A time and space optimal stable population protocol solving exact majority

David Doty
University of California, Davis
doty@ucdavis.edu

Mahsa Eftekhari
University of California, Davis
mhseftekhari@ucdavis.edu

Leszek Gąsieniec
University of Liverpool
l.a.gasieniec@liverpool.ac.uk

Eric Severson
University of California, Davis
eseverson@ucdavis.edu

Przemysław Uznański
University of Wrocław
puznanski@cs.uni.wroc.pl

Grzegorz Stachowiak
University of Wrocław
puznanski@cs.uni.wroc.pl

2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS) | 978-1-6654-2055-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/FOCS52979.2021.00104

***Abstract*—We study population protocols, a model of distributed computing appropriate for modeling well-mixed chemical reaction networks and other physical systems where agents exchange information in pairwise interactions, but have no control over their schedule of interaction partners. The *majority* problem is that of determining in an initial population of n agents, each with one of two opinions A or B, whether there are more A, more B, or a tie. A *stable* protocol solves this problem with probability 1 by eventually entering a configuration in which all agents agree on a correct consensus decision of A, B, or T, from which the consensus cannot change. We describe a protocol solving this problem using $O(\log n)$ states ($\log\log n + O(1)$ bits of memory) and optimal expected time $O(\log n)$. The number of states $O(\log n)$ is known to be optimal for polylogarithmic time stable protocols that are "output dominant" and "monotone" [1]. These are two natural constraints satisfied by our protocol, making it simultaneously time- and state-optimal for that class. We introduce a key technique called a "fixed resolution clock" to achieve partial synchronization.**

Our protocol is *nonuniform*: the transition function has the value $\lceil \log n \rceil$ encoded in it. We show that the protocol can be modified to be uniform, while increasing the state complexity to $\Theta(\log n \log\log n)$.

***Keywords*-majority; population protocols; stable;**

I. Introduction

Population protocols [2] are asynchronous, complete networks that consist of computational entities called *agents* with no control over the schedule of interactions with other agents. In a population of n agents, repeatedly a random pair of agents is chosen to interact, each observing the state of the other agent before updating its own state. They are an appropriate model for electronic computing scenarios such as sensor networks and for "fast-mixing" physical systems such as animal populations [3], gene regulatory networks [4], and chemical reactions [5], the latter increasingly regarded as an implementable "programming language" for molecular engineering, due to recent experimental breakthroughs in DNA nanotechnology [6,7].

Time complexity in a population protocol is defined by *parallel time*: the total number of interactions divided by the population size n, henceforth called simply *time*. This captures the natural timescale in which each individual agent experiences expected $O(1)$ interactions per unit time. All problems solvable with zero error probability by a constant-state population protocol are solvable in $O(n)$ time [8, 9]. The benchmark for "efficient" computation is thus sublinear time, ideally $\mathrm{polylog}(n)$, with $\Omega(\log n)$ time a lower bound on most nontrivial computation, since a simple coupon collector argument shows that is the time required for each agent to have at least one interaction.

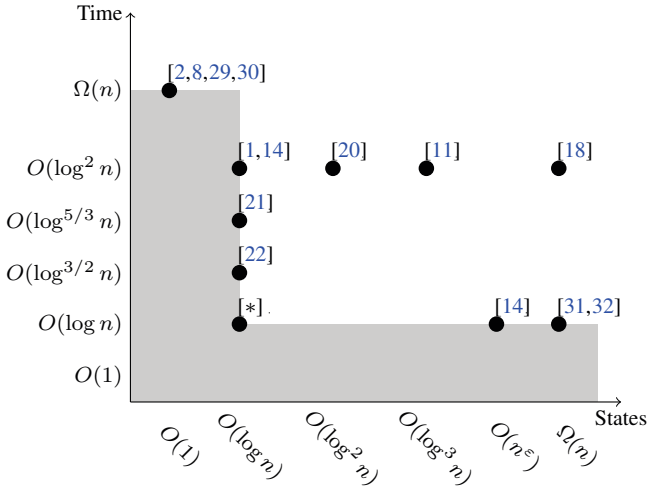
As a simple example of time complexity, suppose we want to design a protocol to decide whether at least one x exists in an initial population of x's and q's. The single transition $x, q \to x, x$ indicates that if agents in states x and q interact, the q agent changes state to x. If x outputs "yes" and q outputs "no", this takes expected time $O(\log n)$ to reach a consensus of all x's (i.e., $O(n \log n)$ total interactions, including null interactions between two x's or between two q's). However, the transitions $x, x \to y, y$; $y, x \to y, y$; $y, q \to y, y$, where x, q output "no" and y outputs "yes", which computes whether at least *two* x's exist, is exponentially slower: expected time $O(n)$. The worst-case input is exactly 2 x's and $n-2$ q's, where the first interaction between the x's takes expected $\binom{n}{2} = \Theta(n^2)$ interactions, i.e., $\Theta(n)$ time.

To have probability 0 of error, a protocol must eventually *stabilize*: reach a configuration where all agents agree on the correct output, which is *stable*, meaning no subsequently reachable configuration can change the output.[1] The original model [2] assumed states and

[1]Technically this connection between probability 1 correctness and reachability requires the number of producible states for any fixed population size n to be finite, which is the case for our protocol.

2575-8454/21/$31.00 ©2021 IEEE
DOI 10.1109/FOCS52979.2021.00104

Table I: Summary of results on the stable exact majority problem in population protocols, including this paper [$*$]. Gray regions are provably impossible: $o(\log\log n)$ state, $o(n)$ time unconditionally [11], $o(\log n)$ state, $O(n^{1-\varepsilon})$ time for monotone, output-dominant protocols [1], and $o(\log n)$ time unconditionally.



transitions are constant with respect to n. However, for important problems such as leader election [10], majority computation [11], and computation of other functions and predicates [12], no constant-state protocol can stabilize in sublinear time with probability 1.[2] This has motivated the study of population protocols whose number of states is allowed to grow with n, and as a result they can solve such problems in polylogarithmic time [1, 11, 14–28].

A. The majority problem in population protocols

Angluin, Aspnes, and Eisenstat [33] showed a protocol they called *approximate majority*, which means that starting from an initial population of n agents with opinions A or B, if $|A - B| = \omega(\sqrt{n}\log n)$ (i.e., the *gap* between the initial majority and minority counts is greater than roughly $\sqrt{n}$), then with high probability the algorithm stabilizes to all agents adopting the majority opinion in $O(\log n)$ time. A tighter analysis by Condon, Hajiaghayi, Kirkpatrick, and Maňuch [34] reduced the required gap to $\Omega(\sqrt{n\log n})$.

Mertzios, Nikoletseas, Raptopoulos, and Spirakis [30], and independently Draief and Vojnović [29], showed a 4-state protocol that solves *exact* majority problem, i.e., it identifies the majority correctly, no matter how small the initial gap.[3] We henceforth refer to this simply as the *majority* problem. The protocol of [29, 30] is also *stable* in the sense that it has probability 1 of getting the correct answer. However, this protocol takes $\Omega(n)$ time in the worst case: when the gap is $O(1)$. Known work on the stable majority problem is summarized in Table I. Gąsieniec, Hamilton, Martin, Spirakis, and Stachowiak [36] investigated $\Omega(n)$ time protocols for majority and the more general "plurality consensus" problem. Blondin, Esparza, Jaax, and Kučera [37] show a similar stable (also $\Omega(n)$ time) majority protocol that also reports if there is a tie.

Alistarh, Gelashvili, and Vojnović [18] showed the first stable majority protocol with worst-case polylogarithmic expected time, requiring $\Omega(n)$ states. A series of positive results reduced the state and time complexity for stable majority protocols [1, 11, 14, 20–22, 31, 32]. Ben-Nun, Kopelowitz, Kraus, and Porat showed the current fastest stable sublinear-state protocol [22] using $O(\log^{3/2} n)$ time and $O(\log n)$ states. The current state-of-the-art protocols use alternating phases of *cancelling* (two biased agents with opposite opinions both become unbiased, preserving the difference between the majority and minority counts) and *splitting* (a.k.a. *doubling*): a biased agent converts an unbiased agent to its opinion; if all biased agents that didn't cancel can successfully split in that phase, then the count difference doubles. The goal is to increase the count difference until it is n; i.e., all agents have the majority opinion. See [15,38] for relevant surveys. Our protocol uses the same framework of cancelling and splitting for $O(\log n)$ phases, but uses *constant* time per phase. This requires a novel phase clock construction, and handling new types of errors introduced by the clock's faster pace.

