Faster Randomized Consensus With an Oblivious Adversary

James Aspnes [*]
Yale University
aspnes@cs.yale.edu

ABSTRACT

Two new algorithms are given for randomized consensus in a shared-memory model with an oblivious adversary. Each is based on a new construction of a conciliator, an object that guarantees termination and validity, but that only guarantees agreement with constant probability. The first conciliator assumes unit-cost snapshots and achieves agreement among n processes with probability $1 - \epsilon$ in $O(\log^* n + \log(1/\epsilon))$ steps for each process. The second uses ordinary multi-writer registers, and achieves agreement with probability $1-\epsilon$ in $O(\log \log n + \log(1/\epsilon))$ steps. Combining these constructions with known results gives randomized consensus for arbitrarily many possible input values using unit-cost snapshots in $O(\log^* n)$ expected steps and randomized consensus for up to $O(\log n \log \log n)$ possible input values using ordinary registers in $O(\log \log n)$ expected steps.

Categories and Subject Descriptors

D.1.3 [**Programming Techniques**]: Concurrent Programming—*Distributed programming*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Theory, Algorithms

Keywords

Consensus, randomization, shared-memory, oblivious adversary

1. INTRODUCTION

In the **consensus** problem, a group of n processes wish to agree on a value, which must be equal to the input of some process. Consensus is known to be impossible to solve deterministically in an asynchronous message-passing [13] or shared-memory [17] model if even one process can fail. However, randomized algorithms can achieve wait-free consensus in bounded expected time.

The cost of consensus is strongly affected by the power of the **adversary scheduler** that chooses at each step which process should carry out the next operation. With an **adaptive adversary**, which can base its decision on the complete state of the system—including internal states of processes—the cost of consensus is well understood. A lower bound of Attiya and Censor [8] shows that $\Omega(n^2)$ expected total steps are needed even for binary consensus, where all inputs are 0 or 1, in a model that provides multi-writer multi-reader registers. Conversely, a matching upper bound was shown in the same paper, and subsequent work has demonstrated that adaptive-adversary consensus can be solved with optimal $O(n)$ expected individual step complexity [6], even if only single-writer registers are available and the inputs are arbitrary [3].

Less well understood is the complexity of randomized shared-memory consensus with an **oblivious adversary** that schedules the sequence of operations in advance without being able to observe the random choices made by the processes. Aumann [10] showed how to achieve $O(\log n)$ expected individual step complexity under a plausible weak-adversary assumption that holds for an oblivious adversary; and a more recent algorithm of Aspnes [4], based on a classic protocol of Chor, Israeli, and Li [12], simultaneously achieves $O(\log n)$ expected individual step complexity and $O(n)$ expected total step complexity under a different weak-adversary assumption.

It is clear that $O(n)$ expected total step complexity is optimal, as each process must take at least one step. A lower bound of Attiya and Censor [9] for oblivious-adversary consensus shows that the probability of failing to terminate in $O(kn)$ total steps is at least $\frac{1}{c^k}$ for some constant k, even with global coins and unit-cost snapshots. This gives a lower bound of $\Omega(n \log(1/\epsilon))$ total steps to reach agreement with probability at least $1 - \epsilon$. However, there is still the possibility that the *expected* individual step complexity even without these assumptions could be as low as $O(1)$.

We do not show such a surprising result here, although our results do show that the cost of oblivious-adversary consensus is much lower than might have been expected given the lack of improvement over the previous fifteen years. We give two algorithms for **conciliators** [4], weak consensus objects that guarantee termination and validity in all executions but guarantee agreement only with constant probability (a more formal definition is given in Section 1.2). These can

[*]Supported in part by NSF grant CCF-0916389.

be alternated with adopt-commit objects [2, 14, 18] to obtain consensus objects that guarantee agreement always, at an expected cost equal to the sum of the costs of the conciliator and the adopt-commit [4].

Our first conciliator, described in Section 2, works in the unit-cost snapshot model, and reaches agreement with constant probability after $O(\log^* n)$ operations per process. The main idea is to use the nesting property of snapshots and the properties of left-to-right maxima of random permutations to reduce an initial set of m values to $O(\log m)$ values on average in each round by assigning a random priority to each value and having each process take the value with highest priority among those it observes in a snapshot. After $O(\log^* m)$ iterations of this process, with constant probability only one value survives.

Because adopt-commit objects can be implemented using $O(1)$ snapshot operations [14], this immediately gives a randomized consensus protocol with $O(\log^* n)$ expected individual step complexity in the unit-cost snapshot model.

Our second conciliator, described in Section 3, works in the ordinary multi-writer register model, and uses a sequence of **sifting** rounds similar to those recently used by Alistarh and Aspnes [1] to implement test-and-set in $O(\log \log n)$ expected steps with an oblivious adversary. Here in each round a multi-writer register is used to eliminate values quickly. Each process chooses randomly whether to write its value to the register (and retain it for the next round), or to read the register and replace its value with the register value if it is not null. By choosing the probability of a write carefully, each round reduces the number of surviving values from m to $O(\sqrt{m})$ on average, giving a unique survivor after $O(\log \log n)$ rounds if all goes well. We give a new analysis of the sifting mechanism that shows that this indeed occurs with constant probability provided the write probabilities are chosen with the needs of the conciliator in mind.

