Efficient Size Estimation and Impossibility of Termination in Uniform Dense Population Protocols

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

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

ABSTRACT

We study *uniform* population protocols: networks of anonymous agents whose pairwise interactions are chosen at random, where each agent uses an *identical* transition algorithm that does not depend on the population size n. Many existing polylog(n) time protocols for leader election and majority computation are nonuniform: to operate correctly, they require all agents to be initialized with an approximate estimate of n (specifically, the value $\lfloor \log n \rfloor$).

Our first main result is a uniform protocol for calculating $\log(n) \pm O(1)$ with high probability in $O(\log^2 n)$ time and $O(\log^4 n)$ states ($O(\log \log n)$ bits of memory). The protocol is not *terminating*: it does not signal when the estimate is close to the true value of $\log n$. If it could be made terminating with high probability, this would allow composition with protocols requiring a size estimate initially. We do show how our main protocol can be indirectly composed with others in a simple and elegant way, based on *leaderless phase clocks*, demonstrating that those protocols can in fact be made uniform.

However, our second main result implies that the protocol *cannot* be made terminating, a consequence of a much stronger result: a uniform protocol for *any* task requiring more than constant time cannot be terminating even with probability bounded above 0, if infinitely many initial configurations are *dense*: any state present initially occupies $\Omega(n)$ agents. (In particular no leader is allowed.) Crucially, the result holds no matter the memory or time permitted.

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**.

KEYWORDS

population protocols; size estimation; termination

ACM Reference Format:
David Doty and Mahsa Eftekhari. 2019. Efficient Size Estimation and Impossibility of Termination in Uniform Dense Population Protocols. In *2019 ACM Symposium on Principles of Distributed Computing (PODC '19), July 29-August 2, 2019, Toronto, ON, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3293611.3331627

*Both authors contributed equally to this research. Authors supported by NSF grants 1619343 and 1844976.

PODC '19, July 29-August 2, 2019, Toronto, ON, Canada

ACM ISBN 978-1-4503-6217-7/19/07…$15.00
https://doi.org/10.1145/3293611.3331627

1 INTRODUCTION

Population protocols [7] are 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 [39], gene regulatory networks [18], and chemical reactions [36], the latter increasingly regarded as an implementable "programming language" for molecular engineering, due to recent experimental breakthroughs in DNA nanotechnology [21, 37].

All problems computable with zero error probability by a constant-state population protocol are computable in $O(n)$ time [9, 26]; the benchmark for "efficient" computation is thus sublinear time, ideally polylog(n). For example, the transition $x, q \to y, y$ (starting with at least as many q as the "input" state x) computes $f(x) = 2x$ in expected time $O(\log n)$, whereas $x, x \to y, q$ computes $f(x) = \lfloor x/2 \rfloor$ exponentially slower: expected time $O(n)$ [20].

Although the original model [7] assumed a set of states and transitions that is constant with respect to n, for important distributed computing problems such as leader election [27], majority computation [2], and computation of other functions and predicates [14] no constant-state protocol can stabilize in sublinear time with probability 1.[1] This motivated the study of protocols in which the set of states and transitions grows with n (essentially adding a nonconstant *memory* to each agent). Such protocols achieve leader election and majority computation using $O(\text{polylog}(n))$ time, keeping the number of states "small": typically $O(\text{polylog}(n))$ [2–4, 15, 17], although $O(\log \log n)$ states suffice for leader election [29].

Unfortunately, many of these sublinear-time protocols [2–4, 15, 17] are *nonuniform*: the set of states and transitions are allowed to depend arbitrarily on n (this is not true of all, see for example recent fast, low-memory leader election protocols [29, 30]). This capability is used to initialize each agent with an approximate estimate of n (the value $\lfloor \log n \rfloor$) required by the protocols.

More desirable would be a *uniform* protocol in which each agent's local algorithm for computing the outputs, given the inputs, has no knowledge of n. Such an algorithm may produce outputs longer

[1] A protocol *stabilizes* when it becomes unable to change the output. A protocol *converges* in a given execution when the output stops changing, though it could take longer to subsequently stabilize. Known time lower bounds [2, 14, 27] are on stabilization, not convergence. Recently Kosowski and Uznanski [31] achieved a breakthrough result, showing $O(1)$-state protocols for leader election and all decision problems computable by population protocols (the *semilinear* predicates), converging with high probability in polylog(n) time, and for any $\epsilon > 0$, probability 1 protocols for the same problems converging in $O(n^\epsilon)$ expected time. The latter protocols require $\Omega(n)$ time to stabilize, as would any constant-state protocol due to the cited time lower bounds.

than its inputs, retaining the ability to use a number of states that grows with the population size. A uniform protocol can be deployed into *any* population without knowing in advance the size, or even a rough estimate thereof.

1.1 Contributions

Nonuniform protocols in the literature [2–4, 15, 17] initialize each agent with the value $\lfloor \log n \rfloor$. Hence we study the problem of computing an approximate estimate of $\log n$.

Our first main result, Theorem 3.1, is a uniform protocol, starting from a configuration where all n agents are in an identical state, that with high probability computes $\log n \pm O(1)$ (storing the value in every agent), using $O(\log^2 n)$ time and $O(\log^4 n)$ states. This answers affirmatively open question 5 of [25]. This is done primarily by generating a sequence of geometric random variables,[2] and propagating the maximum to each agent. However, before the maximum reaches all agents they begin computation; thus we use a restart scheme similar to [29] to reset an agent's computation when it updates to a higher estimate of the max.