Some non-stable protocols solve exact majority with high probability but have a positive probability of incorrectness. Berenbrink, Elsässer, Friedetzky, Kaaser, Kling, and Radzik [14] showed a protocol that with initial gap α uses $O(s + \log\log n)$ states and WHP converges in $O(\log n \log_s(\frac{n}{\alpha}))$ time.[4] With $\alpha = 1$ and $s = O(1)$, the protocol uses $O(\log\log n)$ states and converges in $O(\log^2 n)$ time. Kosowski and Uznański [13] showed a protocol using $O(1)$ states and converging in $O(\text{polylog}(n))$ time with high probability.

[2]These problems have $O(1)$ state, sublinear time *converging* protocols [13]. A protocol *converges* when it reaches the correct output without subsequently changing it—though it may remain change*able* for some time after converging—whereas it *stabilizes* when the output becomes unchangeable. See [10,14] for a discussion of the distinction between stabilization and convergence. In this paper we consider only stabilization time.

[3]The 4-state protocol doesn't identify ties, (gap $= 0$), but this can be handled with 2 more states; see Stable-Backup in [35].

[4]This protocol is SIMPLEMAJORITY in [14], which they then build on to achieve multiple stable protocols. The stable protocols require either $\Omega(n)$ stabilization time or $\Omega(\log n)$ states to achieve sublinear stabilization time.

On the negative side, Alistarh, Aspnes, and Gelashvili [1] showed that any stable majority protocol taking (roughly) less than linear time requires $\Omega(\log n)$ states if it also satisfies two conditions (satisfied by all known stable majority protocols, including ours): *monotonicity* and *output dominance*. These concepts are discussed in Section V. In particular, the $\Omega(\log n)$ state bound of [1] applies only to *stable* (probability 1) protocols; the high probability protocol of [14], for example, uses $O(\log \log n)$ states and $O(\log^2 n)$ time.

B. Our contribution

We show a stable population protocol solving the exact majority problem in optimal $O(\log n)$ time (in expectation and with high probability) that uses $O(\log n)$ states. Our protocol is both monotone and output dominant (see Section V or [1] for discussion of these definitions), so by the $\Omega(\log n)$ state lower bound of [1], our protocol is both time and space optimal for the class of monotone, output-dominant stable protocols.

A high-level overview of the algorithm is given in Sections III-A and III-B, with a full formal description given in [35]. Like most known majority protocols using more than constant space (the only exceptions being in [14]), our protocol is *nonuniform*: agents have an estimate of the value $\lceil \log n \rceil$ embedded in the transition function and state space. Section IV describes how to modify our main protocol to make it uniform, retaining the $O(\log n)$ time bound, but increasing the state complexity to $O(\log n \log \log n)$ in expectation and with high probability. That section discusses challenges in creating a uniform $O(\log n)$ state protocol.

II. Preliminaries

We write $\log n$ to denote $\log_2 n$, and $\ln n$ to denote the natural logarithm. We write $x \sim y$ to denote that x and y are asymptotically equivalent (implicitly in the population size n), meaning $\lim_{n\to\infty} \frac{x(n)}{y(n)} = 1$.

A. Population protocols

A *population protocol* is a pair $\mathcal{P} = (\Lambda, \delta)$, where Λ is a finite set of *states*, and $\delta : \Lambda \times \Lambda \to \Lambda \times \Lambda$ is the *transition function*.[5] In this paper we deal with *nonuniform* protocols in which a different Λ and δ are allowed for different population sizes n (one for each possible value of $\lceil \log n \rceil$), but we abuse terminology and refer to the whole family as a single protocol. In all cases (as with similar nonuniform protocols), the nonuniformity is used to embed the value $\lceil \log n \rceil$ into each agent; the transitions are otherwise "uniformly specified". See Section IV for more discussion of uniform protocols.

A *configuration* $\mathbf{c}$ of a population protocol is a multiset over Λ of size n, giving the states of the n agents in the population. For a state $s \in \Lambda$, we write $\mathbf{c}(s)$ to denote the count of agents in state s. A *transition* (a.k.a., *reaction*) is a 4-tuple $\alpha = (r_1, r_2, p_1, p_2)$, written $\alpha : r_1, r_2 \to p_1, p_2$, such that $\delta(r_1, r_2) = (p_1, p_2)$. If an agent in state r_1 interacts with an agent in state r_2, then they change states to p_1 and p_2. For every pair of states r_1, r_2 without an explicitly listed transition $r_1, r_2 \to p_1, p_2$, there is an implicit *null* transition $r_1, r_2 \to r_1, r_2$ in which the agents interact but do not change state. For our main protocol, we specify transitions formally with pseudocode that indicate how agents alter each independent field in their state. We say a configuration $\mathbf{d}$ is *reachable* from a configuration $\mathbf{c}$ if applying 0 or more transitions to $\mathbf{c}$ results in $\mathbf{d}$.

B. Stable majority computation

There are many modes of computation considered in population protocols: computing integer-valued functions [9, 12, 39] where the number of agents in a particular state is the output, Boolean-valued predicates [8,40] where each agent outputs a Boolean value as a function of its state and the goal is for all agents eventually to have the correct output, problems such as leader election [10, 11, 19, 25–28], and generalizations of predicate computation, where each agent individually outputs a value from a larger range, such as reporting the population size [16,23,24]. Majority computation is Boolean-valued if computing the predicate "$A \geq B$?", where A and B represent the initial counts of two opinions A and B. We define the slightly generalized problem that requires recognizing when there is a tie, so the range of outputs is $\{\mathsf{A}, \mathsf{B}, \mathsf{T}\}$.

Formally, if the set of states is Λ, the protocol defines a disjoint partition of $\Lambda = \Lambda_{\mathsf{A}} \cup \Lambda_{\mathsf{B}} \cup \Lambda_{\mathsf{T}}$. For $u \in \{\mathsf{A}, \mathsf{B}, \mathsf{T}\}$, if $a \in \Lambda_u$ for all $a \in \mathbf{c}$, we define *output* $\phi(\mathbf{c}) = u$ (i.e., all agents in $\mathbf{c}$ agree on the output u). Otherwise $\phi(\mathbf{c})$ is undefined (i.e., agents disagree on the output). We say $\mathbf{o}$ is *stable* if $\phi(\mathbf{o})$ is defined and, for all $\mathbf{o}_2$ such that $\mathbf{o} \Rightarrow \mathbf{o}_2$, $\phi(\mathbf{o}) = \phi(\mathbf{o}_2)$, i.e., the output cannot change.

The protocol identifies two special *input states* $A, B \in \Lambda$. A *valid* initial configuration $\mathbf{i}$ satisfies $a \in \{A, B\}$ for all $a \in \mathbf{i}$. We say the *majority opinion* of $\mathbf{i}$ is $M(\mathbf{i}) = \mathsf{A}$ if $\mathbf{i}(A) > \mathbf{i}(B)$, $M(\mathbf{i}) = \mathsf{B}$ if $\mathbf{i}(A) < \mathbf{i}(B)$, and $M(\mathbf{i}) = \mathsf{T}$ if $\mathbf{i}(A) = \mathbf{i}(B)$. The protocol *stably*

[5] To understand the full generality of our main protocol, we include randomized transitions in our model. However, there is only one type of randomized transition in the protocol (the "drip reactions" of Phase 3 described in Section III-B), parameterized by probability p, and in fact we prove the protocol works even when these transitions are deterministic, i.e., when $p = 1$.

computes majority if, for any valid initial configuration $\mathbf{i}$, for all $\mathbf{c}$ such that $\mathbf{i} \Rightarrow \mathbf{c}$, there is a stable $\mathbf{o}$ such that $\mathbf{c} \Rightarrow \mathbf{o}$ and $\phi(\mathbf{o}) = M(\mathbf{i})$. Let $O_\mathbf{i} = \{\mathbf{o} : \phi(\mathbf{o}) = M(\mathbf{i})\}$ be the set of all correct, stable configurations. In other words, for any reachable configuration, it is possible to reach a correct, stable configuration, or equivalently reach a strongly connected component in $O_\mathbf{i}$.