Using this second conciliator gives $O(\log \log n)$ expected individual step complexity for randomized consensus in a multi-writer register model with an oblivious adversary, provided the range of input values is small. When the range of input values is large, the best currently known construction of an adopt-commit object [7] dominates the cost.

For both conciliators, because the oblivious adversary can't see processes' coin-flips or states, we can have each process generate a sequence of random bits associated with its input value, which then propagate along with the input value as it is adopted by other processes; we call this combination of an input value and random bits a **persona**. This allows all copies of a persona to be treated the same way in each round regardless of which processes hold them, making the number of surviving distinct personae– instead of the number of surviving processes—the relevant measure of progress.

A straightforward implementation of either conciliator leads to greater total expected step complexity than the optimal $O(n)$ bound achieved in [4]. We show (in Section 4) how to embed the $O(\log \log n)$ conciliator in a conciliator based on the consensus protocol of Chor, Israeli, and Li [12] to get a conciliator with both expected $O(\log \log n)$ individual step complexity and expected $O(n)$ total step complexity. The same technique also works for the $O(\log^* n)$ snapshot-based conciliator.

1.1 Model

We consider a standard asynchronous shared-memory model, in which n processes communicate by executing operations on shared-memory objects. These objects are either **snapshot objects**, which support an update operation and a read operation that returns a vector of the values of the most recent update operations for every process; or **atomic multi-writer multi-reader registers**, which support a write operation and a read operation that returns the value of the most recent write. In either case we treat all operations as taking one step. We do not assume any limitation on the size of registers.

Timing is controlled by an **oblivious adversary**, which specifies a **schedule** consisting of a sequence of process ids. At each step, the next process in the schedule executes one operation of its choosing. We assume that once a process has finished its protocol, any steps allocated to it become no-ops; these no-ops are not included when computing the complexity of the algorithm. Any coin-flips done by the processes are independent of the schedule chosen by the adversary.

1.2 Consensus, conciliators, and adopt-commit objects

In a **consensus** protocol, each process starts with an input and eventually decides on an output, subject to the requirements:

- **Termination.** With probably 1, each nonfaulty process decides after finitely many steps.
- **Validity.** The output value of each process is equal to the input value of some process.
- **Agreement.** All processes choose the same output value.

A conciliator [4] keeps the termination and validity conditions, but replaces agreement with:

- **Probabilistic agreement.** There is a fixed **agreement probability** $\delta > 0$ such that, for any adversary strategy, the probability that all return values are equal is at least δ.

The idea is that a conciliator attempts to produce agreement, but cannot guarantee or detect that it occurs.

An **adopt-commit protocol** [14] or **adopt-commit object** [2, 18] detects agreement, but does not create it. Our definition of an adopt-commit object uses the terminology of [4, 7]: an adopt-commit object provides a single operation $\texttt{AdoptCommit}(v)$ that returns (commit, v') or (adopt, v') for some v', subject to termination, validity (v' must equal some operation's input) and two new conditions that replace agreement:

- **Convergence.** If all operations have the same input value v, all operations return (commit, v).
- **Coherence.** If any operation returns (commit, v), then all operations return either (commit, v) or (adopt, v).

It is shown in [4] that an alternating sequence of conciliators and adopt-commit objects implements consensus, assuming each process decides immediately on v when it sees (commit, v). The idea is that some conciliator eventually produces agreement on some common value v, and the acceptance condition on the following adopt-commit means that all processes decide on that value. The additional conditions on conciliators and adopt-commit objects are used to show that validity and agreement hold. The cost of consensus using this technique is, on average, asymptotically equal to the sum of the cost of a conciliator and an adopt-commit object, because, on average, only a constant number of these objects are accessed by each process.

1.3 Notation

We use log for base-2 logarithm and ln for base-e logarithm. The **iterated logarithm function** $\log^* n$ is defined by the recurrence $\log^* n = 0$ for $n \leq 1$ and $\log^* n = 1 + \log^* \log n$ for larger n.

2. CHEAP CONSENSUS WITH CHEAP SNAPSHOTS

Here we show how to solve consensus for an unbounded range of input values in $O(\log^* n)$ expected steps using unit-cost atomic snapshot operations. We assume that an oblivious adversary schedules the order of these snapshot operations independently of the random choices made by the algorithm. Whether this is a reasonable assumption in practice is a tricky question [16], but our algorithm does demonstrate that any potential oblivious-adversary consensus lower bound greater than $\Omega(\log^* n)$ will hold only in a less benevolent model.

