{K}}$ and consider the martingale $\left\{\sum_{j=1}^{i} Y_j, \mathcal{F}_i\right\}$. To apply the theorem we need to compute a bound on the value L_N.

We begin by getting a bound on the first term $\sum \mathrm{E}\left[|Y_i|^4\right]$. We have

$$\sum_{i=1}^{N} \mathrm{E}\left[|Y_i|^4\right] = \mathrm{E}\left[\sum_{i=1}^{N} |Y_i|^4\right] = \frac{1}{K^2}\,\mathrm{E}\left[\sum_{i=1}^{N} |\kappa_i X_i|^4\right] = \frac{1}{K^2}\,\mathrm{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{T_i} j^{4a}\right] \tag{12}$$

Now,

$$\sum_{j=1}^{T_i} j^{4a} \le \int_0^{T_i} j^{4a}\,dj + T_i^{4a} = \frac{T_i^{4a+1}}{4a+1} + T_i^{4a}.$$

Consider the two parts of this bound separately. Define $\psi(x) = x^A/A, \chi(x) = \frac{x^{4a+1}}{4a+1}$, then $\chi\psi^{-1}(y) = \frac{(Ay)^{(4a+1)/A}}{4a+1}$ is convex, $\chi(0) = 0$, and hence $\sum_{i=1}^{n} \frac{T_i^{4a+1}}{4a+1}$ is at most $\frac{(n^{1/A}T_K)^{4a+1}}{4a+1}$ using Lemma 5.3.

Similarly, let $\chi'(x) = x^{4a}$. Here the convexity of $\chi'\psi^{-1}$ depends on the value of a. If $a \ge \frac{1}{2}$ then $\chi'\psi^{-1}(y) = (Ay)^{4a/A}$ is convex (since $4a/A = 4a/(2a+1) \ge 1$), and thus (again by Lemma 5.3) $\sum_{i=1}^{n} T_i^{4a} \le (n^{1/A}T_K)^{4a} = n^{4a/A}T_K^{4a} \le n^{(4a+1)/A}T_K^{4a}$. If $a \le \frac{1}{2}$ then $\chi'\psi^{-1}(y)$ is concave (since now $4a/A \le 1$), and thus by Lemma 5.1 $\sum_{i=1}^{n} T_i^{4a} \le nT_K^{4a} \le n^{(4a+1/A)}T_K^{4a}$.

Plugging everything back into (12) gives

$$\sum_{i=1}^{N} \mathrm{E}\left[|Y_i|^4\right] \le \frac{n^{(4a+1)/A}T_K^{4a}}{K^2} + \frac{(n^{1/A}T_K)^{4a+1}}{K^2(4a+1)}. \tag{13}$$

For the second term $\mathrm{E}\left[|V_N^2 - 1|^2\right]$, observe that

$$V_N^2 = \sum_{i=1}^{N} \mathrm{E}\left[Y_i^2 \mid \mathcal{F}_{i-1}\right] = \frac{1}{K}\sum_{i=1}^{N} \mathrm{E}\left[(\kappa_i X_i)^2 \mid \mathcal{F}_{i-1}\right],$$

which is just $1/K$ times the sum of the squares of the weights $|X_i|$ of the common votes. But the total variance of the common votes can differ from K by at most the variance of the first vote X_i for which $\kappa_i = 0$. Since the processor that casts this vote can have cast at most $n^{1/A}T_K$ votes beforehand, the variance of this vote is at most $\left(n^{1/A}T_K + 1\right)^{2a}$, giving the bound

