A Simple Population Protocol for Fast Robust Approximate Majority

Dana Angluin[1], James Aspnes[1,⋆], and David Eisenstat[2,⋆⋆]

[1] Yale University, Department of Computer Science
{dana.angluin, james.aspnes}@yale.edu
[2] Princeton University, Department of Computer Science
deisenst@cs.princeton.edu

Abstract. We describe and analyze a 3-state one-way population protocol for approximate majority in the model in which pairs of agents are drawn uniformly at random to interact. Given an initial configuration of x's, y's and blanks that contains at least one non-blank, the goal is for the agents to reach consensus on one of the values x or y. Additionally, the value chosen should be the majority non-blank initial value, provided it exceeds the minority by a sufficient margin. We prove that with high probability n agents reach consensus in $O(n \log n)$ interactions and the value chosen is the majority provided that its initial margin is at least $\omega(\sqrt{n \log n})$. This protocol has the additional property of tolerating Byzantine behavior in $o(\sqrt{n})$ of the agents, making it the first known population protocol that tolerates Byzantine agents. Turning to the register machine construction from [2], we apply the 3-state approximate majority protocol and other techniques to speed up the per-step parallel time overhead of the simulation from $O(\log^4 n)$ to $O(\log^2 n)$. To increase the robustness of the phase clock at the heart of the register machine, we describe a consensus version of the phase clock and present encouraging simulation results; its analysis remains an open problem.

1 Introduction

Population protocols [1] model distributed systems in which individual agents are extremely limited, in fact finite-state, and complex behavior of the system as a whole emerges from the rules governing pairwise interaction of the agents. Such models have been defined and used in other fields, for example, statistics, epidemiology, physics and chemistry; understanding their behavior is a fundamental scientific problem. The new perspective we bring as computer scientists is to ask what computational behaviors these systems can exhibit. In addition to fundamental scientific knowledge, answers may provide novel designs for distributed computational systems at many scales.

⋆ Supported in part by NSF grant CNS-0435201.

⋆⋆ Supported in part by a Gordon Y. S. Wu Fellowship and a National Defense Science and Engineering Graduate Fellowship.

Chemists have defined a standard model of small molecules in a well-mixed solution, in which the molecules are agents, the state of an agent represents the chemical species of the molecule, and interaction rules specify the probable products of a collision between two molecules; the sequence of collisions is determined by uniform random draws of a pair of agents to interact [4, 5]. In [1] it is shown that this model in principle permits the design of a "computer in a beaker," that is, we can design interaction rules that allow a population of n molecules to simulate the behavior of a register machine with a constant number of registers holding numbers of magnitude $O(n)$ for $poly(n)$ steps with error probability $1/\,poly(n)$ in parallel time that is a factor of $poly(n)$ larger than the number of simulated instructions. In [2] we have shown that a careful analysis of the properties of epidemics permits us to design a much more efficient simulation, in which the per-step slowdown factor is $O(\log^4 n)$ parallel time.

The register machine of [2] has several shortcomings: it requires an initial configuration with a designated leader, it does not tolerate faults, and $O(\log^4 n)$ is still fairly slow. One goal of this paper is to develop more tools for the design of fast robust population protocols, with improvement of the register machine as a critical testbed. The main tool we develop is our 3-state protocol for approximate majority; this is a protocol that rapidly takes a configuration of x's, y's and b's to a configuration that is all x's or all y's. The final value represents the majority value (among x's and y's) in the initial configuration, provided it exceeds the minority value by a sufficient margin, namely $\omega(\sqrt{n \log n})$. Moreover, we show that is robust to $o(\sqrt{n})$ Byzantine agents, the first population protocol that provably tolerates any Byzantine agents. With a sufficiently redundant representation of register values, this protocol gives a fast comparison operation, which, when combined with other techniques, reduces the slowdown factor of the simulation to $O(\log^2 n)$ parallel time. Though the approximate majority protocol has only 3 states, its analysis is nontrivial; we expect that the protocol and its analysis will find other applications in the design of fast robust population protocols.

The register machine of [2] has at its heart the construction of a phase clock that causes the agents to move together rapidly through a fixed cycle of phases; together meaning that no two agents are more than a very few phases apart, and rapidly meaning the parallel time to complete a cycle is $O(\log n)$. We would like to avoid the need for a designated leader or leaders in the initial configuration to synchronize the phase clock, and we would like the phase clock to be robust. In Section 8 we describe an apparently robust consensus variant of the phase clock, and a protocol to start it from a uniform initial configuration. We give simulation results suggesting that it performs well. However, the problem of analyzing it formally remains open; more tools are necessary.