The basic idea of the protocol is to have a sequence of rounds, where in each round each process writes its current preference, takes a snapshot, and adopts the value out of those it sees with the highest random priority for this round. We will show that on average, if m values enter a round, $O(\log m)$ leave; thus after $O(\log^* n)$ rounds, the expected number of survivors is $O(1)$, and becomes 1 with probability at least $1 - \epsilon$ after $O(\log(1/\epsilon))$ additional rounds. To make this work, it is necessary for all copies of a given input value to share the same priority in each round. This is done by generating a vector of priorities for each process at the start of the protocol, which is then carried along with the process's input value as other processes adopt it. We refer to the combined input value and priority vector as a **persona**; the goal of the algorithm is to leave all processes with the same persona (and thus the same input value).

Pseudocode is given in Algorithm 1.

```
1 procedure conciliator(input)
2     Let priority be a vector of $\log^* n + \lceil \log(1/\epsilon) \rceil$
      independent uniform random variables in $[0, 1]$
3     persona ← ⟨input, priority⟩
4     for $i \leftarrow 1 \ldots \log^* n + \lceil \log(1/\epsilon) \rceil$ do
5         $A_i$[myId] ← persona
6         $S \leftarrow$ snapshot($A_i$)
7         Let $j$ maximize $S[j]$.priority[$i$] over all
          non-null $S[j]$
8         persona ← $S[j]$
9     return persona.input
```

Algorithm 1: Priority-based conciliator

To simplify the analysis, we assume that the priority values are uniformly chosen from the real interval $[0, 1]$. This allows us to assume that ties occur only with probability 0. More realistically, a smaller range of priorities could be used, at the cost of a small probability that the analysis fails.

Let Y_i be the number of distinct personae remaining after i rounds of the algorithm. Let $X_i = Y_i - 1$ be the number of **excess personae** remaining after i rounds. We will show that $\mathrm{E}[X_i]$ converges rapidly to 0. Markov's inequality can then be used to bound the probability that more than one value survives.

Lemma 1. *Let X_i be the number of excess personae remaining after i rounds of Algorithm 1. Then*

$$\mathrm{E}[X_{i+1}|X_i] \leq \min(\ln(X_i + 1), X_i/2). \quad (1)$$

Proof. For a persona to survive round $i + 1$, it must be the highest-priority persona in some view obtained by taking a snapshot of A_{i+1}. Recall that $Y_i = X_i + 1$ gives the number of distinct personae remaining after previous rounds, and that all copies of a particular persona share the same priority.

Define a view as the set of personae that appear in the result of some `snapshot` operation. Order the personae $a_1, a_2, \ldots, a_{Y_i}$ written to A_{i+1} by increasing size of the smallest view V_j that contains each a_j, breaking ties arbitrarily. Because each write to A can only add new personae, each view is a subset of any larger views. Furthermore, adding more personae to a view can only decrease the probability that any particular persona has the highest priority. It follows that a_j has the highest priority in any view (and thus survives) if and only if it has the highest priority in the smallest view that contains it, V_j.

We now wish to argue that each member of V_j is equally likely to have the highest priority, so that the probability that a_j in particular has the highest priority is exactly $1/|V_j|$. We must be a little bit careful here: while the adversary's schedule determine the order in which *processes* write and read the snapshot object, it does not determine the order in which *personae* appear, because this depends on the assignment of personae to processes, which in turn depends on the outcome of previous rounds. But the order of personae *is* fully determined by the combination of the adversary's schedule and the priorities in all rounds $i' < i + 1$. Since the priorities in round $i + 1$ are independent of these variables, the chance that a_j has the highest priority in its view is in fact $1/|V_j| \leq 1/j$.

Summing $1/j$ over all j gives a harmonic series, showing $\mathrm{E}[Y_{i+1}|Y_i] \leq H_{Y_i} \leq \ln Y_i + 1$. So $\mathrm{E}[X_{i+1}|X_i] \leq \ln Y_i = \ln(X_i+1)$.[1] However, because each term after the first in the harmonic series is less than or equal to 1/2, we also have the cruder, but more accurate for small X_i, bound $\mathrm{E}[Y_{i+1}|Y_i] \leq 1+(Y_i-1)/2 = 1+X_i/2$, or $\mathrm{E}[X_{i+1}|X_i] \leq X_i/2$. Combining these bounds gives the claimed bound (1). □

Iterating Lemma 1, plus an application of Jensen's inequality, shows that the algorithm works as advertised.

Theorem 2. *Algorithm 1 implements a conciliator with agreement probability $1 - \epsilon$ and $O(\log^* n + \log(1/\epsilon))$ individual step complexity.*

[1]What we are doing here is very similar to counting **left-to-right maxima** or **outstanding values** of a random permutation. There is an extensive literature on the distribution of the number left-to-right maxima, going back to a classic paper of Rényi [19], but for our purposes a simple linearity-of-expectation bound is enough.

PROOF. Termination and validity are immediate from inspection of the code. So we concentrate on showing probabilistic agreement, specifically that the set of surviving personae converges to a singleton with probability at least $1-\epsilon$.

Let X_i be the number of excess personae after i rounds as in Lemma 1.