C. Time complexity

In any configuration the next interaction is chosen by selecting a pair of agents uniformly at random and applying an applicable transition, with appropriate probabilities for any randomized transitions. Thus the sequence of transitions and configurations they reach are random variables. To measure time we count the total number of interactions (including null transitions such as $a, b \to a, b$ in which the agents interact but do not change state), and divide by the number of agents n. In the population protocols literature, this is often called "parallel time": n interactions among a population of n agents equals one unit of time.

If the protocol stably computes majority, then for any valid initial configuration $\mathbf{i}$, the probability of reaching a stable, correct configuration, $\mathbb{P}[\mathbf{i} \Rightarrow O_\mathbf{i}] = 1$. [6] We define the *stabilization time* S to be the random variable giving the time to reach a configuration $\mathbf{o} \in O_\mathbf{i}$.

When discussing random events in a protocol of population size n, we say event E happens *with high probability* if $\mathbb{P}[\neg E] = O(n^{-c})$, where c is a constant that depends on our choice of parameters in the protocol, where c can be made arbitrarily large by changing the parameters. In other words, the probability of failure can be made an arbitrarily small polynomial. For concreteness, we will write a particular polynomial probability such as $O(n^{-2})$, but in each case we could tune some parameter (say, increasing the time complexity by a constant factor) to increase the polynomial's exponent. We say event E happens *with very high probability* if $\mathbb{P}[\neg E] = O(n^{-\omega(1)})$, i.e., if its probability of failure is smaller than any polynomial probability.

III. Nonuniform majority algorithm description

The following is the main theorem of this paper.

Theorem III.1. *There is a nonuniform population protocol Nonuniform-Majority, using $O(\log n)$ states, that stably computes majority in $O(\log n)$ stabilization time, both in expectation and with high probability.*

[6] Since population protocols have a finite reachable configuration space, this is equivalent to the stable computation definition that for all $\mathbf{c}$ reachable from $\mathbf{i}$, there is a $\mathbf{o}' \in O_\mathbf{i}$ reachable from $\mathbf{c}$.

A. High-level overview of algorithm

In this overview we use "pseudo-transitions" such as $A, B \to \mathcal{O}, \mathcal{O}$ to describe agents updating a portion of their states, while ignoring other parts of the state space.

Each agent initially has a `bias`: $+1$ for opinion A and -1 for opinion B, so the population-wide sum $g = \sum_v v.\texttt{bias}$ gives the *initial gap* between opinions. The majority problem is equivalent to determining $\text{sign}(g)$. Transitions redistribute biases among agents but, to ensure correctness, maintain the population-wide g as an invariant. Biases change through *cancel reactions* $+\frac{1}{2^i}, -\frac{1}{2^i} \to 0, 0$ and *split reactions* $\pm\frac{1}{2^i}, 0 \to \pm\frac{1}{2^{i+1}}, \pm\frac{1}{2^{i+1}}$, down to a minimum $\pm\frac{1}{2^L}$. The constant $L = \lceil \log_2(n) \rceil$ ensures $\Theta(\log n)$ possible states. The *gap* is defined to be $\sum_v \text{sign}(v.\texttt{bias})$, the difference in counts between majority and minority biases. Note the gap should grow over time to spread the correct majority opinion to the whole population, while the invariant g should ensure correctness of the final opinion.

The cancel and split reactions average the bias value between both agents, but only when the average is also a power of 2, or 0. If we had averaging reactions between all pairs of biases (also allowing, e.g., $\frac{1}{2}, \frac{1}{4} \to \frac{3}{8}, \frac{3}{8}$), then all biases would converge to $\frac{g}{n}$, but this would use too many states.[7] With our limited set $\{0, \pm\frac{1}{2}, \pm\frac{1}{4}, \ldots, \pm\frac{1}{2^L}\}$ of possible biases, allowing all cancel and split reactions simultaneously does not work. Most biases appear simultaneously across the population, reducing the count of each bias, which slows the rate of cancel reactions. Then the count of unbiased 0 agents is reduced, which slows the rate of split reactions, see Fig. 1a. Also, there is a non-negligible probability for the initial minority opinion to reach a much greater count, if those agents happen to do more split reactions, see Fig. 1b. Thus using only the count of positive versus negative biases will not work to solve majority even with high probability.

To solve this problem, we partially synchronize the unbiased agents with a field `hour`, adding $\log n$ states $0_0, 0_1, 0_2, \ldots, 0_L$. The new split reactions

$$\pm\frac{1}{2^i}, 0_h \to \pm\frac{1}{2^{i+1}}, \pm\frac{1}{2^{i+1}} \quad \text{if } h > i$$

will wait until $\texttt{hour} \geq h$ before doing splits down to $\texttt{bias} = \pm\frac{1}{2^h}$. We could use existing phase clocks to perfectly synchronize `hour`, by making each hour use $\Theta(\log n)$ time, enough time for all opinionated agents to split. Then WHP all agents would be in states $\{0_h, +\frac{1}{2^h}, -\frac{1}{2^h}\}$ by the end of hour h, see Fig. 1c. The

[7] This was effectively the approach used for majority in [18,32], for an $O(n)$ state, $O(\log n)$ time protocol.

invariant $g = \sum_v v.\texttt{bias}$ implies that all minority opinions would be eliminated by hour $\lceil \log_2 \frac{1}{g} \rceil \leq L$. This would give an $O(\log n)$-state, $O(\log^2 n)$-time majority algorithm, essentially equivalent to [1, 14].

The main idea of our algorithm is to use these rules with a faster clock using only $O(1)$ time per hour. The `hour` field of unbiased agents is synchronized to a separate subpopulation of clock agents, who use a field `minute`, with k consecutive minutes per hour. Minutes advance by *drip reactions* $C_i, C_i \to C_i, C_{i+1}$, and catch up by *epidemic reactions* $C_i, C_j \to C_{\max(i,j)}, C_{\max(i,j)}$. See Fig. 2 for an illustration of the clock `minute` and `hour` dynamics.

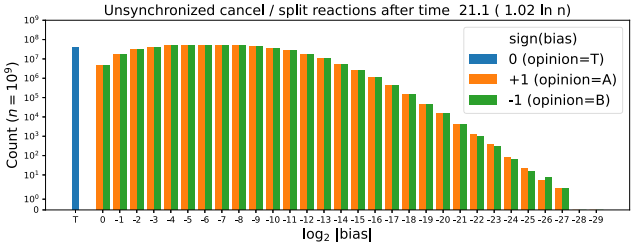
Since $O(1)$ time per hour is not sufficient to bring all agents up to the current hour before advancing to the next, we now have only a large constant fraction of agents, rather than all agents, synchronized in the current hour. Still, we prove this looser synchronization keeps the values of `hour` and `bias` relatively concentrated, so by the end of this phase, we reach a configuration as shown in Fig. 1d. Most agents have the majority opinion (WLOG positive), with three consecutive biases $+\frac{1}{2^l}, +\frac{1}{2^{l+1}}, +\frac{1}{2^{l+2}}$.

Detecting ties.: This algorithm gives an elegant way to detect a tie with high probability. In this case, $g = 0$, and with high probability, all agents will finish the phase with $\texttt{bias} \in \{0, \pm\frac{1}{2^L}\}$. Checking this condition *stably* detects a tie (i.e., with probability 1, if this condition is true, then there is a tie), because for any nonzero value of g, there must be some agent with $|\texttt{bias}| > \frac{1}{2^L}$.

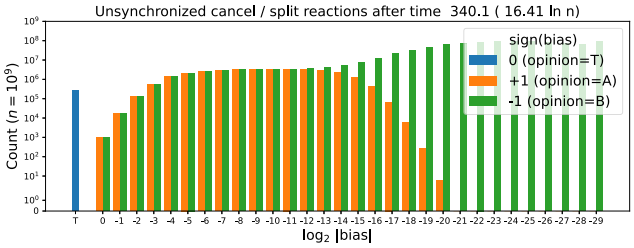
Cleanup Phases.: We must next eliminate all minority opinions, while still relying on the invariant $g = \sum_v v.\texttt{bias}$ to ensure correctness. Note that is it possible with low probability to have a greater count of minority opinions (with smaller values of `bias`), so only relying on counts of positive and negative biases would give possibilities of error.

We first remove any minority agents with large bias, by using an additional subpopulation of Reserve agents that enable additional split reactions for large values of $|\texttt{bias}| > \frac{1}{2^l}$. Then after cancel reactions with the bulk of majority agents, the only minority agents left must have $|\texttt{bias}| < \frac{1}{2^{l+2}}$.