2 Model

A **population protocol** consists of a finite set of states Q, a finite set of input symbols $X \subseteq Q$, a finite set of output symbols Y, an output function $\gamma : Q \to Y$, and a **joint transition function** $\delta : Q \times Q \to Q \times Q$. A population protocol is executed by a fixed finite **population** of agents with states in Q. For convenience, we assume that each agent has an identity $v \in V$, but agents do not know their own identities or others'.

Initially, each agent is assigned a state according to an **input** $x : V \to X$ that maps agent identities to input symbols. In the general population protocol model, there is an **interaction graph**, a directed graph $G = (V, A)$ without self-loops, whose arcs indicate the possible agent interactions that may take place. (G is directed because we assume that interacting agents are able to break symmetry.) In this paper, G will always be a complete graph.

During each execution step, an arc (v, w) is chosen uniformly at random from A. The "source" agent v is the **initiator**, and the "sink" agent w is the **responder**. These agents update their states jointly according to δ: if v is in state q_v and w is in state q_w, the state of v becomes $\delta_1(q_v, q_w)$, the state of w becomes $\delta_2(q_v, q_w)$, where δ_i gives the i^{th} coordinate of the output of δ. The states of all other agents are unchanged. For any given V, a population protocol **computes** a (possibly partial) function $f : X^V \to Y$ in ℓ steps with error probability ϵ if for all $x \in f^{-1}(Y)$, the configuration $c : V \to Q$ after ℓ steps satisfies the following properties with probability $1 - \epsilon$.

- All agents agree on the correct output: for all $v \in V$, $f(x) = \gamma(c(v))$.
- This remains true with probability 1 in the future.

We are interested in the guarantees one can make about a fixed protocol over a family of functions f defined for all finite populations.

Although we have described the population protocol model in a sequential light, in which each step is a single pairwise interaction, interactions between pairs involving different agents are independent and may be thought of as occurring in parallel. In measuring the speed of population protocols, then, we define 1 unit of **parallel time** to be $|V|$ steps. The rationale is that in expectation, each agent initiates 1 interaction per parallel time unit; this corresponds to the chemists' idealized assumption of a well-mixed solution.

2.1 Byzantine Agents

We extend the basic randomized population protocol model described above to allow Byzantine behavior from some of the agents. In addition to the n normal agents we allow a population to include z Byzantine agents. For each interaction, an ordered pair of agents is selected uniformly at random from the population of normal and Byzantine agents. A Byzantine agent may simulate any normal agent state in an interaction, and its choice of state may depend on both the global configuration and the identity of the specific agent it encounters. The state of Byzantine agents is not meaningful and so is not included in the description of a configuration. We first describe our protocol and analyze its behavior without Byzantine agents.

3 A 3-State Approximate Majority Protocol

We analyze the behavior of the following population protocol with states $Q = \{b, x, y\}$. The state b is the **blank** state. Row labels give the initiator's state and column labels the responder's state.

$$\begin{array}{cccc} & x & b & y \\ x & (x,x) & (x,x) & (x,b) \\ b & (b,x) & (b,b) & (b,y) \\ y & (y,b) & (y,y) & (y,y) \end{array}$$

Note that this protocol is **one-way**: every interaction changes at most the responder's state; thus it can be implemented with one-way communication. Only the interactions xb, yb, xy, and yx change the responder's state; we may think of these as the only interactions that consume energy. The **blank configuration** of all b's is stable, but cannot be reached from any non-blank configuration because no interaction can eliminate the last x or y. The configurations of all x's and all y's are stable, and every non-blank configuration can reach at least one of them.

An intuitive description of the process is that agents in state b are undecided, while initiators in states x and y are attempting to convert responders that they meet to adopt their respective states. Such an initiator immediately converts an undecided responder, but only succeeds in reducing an opposing responder to undecided status. The process may also be thought of as two competing epidemics, x's and y's, with the ability to reverse each other's progress.

In Sections 4 and 5, we show that with high probability this protocol (a) converges from any non-blank configuration to a stable configuration in $O(n \log n)$ interactions; and (b) correctly computes the initial majority x or y value provided $\omega(\sqrt{n \log n})$ more agents carry this value in the starting configuration than carry the opposing value. In Section 6, we show that it can tolerate $o(\sqrt{n})$ Byzantine agents; the formal definition of this property is given there.

4 Convergence

We show that, from any non-blank initial configuration, the 3-state approximate majority protocol converges to either all x tokens or all y tokens within $O(n \log n)$ interactions with high probability. We divide the space of non-blank configurations into four regions: three corners, where most tokens are b, x, or y, and a central region where the tokens are more evenly balanced. We show that the number of interactions in each region is bounded by $O(n \log n)$ w.h.p., by constructing a family of supermartingales of the form $M = e^{aS/n} f(x, y)$ where $a > 0$ is a constant, S counts the number of interactions of a particular type and f is a potential function defined across the entire space of configurations. (We overload x, y and b to denote the number of each token in a configuration.)