Let $f(x) = \min(\ln(x+1), x/2)$. Then (1) says that $\mathrm{E}[X_{i+1}|X_i] \le f(X_i)$. Note that f is the minimum of increasing concave functions over $[0,\infty)$ and is thus increasing and concave itself. Because f is concave over $[0,\infty)$, we have that, for any random variable $X \ge 0$, $\mathrm{E}[f(X)] \le f(\mathrm{E}[X])$ by Jensen's inequality.

It follows that $\mathrm{E}[X_{i+1}] = \mathrm{E}[\mathrm{E}[X_{i+1}|X_i]] \le \mathrm{E}[f(X_i)] \le f(\mathrm{E}[X_i])$, and in general we have $\mathrm{E}[X_i] \le f^{(i)}(X_0) < f^{(i)}(n)$, where $f^{(i)}$ is the i-fold composition of f defined recursively by $f^{(0)}(x) = x$ and $f^{(i+1)} = f \circ f^{(i)}$.

It is not hard to show that $f(x) \le \log x$ for $x \ge 2$. Since $\log^{(i)} n \ge 2$ for $i \le \log^* n - 1$, we have $f^{(i)}(n) \le \log^{(i)} n$ for $i \le \log^* n$, and in particular have $f^{(\log^* n)}(n) \le 1$. Since $f(x) \le x/2$ always, applying f an additional $\lceil \log(1/\epsilon)\rceil$ times gives $f^{(\log^* n + \lceil \log(1/\epsilon)\rceil)}(n) \le \epsilon$ for any $\epsilon > 0$.

This gives $\mathrm{E}[X_{\log^* n + \lceil \log(1/\epsilon)\rceil}] \le \epsilon$. But $X_{\log^* n + \lceil \log(1/\epsilon)\rceil}$ is a non-negative integer-valued random variable, so applying Markov's inequality we have $\Pr[X_{\log^* n + \lceil \log(1/\epsilon)\rceil} > 0] = \Pr[X_{\log^* n + \lceil \log(1/\epsilon)\rceil} \ge 1] \le \epsilon$. $\square$

As noted in the introduction, alternating copies of Algorithm 1 with adopt-commit objects immediately gives an oblivious-adversary consensus protocol with $O(\log^* n)$ expected individual step complexity in the unit-cost snapshot model.

3. CHEAP CONSENSUS WITH MULTI-WRITER REGISTERS

Algorithm 2 gives an implementation of a conciliator with agreement probability $1-\epsilon$ in which each process takes $O(\log\log n + \log(1/\epsilon))$ steps.

```
1  procedure conciliator(input)
2      Let chooseWrite be a vector of
       $\lceil \log\log n\rceil + \lceil \log_{4/3}(8/\epsilon)\rceil$ independent random
       Boolean variables with
       $\Pr[\mathsf{chooseWrite}[i] = 1] = p_i$
3      persona ← ⟨input, chooseWrite⟩
4      for $i \leftarrow 1 \ldots \lceil \log\log n\rceil + \lceil \log_{4/3}(8/\epsilon)\rceil$ do
5          if persona.chooseWrite[$i$] = 1 then
6              $r_i$ ← persona
7          else
8              $v \leftarrow r_i$
9              if $v \ne \bot$ then
10                 persona ← $v$
11     return persona.input
```

Algorithm 2: Sifting conciliator

The basic mechanism is similar to the **sift** protocol used to reduce the number of participants in test-and-set in [1], where processes drop out when they see other processes that are not dropping out. The main difference is that in the new protocol, a process that sees another process adopts that process's persona (preferred value and random bits) and continues with the algorithm instead of dropping out.

In each round i, a process carries out exactly one operation, choosing randomly whether to write or read the register r_i. If the process writes, its persona is retained in the next round. If it reads, its persona is retained only if it sees an empty register; otherwise, it adopts whatever persona it sees. As in Algorithm 1, these random choices are controlled by vectors of random bits generated at the start of the protocol that propagate along with the input values, so that all processes with the same persona in round i take the same action.

The probabilities of each event vary from round to round and are carefully tuned to reduce the expected number of excess personae as quickly as possible. The specific probabilities used will be given after proving the following lemma.

LEMMA 3. *Let Y_i be the number of distinct personae that survive the first i rounds of Algorithm 2 and let $X_i = Y_i - 1$ be the number of excess personae. Then*

$$\mathrm{E}[X_{i+1}|X_i] < \min\begin{cases} p_{i+1}X_i + \frac{1}{p_{i+1}}, \\ (1 - p_{i+1} + p_{i+1}^2)X_i. \end{cases}$$

PROOF. The first case of the min is more useful when X_i is large. To obtain it, we will first compute a bound on $\mathrm{E}[Y_{i+1}|Y_i]$ and then manipulate it to obtain the stated bound on $\mathrm{E}[X_{i+1}|X_i]$.