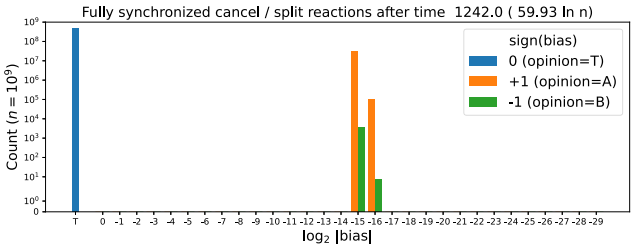
To then remove minority agents with small bias, we allow agents with larger bias to "consume" agents with smaller bias, such as an interaction between agents $+\frac{1}{4}$ and $-\frac{1}{256}$. Here the positive agent can be thought to hold the entire bias $+\frac{1}{4} - \frac{1}{256} = +\frac{63}{256}$, but since this value is not in the allowable states, it can only store that its bias is in the range $+\frac{1}{8} \leq \texttt{bias} \leq +\frac{1}{4}$. Without



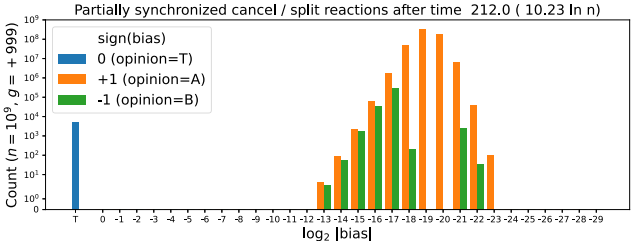
(a) Cancel/split reactions with no synchronization. All states become present, many in about equal counts. Rate of cancel reactions and fraction of 0 agents are $\Theta(\frac{1}{\log n})$.



(b) Later snapshot of the simulation in Fig. 1a. The initial minority B now has a much larger count, because those agents happened to undergo more split reactions.



(c) Cancel/split reactions, fully synchronized into $O(\log n)$ time hours, at the beginning of hour 16. All minority are eliminated by hour $\log n$ in $O(\log^2 n)$ time.



(d) Main phase of our protocol, split reactions partially synchronized using the clock in Fig. 2, at the end of this $O(\log n)$ time phase. Most agents are left with $\texttt{bias} \in \{+\frac{1}{2^{18}}, +\frac{1}{2^{19}}, +\frac{1}{2^{20}}\}$. Later phases eliminate the remaining minority agents.

Figure 1: Cancel / split reactions with no synchronization (1a,1b), perfect synchronization (1c), and partial synchronization (1d) via the fixed-resolution phase clock of our main protocol. Plots generated from [41].

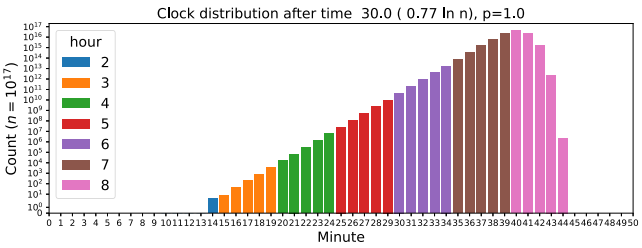


Figure 2: Clock rules of our protocol, showing a travelling wave distribution over minutes, on a larger population of size $n = 10^{17}$ to emphasize the distribution. The distribution's back tail decays exponentially, and its front tail decays doubly exponentially. A large constant fraction of agents are in the same two consecutive `hour`'s (here 7 and 8). Plot generated from [41].

knowing its exact `bias`, this agent cannot participate in future averaging interactions. However, with high probability there are sufficient majority agents to eliminate all remaining minority via these *consumption reactions*.

A final phase checks for the presence of both positive and negative `bias`, and if one has been completely eliminated, it stabilizes to the correct output. In the case where both are present, this is a detectable error, where we can move to a slow, correct backup that uses the original inputs. Due to the low probability of this case, it contributes negligibly to the total expected time.

B. Intuitive description of each phase

Our full protocol is broken up into 11 consecutive phases. We describe each phase intuitively. Full pseudocode is given in [35]. Note that some further separation of phases was done to create more straightforward proofs of correctness, so simplicity of the proofs was optimized over simplicity of the full protocol pseudocode. It is likely possible to have simpler logic that still solves majority via the same strategy.

Phase 0: "Population splitting" [38] divides agents into roles used in subsequent phases: Main, Reserve, Clock. In timed phases (those not marked as *Untimed* or *Fixed-resolution clock*, including the current phase), Clock agents count from $\Theta(\log n)$ to 0 to cause the switch to the next phase after $\Theta(\log n)$ time.

"Standard" population splitting uses reactions such as $x, x \to r_1, r_2$ to divide agents into two roles r_1, r_2. This takes $\Theta(n)$ time to converge, which can be decreased to $\Theta(\log n)$ time via $r_1, x \to r_1, r_2$ and $r_2, x \to r_2, r_1$, while maintaining that $\#r_1$ and $\#r_2$ are both $n/2 \pm \sqrt{n}$ WHP. However, since all agents initially have an opinion, but Clock and Reserve agents do not hold an opinion, agents that adopt role Clock or Reserve must first pass off their opinion to a Main agent.

From each interacting pair of unassigned agents, one will take the Main role and hold the opinions of both agents, interpreting A as $+1$ and B as -1. This Main agent will then be allowed to take at most one other opinion (in an additional reaction that enables rapid convergence of the population splitting), and holding 3 opinions can end up with a bias in the range $\{-3, -2, -1, 0, +1, +2, +3\}$.

Phase 1: Agents do "integer averaging" [25] of biases in the set $\{-3, \ldots, +3\}$ via reactions $i, j \to \lfloor \frac{i+j}{2} \rfloor, \lceil \frac{i+j}{2} \rceil$. Although taking $\Theta(n)$ time to converge in some cases, this process is known [42] to result in three consecutive values in $O(\log n)$ time. If those three values are detected to be $\{-1, 0, +1\}$ in the next phase, the algorithm continues.

Phase 2: (*Untimed*) Agents propagate the set of opinions (signs of biases) remaining after Phase 1 to detect if only one opinion remains. If so, we have converged on a majority consensus, and the algorithm halts here. At this point, this is essentially the exact majority protocol of [30], which takes $O(\log n)$ time with an initial gap $\Omega(n)$, but longer for sublinear gaps (e.g., $\Omega(n)$ time for a gap of 1). Thus, if agents proceed beyond this phase (i.e., if both opinions A and B remain at this point), we will use later that the gap was smaller than $0.025 \cdot \#\text{Main}$. With low probability both opinions remain but some agent has $|\texttt{bias}| > 1$, in which case we proceed directly to a slow stable backup protocol in Phase 10.

Phase 3: (*Fixed-resolution clock*) The key goal at this phase is to use cancel and split reactions to average the bias across the population to give almost all agents the majority opinion. Biased agents hold a field $\texttt{exponent} \in \{-L, \ldots, -1, 0\}$, describing the magnitude $|\texttt{bias}| = 2^{\texttt{exponent}}$, a quantity we call the agent's *mass*. Cancel reactions eliminate opposite biases $+\frac{1}{2^i}, -\frac{1}{2^i} \to 0, 0$ with the same `exponent`; cancel reactions strictly reduce total mass. Split reactions $\pm\frac{1}{2^i}, 0 \to \pm\frac{1}{2^{i+1}}, \pm\frac{1}{2^{i+1}}$ give half of the bias to an unbiased agent, decrementing the `exponent`; split reactions preserve the total mass. The unbiased $\mathcal{O}$ agents, with $\texttt{role} = \text{Main}$, $\texttt{opinion} = \texttt{bias} = 0$, act as the fuel for split reactions. We want to obtain tighter synchronization in the exponents than Fig. 1a, approximating the ideal synchronized behavior of the $O(\log^2 n)$ time algorithm of Fig. 1c while using only $O(\log n)$ time. To achieve this, the Clock agents run a "fixed resolution" clock that keeps them roughly synchronized (though not perfectly; see Fig. 2) as they count their "minutes" from 0 up

to $L' = kL$, using $O(1)$ time per minute. This is done via "drip" reactions $C_i, C_i \to C_i, C_{i+1}$ (when minute i gets sufficiently populated, pairs of C_i agents meet with sufficient likelihood to increment the minute) and $C_j, C_i \to C_j, C_j$ for $i < j$ (new higher minute propagates by epidemic).[8] If randomized transitions are allowed, by lowering the probability p of the drip reaction, the clock rate can be slowed by a constant factor (see Fig. 3). Although we prove a few lemmas about this generalized clock, and some of our simulation plots use $p < 1$, our proofs work even for $p = 1$, i.e., a deterministic transition function, although this requires constant-factor more states (by increasing the number of "minutes per hour", explained next).