$$|V_N^2 - 1|^2 \le \frac{1}{K^2}\left(n^{1/A}T_K + 1\right)^{4a}. \tag{14}$$

Combining (13) and (14) gives

$$\begin{aligned} L_N &\leq \frac{n^{(4a+1)/A}T_K^{4a}}{K^2} + \frac{(n^{1/A}T_K)^{4a+1}}{K^2(4a+1)} + \frac{\left(n^{1/A}T_K+1\right)^{4a}}{K^2} \\ &= \frac{n^{(4a+1)/A}T_K^{4a}}{K^2} + \frac{n^{(4a+1)/A}T_K^{4a+1}}{K^2(4a+1)} + \frac{n^{4a/A}T_K^{4a}(1+n^{-1/A}T_K^{-1})^{4a}}{K^2} \\ &\leq A^2 n^{-1/A}T_K^{-2} + \frac{A^2 n^{-1/A}T_K^{-1}}{4a+1} + A^2 n^{-2/A}T_K^{-2}\exp(4an^{-1/A}T_K^{-1}) \\ &\leq 2A^2 n^{-1/A}T_K^{-1} + e^{1/2}A^2 n^{-1/A}T_K^{-1} \\ &< \frac{4A^2}{n^{1/A}T_K} \end{aligned}$$

The third-to-last step uses the approximation $(1+x)^b \leq e^{bx}$ for non-negative b and x. The resulting exponential term is serendipitously bounded by $e^{1/2}$ if (10) holds, since $2a < A \leq A^2$ implies $4an^{-1/A}T_K^{-1} < 2A^2(n^{1/A}T_K)^{-1} \leq 2/4$.

A more direct application of (10) shows that $L_N \leq 1$, and thus Theorem 4.3 applies. Hence

$$\begin{aligned} \left|\Pr\left[\sum \kappa_i X_i \leq \sqrt{K}\right] - \Phi(1)\right| &= \left|\Pr\left[\sum_{i=1}^{N} Y_i \leq 1\right] - \Phi(1)\right| \\ &\leq C\left(\frac{4A^2}{n^{1/A}T_K}\right)^{1/5} \\ &\leq C_1\left(\frac{A^2}{n^{1/A}T_K}\right)^{1/5}. \end{aligned}$$

□

5.3. Extra votes. In this section we examine the extra votes from the point of view of a particular processor P.

Recall that ζ_i^P is defined to be 1 if the vote X_i is cast by some processor Q before P's final read of Q's register and 0 otherwise. Clearly, $\zeta_i^P \geq \kappa_i$ since P could not have started its final read until the total variance exceeded K. As discussed above, both ζ_i^P and κ_i are $\mathcal{F}_{i-1}$-measurable. Thus $\xi_i = \zeta_i^P - \kappa_i$ is a $0-1$ random variable that is $\mathcal{F}_{i-1}$-measurable, and $\left\{S_i^P = \sum_{j=1}^{i} \xi_j X_j, \mathcal{F}_i\right\}$ is a martingale by Theorem 4.1.

Define $\Delta = n(gT_K)^a$. The following lemma shows a bound on the tails of $\sum \xi_i X_i$.

LEMMA 5.7. *If*

$$g^a \leq \frac{1}{2}\sqrt{\frac{T_K}{nA}}, \tag{15}$$

and

$$g^A \leq 1 + \frac{1}{8\log(10n)}, \tag{16}$$

[8]

then for each processor P,

$$\Pr\left[\sum(\zeta_i^P - \kappa_i)X_i \leq \Delta - \sqrt{K}\right] \leq \frac{1}{10n}. \tag{17}$$

[8] By $\log(x)$ we will always mean the natural logarithm of x.

Proof. The proof uses Corollary 4.6, so we proceed by showing that its premises (stated in Theorem 4.5) are satisfied for $\{\sum \xi_i X_i, \mathcal{F}_i\}$.

By Corollary 5.5, X_i and thus $\xi_i X_i$ is zero for $i > n(T_K + c + 1)$. So $\sum \xi_i X_i = S_M^P$ where $M = n(T_K + c + 1)$.