Order the personae $a_1, \ldots, a_{Y_i}$ that appear as the persona of at least one process leaving round i by the order in which a process carrying each persona is first scheduled to write or read r_{i+1}. Observe that the assignment of personae to processes in round $i+1$ is determined by the chooseWrite bits for rounds 1 through i and the schedule chosen by the adversary; both are independent of the round-$(i+1)$ chooseWrite bits.

For each persona a_j, it survives round $i+1$ if (a) some process sees a 1 in the corresponding chooseWrite$[i+1]$ bit and writes a_j; (b) some process reads a_j and adopts it; or (c) some process sees a 0 in the corresponding chooseWrite$[i+1]$ bit but sees $\bot$ when it reads. If case (b) occurs, so does case (a), since some process had to first write a_j to r_{i+1}. Case (c) occurs if persona.chooseWrite$[i+1]$ is 0 for the first process to write a_j and for all process with persona $a_{j'}$ for $j' < j$.

The probabilities of these events are p for case (a) and $(1-p)^j$ for case (c). Summing these probabilities over all j gives $\mathrm{E}[Y_{i+1}|Y_i] \le p_{i+1}Y_i + 1/p_{i+1} - 1$, where $1/p_{i+1} - 1 = \sum_{j=1}^{\infty}(1-p)^j$ is an upper bound on the terms from case (c).

Substituting in $X_{i+1} = Y_{i+1} - 1$ and $Y_i = X_i + 1$ gives

$$\begin{aligned} \mathrm{E}[X_{i+1}|X_i] &\le p_{i+1}(X_i+1) + 1/p_{i+1} - 2 \\ &= p_{i+1}X_i + 1/p_{i+1} + p_{i+1} - 2 \\ &< p_{i+1}X_i + 1/p_{i+1}. \end{aligned}$$

The second case of the min becomes useful when X_i is small, where the cavalier unbounding of bounded sums and dropping of small terms in the previous analysis causes trouble. For this bound, we consider separately the cases where the first process q reads or writes, and assume, for the sake of obtaining a simple upper bound, that all personae survive if q reads. Somewhat more formally, we have:

$$\begin{aligned}\mathrm{E}[X_{i+1}|X_i] &= (1-p_{i+1})\,\mathrm{E}[X_{i+1}|X_i, q \text{ reads}] \\ &\quad + p_{i+1}\,\mathrm{E}[X_{i+1}|X_i, q \text{ writes}] \\ &\le (1-p_{i+1})X_i + p_{i+1}^2 X_i \\ &= (1-p_{i+1}+p_{i+1}^2)X_i.\end{aligned}$$

□

Now let use choose the probabilities p_i.

The first bound in Lemma 3 is minimized by letting $p_1 = 1/\sqrt{X_0}$; this gives $\mathrm{E}[X_1] \le 2\sqrt{X_0} \le \sqrt{n-1}$.

Iterating this procedure in subsequent rounds gives a recurrence $x_0 = n-1$, $p_{i+1} = 1/\sqrt{x_i}$, $x_{i+1} = p_i x_i + 1/p_i = 2\sqrt{x_i}$, whose solution gives

$$x_i = 2^{2-2^{-i+1}}(n-1)^{2^{-i}} \tag{2}$$

and

$$p_i = 2^{1-2^{-i+1}}(n-1)^{-2^{-i}}. \tag{3}$$

We will use these values of p_i for the first $\lceil \log\log n\rceil$ iterations, obtaining:

Lemma 4. *Let X_i be the number of distinct personae that survive the first i rounds of Algorithm 2 using p_i as defined above for $i = 1 \dots \log\log n$. Let x_i also be defined as above. Then $\mathrm{E}[X_i] < x_i$ for all i in $1 \dots \lceil \log\log n\rceil$.*

Proof. The proof is by induction on i, using Lemma 3 and the x_i recurrence at each step:

$$\begin{aligned}\mathrm{E}[X_{i+1}] &= \mathrm{E}[\mathrm{E}[X_{i+1}|X_i]] \\ &\le \mathrm{E}[p_{i+1}X_i + 1/p_{i+1}] \\ &= p_{i+1}\,\mathrm{E}[X_i] + 1/p_{i+1} \\ &\le p_{i+1}x_i + 1/p_{i+1} \\ &= 2\sqrt{x_i} \\ &= x_{i+1}.\end{aligned}$$

□

For $i = \lceil \log\log n\rceil$, this gives

$$\begin{aligned}x_{\lceil \log\log n\rceil} &= 2^{2-2^{-\lceil \log\log n\rceil+1}}(n-1)^{2^{-\lceil \log\log n\rceil}} \\ &< 4n^{1/\log n} \\ &= 8.\end{aligned}$$

For $i > \lceil \log\log n\rceil$, we switch to $p_i = 1/2$, which minimizes the coefficient $1 - p_i + p_i^2$ in the second case of the Lemma 3 bound. Now we have:

Lemma 5. *Let X_i be the number of distinct personae that survive the first i rounds of Algorithm 2. Let $p_i = 2^{1-2^{-i+1}}(n-1)^{-2^{-i}}$ for $i = 1 \dots \lceil \log\log n\rceil$ and $1/2$ for larger i. Let $j > 0$. Then $\mathrm{E}[X_{\lceil \log\log n\rceil+j}] \le 8 \cdot (3/4)^j$.*

Proof. We have just shown that $\mathrm{E}[X_{\lceil \log\log n\rceil}] \le 8$, and each subsequent round multiplies the bound by $(1 - 1/2 + (1/2)^2) = 3/4$. □

Plugging in the number of iterations from the algorithm gives:

Theorem 6. *Algorithm 2 implements a conciliator with agreement probability $1-\epsilon$ and $O(\log\log n + \log(1/\epsilon))$ individual step complexity.*

Proof. Again termination and validity are easy, so we concentrate on probabilistic agreement.

From Lemma 5, after $\lceil \log\log n\rceil + \lceil \log_{4/3}(8/\epsilon)\rceil$ rounds, the expected number of excess personae X is at most

$$8 \cdot (3/4)^{\log_{4/3}(8/\epsilon)} = 8 \cdot (\epsilon/8) = \epsilon.$$

So the probability that X is nonzero is bounded by ϵ using Markov's inequality. □

The dependence on ϵ in the running time is necessary, due to the oblivious-adversary lower bound of Attiya and Censor [9]. Whether the $\log\log n$ part can be further improved is open.

To extend Algorithm 2 to an algorithm for consensus, we can alternate it with adopt-commit objects as described in [4]. For constant ϵ, this gives an expected individual step complexity equal to $O(\log\log n)$ plus the cost of the adopt-commit. Unfortunately, the best currently-known implementation of adopt-commit [7] is relatively expensive, requiring $\Theta(\log m/\log\log m)$ steps from each process if run with m possible input values. This means that we can only get $O(\log\log n)$-step consensus if the number of values m is $O(\log n \log\log n)$. It is possible that further improvements in randomized adopt-commit implementations (for an oblivious adversary) might increase this limit.

4. LINEAR EXPECTED TOTAL WORK

Algorithm 2 requires $\Theta(n\log\log n)$ additional total steps to achieve a constant probability of agreement in the worst case. With some tinkering, we can reduce this to $O(n)$ while keeping the $O(\log\log n)$ individual step complexity.

The essential idea is to embed Algorithm 2 in an outer conciliator algorithm extracted from the consensus procotocol of Chor, Israeli, and Li [12]. In the outer algorithm, there is a single register proposal that is initially $\perp$. At each step, a process reads proposal and returns its value if it is not $\perp$; otherwise, it writes its input to proposal with probability $\frac{1}{4n}$ and executes a step of Algorithm 2 otherwise.

In isolation, the Chor-Israeli-Li conciliator (CIL) works because some process writes proposal after $4n$ attempts on average, and once some process writes proposal, each of the remaining $n-1$ processes has at most a $\frac{1}{4n}$ chance of overwriting the register before reading a non-null value and leaving. If no process overwrites the first value (which occurs, by the union bound, with probability at least $1 - \frac{n-1}{4n} > 3/4$), all processes will agree on it.

Embedding Algorithm 2 into CIL means that some processes may return a value obtained from Algorithm 2 while others return a value obtained from proposal. To reconcile these two values, we use a final combining stage that in effect implements a simple two-valued conciliator. First, an adopt-commit object forces each process to decide on the unique value if only one is present; in not, a shared coin chooses between the competing values otherwise. Using the persona technique again, we implement this shared coin by associating a random bit with each input, and having the combining stage use this bit. We show that, with constant probability, bit associated with the combining-stage inputs are equal to each other and the outcome of the adopt-commit, giving agreement on the final output value.[2]

[2]A similar technique was used in [5] to combine interleaved shared-coin algorithms into a single shared coin, but this re-

Pseudocode for the entire procedure is given in Algorithm 3.

```
procedure conciliator(input)
    Choose coin uniformly from $\{0, 1\}$
    Initialize an instance of Algorithm 2 with input
    $\langle$input, coin$\rangle$ and $\epsilon = 1/4$.
    while proposal $= \perp$ do
        with probability $\frac{1}{4n}$ do
            proposal $\leftarrow \langle$input, coin$\rangle$
        else
            Run one step of the Algorithm 2 instance
            if Algorithm 2 returns $v$ then
                $r[0] \leftarrow v$.input
                return combine$(0, v$.coin$)$
    $v \leftarrow$ proposal
    $r[1] \leftarrow v$.input
    return combine$(1, v$.coin$)$
procedure combine$(i$, coin$)$
    $\langle$decide, $b\rangle \leftarrow$ AdoptCommit$(i)$
    if decide $\neq$ commit then
        $b \leftarrow$ coin
    return $r[b]$
```

Algorithm 3: CIL conciliator with embedded sifter

Theorem 7. *Algorithm 3 implements a conciliator with $O(\log\log n)$ worst-case individual step complexity, $O(n)$ expected total step complexity, and agreement probability* $1/8$.