Now the $\mathcal{O}$ agents will use $\Theta(\log n)$ states to store an "hour", coupled to the C clock agents via $C_{\lfloor i/k \rfloor}, \mathcal{O}_j \to C_{\lfloor i/k \rfloor}, \mathcal{O}_{\lfloor i/k \rfloor}$ if $\lfloor i/k \rfloor > j$, i.e., every consecutive k Clock minutes corresponds to one Main hour, and clock agents drag $\mathcal{O}$ agents up to the current hour. Our proofs require $k = 45$ minutes per hour when $p = 1$, but smaller values of k work in simulation. For example, the simulation in Fig. 1d showing intended behavior of this phase used only $k = 3$ minutes per hour with $p = 1$. This clock synchronizes the `exponents` because agents with `exponent` $= -i$ can only split down to `exponent` $= -(i+1)$ with an $\mathcal{O}$ agent that has `hour` $\geq i + 1$. This prevents the biased agents from doing too many splits too quickly. As a result, during hour i, most of the biased agents have $|\texttt{bias}| = \frac{1}{2^i}$, so the cancel reactions $+\frac{1}{2^i}, -\frac{1}{2^i} \to 0, 0$ happen at a high rate, providing many $\mathcal{O}$ agents as "fuel" for future split reactions. We tune the constants of the clock to ensure hour i lasts long enough to bring most biased agents down to `exponent` $= -i$ via split reactions and then let a good fraction do cancel reactions.

[8]This clock is similar to the power-of-two-choices leaderless phase clock of [1], where the agent with smaller (or equal) minute increments their clock ($C_j, C_i \to C_j, C_{i+1}$ for $i \leq j$), but increasing the smaller minute by only 1. Similarly to our clock, the maximum minute can increase only with both agents at the same minute. A similar process was analyzed in [43], and in fact was shown to have the key properties needed for our clock to work—an exponentially-decaying back tail and a double-exponentially-decaying front tail—so it seems likely that a power-of-two-choices clock could also work with our protocol.

The randomized variant of our clock with drip probability p is also similar to the "junta-driven" phase clock of [26], but with a linear number $2pn$ of agents in the junta, using $O(1)$ time per minute, rather than the $O(n^\epsilon)$-size junta of [26], which uses $O(\log n)$ time per minute. There, smaller minutes are brought up by epidemic, and only an agent in the junta seeing another agent at the same minute will increment. The epidemic reaction is exactly the same in both rules. The probability p of a drip reaction can be interpreted as the probability that one of the agents is in the junta. For similar rate of $O(1)$ time per minute phase clock construction see also Dudek and Kosowski work [44].

The key property at the conclusion of this phase is that unless there is a tie, WHP most majority agents end up in three consecutive exponents $-l, -(l+1), -(l+2)$, with a negligible mass of any other Main agent (majority agents at lower/higher exponents, minority agents at any exponent, or $\mathcal{O}$ agents).[9] Phases 5-7 use this fact to quickly push the rest of the population to a configuration where *all* minority agents have exponents strictly below $-(l+2)$; Phase 8 then eliminates these minority agents quickly.

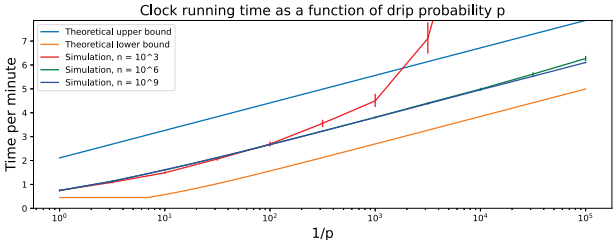


Figure 3: The theoretical upper and lower bounds for the time of one clock minute (from[35, Theorem 6.8]), along with samples from simulation. For each value of n, 100 minute times were sampled, taking $t_{i+1}^{0.1} - t_i^{0.1}$ for $i = 9, \ldots, 18$ over 10 independent trials. All our proofs assume p is constant, and for any fixed value of p, will only hold for sufficiently large n. The case $n = 10^3$ shows that when $p = O(1/n)$, the bounds no longer hold. This is to be expected because the expected number of drips becomes too small for large deviation bounds to still hold.

Phase 4: (*Untimed*) The special case of a tie is detected by the fact that, since the total bias remains the initial gap g, if all biased agents have minimal exponent $-L$, g has magnitude less than 1:

$$|g| = \left| \sum_{a.\texttt{role}=\text{Main}} a.\texttt{bias} \right| \leq \sum_{a.\texttt{role}=\text{Main}} |a.\texttt{bias}|$$
$$\leq \sum_{a.\texttt{role}=\text{Main}} \frac{1}{2^L} < \frac{n}{2^{\lceil \log_2(n) \rceil}} \leq 1.$$

The initial gap g is integer valued, so $|g| < 1 \implies g = 0$. Thus this condition implies there is a tie with probability 1; the converse that a tie forces all biased agents to exponent $-L$ holds with high probability. If only exponent $-L$ is detected, the algorithm halts here with all agents reporting output T. Otherwise, the algorithm proceeds to the next phase.

Phase 5, Phase 6: Using the key property of Phase 3, these phases WHP pull all biased agents above exponent $-l$ down to exponent $-l$ or below using the Reserve

[9]l is defined such that if all biased agents were at exponent $-l$, the difference in counts between majority and minority agents would be between $0.4 \cdot \#\text{Main}$ and $0.8 \cdot \#\text{Main}$.

R agents. The R's activate themselves in Phase 5 by sampling the exponent of the first biased agent they meet. This ensures WHP that sufficiently many Reserve agents exist with exponents $-l, -(l+1), -(l+2)$ (distributed similarly to the agents with those exponents). Then in Phase 6, they act as fuel for splits, via $R_i, \pm\frac{1}{2^j} \to \pm\frac{1}{2^{j+1}}, \pm\frac{1}{2^{j+1}}$ when $|i| > |j|$. The reserve agents, unlike the $\mathcal{O}$ agents in Phase 3, do not change their exponent in response to interactions with Clock agents. Thus sufficiently many reserve agents will remain to allow the small number of biased agents above exponent $-l$ to split down to exponent $-l$ or below.

Phase 7: This phase allows more general reactions to distribute the dyadic biases, allowing reactions between agents up to two exponents apart, to eliminate the opinion with smaller exponent: $\frac{1}{2^i}, -\frac{1}{2^{i+1}} \to \frac{1}{2^{i+1}}, 0$ and $\frac{1}{2^i}, -\frac{1}{2^{i+2}} \to \frac{1}{2^{i+1}}, \frac{1}{2^{i+2}}$ (and the equivalent with positive/negative biases swapped). Since all agents have exponent $-l$ or below, and many more majority agents exist at exponents $-l, -(l+1), -(l+2)$ than the total number of minority agents anywhere, these (together with standard cancel reactions $\frac{1}{2^i}, -\frac{1}{2^i} \to 0, 0$) rapidly eliminate all minority agents at exponents $-l, -(l+1), -(l+2)$, while maintaining $\Omega(n)$ majority agents at exponents $\geq -(l+2)$ and $< 0.01n$ total minority agents, now all at exponents $\leq -(l+3)$.

Phase 8: This phase eliminates the last minority agents, while ensuring that if any error occurred in previous phases, some majority agents remain, to allow detecting the error by the presence of both opinions.[10] The biased agents add a Boolean field `full`, initially False, and *consumption* reactions that allow an agent at a larger exponent i to consume (set to mass 0 by setting it to be $\mathcal{O}$) an agent at an arbitrarily smaller exponent $j < i$. Now the remaining agent represents some non-power-of-two mass $m = 2^i - 2^j$, which it lacks sufficient memory to track exactly. Thus setting the flag `full` = True corresponds to the agent having an uncertain mass m in the range $2^{i-1} \leq m < 2^i$. Because of this uncertainty, full agents are not allowed to consume other smaller levels. However, there are more than enough high-exponent majority agents by this phase to consume all remaining lower exponent minority agents.