Set $w_i = |\xi_i X_i|$. Then the first premise of Corollary 4.6 follows from the fact that for each i, ξ_i and $|X_i|$ are both $\mathcal{F}_i$-measurable. The second premise is immediate. For the third premise, notice that

$$\sum (|\xi_i X_i|)^2 = \sum \xi_i X_i^2 = \sum \zeta_i^P X_i^2 - \sum \kappa_i X_i^2 \leq \sum X_i^2 - \sum \kappa_i X_i^2.$$

The first term is

$$\sum X_i^2 = \sum_{i=1}^{n} \sum_{j=1}^{U_i} j^{2a}.$$

The second term is

$$\sum \kappa_i X_i^2 \geq K - t^{2a}$$

for some t which is at most U_i for some i. Thus

$$\begin{aligned} \sum (|\xi_i X_i|)^2 &\leq -K + t^{2a} + \sum_{i=1}^{n} \sum_{j=1}^{U_i} j^{2a} \\ &< -K + \sum_{i=1}^{n} \sum_{j=1}^{U_i+1} j^{2a} \\ &\leq -K + \sum_{i=1}^{n} (U_i + 2)^A / A. \end{aligned} \tag{18}$$

Let $\chi(x) = (x+2)^A / A$. Then

$$\chi\left(\psi^{-1}(y) + c + 1\right) = \frac{\left((Ay)^{1/A} + c + 3\right)^A}{A} \tag{19}$$

We can treat this function as an instance of a class of functions of the form $(x^p + C)^q$, where x, p, q, C are all non-negative, whose concavity (or lack thereof) can be determined by finding the sign of the second derivative:

$$\begin{aligned} \operatorname{sgn}\left[\frac{d^2}{dx^2}(x^p + C)^q\right] &= \operatorname{sgn}\left[\frac{d}{dx} q(x^p + C)^{q-1} p x^{p-1}\right] \\ &= \operatorname{sgn}\left[q(q-1)(x^p + C)^{q-2} p^2 x^{2p-2} + q(x^p + C)^{q-1} p(p-1) x^{p-2}\right] \\ &= \operatorname{sgn}\left[q(x^p + C)^{q-2} p x^{p-2} \left[(q-1)px^p + (x^p + C)(p-1)\right]\right] \\ &= \operatorname{sgn}\left[(q-1)px^p + (x^p + C)(p-1)\right] \\ &= \operatorname{sgn}\left[(pq-1)x^p + C(p-1)\right] \end{aligned}$$

In the particular case we are interested in, $p = 1/A$, $q = A$, and $C = c + 3$. Since $pq - 1 = 0$ the first term vanishes and the sign is equal to the sign of $1/A - 1$, which is

less than or equal to zero since $A \geq 1$. Thus the function $\left(x^{1/A} + c + 3\right)^A$ is concave, and since concavity is preserved by linear transformations $\frac{\left((Ay)^{1/A}+c+3\right)^A}{A}$ is concave as well.

Lemma 5.4 now gives

$$\sum_{i=1}^{n} \frac{(U_i+2)^A}{A} \leq n\chi(T_K + c + 1) = \frac{n(T_K + c + 3)^A}{A} \leq \frac{n(gT_K)^A}{A}. \tag{20}$$

It follows from (18) and (20) that

$$\sum(|\xi_i X_i|)^2 \leq \frac{n(gT_K)^A}{A} - K = K(g^A - 1)$$

Applying (5) from Corollary 4.6 now yields, for all $\lambda > 0$,

$$\Pr\left[S_M^P \leq -\lambda\right] \leq e^{-\lambda^2/(2K(g^A-1))}. \tag{21}$$

If (15) holds, then $\Delta \leq \frac{\sqrt{K}}{2}$. So

$$\begin{aligned} \Pr\left[\sum \xi_i X_i \leq \Delta - \sqrt{K}\right] &\leq \Pr\left[S_M^P \leq -\frac{\sqrt{K}}{2}\right] \\ &\leq e^{-K/(8K(g^A-1))} \\ &= e^{-1/(8(g^A-1))}. \end{aligned}$$

But if (16) holds then

$$g^A - 1 \leq \frac{1}{8\log(10n)}$$

and, since $\log(10n) > 0$ and $g > 1$,

$$-\frac{1}{8(g^A-1)} \leq -\log(10n)$$

from which it follows that

$$e^{-1/8(g^A-1)} \leq e^{-\log(10n)} = \frac{1}{10n}.$$

□

5.4. Written votes vs. decided votes. In this section we show that the difference between $\sum \zeta_i^P X_i$ and the total vote actually read by P is bounded by $\Delta = n(gT_K)^a$.