Proof. Termination and the individual step complexity bound hold because no process can execute Line 8 more than $O(\log\log n)$ times without leaving the main loop. This implies that the main loop is also executed $O(\log\log n)$ times, because at most one iteration can skip Line 8 without writing proposal and causing the loop to finish anyway. The binary adopt-commit object and additional work in combine cost at most $O(1)$ additional steps.

For expected total steps, observe that the total expected number of iterations of the main loop is bounded by $4n$ across all processes, because each such iteration has an independent $\frac{1}{4n}$ probability of shutting the protocol down. Additional operations outside the main loop cost at most $O(n)$ more total steps.

For validity, observe that if the adopt-commit value returns $\langle$commit, $b\rangle$, then b is the index of a register to which some process's input (as returned from Algorithm 2 or written directly to proposal) has been written. If it returns $\langle$adopt, $-\rangle$, then both indices have appeared in inputs to the adopt-commit (by acceptance), so both registers have been initialized. In either case validity holds.

Probabilistic agreement is messier. We begin by arguing that the probability that Algorithm 2 returns a unique pair v is not affected by the embedding.

Fix a schedule S for the execution of Algorithm 3. Let $\sigma_1, \sigma_2, \ldots, \sigma_n$ be the sequence of coin-flips used in Line 5 by processes $1, 2, \ldots, n$. Observe that S and $\sigma_1 \ldots \sigma_n$ between them determine the schedule for Algorithm 2: an easy way to see this is that if we replace Line 8 with an operation that emits the current process id, we compute the induced schedule based only on S and $\sigma_1 \ldots \sigma_n$. Because S and $\sigma_1 \ldots \sigma_n$ are independent of the input to Algorithm 2 and any coin-flips used inside it, the induced schedule is also independent of these quantities.

Conditioning on some fixed induced schedule, we have from Theorem 6 that Algorithm 2 violates probabilistic agreement with probability at most $1/4$. This continues to hold when we remove the conditioning by averaging over all values of $\sigma_1 \ldots \sigma_n$.

A second contribution to our error budget is the probability that the outer CIL mechanism produces more than one output. From the earlier discussion, this occurs with probability at most $\frac{n-1}{4n} < 1/4$.

It follows that the probability that either Algorithm 2 or the proposal register yield more than one output is less than $1/4 + 1/4 = 1/2$. If both conciliators produce a unique output, then combine produces agreement with independent probability at least $1/4$. To see this, let b_{AC} be the unique bit tagged with commit by the adopt-commit object (if any), and b_0 and b_1 the coin bits for $i = 0$ and $i = 1$. The probability that $b_0 = b_1$ is at least $1/2$ (it may be 1 if both correspond to a random bit supplied by the same process); if b_{AC} exists, the probability this common bit equals it is also $1/2$, because b_{AC} does not depend on any of the coin values. Multiplying these probabilities gives at least a $1/4$ chance that all processes choose the same b, which gives the at least $1/8$ probability of agreement for the algorithm as a whole. $\square$

Essentially the same argument shows that replacing Algorithm 2 in Algorithm 3 with Algorithm 1 similarly gives an oblivious-adversary conciliator for the unit-cost snapshot model with $O(\log^* n)$ worst-case individual step complexity and $O(n)$ expected total step complexity. A similar argument is likely to work on any conciliator that is "oblivious" in the sense that it only copies its input values without examining them.

As with the algorithms in previous sections, we can use Algorithm 3 to obtain consensus by alternating it with adopt-commit objects. The resulting protocol takes an expected $O(\log\log n)$ individual steps and expected $O(n)$ total steps, for consensus on $O(1)$ possible inputs, with the input space limit as before reflecting the limitations of currently-known adopt-commit objects.

5. CONCLUSIONS

Under the assumption of an oblivious adversary, we have shown how to reduce the expected individual step complexity of consensus from $O(\log n)$ to $O(\log\log n)$ in the standard multi-writer register model and $O(\log^* n)$ in the practically irrelevant but theoretically significant unit-cost snapshot model. Many open problems remain.

Strength of the adversary.

Our results exploit the limitations of the oblivious adversary in several ways. As in previous protocols of Chandra [11] and Aumann [10], we pre-flip coins that are later shared between many processes; this requires assuming at minimum a **content-oblivious** adversary that cannot see the contents of registers or the internal states of processes.

sult depends on the output of a shared coin not always being under the control of the adversary. Unlike the shared-coin case, it is not clear that interleaving arbitrary conciliators will always give a conciliator.

We also choose between different operations probabilistically as in the protocol of Chor, Israeli, and Li [12], requiring a **weak adversary** that cannot prevent this. As the oblivious adversary is weaker than both these adversaries, our algorithms work for an oblivious adversary, but it would be interesting to examine more closely exactly what properties of the adversary are needed for either these algorithms or sublogarithmic consensus in general.

Gap between upper and lower bounds.