Specifically, we let τ_* be the stopping time at which the protocol converges, and let $\tau = \min(\tau_*, kn \log n)$ for some fixed k. Assuming f does not vary too much over the space of configurations, we can use the supermartingale property $\mathrm{E}[M_\tau] \leq M_0$ to show that $e^{aS_\tau/n}$ is small, and then use Markov's inequality to get the bound on S_τ. Summing the bounds for each region then gives the total bound on the number of interactions. Though it would seem that truncating at time $kn \log n$ assumes what we are trying to prove, in fact we show that with high probability the total number of interactions is much less than $kn \log n$, implying that we do in fact converge by the given time bound.

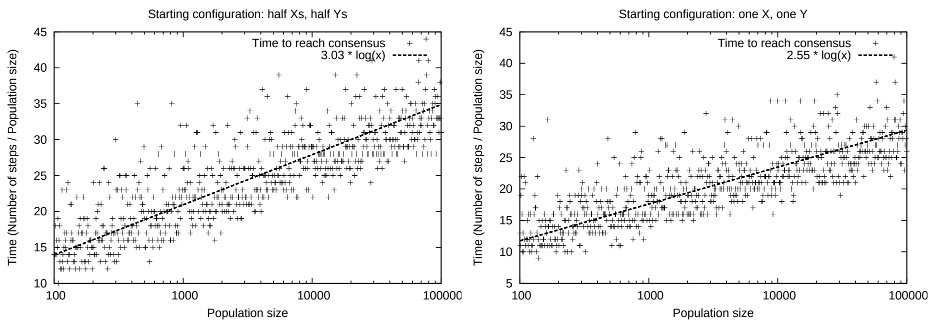
The resulting proof requires a careful selection of f. To keep the argument at least locally simple, we construct separate potential functions to bound different classes of operations, based on the type of interaction that occurs and which region of the configuration space it occurs in. The reason for this classification is that the behavior of the protocol is qualitatively different in different regions of the configuration space. When most tokens are blank, the protocol acts like an epidemic, with non-blank tokens rapidly infecting blank tokens. When most tokens carry the same non-blank value, the protocol acts like coupon collector, with the limit on convergence being the time for the few remaining minority tokens to be converted to the majority value. In the central region, where no token type predominates, the protocol acts like a random walk with increasing bias away from the center. Unfortunately, in none of these configurations does the protocol act *enough* like the analogous well-known stochastic processes to permit a direct reduction to previous results, and the behavior in border areas blends smoothly between one form and another. The supermartingale/potential function approach allows separate arguments designed for each region to be blended smoothly together. Unfortunately, this still requires considerable calculation to verify that each potential function does what it is supposed to. In this extended abstract, the detailed calculations are omitted for reasons of space.

The reader may be surprised to find that such a simple protocol requires such a lengthy proof. Despite substantial efforts, we were unable to apply more powerful tools to this problem. Part of the reason is that we are trying to obtain exact asymptotic bounds on a system much of whose interesting behavior occurs when particular tokens are very rare or when the behavior of the protocol is highly random (e.g., with evenly balanced numbers of x and y tokens); this (together with the fact that the corresponding systems of differential equations do not have closed-form solutions) appears to rule out arguments based on classical techniques involving reduction to a continuous process in the limit (e.g., [6, 7]). Similarly, approaches based on direct computation of hitting times or eigenvalues of the resulting Markov chain would appear to require substantially more work than a direct potential function argument.

It is possible that such difficulties are an inherent property of randomized population protocols. The ability to construct register machines using such protocols [1, 2] suggests that analysis of an arbitrary protocol for arbitrarily large populations quickly enters the realm of undecidability. But we cannot rule out the possibility that a more sophisticated approach might give an easier proof of the convergence rate for the particular protocols we are interested in.

Our results are stated using explicit constant factors. The reader should be warned that in many cases these are gross overestimates, and that from simulation we observe that the expected number of interactions to convergence seems to be less than $4n \log n$ from two challenging initial configurations (see Figure 1.) The first of these, an initial population evenly divided between x and y with no blank tokens, can be shown numerically for reasonably small n to be the configuration that maximizes expected convergence time.

Fig. 1. Simulation results: parallel time of approximate majority from two initial conditions



The full convergence bound is stated below.