LEMMA 5.8. *Let R_P be the sum of the votes read during P's final read. Then*

$$\left|\sum \zeta_i^P X_i - R_P\right| \leq n(T_k + c + 1)^a \leq n(gT_K)^a = \Delta \tag{22}$$

Proof. Suppose $\zeta_i^P = 1$, and suppose X_i is decided by processor P_j. If the vote X_i is not included in the value read by P, it must have been decided before P's read of P_j's register but written afterwards. Because each vote is written out before

the next vote is decided there can be at most one vote from P_j which is included in $\sum \zeta_i^P X_i$ but is not actually read by P. This vote has weight at most U_j^a. So we have $\left|\sum \zeta_i^P X_i - R_P\right| \leq \sum_{i=1}^n U_i^a$.

Now let $\chi(x) = x^a$. Then $\chi\left(\psi^{-1}(y) + c + 1\right) = \left((Ay)^{1/A} + c + 1\right)^a$. The concavity of this function can be shown using the argument applied to (19) in Lemma 5.7: the sign of its second derivative will be equal to the sign of $(pq-1)x^p + C(p-1)$ where $x = Ay$, $p = 1/A$, $q = a$, and $C = c + 1$. Since Ay and $c + 1$ are both non-negative and a/A and $1/A$ are both less than or equal to 1, both terms are non-positive and thus $\left((Ay)^{1/A} + c + 1\right)^a$ is concave. The rest follows from Lemma 5.4. □

5.5. Choice of parameters. Let us summarize the proof of correctness in a single theorem:

THEOREM 5.9. *Define*

$$\begin{aligned} A &= 2a + 1 \\ T_K &= \left(\frac{AK}{n}\right)^{1/A} \\ g &= 1 + \frac{c+3}{T_K} \end{aligned}$$

and suppose that all of the following hold:

$$g^a \leq \frac{1}{2}\sqrt{\frac{T_K}{nA}} \tag{23}$$

$$g^A \leq 1 + \frac{1}{8\log(10n)} \tag{24}$$

$$\frac{4A^2}{n^{1/A}T_K} \leq 1 \tag{25}$$

Then the protocol implements a shared coin with agreement parameter at least

$$1 - \left[\Phi(1) + C_1\left(\frac{A^2}{n^{1/A}T_K}\right)^{1/5} + 1/10\right] \tag{26}$$

where C_1 is the constant from Lemma 5.6.

Proof. To show that the agreement parameter is at least (26) we must show that for each $z \in \{0, 1\}$ the probability that all processors decide z is at least (26). Without loss of generality let us consider only the probability that all processors decide 1; the case of all processors deciding 0 follows by symmetry.

The essential idea of the proof is as follows. With at least a constant probability, the total common vote is at least $\sqrt{K}$ (Lemma 5.6). The "drift" added to this total by the extra votes for any single processor P is small with high probability (Lemma 5.7). Thus even after adding in the extra votes for P, the total will be large enough that the offset $\Delta = n(gT_K)^a$ caused by votes that are generated but not written out in time for P's final read will not push it over the line (Lemma 5.8).

More formally, we wish to show that the event

- $\sum \kappa_i X_i > \sqrt{K}$, and
- For each P, $\sum(\zeta_i^P - \kappa_i)X_i > \Delta - \sqrt{K}$

occurs with probability at least (26). Since this event implies that for all P, $\sum \zeta_i^P X_i > \Delta$, by Lemma 5.8 we have that each P reads a value greater than 0 during its final read and thus decides 1.