[10]A naïve idea to reach a consensus at this phase is to allow cancel reactions $\frac{1}{2^i}, -\frac{1}{2^j} \to 0, 0$ between arbitrary pairs of exponents with opposite opinions. However, this has a positive probability of erroneously eliminating the majority. This is because the majority, while it necessarily has larger mass than the minority at this point, could have smaller *count*. For example, we could have 16 A's with `exponent` $= -2$ and 32 B's with `exponent` $= -5$, so A's have mass $16 \cdot 2^{-2} = 4$ and B's have smaller mass $32 \cdot 2^{-5} = 1$, but larger count than A.

Crucially, agents that have consumed another agent and set `full` = True may themselves then be consumed by a third agent (with `full` = False) at an even larger exponent. This is needed because a minority agent at exponent $i \leq -(l+3)$ may consume a (rare) majority agent at exponent $j < i$, but the minority agent itself can be consumed by another majority agent with exponent $k > i$.

Phase 9: (*Untimed*) This is identical to Phase 2: it detects whether both biased opinions A and B remain. If not (the likely case), the algorithm halts, otherwise we proceed to the next phase.

Phase 10: (*Untimed*) Agents execute a simple, slow stable majority protocol [37], similar to that of [29,30] but also handling ties. This takes $\Theta(n \log n)$ time, but the probability that an earlier error forces us to this phase is $O(1/n^2)$, so it contributes negligibly to the total expected time.

IV. Uniform, Stable Protocols for Majority Using More States

The algorithm described in Section III is *nonuniform*: the set of transitions used for a population of size n depends on the value $\lceil \log n \rceil$. A uniform protocol [16, 23, 45] is a single set of transitions that can be used in any population size. Since it is "uniformly specified", the transition function is formally defined by a linear-space Turing machine, where the space bound is the maximum space needed to read and write the input and output states. The original model [2] used $O(1)$ states and transitions for all n and so was automatically uniform, but many recent $\omega(1)$ state protocols are nonuniform. With the exception of the uniform variant in [14], all $\omega(1)$ state stable majority protocols are nonuniform [1, 11, 18, 20–22]. The uniform variant in [14] has a tradeoff parameter s that, when set to $O(1)$ to minimize the states, uses $O(\log n \log\log n)$ states and $O(\log^2 n)$ time.

In this section we show that there is a way to make Nonuniform-Majority in Section III uniform, retaining the $O(\log n)$ time bound, but the expected number of states increases to $\Theta(\log n \log\log n)$.[11]

A. Main idea of $O(\log n \log\log n)$ state uniform majority (not handling ties)

Since Nonuniform-Majority uses the hard-coded value $L = \lceil \log n \rceil$, to make the algorithm uniform, we require a way to estimate $\log n$ and store it in a field L (called `logn` below) of each agent. For correctness and

[11]We say "expected" because this protocol has a non-zero probability of using an arbitrarily large number of states. The number of states will be $O(\log n \log\log n)$ in expectation and with high probability.

speed, it is only required that `logn` be within a constant multiplicative factor of $\log n$.

Gąsieniec and Stachowiak [26] show a uniform $O(\log \log n)$ state, $O(\log n)$ time protocol (both bounds in expectation and with high probability) that computes and stores in each agent a value $\ell \in \mathbb{N}^+$ that, with high probability, is within additive constant $O(1)$ of $\lceil \log \log n \rceil$ (in particular, WHP $\ell \geq \lfloor \log \log n \rfloor - 3$ [14, Lemma 8]), so $2^\ell = \Theta(\log n)$. (This is the so-called *junta election* protocol used as a subroutine for a subsequent leader election protocol.) Furthermore, agents approach this estimate from below, propagating the maximum estimate by epidemic $\ell', \ell \to \ell, \ell$ if $\ell' < \ell$. This gives an elegant way to compose the size estimation with a subsequent nonuniform protocol $\mathcal{P}$ that requires the estimate: agents store their estimate `logn` of $\log n$ and use it in $\mathcal{P}$. Whenever an agent's estimate `logn` updates—always getting larger—it simply resets $\mathcal{P}$, i.e., sets the entire state of $\mathcal{P}$ to its initial state. We can then reason as though all agents actually started with their final convergent value of `logn`.[12]

To make our protocol uniform, but remove its correctness in the case of a tie, as we explain below, all agents conduct this size estimation, stored in the field `logn`, in parallel with the majority protocol $\mathcal{P}$ of Section III. Each agent resets $\mathcal{P}$ to its initial state whenever `logn` updates. This gives the stated $O(\log n)$ time bound and $O(\log n \log \log n)$ state bound. Note that in Phase 0, agents count from $\texttt{counter} = \Theta(\log n)$ down to 0. It suffices to set the constant in the Θ sufficiently large that all agents with high probability receive by epidemic the convergent final value of `logn` significantly before any agent with the same convergent estimate counts to 0.

Acknowledging that, with small probability, the estimate of $\log n$ could be too low for Phase 4 to be correct, we simply remove Phase 4 and do not attempt to detect ties. So if we permit undefined behavior in the case of a tie (as many existing fast majority protocols do), then this modification of the algorithm otherwise retains stably correct, $O(\log n)$ time behavior, while increasing the state complexity to $O(\log n \log \log n)$.

B. How to stably compute ties

With low but positive probability, the estimate of $\log n$ could be too small. For most phases of the algorithm, this would merely amplify the probability of error events (e.g., Phase 1 doesn't last long enough for agents to converge on biases $\{-1, 0, +1\}$) that later phases are designed to handle. However, the correctness of Phase 4 (which detects ties) requires agents to have split through at least $\log n$ exponents in Phase 3. Since the population-wide bias doubles each time the whole population splits down one exponent, the only way for the whole population to split through $\log n$ exponents is for there to be a tie (i.e., the population-wide bias is 0, so can double unboundedly many times). In this one part of the algorithm, for correctness we require the estimate to be at least $\log n$ with probability 1. (It can be much greater than $\log n$ without affecting correctness; an overestimate merely slows down the algorithm.)

To correct this error, we will introduce a stable backup size estimate, to be done in Phase 4. Note that there are only a constant number of states with $\texttt{phase} = 4$: Clock agents do not store a counter in this phase, and Main agents that stay in this phase must have $\texttt{bias} \in \left\{0, \pm\frac{1}{2^L}\right\}$. Thus we can use an additional $\Theta(\log n)$ states for the agents that are currently in $\texttt{phase} = 4$ to stably estimate the population size. If they detect that their estimate of L was too small, they simply go to the stable backup Phase 10.

Stable computation of $\lfloor \log n \rfloor$.: The stable computation of $\log n$ has all agents start in state L_0, where the subscript represents the agent's estimate of $\lfloor \log n \rfloor$. We have the following transitions: for each $i \in \mathbb{N}$, $L_i, L_i \to L_{i+1}, F_{i+1}$ and, for each $0 \leq j < i$, $F_i, F_j \to F_i, F_i$. Among the agents in state L_i, half make it to state L_{i+1}, reaching a maximum of L_k at $k = \lfloor \log_2 n \rfloor$.[13] All remaining F agents receive the maximum value k by epidemic. A very similar protocol was analyzed in [24, Lemma 12]. A time analysis of this protocol is in the full version of this paper [35].

C. Challenges for $O(\log n)$ state uniform algorithm

It is worth discussing some ideas for adjusting the uniform protocol described above to attempt to reduce its space complexity to $O(\log n)$ states. The primary challenge is to enable the population size n to be estimated without storing the estimate in any agent that participates in the main algorithm (i.e., the agent has role Main, Clock, or Reserve). If agents in the main algorithm do not store the size, then by [23, Theorem 4.1] they will provably go haywire initially,

[12] One might hope for a stronger form of composition, in which the size estimation *terminates*, i.e., sets an initially False Boolean flag to True only if the size estimation has converged, in order to simply prohibit the downstream protocol $\mathcal{P}$ from starting with an incorrect estimate of $\log n$. However, when A and B are both initially $\Omega(n)$, this is impossible; $\Omega(n)$ agents will set the flag to True in $O(1)$ time, long before the size estimation converges [23, Theorem 4.1].

[13] More generally, the unique stable configuration encodes the full population size n in binary in the following distributed way: for each position i of a 1 in the binary expansion of n, there is one agent L_i. Thus, these remaining agents lack the space to participate in propagating the value $k = \lfloor \log n \rfloor$ by epidemic, but there are $\Omega(n)$ followers F to complete the epidemic quickly.

with agents in every phase, totally unsynchronized, and require the size estimating agents to reset them after having converged on a size estimate that is $\Omega(\log n)$.