Theorem 1. *Let τ_* be the time at which $x = n$ or $y = n$ first holds. Then for any fixed $c > 0$ and sufficiently large n,*

$$\Pr\left[\tau_* \geq 6754 n \log n + 6759 c n \log n\right] \leq 5 n^{-c}.$$

Proof. We give here a brief sketch only. First, we show that the potential function $f = \frac{1}{(x-y)^2+2n}$ is reduced by $-\Theta(1/n)$ of its previous value on average conditioned on an xb or a yb interaction, and that it rises by a smaller relative amount conditioned on an xy or a yx interaction. It follows that $f \cdot e^{O(S-\alpha T)/n}$ is a supermartingale, where S counts xy and yx interactions, T counts xb and yb interactions, and $\alpha < 1$. The factor of $O(n)$ drop in f between the maximum initial $f = 1/(2n)$ and final $f = 1/(n^2+2n)$ allows only a similar expected rise in $e^{(S-\alpha T)/n}$ and thus (with high probability) only an $O(n \log n)$ rise in $S - \alpha T$. Since all but at most $n - 1$ initial blank tokens destroyed in T must have previously been created in S, $T \leq S + n$, giving an $O(n \log n)$ bound on, in turn: (a) S alone; (b) $S + T$; (c) all interactions in the central region (where xy, yx, xb, or yb interactions are likely). Separate potential functions are used to bound the remaining interactions in the corners. $\square$

5 Correctness of Approximate Majority

Not only does the 3-state protocol converge quickly, but it also converges to the dominant non-blank value in its input if there is a large enough initial majority.

Theorem 2. *With high probability, the 3-state approximate majority protocol converges to the initial majority value if the difference between the initial majority and initial minority populations is $\omega(\sqrt{n \log n})$.*

Proof. Without loss of generality, assume that the initial majority value is x. We consider a coupled process (u_t, u'_t) where $u_t = (x_t - y_t)$ and u'_t is the sum of a series of fair ± 1 coin flips. Initially $u'_0 = u_0$. Later values of u'_t are specified by giving a joint distribution on $(\Delta u, \Delta u')$. We do so as follows. Let p be the probability that $\Delta u = 1$ and q the probability that $\Delta u = -1$. Then let

$$(\Delta u, \Delta u') = \begin{cases} (0,0) & \text{with probability } 1 - p - q, \\ (1,1) & \text{with probability } \frac{1}{2}(p+q), \\ (1,-1) & \text{with probability } p - \frac{1}{2}(p+q), \\ (-1,-1) & \text{with probability } q. \end{cases}$$

The probability in the third case is non-negative if $p/(p+q) = \Pr[\Delta u = 1 | \Delta u \neq 0] \geq \frac{1}{2}$. This holds as long as $u \geq 0$; should u ever drop to zero, we end the process.

Observe that unless this event happens, we have $u_t \geq u'_t$. We can also verify by summing the cases that Δu rises with probability exactly p and drops with probability exactly q; and that $\Delta u'$ rises or drops with equal probability $\frac{1}{2}(p+q)$. So we have $\mathrm{E}[\Delta u'] = 0$ and that $|\Delta u'| \leq 1$, the preconditions for Azuma's inequality.

Theorem 1 shows that the process converges before $O(n \log n)$ interactions with high probability. Suppose the process converges at some time $\tau = O(n \log n)$. Then by Azuma's inequality we have that $|u'_\tau - u'_0| = O(\sqrt{n \log n})$ throughout this interval with high probability. So if $u'_0 = u_0 = \omega(\sqrt{n \log n})$, it follows that $u_0 \geq u'_0 \geq 0$ throughout the execution, and in particular that the process does not terminate before convergence and that u is non-negative at convergence. But this excludes the $y = n$ case, so the process converges to the initial majority value. $\square$

6 Tolerating Byzantine Agents

In this section, we show that the 3-state approximate majority protocol can tolerate up to $o(\sqrt{n})$ Byzantine agents, computing the correct majority value in $O(n \log n)$ time with high probability despite their interference. However, to do so we must both assume a somewhat larger initial majority, and slightly relax the criterion for convergence.

The issue with convergence is that Byzantine agents can always pull the normal agents out of a converged configuration. For example, if all normal agents are in the x state, any encounter with a Byzantine initiator can shift the normal agent to a b state, and a second encounter can shift it to a y state, even though there are no normal y agents in the population. So we must accept a small number of normal agents that do not have the correct value.

But in fact the situation is worse: if we run long enough, there exists a trajectory with nonzero probability that takes us to the blank configuration, which is stable. So we must also accept a small probability that we reach the blank configuration quickly, and the assurance that we reach it with probability 1 after a very long time. However, we can show that with high probability neither outcome occurs within a polynomial number of steps.

Our technique is to adjust the potential functions used by the non-Byzantine process to account for Byzantine transitions. We then use these adjusted potential functions to show that (a) strong pressure exists to keep the process out of the high-b corner and in the high-x and high-y corners, and (b) the number of interactions (including Byzantine interactions) to reach the x or y corner is still small.