There is still a gap between our upper bounds of $O(\log\log n + \log(1/\epsilon))$ and $O(\log^* n + \log(1/\epsilon))$ for conciliators with agreement probability $1-\epsilon$ and the $\Omega(1/\epsilon)$ lower bound of Attiya and Censor [9], leaving open the possibility that the dependence on n could be further reduced. It may also be the case that this dependence is necessary, and that a lower bound could be shown that mirrors the structure of the upper bound: showing that $\Omega(n^c)$ or $\Omega(\log n)$ values remain after one layer of computation in the multi-writer register or unit-cost snapshot models, respectively, in some class of executions.

Gap between consensus and test-and-set.

Many of our techniques are similar to techniques recently used for oblivious-adversary test-and-set [1, 15]. Our protocol for consensus using cheap snapshots has the same $O(\log^* n)$ expected individual step complexity as the best currently known protocol for test-and-set *without* using snapshots, due to Giakkoupis and Woelfel [15], which was developed concurrently with and independently of the present work. In the standard multi-writer register model, our protocol follows both the structure and the $O(\log\log n)$ complexity the *previous* best known test-and-set protocol of Alistarh and Aspnes [1], but this is significantly slower than the newer $O(\log^* n)$ protocol. Unfortunately, the specific technique used by Giakkoupis and Woelfel for test-and-set does not generalize immediately to consensus, because it exploits the fact that a loser in a test-and-set protocol can leave immediately without determining the identity of any specific survivor, so long as it can determine that at least one other process survives. But it may be that some similar technique could yield further improvements for oblivious-adversary consensus in the multi-writer register model.

6. REFERENCES

[1] Dan Alistarh and James Aspnes. Sub-logarithmic test-and-set against a weak adversary. In *Distributed Computing: 25th International Symposium, DISC 2011*, volume 6950 of *Lecture Notes in Computer Science*, pages 97–109. Springer-Verlag, September 2011.

[2] Dan Alistarh, Seth Gilbert, Rachid Guerraoui, and Corentin Travers. Of choices, failures and asynchrony: The many faces of set agreement. In Yingfei Dong, Ding-Zhu Du, and Oscar H. Ibarra, editors, *ISAAC*, volume 5878 of *Lecture Notes in Computer Science*, pages 943–953. Springer, 2009.

[3] James Aspnes. Randomized consensus in expected $O(n^2)$ total work using single-writer registers. In *Distributed Computing: 25th International Symposium, DISC 2011*, volume 6950 of *Lecture Notes in Computer Science*, pages 363–373. Springer-Verlag, September 2011.

[4] James Aspnes. A modular approach to shared-memory consensus, with applications to the probabilistic-write model. *Distributed Computing*, 25(2):179–188, May 2012.

[5] James Aspnes, Hagit Attiya, and Keren Censor. Combining shared coin algorithms. *Journal of Parallel and Distributed Computing*, 70(3):317–322, March 2010.

[6] James Aspnes and Keren Censor. Approximate shared-memory counting despite a strong adversary. In *SODA '09: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 441–450, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[7] James Aspnes and Faith Ellen. Tight bounds for anonymous adopt-commit objects. In *23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 317–324, June 2011.

[8] Hagit Attiya and Keren Censor. Tight bounds for asynchronous randomized consensus. *Journal of the ACM*, 55(5):20, October 2008.

[9] Hagit Attiya and Keren Censor-Hillel. Lower bounds for randomized consensus under a weak adversary. *SIAM J. Comput.*, 39(8):3885–3904, 2010.

[10] Yonatan Aumann. Efficient asynchronous consensus with the weak adversary scheduler. In *PODC '97: Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 209–218, New York, NY, USA, 1997. ACM.

[11] Tushar Deepak Chandra. Polylog randomized wait-free consensus. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 166–175, Philadelphia, Pennsylvania, USA, 23–26 May 1996.

[12] Benny Chor, Amos Israeli, and Ming Li. Wait-free consensus using asynchronous hardware. *SIAM J. Comput.*, 23(4):701–712, 1994.

[13] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.

[14] Eli Gafni. Round-by-round fault detectors: Unifying synchrony and asynchrony (extended abstract). In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing*, pages 143–152, 1998.

[15] George Giakkoupis and Philipp Woelfel. On the time and space complexity of randomized test-and-set. To appear, PODC 2012.

[16] Wojciech M. Golab, Lisa Higham, and Philipp Woelfel. Linearizable implementations do not suffice for randomized distributed computation. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 373–382. ACM, 2011.

[17] Michael C. Loui and Hosame H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. *Advances in Computing Research*, pages 163–183, 1987.

[18] Achour Mostefaoui, Sergio Rajsbaum, Michel Raynal, and Corentin Travers. The combined power of conditions and information on failures to solve asynchronous set agreement. *SIAM Journal on Computing*, 38(4):1574–1601, 2008.

[19] Alfred Rényi. Théorie des éléments saillants d'une suite d'observations. *Annales scientifiques de l'Université de Clermont-Ferrand 2, série Mathématiques*, 8(2):7–13, 1962.