It will be easiest to compute an upper bound on the probability that this event does *not* occur. For the event not to occur, we must have either $\sum \kappa_i X_i \le \sqrt{K}$ or $\sum(\zeta_i^P - \kappa_i)X_i \le \Delta - \sqrt{K}$ for some P. But as the probability of a union of events never exceeds the sum of the probabilities of the events, the probability of failing in any of these ways is at most

$$\begin{aligned} & \Pr\left[\sum \kappa_i X_i \le \sqrt{K}\right] + \sum_P \Pr\left[\sum(\zeta_i^P - \kappa_i)X_i \le \Delta - \sqrt{K}\right] \\ \le\ & \left[\Phi(1) + C_1\left(\frac{A^2}{n^{1/A}T_K}\right)^{1/5}\right] + n\frac{1}{10n} \end{aligned} \tag{27}$$

by Lemmas 5.6 and 5.7. So the probability that some processor decides 0 is at most (27), and thus the probability that all processors decide 1 is at least 1 minus (27). □

The running time of the protocol is more easily shown:

THEOREM 5.10. *No processor executes more than* $(AK)^{1/A}(2 + n/c) + 2c + 2n$ *register operations during an execution of the shared coin protocol.*

Proof. First consider the maximum number of votes a processor can cast. After $(AK)^{1/A}$ votes the total variance of the processor's votes will be

$$\sum_{x=1}^{(AK)^{1/A}} x^{2a} > \int_0^{(AK)^{1/A}} x^{2a}\,dx = \frac{\left((AK)^{1/A}\right)^A}{A} = K,$$

so after at most an additional c votes the processor will execute line 11 of Figure 1 and see a total variance greater than K. Thus each processor casts at most $(AK)^{1/A} + c$ votes. But each vote costs 1 write operation in line 8, and every c votes costs n reads in line 11, to which must be added a one-time cost of n reads in line 13. The total number of operations is thus at most $\left((AK)^{1/A} + c\right)\left(1 + \lceil n/c \rceil\right) + n \le ((AK)^{1/A} + c)(2 + n/c) + n = (AK)^{1/A}(2 + n/c) + 2c + 2n$. □

It remains only to find values for a, K, and c which give both a constant agreement parameter and a reasonable running time. As a warm-up, let us consider what happens if we emulate the protocol of Bracha and Rachman [8]:

THEOREM 5.11. *If* $a = 0$, $K = 4n^2$, *and* $c = \frac{n}{4\log n} - 3$, *then for* n *sufficiently large the protocol implements a shared coin with agreement parameter at least* 0.05 *in which each processor executes at most* $O(n^2 \log n)$ *operations.*

Proof. For the agreement parameter, we have $A = 1$, $T_K = 4n$, and $g = 1 + 1/(16\log n)$. Then (23) holds since $g^a = 1 \le \frac{1}{2}\sqrt{T_K/nA} = 1$. Furthermore,

$$\begin{aligned} \left(1 + \frac{1}{8\log(10n)}\right)^{1/A} &= 1 + \frac{1}{8(\log n + \log 10)} \\ &\ge 1 + \frac{1}{16\log n} \end{aligned}$$

when $n \ge 10$. Thus (24) holds. The remaining inequality (25) holds for $n \ge 1$, so by Theorem 5.9 we have a probability of failure of at most

$$\begin{aligned} & \Phi(1) + C_1\left(\frac{1}{4n^2}\right)^{1/5} + 1/10 \\ & \le 0.842 + O\left(\frac{1}{n^{2/5}}\right) + 0.1 \end{aligned}$$

which is not more than $0.942 + \epsilon$ for n sufficiently large. In particular for n greater than some n_0 this quantity is at most 0.95, and the agreement parameter is thus at least $1 - 0.95$.

The running time is immediate from Theorem 5.10. □

Now consider what happens if a is not restricted to be a constant 0.