6.1 Biased-Walk Barriers

Let us begin by showing that it is difficult even for Byzantine agents to force the protocol into a configuration with a low value of $v_t = x_t + y_t$.

Observe that if the Byzantine agents attempt to minimize v, v nonetheless increases at each interaction with likelihood proportional to vb and decreases with likelihood proportional to $2xy + zv$. So the probability of an increase conditioned on any change in v is $vb/(vb + 2xy + zv) \geq vb/(vb + v^2/2 + zv) = b/(b + z + v/2) \geq b/n$ provided $z \leq v/2$. For large b and small z this gives a random-walk behavior that is strongly biased upwards.

Suppose $\sqrt{n} \leq v \leq n/8$. Then $b \geq (7/8)n$ and $z = o(\sqrt{n}) \ll v/2$, so $\Pr[\Delta v = 1 | \Delta v \neq 0] \geq 7/8$. We wish to bound the probability starting from some initial v_0 in this range that v reaches $\sqrt{n}$ before it reaches $n/8$. Though the probability that v rises or falls changes over the interval, the position of v can be lower-bounded by the position of a coupled variable v' that moves according to a biased random walk with fixed probability $p = 7/8$ of increasing by 1 and $q = 1/8$ of decreasing by 1. From the standard analysis of the gambler's ruin problem,[3] we have that $(q/p)^{v'_t}$ is a martingale, and thus that the quantity

$$\Pr[v' \text{ reaches } \sqrt{n} \text{ before } n/8](q/p)^{\sqrt{n}} + \Pr[v' \text{ reaches } n/8 \text{ before } \sqrt{n}](q/p)^{n/8}$$

is equal to $(q/p)^{v_0}$. Because $(q/p)^{n/8} = (1/7)^{n/8}$ is exponentially small, it makes sense to ignore the second addend, leaving

$$\Pr[v' \text{ reaches } \sqrt{n} \text{ before } n/8](q/p)^{\sqrt{n}} < (q/p)^{v_0}$$

or

$$\Pr[v' \text{ reaches } \sqrt{n} \text{ before } n/8] < (q/p)^{v_0 - \sqrt{n}}.$$

It follows that if $v_0 \geq \sqrt{n} + c \log_7 n$, then the probability that v drops to $\sqrt{n}$ before reaching $n/8$ is bounded by n^{-c}. Once v reaches $n/8$, further drops to $\sqrt{n}$ become exponentially improbable even conditioned on starting at $v = n/8 - 1$. We thus have:

Lemma 1. *Fix $c > 0$. Let $z = o(\sqrt{n})$ and let $v_0 \geq \sqrt{n} + c \log_7 n$. Then for sufficiently large n, the probability that $v_t \leq \sqrt{n}$ for any $t < e^{n/8} n^{-c}$ is less than $2n^{-c}$.*

Proof. The probability that v reaches $\sqrt{n}$ before reaching $n/8$ for the first time is at most n^{-c}. For each subsequent drop to $n/8 - 1$, there is a probability of at most $(1/7)^{n/8 - 1 - \sqrt{n}} \leq \exp(-n/8))$ that v reaches $\sqrt{n}$ before returning to $n/8$.

[3] See, for example, [3, §XIV.2].

Since each such excursion below $n/8$ involves at least one interaction, $e^{n/8}n^{-c}$ interactions gives at most an expected n^{-c} drops to $\sqrt{n}$ for a total probability of reaching $v = \sqrt{n}$ bounded by $2n^{-c}$. □

We can apply a similar analysis to the x and y corners, but here the protocol drifts toward the all-x or all-y configuration instead of away from it. Here we track $3y + b$ for the x corner and $3x + b$ for the y corner. Because these functions can change by more than just ± 1, the simple random walk analysis becomes more difficult. Instead, we proceed by showing that $\exp(3y + b)$ is a supermartingale, and bound the probability of moving from $2\sqrt{n}$ to $3\sqrt{n}$ by $\exp(-\sqrt{n})$, the inverse of the change in $\exp 3y + b$.

Formally, we have:

Lemma 2. *Fix $c > 0$. Let $z = o(\sqrt{n})$ and let $3y_0 + b_0 \leq 2\sqrt{n}$. Then for sufficiently large n, the probability that $3y_t + b_t \geq 3\sqrt{n}$ for any $t < e^{\sqrt{n}-1}n^{-c}$ is less than n^{-c}.*

Proof. (Omitted for reasons of space.) □

6.2 Convergence Time with Byzantine Agents

The convergence time is given in the following theorem.