The following method would let the Size agents reset main algorithm agents, without actually having to store an estimate of $\log n$ in the algorithm agents, but it only works with high probability. The size estimating agents could start a junta-driven clock as in [26], which is reset whenever they update their size estimate. Then, as long as there are $\Omega(n)$ Size agents, they could for a phase timed to last $\Theta(\log n)$ time, reset the algorithm agents by *direct* communication (instead of by epidemic). This could put the algorithm agents in a quiescent state where they do not interact with each other, but merely wait for the Size agents to exit the resetting phase, indicating that the algorithm agents are able to start interacting again. Since there are $\Omega(n)$ size-estimating agents, each non-size-estimating agent will encounter at least one of them with high probability in $O(\log n)$ time.

The problem is that the reset signal is not guaranteed to reach every algorithm agent. There is some small chance that a Main agent with a bias different from its `input` does not encounter a Size agent in the resetting phase, so is never reset. The algorithm from that point on could reach an incorrect result when the agent interacts with properly reset agents, since the sum of biases across the population has changed. In our algorithm, by "labeling" each reset with the value `logn`, we ensure that no matter what states the algorithm agents find themselves in during the initial chaos before size computation converges, every one of them is guaranteed to be reset one last time with the same value of `logn`. The high-probability resetting described above could potentially work to create a high probability uniform protocol using $O(\log n)$ time and states, though we have not thoroughly explored the possibility.

But it seems difficult to achieve probability-1 correctness using the technique of "reset the whole majority algorithm whenever the size estimate updates," without multiplying the state complexity by the number of possible values of `logn`. Since we did not need $\lfloor \log n \rfloor$ exactly, but only a value that is $\Theta(\log n)$, we paid only $\Theta(\log \log n)$ multiplicative overhead for the size estimate, but it's not straightforward to see how to avoid this using the resetting technique.

V. Conclusion

There are some remaining open problems concerning the majority problem for population protocols.

Uniform $O(\log n)$-time, $O(\log n)$-state majority protocol: Our main $O(\log n)$ state protocol, described in Section III, is *nonuniform*: all agents have the value $\lceil \log n \rceil$ encoded in their transition function. The uniform version of our protocol described in Section IV uses $O(\log n \log \log n)$ states. It remains open to find a uniform protocol that uses $O(\log n)$ time and states.

$O(\log n)$ time, $o(\log n)$ state non-stable protocol: Berenbrink, Elsässer, Friedetzky, Kaaser, Kling, and Radzik [14] showed a non-stable majority protocol (i.e., it has a positive probability of error) using $O(\log \log n)$ states and converging in $O(\log^2 n)$ time. (See Section I for more details.) Is there a protocol with $o(\log n)$ states solving majority in $O(\log n)$ time?

Unconditional $\Omega(\log n)$ state lower bound for stable majority protocols: The lower bound of $\Omega(\log n)$ states for (roughly) sublinear time majority protocols shown by Alistair, Aspnes, and Gelashvili [1] applies only to stable protocols satisfying two conditions: *monotonicity* and *output dominance*.

Recall that a *uniform* protocol is one where a single set of transitions works for all population sizes; nonuniform protocols typically violate this by having an estimate of the population size (e.g., the integer $\lceil \log n \rceil$) embedded in the transition function. Monotonicity is a much weaker form of uniformity satisfied by nearly all known nonuniform protocols. While allowing different transitions as the population size grows, monotonicity requires that the transitions used for population size n must also be correct for all smaller population sizes $n' < n$ (i.e., an *overestimate* of the size cannot hurt), and furthermore that the transitions be no slower on populations of size n' than on populations of size n. Our nonuniform protocol Nonuniform-Majority is monotone; for an explanation, see the conclusion of the full version of this paper [35].

Output dominance references the concept of a *stable* configuration $\mathbf{c}$, in which all agents have a consensus opinion that cannot change in any configuration subsequently reachable from $\mathbf{c}$. A protocol is *output dominant* if in any stable configuration $\mathbf{c}$, adding more agents with states already present in $\mathbf{c}$ maintains the property that every reachable stable configuration has the same output (though it may disrupt the stability of $\mathbf{c}$ itself). This condition holds for all known stable majority protocols, including that described in this paper: they obey the stronger condition that adding states already present in $\mathbf{c}$ does not even disrupt its stability. In such protocols, two agents with the same opinion cannot create the opposite opinion, so stabilization happens exactly when all agents first converge on a consensus output. For a more detailed explanation, see the conclusion of the full version of this paper [35].

Monotonicity can be seen as a natural condition that all "reasonable" non-uniform protocols must satisfy,

but output dominance arose as an artifact that was required for the lower bound proof strategy of [1]. It remains open to prove an unconditional (i.e., removing the condition of output dominance) lower bound of $\Omega(\log n)$ states for *any* stable monotone majority protocol taking polylogarithmic time, or to show a stable polylogarithmic time monotone majority protocol using $o(\log n)$ states, which necessarily violates output dominance. If the unconditional lower bound holds, then our protocol is simultaneously optimal for both time and states. Otherwise, it may be possible to use $o(\log n)$ states to stably compute majority in polylogarithmic stabilization time with a non-output-dominant protocol.

We close with questions unrelated to majority.

Fast population protocol for parity: The majority problem "$X_1 > X_2$?" is a special case of a *threshold* predicate, which asks whether a particular weighted sum of inputs $\sum_{i=1}^{k} w_i X_i > c$ exceeds a constant c. (For majority, $w_1 = 1, w_2 = -1, c = 0$; our protocol extends straightforwardly to other values of w_i, though not other values of c.) The threshold predicates, together with the *mod* predicates, characterize the *semilinear* predicates, which are precisely the predicates stably computable by $O(1)$ state protocols [40] with no time constraints (though $\Theta(n)$ time to stabilize is sufficient for all semilinear predicates [8] and necessary for "most" [12]). A representative mod predicate is *parity*: asking whether an odd or even number of agents exist. Like majority, parity is solvable by a simple protocol in $O(n)$ time [2], and it is known to require $\Omega(n)$ time for any $O(1)$ state protocol to stabilize [12]. Techniques from [11] can be used to show that stabilization requires close to linear time even allowing up to $\frac{1}{2}\log\log n$ states. An interesting open question is to consider allowing $\omega(1)$ states in deciding parity. Can it then be decided in polylogarithmic time?

Fast population protocols for function computation: The transition $X, Q \to Y, Y$, starting with sufficiently many excess agents in state Q, computes the function $f(x) = 2x$, because if we start with x agents in state X, in time $O(\log n)$, $2x$ agents adopt state Y [39]. The similar transition $X, X \to Y, Q$ computes $f(x) = \lfloor x/2 \rfloor$, but using time $\Theta(n)$, as does any $O(1)$-state protocol computing any linear function with a coefficient not in $\mathbb{N}$, as well as "most" non-linear functions such as $\min(x_1, x_2)$ (computable by $X_1, X_2 \to Y, Q$) and $\max(x_1, x_2)$ [12]. Can such functions be computed in sublinear time by using $\omega(1)$ states?

Acknowledgments. Doty, Eftekhari, and Severson were supported by NSF awards 1900931 and 1844976.

References

[1] D. Alistarh, J. Aspnes, and R. Gelashvili, "Space-optimal majority in population protocols," in *SODA 2018: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2018, pp. 2221–2239.

[2] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta, "Computation in networks of passively mobile finite-state sensors," *Distributed Computing*, vol. 18, no. 4, pp. 235–253, March 2006.

[3] V. Volterra, "Variazioni e fluttuazioni del numero d'individui in specie animali conviventi," *Memoria della Reale Accademia Nazionale dei Lincei*, vol. 2, pp. 31–113, 1926.

[4] J. M. Bower and H. Bolouri, *Computational modeling of genetic and biochemical networks*. MIT press, 2004.

[5] D. Soloveichik, M. Cook, E. Winfree, and J. Bruck, "Computation with finite stochastic chemical reaction networks," *Natural Computing*, vol. 7, no. 4, pp. 615–633, 2008.

[6] Y.-J. Chen, N. Dalchau, N. Srinivas, A. Phillips, L. Cardelli, D. Soloveichik, and G. Seelig, "Programmable chemical controllers made from DNA," *Nature Nanotechnology*, vol. 8, no. 10, pp. 755–762, 2013.