THEOREM 5.12. *If* $a = (\log n - 1)/2$, $K = (16n \log n)^{\log n}(n/\log n)$, *and* $c = (n/\log n) - 3$, *then for* n *sufficiently large the protocol implements a shared coin with constant agreement parameter in which each processor executes at most* $O(n \log^2 n)$ *operations.*

Proof. We have $A = \log n$, $T_K = 16n \log n$, and $g = 1 + \frac{1}{16 \log^2 n}$.

We want to apply Theorem 5.9, so first we verify that its premises are satisfied. To show (23), compute

$$g^a = \left(1 + \frac{1}{16 \log^2 n}\right)^{(\log n - 1)/2} \leq e^{(\log n - 1)/(32 \log^2 n)} \leq e^{1/(32 \log n)}$$

which for $n \geq 2$ will be less than $\frac{1}{2}\sqrt{T_K/nA} = 2$. To show (24), note that

$$g^A = \left(1 + \frac{1}{16 \log^2 n}\right)^{\log n} \leq e^{1/(16 \log n)}$$

and thus $\log(g^A) \leq 1/(16 \log n)$. But

$$\begin{aligned} \log\left(1 + \frac{1}{8 \log(10n)}\right) &\geq \frac{1}{8 \log(10n)} - \frac{1}{128 \log^2(10n)} \\ &= \frac{1}{8(\log n + \log 10)} - \frac{1}{128(\log n + \log 10)^2} \end{aligned}$$

(using the approximation $\log(1 + x) \geq x - \frac{1}{2}x^2$). For sufficiently large n this quantity exceeds $1/(16 \log n)$ and (24) holds. The remaining constraint (25) is easily verified, and thus Theorem 5.9 applies and the agreement parameter is at least

$$\begin{aligned} 1 - \left[\Phi(1) + C_1 \left(\frac{\log^2 n}{n^{1/\log n}(16n \log n)}\right)^{1/5} + 1/10\right] \\ \geq 1 - \left(0.842 + O\left(\left(\frac{\log n}{n}\right)^{1/5}\right) + 0.10\right) \end{aligned}$$

which is at least 0.05 for sufficiently large n. Thus the protocol gives a constant agreement parameter.

Now by Theorem 5.10, the number of operations executed by any single processor is at most $(AK)^{1/A}(2 + n/c) + 2c + 2n$, or

$$(\log n)^{1/\log n}(16n \log n)(n/\log n)^{1/\log n} O(\log n) + O(n)$$

which is $O(n \log^2 n)$. □

6. Discussion. This paper presents the first randomized consensus algorithm which achieves a nearly optimal worst-case bound on the expected number of operations a processor needs to execute. To achieve this we construct a weak shared coin protocol based on random voting where the weight of votes cast by a processor

increases with the number of votes it has already cast. The consensus protocol can then be constructed around it using the established techniques of Aspnes and Herlihy [4] with only a constant-factor increase in the number of operations done by each processor.[9]

This work leads to several interesting questions. First, our voting scheme implicitly gives higher priority to operations done by processors that have already performed many operations. Such implicit priority granting may yield faster algorithms for other shared memory problems, such as approximate agreement or randomized resource allocation.

Also, although our solution improves significantly on the worst-case expected bound on the number of operations a single processor is required to perform in order to achieve consensus, the total number of operations done by all of the processors together is slightly larger (by a factor of $\log n$) than in the unweighted-voting protocol of Bracha and Rachman [8]. It is of theoretical interest whether there is an inherent trade-off here.

7. Acknowledgments. We would like to thank Serge Plotkin and David Applegate for their many useful suggestions.

REFERENCES

[1] K. Abrahamson, *On achieving consensus using a shared memory*, in Proceedings of the Seventh ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Aug. 1988, pp. 291–302.

[2] N. Alon and J. H. Spencer, *The Probabilistic Method*, John Wiley and Sons, 1992.