Theorem 3. *Let τ be the time at which $x \geq n - \sqrt{n}$, $y \geq n - \sqrt{n}$, or $v \leq \sqrt{n}$ first holds. Let v_0 be the initial number of x's and y's. Then for any fixed $c > 0$ and sufficiently large n, if $v_0 \geq \sqrt{n} + c \log_7 n$, then*

$$\Pr\left[\tau \geq 6754 n \log n + 6759 c n \log n \text{ or } v_\tau \leq \sqrt{n}\right] = n^{-c+o(1)}.$$

Proof. (Proof omitted for reasons of space.) □

Note that once we are in the x or y corner, Lemma 2 tells us that we remain there with high probability for exponential time. So we have a complete characterization of the convergence behavior of the 3-state majority protocol with $o(\sqrt{n})$ Byzantine agents. It is also not hard to see that the proof of Theorem 2 also continues to hold for $z = o(\sqrt{n})$, provided we increase the size of the initial majority to $\omega(\sqrt{n}\log n)$ to compensate for the offset of $o(\sqrt{n}\log n)$ generated by Byzantine interactions.

7 Speeding Up the Register Machine Construction

In this section we show how to use the 3-state approximate majority protocol and other techniques to speed up the register machine construction in [2] so that it has per-step parallel time overhead of $O(\log^2 n)$ instead of $O(\log^4 n)$. The original construction is based on a single agent representing a finite-state controller operating via commands spread by epidemics on register values represented in unary by tokens scattered across the population. A major bottleneck in [2] is the difficulty of carrying out exact comparisons (performed in $O(\log^2 n)$ time using $O(\log n)$ rounds that alternate cancellation with amplification) and of performing subtractions (done in $O(\log^3 n)$ using addition, comparison, and binary search). Our approximate majority protocol gives a simpler and faster implementation of comparison, provided we pad out the register values to avoid near-ties. A further adjustment to the representation gives cheap subtraction.

Because space does not permit us to repeat the description of the original construction here, we refer the reader to [2] for details.

We begin by replacing the original $O(\log^2 n)$ parallel time comparison operation by our new $O(\log n)$ parallel time approximate majority protocol. To ensure that a comparison is correct with high probability, we need to ensure that the register values being compared differ by $\omega(\sqrt{n \log n})$. We guarantee this by having registers hold values that are multiples of $n^{2/3}$; three such registers are sufficient to represent $n = (n^{1/3})^3$ different values, thinking of them as the high, middle and low order **wide-digits** of a number in base $n^{1/3}$. Thus, to compare two wide digits, say A and B, we do an approximate majority comparison of $A + (1/2)n^{2/3}$ with B; if the result is that A is in the majority, then we conclude that $B \leq A$, otherwise that $A < B$. To compare two registers composed of $O(1)$ wide-digits it suffices to proceed digit by digit.

The subtraction operation of [2] requires $O(\log n)$ rounds of binary search where the $O(\log^2 n)$ parallel time comparison operation dominates the cost of each round. Though we could replace these comparisons with our faster comparison operation and reduce the cost of subtraction to $O(\log^2 n)$, we can obtain a still better reduction to $O(\log n)$ parallel time by use the logician's construction of the integers from the natural numbers: the value A in a register is represented by the difference $A_+ - A_-$ of values in two different registers. To compute $C \leftarrow A + B$, we compute $C_+ \leftarrow A_+ + B_+$ and $C_- \leftarrow A_- + B_-$. To compute $C \leftarrow A - B$, we compute $C_+ \leftarrow A_+ + B_-$ and $C_- \leftarrow A_- + B_+$. These operations both take parallel time $O(\log n)$, because addition is already $O(\log n)$ in the previous construction. An additional clock cycle of cancellation keeps the $+$ and $-$ components from overflowing.

For registers with this balanced representation, we must revisit comparison. To compare A with B, we compare $(A_+ + B_-)$ with $(A_- + B_+)$. Since these differ by a multiple of $n^{2/3}$, our previous comparison method works. The result is that subtraction can be done with $O(1)$ additions and comparisons, which gives parallel time of $O(\log n)$.

The most expensive operation in [2] is division by a constant, which is based on $O(\log n)$ rounds of binary search in which subtraction dominates. The improved cost of subtraction immediately reduces the parallel time for division to $O(\log^2 n)$ without any change to the division algorithm.

The remaining issue is how to convert the input values in the registers, which are represented in simple unary, into the wide-digits representation. We use the previous machine operations to create a reference value of magnitude $\Theta(n^{2/3})$ in a register and the usual base-conversion algorithms to extract the wide digits of each input register value and store them multiplied by the reference value; this initialization takes polylogarithmic parallel time [2], after which the per-step overhead of simulating the register machine is $O(\log^2 n)$. Thus, for simulating the register machine specified in [2], we have the following improvement.