[7] N. Srinivas, J. Parkin, G. Seelig, E. Winfree, and D. Soloveichik, "Enzyme-free nucleic acid dynamical systems," *Science*, vol. 358, no. 6369, p. eaal2052, 2017.

[8] D. Angluin, J. Aspnes, and D. Eisenstat, "Fast computation by population protocols with a leader," *Distributed Computing*, vol. 21, no. 3, pp. 183–199, September 2008.

[9] D. Doty and M. Hajiaghayi, "Leaderless deterministic chemical reaction networks," *Natural Computing*, vol. 14, no. 2, pp. 213–223, 2015.

[10] D. Doty and D. Soloveichik, "Stable leader election in population protocols requires linear time," *Distributed Computing*, vol. 31, no. 4, pp. 257–271, 2018.

[11] D. Alistarh, J. Aspnes, D. Eisenstat, R. Gelashvili, and R. L. Rivest, "Time-space trade-offs in population protocols," in *SODA 2017: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2017, pp. 2560–2579.

[12] A. Belleville, D. Doty, and D. Soloveichik, "Hardness of computing and approximating predicates and functions with leaderless population protocols," in *ICALP 2017: 44th International Colloquium on Automata, Languages, and Programming*, vol. 80, 2017, pp. 141:1–141:14.

[13] A. Kosowski and P. Uznański, "Population protocols are fast," *CoRR*, vol. abs/1802.06872, 2018. [Online]. Available: http://arxiv.org/abs/1802.06872

[14] P. Berenbrink, R. Elsässer, T. Friedetzky, D. Kaaser, P. Kling, and T. Radzik, "Time-space trade-offs in population protocols for the majority problem," *Distributed Computing*, Aug 2020. [Online]. Available: http://dx.doi.org/10.1007/s00446-020-00385-0

[15] R. Elsässer and T. Radzik, "Recent results in population protocols for exact majority and leader election," *Bulletin of EATCS*, vol. 3, no. 126, 2018.

[16] D. Doty, M. Eftekhari, O. Michail, P. G. Spirakis, and M. Theofilatos, "Brief announcement: Exact size counting in uniform population protocols in nearly logarithmic time," in *DISC 2018: 32nd International Symposium on Distributed Computing*, 2018.

[17] J. Burman, H.-L. Chen, H.-P. Chen, D. Doty, T. Nowak, E. Severson, and C. Xu, "Time-optimal self-stabilizing leader election in population protocols," in *PODC 2021: Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, 2021.

[18] D. Alistarh, R. Gelashvili, and M. Vojnović, "Fast and exact majority in population protocols," in *PODC 2015: Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*. ACM, 2015, pp. 47–56.

[19] P. Berenbrink, D. Kaaser, P. Kling, and L. Otterbach, "Simple and Efficient Leader Election," in *1st Symposium on Simplicity in Algorithms (SOSA 2018)*, vol. 61, 2018, pp. 9:1–9:11.

[20] A. Bilke, C. Cooper, R. Elsässer, and T. Radzik, "Brief announcement: Population protocols for leader election and exact majority with $O(\log^2 n)$ states and $O(\log^2 n)$ convergence time," in *PODC 2017: Proceedings of the ACM Symposium on Principles of Distributed Computing*. ACM, 2017, pp. 451–453.

[21] P. Berenbrink, R. Elsässer, T. Friedetzky, D. Kaaser, P. Kling, and T. Radzik, "A population protocol for exact majority with $O(\log^{5/3} n)$ stabilization time and $\Theta(\log n)$ states," in *DISC 2018: Proceedings of the 32nd International Symposium on Distributed Computing*, vol. 10, 2018, pp. 1—18.

[22] S. Ben-Nun, T. Kopelowitz, M. Kraus, and E. Porat, "An $O(\log^{3/2} n)$ parallel time population protocol for majority with $O(\log n)$ states," in *PODC 2020: Proceedings of the 39th Symposium on Principles of Distributed Computing*, 2020, pp. 191–199.

[23] D. Doty and M. Eftekhari, "Efficient size estimation and impossibility of termination in uniform dense population protocols," in *PODC 2019: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 34–42.

[24] P. Berenbrink, D. Kaaser, and T. Radzik, "On counting the population size," in *PODC 2019: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, p. 43–52.

[25] D. Alistarh and R. Gelashvili, "Polylogarithmic-time leader election in population protocols," in *42nd International Colloquium on Automata, Languages, and Programming*, vol. 9135, 2015, pp. 479 – 491.

[26] L. Gąsieniec and G. Stachowiak, "Fast space optimal leader election in population protocols," in *SODA 2018: ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2018, pp. 2653–2667.

[27] L. Gąsieniec, G. Stachowiak, and P. Uznański, "Almost logarithmic-time space optimal leader election in population protocols," in *31st ACM Symposium on Parallelism in Algorithms and Architectures*, 2019, pp. 93–102.

[28] P. Berenbrink, G. Giakkoupis, and P. Kling, "Optimal time and space leader election in population protocols," in *STOC 2020: Proceedings of the 52nd ACM SIGACT Symposium on Theory of Computing*, 2020, p. 119–129.

[29] M. Draief and M. Vojnović, "Convergence speed of binary interval consensus," *SIAM Journal on control and Optimization*, vol. 50, no. 3, pp. 1087–1109, 2012.

[30] G. B. Mertzios, S. E. Nikoletseas, C. L. Raptopoulos, and P. G. Spirakis, "Determining majority in networks with local interactions and very small local memory," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2014, pp. 871–882.

[31] Y. Mocquard, E. Anceaume, J. Aspnes, Y. Busnel, and B. Sericola, "Counting with population protocols," in *14th IEEE International Symposium on Network Computing and Applications*, 2015, pp. 35–42.

[32] Y. Mocquard, E. Anceaume, and B. Sericola, "Optimal proportion computation with population protocols," in *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, 2016, pp. 216–223.

[33] D. Angluin, J. Aspnes, and D. Eisenstat, "A simple population protocol for fast robust approximate majority," *Distributed Computing*, vol. 21, no. 2, pp. 87–102, 2008.

[34] A. Condon, M. Hajiaghayi, D. Kirkpatrick, and J. Maňuch, "Approximate majority analyses using tri-molecular chemical reaction networks," *Natural Computing*, vol. 19, no. 1, pp. 249–270, 2020.

[35] D. Doty, M. Eftekhari, L. Gąsieniec, E. Severson, G. Stachowiak, and P. Uznański, "A time and space optimal stable population protocol solving exact majority," arXiv, Tech. Rep. 2106.10201, 2021. [Online]. Available: https://arxiv.org/abs/2106.10201

[36] L. Gąsieniec, D. Hamilton, R. Martin, P. G. Spirakis, and G. Stachowiak, "Deterministic population protocols for exact majority and plurality," in *20th International Conference on Principles of Distributed Systems*, 2016.

[37] M. Blondin, J. Esparza, S. Jaax, and A. Kučera, "Black ninjas in the dark: Formal analysis of population protocols," in *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, ser. LICS '18, 2018, p. 1–10.

[38] D. Alistarh and R. Gelashvili, "Recent algorithmic advances in population protocols," *ACM SIGACT News*, vol. 49, no. 3, pp. 63–73, 2018.

[39] H.-L. Chen, D. Doty, and D. Soloveichik, "Deterministic function computation with chemical reaction networks," *Natural Computing*, vol. 13, no. 4, pp. 517–534, 2013.

[40] D. Angluin, J. Aspnes, and D. Eisenstat, "Stably computable predicates are semilinear," in *PODC 2006: 25th annual ACM Symposium on Principles of Distributed Computing*, 2006, pp. 292–299.

[41] https://github.com/UC-Davis-molecular-computing/ppsim/blob/main/examples/majority.ipynb.

[42] Y. Mocquard, B. Sericola, F. Robin, and E. Anceaume, "Stochastic analysis of average based distributed algorithms," 2020.

[43] P. Berenbrink, A. Czumaj, A. Steger, and B. Vöcking, "Balanced allocations: The heavily loaded case," *SIAM Journal on Computing*, vol. 35, no. 6, pp. 1350–1385, 2006.

[44] B. Dudek and A. Kosowski, "Universal protocols for information dissemination using emergent signals," in *STOC 2018: Proceedings of the 50th ACM SIGACT Symposium on Theory of Computing*, 2018, pp. 87–99.

[45] I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P. G. Spirakis, "Passively mobile communicating machines that use restricted space," *Theoretical Computer Science*, vol. 412, no. 46, pp. 6469–6483, October 2011.