[3] J. Aspnes, *Time- and space-efficient randomized consensus*, Journal of Algorithms, 14 (1993), pp. 414–431.

[4] J. Aspnes and M. Herlihy, *Fast randomized consensus using shared memory*, Journal of Algorithms, 11 (1990), pp. 441–461.

[5] H. Attiya, D. Dolev, and N. Shavit, *Bounded polynomial randomized consensus*, in Proceedings of the Eighth ACM Symposium on Principles of Distributed Computing, Aug. 1989, pp. 281–294.

[6] P. Billingsley, *Probability and Measure*, John Wiley and Sons, second ed., 1986.

[7] G. Bracha and O. Rachman, *Approximated counters and randomized consensus*, Tech. Report 662, Technion, 1990.

[8] ———, *Randomized consensus in expected $O(n^2 \log n)$ operations*, in Proceedings of the Fifth International Workshop on Distributed Algorithms, Springer-Verlag, 1991.

[9] H. Chernoff, *A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations*, Annals of Mathematical Statistics, 23 (1952), pp. 493–407.

[10] B. Chor, A. Israeli, and M. Li, *Wait-free consensus using asynchronous hardware.*, SIAM Journal on Computing, 23 (1994), pp. 701–712. Preliminary version appears in *Proceedings of the 6th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing,* pages 86-97, 1987.

[11] B. Chor and L. Moscovici, *Solvability in asynchronous environments*, in 30th Annual Symposium on Foundations of Computer Science, Oct. 1989, pp. 422–427.

[12] D. Dolev, C. Dwork, and L. Stockmeyer, *On the minimal synchronism needed for distributed consensus*, J. Assoc. Comput. Mach., 34 (1987), pp. 77–97.

[13] C. Dwork, M. Herlihy, S. Plotkin, and O. Waarts, *Time-lapse snapshots*, in Proceedings of Israel Symposium on the Theory of Computing and Systems, 1992.

[14] C. Dwork, M. Herlihy, and O. Waarts, *Bounded round numbers*, in Proceedings of the 12th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Aug. 1993, pp. 53–64.

[9] Due to the unbounded round structure of [4], the resulting consensus protocol assumes unbounded registers. We believe these unbounded registers can be eliminated using the bounded round numbers construction of Dwork, Herlihy and Waarts [14].

[15] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. 2, John Wiley and Sons, second ed., 1971.
[16] M. J. Fischer, N. A. Lynch, and M. S. Paterson, *Impossibility of distributed commit with one faulty process*, J. Assoc. Comput. Mach., 32 (1985), pp. 374–382.
[17] P. Hall and C. Heyde, *Martingale Limit Theory and Its Application*, Academic Press, 1980.
[18] P. R. Halmos, *Invariants of certain stochastic transformations: The mathematical theory of gambling systems*, Duke Mathematical Journal, 5 (1939), pp. 461–478.
[19] G. Hardy, J. Littlewood, and G. Pólya, *Inequalities*, Cambridge University Press, second ed., 1952.
[20] M. Herlihy, *Wait-free synchronization*, ACM Trans. Prog. Lang. Syst., 13 (1991), pp. 124–149.
[21] P. Kopp, *Martingales and Stochastic Integrals*, Cambridge University Press, 1984.
[22] M. C. Loui and H. H. Abu-Amara, *Memory requirements for agreement among unreliable asynchronous processes*, in Advances in Computing Research, F. P. Preparata, ed., vol. 4, JAI Press, 1987.
[23] S. A. Plotkin, *Sticky bits and universality of consensus*, in Proceedings of the Eighth ACM Symposium on Principles of Distributed Computing, Aug. 1989, pp. 159–176.
[24] M. Saks, N. Shavit, and H. Woll, *Optimal time randomized consensus — making resilient algorithms fast in practice*, in Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms, 1991, pp. 351–362.
[25] J. Spencer, *Ten Lectures on the Probabilistic Method*, Society for Industrial and Applied Mathematics, 1987.