Theorem 4. *A probabilistic population of n agents with a designated leader can simulate the steps of the virtual register machine defined in [2], such that the probability that any single step in the simulation fails or takes more than $O(n \log^2 n)$ interactions can be made $O(n^{-c})$ for any fixed c.*

8 A More Robust Phase Clock?

The fault tolerance of the 3-state approximate majority protocol and the inherent redundancy of the wide-digit representation of register values is encouraging: perhaps there is a fast and provably robust version of the register machine construction. However, there is a second component of our register machine that must be made more robust. This is the **phase clock**, a subprotocol used to count off intervals of $\Theta(\log n)$ parallel time so that the leader can estimate when an epidemic has finished propagating.

The phase clock of [2] is a protocol with m states and one designated leader. The state of an agent represents the phase that it is in. A responder in phase i adopts the phase of any initiator in phases $i{+}1 \bmod m$ through $i{+}m/2 \bmod m$, but ignores initiators in other phases. New phases are triggered by the leader agent. When the leader responds to an initiator in its own phase, that leader moves to the next phase. Counting the number of interactions between the event that the leader enters phase 0 until the event that the leader next enters phase 0 as a **round**, it is shown that with high probability each of a polynomial number of rounds takes parallel time $\Theta(\log n)$ with inverse polynomial probability of error. Note that although this protocol also works when given $\Theta(n^{1-\epsilon})$ designated leaders, it requires an initial configuration with the appropriate number of designated leaders, and is not robust to errors.

We now describe (a) a method for quickly starting the phase clock from a uniform initial state, and (b) a consensus variant of the phase clock that appears to be more robust. Our present techniques are not sufficient to analyze the resulting algorithm, so we must make do with simulation results.

Simulation results suggest that the following protocol can start up a phase clock with high probability in $\Theta(\log n)$ parallel time. This protocol is one-way, so for brevity we identify δ with δ_2.

This protocol is the semidirect product of several components. The first component allows us to make approximate coin tosses. The states are $Q_{\text{coin}} = \{0, 1\}$, with initial value $x_{\text{coin}} = 0$, and the transition function is $\delta_{\text{coin}}(q, q') = 1 - \pi_{\text{coin}}(q)$. Starting from any configuration, this protocol rapidly converges towards an equal proportion of agents in each state. The second component counts the number of consecutive coin values equal to 1 the agent has seen immediately prior, up to a maximum of $\ell > 0$. The states are $Q_{\text{count}} = \{0, 1, \ldots, \ell\}$, $x_{\text{count}} = 0$, and the transition function $\delta_{\text{count}}(q, q') = \pi_{\text{coin}}(q)(\min\{\pi_{\text{count}}(q') + 1, \ell\})$.

The third component approximates an exponential decay process. There is a parameter $k_1 \leq \ell$. The states are $Q_{\text{decay1}} = \{0, 1\}$, $x_{\text{decay1}} = 1$, and the update function is $\delta_{\text{decay1}}(q, q') = [\pi_{\text{count}}(q) < k_1]\pi_{\text{decay1}}(q')$. The idea is that for all $0 < \alpha < 1$ and $0 < c$, we can find k_1 such that with high probability, there is a period of $cn \log n$ steps where the number of agents with decay1 value of 1 is between 1 and n^α. In this period, using agents with decay1 value of 1 as temporary leaders, we can run a *disposable* phase clock that functions correctly only for a constant number of phases before all the values of decay1 become 0. This phase clock is used to choose a stable leader population of size $\Theta(n^{1-\epsilon})$, which in turn supports a second copy of the phase clock that runs for polynomially many steps.

Our consensus variant of the phase clock from [2] works as follows. In addition to the phases $0, 1, \ldots, \phi - 1$, we have a blank "phase". Thus $Q_{\text{phase1}} = \{b, 0, 1, \ldots, \phi - 1\}$. $x_{\text{phase1}} = 0$. If x is a nonblank phase, then let $\operatorname{succ}(x) = (x + 1) \bmod \phi$ be the successor phase of x. We have

$$\delta_{\text{phase1}}(q, q') = \begin{cases} p' & \text{if } p = b \\ p & \text{if } p' = b \\ p' & \text{if } p' = p \neq b \text{ and } \pi_{\text{decay1}}(q) = 0 \\ \operatorname{succ}(p') & \text{if } p' = p \neq b \text{ and } \pi_{\text{decay1}}(q) = 1 \\ p' & \text{if } p, p' \neq b \text{ and } p' = \operatorname{succ}(p) \\ p & \text{if } p, p' \neq b \text{ and } \operatorname{succ}(p') = p \\ b & \text{otherwise,} \end{cases}$$

where $p = \pi_{\text{phase1}}(q)$ and $p' = \pi_{\text{phase1}}(q')$. If the phase of the initiator is blank or one behind the responder's, the responder's phase is unchanged. If the phase of the responder is blank, it copies the phase of the initiator. If the phases are nonblank and equal, the responder increments its phase if and only if the initiator has decay value 1 (temporary leader status.) If the initiator's phase is one more than the responder's, the responder increments its phase. In all other cases, the responder sets its phase to blank. In summary, we are following a multiple-valued generalization of the 3-state majority algorithm *except* when the phases are nonblank and within distance 1 of one another. In this case, we revert to behavior like that of the original phase clock.

Once the disposable phase clock is running, it is used to select the real phase clock's leaders. This is accomplished by having another exponential decay process that is reset by the disposable phase clock each complete cycle. Thus we need a way to detect approximately the onset of each cycle. Our criterion is for each agent to keep a local "maximum" of the phases it has been in, and perform the reset when this maximum wraps around. Formally, $Q_{\text{max}} = \{0, 1, \ldots, \phi - 1\}$, $x_{\text{max}} = 0$, and

$$\delta_{\text{max}}(q, q') = \begin{cases} p' & \text{if } p' \neq b \text{ and } (p' - \pi_{\text{max}}(q')) \bmod \phi \leq \phi/2 \\ \pi_{\text{max}}(q') & \text{otherwise,} \end{cases}$$

where $p' = \delta_{\text{phase1}}(q, q')$. Since the last cycle may be partial, we also need a one-value history for the decay process. Now $Q_{\text{decay2}} = \{0, 1\} \times \{0, 1\}$, $x_{\text{decay2}} = (1, 1)$, and

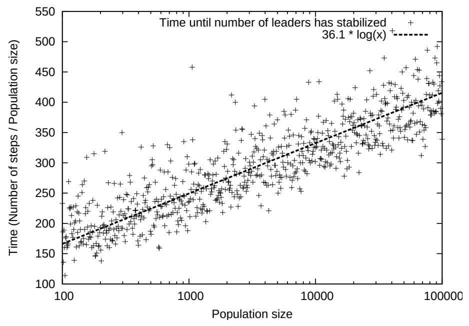
$$\delta_{\text{decay2}}(q, q') = \begin{cases} (1, y) & \text{if } \delta_{\text{max}}(q, q') < \pi_{\text{max}}(q') \\ ([\pi_{\text{count}}(q) < k_2]y, y') & \text{otherwise,} \end{cases}$$

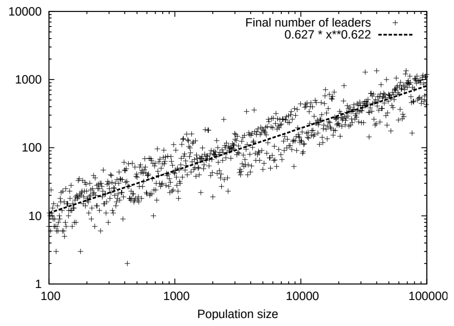
where $(y, y') = \delta_{\text{decay2}}(q')$. The final set of leaders are those agents with $y' = 1$ when the disposable phase clock stops running. A second copy of the consensus phase clock, running from the initial configuration using $y' = 1$ to designate leaders, rapidly converges in simulation to correct robust phase clock behavior when the number of leaders becomes appropriate.

We implemented the disposable phase clock leader election protocol and tried it once on each value of $\lfloor 1.01^n \rfloor$ between 100 and 100000 for integers n, with every agent in the same initial state. There are three parameters to tune: ϕ, k_1, and k_2. The protocol is not very sensitive to the settings of these parameters, but the setting $\phi = 9$, $k_1 = 5$, and $k_2 = 4$ worked better than many others.

The results are depicted in Figure 2. As can be seen, it seems that the protocol generally leaves $\Theta(n^{1-\epsilon})$ leaders and completely converges in $O(\log n)$ time.

Fig. 2. Simulation results: parallel time of leader election and final number of leaders





9 Acknowledgments

The second author would like to thank Joanna Ellman-Aspnes for suggestions that helped overcome an obstacle in the proof of the main lemma bounding convergence. The authors would like to thank the DISC 2007 reviewers for their helpful comments.

References

1. Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, pages 235–253, March 2006.
2. Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. In *Distributed Computing: 20th International Symposium, DISC 2006: Stockholm, Sweden, September 2006: Proceedings*, pages 61–75, September 2006.
3. William Feller. *An Introduction to Probability and its Applications*, volume 1. John Wiley and Sons, third edition, 1958.
4. Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.
5. Daniel T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, 188:404–425, 1992.
6. Thomas G Kurtz. *Approximation of Population Processes*. Number 36 in CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, 1981.
7. Nicholas C. Wormald. Differential equations for random processes and random graphs. *Annals of Applied Probability*, 5(4):1217–1235, November 1995.