One might hope to use this protocol as a subroutine to "uniformize" existing nonuniform protocols for leader election and majority [2–4, 15, 17].[3] Suppose the size-estimating protocol could be made terminating, eventually producing a termination "signal" that with high probability does not appear until the size estimate has converged. This would allow composition with other protocols requiring the size estimate. It has been known since the beginning of the population protocol model [7] that termination cannot be guaranteed with probability 1. However, leader-driven protocols can be made terminating with high probability, including simulation of register machines [9] or exact population size counting [32].

Our second main result, Theorem 4.1, shows that this is impossible to do with our leaderless size-estimation protocol and a very wide range of others. This answers negatively open questions 1-3 of [25]. The production of such a terminating signal cannot be delayed, even with probability bounded above 0, by more than $O(1)$ time in any uniform protocol where, for some $\alpha > 0$, infinitely many valid initial configurations are *α-dense*, meaning that each state present is the state of at least αn agents. This holds even for randomized protocols with a nondeterministic transition function. (Because this is an impossibility result, the fact that it holds for both deterministic and randomized protocols makes it stronger than if it held only for deterministic protocols.) Since virtually all nontrivial computation with population protocols requires $\Omega(\log n)$ time[4] (including leader election, and computation of predicates and functions such as majority and $g(x) = 2x$), this implies that no uniform terminating protocol can solve these problems from dense initial configurations.

The hypothesis of density is crucial: with a *leader*, high-probability termination is possible in a uniform protocol [9]. The hypothesis of uniformity is also crucial: if each agent can *initially* store a value $f(n)$, then a termination signal can be delayed until some agent experiences $f(n)$ interactions, an event whose expected time grows unboundedly with n if f grows sufficiently fast. This result uses a density argument similar to that used previously to show time lower bounds, which assume a state set of size $O(1)$ [14, 23, 27] or $\leq \frac{1}{2} \log \log n$ [2]. In contrast, our argument holds for *any* state set size, by showing that a particular subset of states is produced in constant time w.h.p., and using a careful argument to show that this subset necessarily contains the termination signal.

Despite this difficulty in directly composing size estimation with a downstream protocol (or several stages/subprotocols composed in series), we present a general and simple method of composition (via restarting), based on a "leaderless phase clock" using a weaker log population size estimate s (called `logSize2` in the pseudocode in Section 3.2) obtained initially (where $\log n - \log \ln n \leq s \leq 2 \log n$ w.h.p.).[5] Based on s and the expected convergence time of the downstream protocol, each agent once per interaction increments a counter c, from 0 up to $f(s)$, and the first agent to reach $f(s)$ signals the entire population to terminate (or move to the next stage). $f(s)$ is chosen large enough that no agent reaches $f(s)$ before the downstream protocol converges. The entire downstream protocol is reset if the initial size estimate s changes. With the above scheme, agents need to store the variables s, c, and possibly also $f(s)$ (in our case $f(s) = O(s)$ so it need not be stored explicitly, but if $f(s) = \text{poly}(s)$, for example, $f(s)$ may need to be stored separately from s). If the downstream protocol requires $t(n)$ time to converge, then agents also set their threshold $f(s) > t(n)$ (where $f(s)$ is "large" compared to $t(n)$). This requires $O(f(s)^2 \cdot \log n)$ states will be added to the state complexity of the protocol, or $O(\log^2 n)$ if $f(s) = O(\log n)$ (as in our case) since $f(s)$ need not be stored explicitly. To compose multiple downstream stages/subprotocols in series, we also need a way to compute and possibly store the number K of stages (in our case $K = \Theta(\log n)$, also chosen as a constant times s, so K need not be stored explicitly), and we need to store an index indicating which stage we are on. For K stages, this multiplies the state complexity by K if $K = O(\log n)$ and K^2 otherwise (since K must be stored explicitly in the latter case).

1.2 Related work

The work of this paper was inspired by recent work on nonuniform polylog time leader election/majority [2–4, 6, 15, 17, 38]; the fact that those protocols require an approximate size estimate is the direct motivation for seeking a protocol that can compute such an estimate (though unfortunately due to Theorem 4.1, composition of our protocol with these is not totally straightforward).

Some nonuniform protocols crucially rely on an estimate of $\log n$ (e.g. [2–4, 15, 17, 38]) for *correctness*. Other nonuniform protocols are more robust, using the estimate merely to allow the protocol to have a finite number of states. For example, Alistarh and Gelashvili [4] show a $O(\log^3 n)$-time protocol for leader election in which leaders increment a counter on each interaction. The uniform variant of that protocol, with no estimate of $\log n$, is correct with

[2] To our knowledge, this constitutes the first analysis of sums of independent random variables, each of which is a maximum of geometric random variables. Standard Chernoff and other tail bounds generally used for bounded random variables fail in this case. We apply the theory of sub-exponential random variables [35] to obtain strong bounds on the moment-generating function of a maximum of geometric random variables in order to obtain the required Chernoff bounds.

[3] Some protocols for leader election [29, 30] are uniform, but other protocols [2, 4, 15, 17] have the benefit of simplicity and may possibly be easier to reason about and compose with other protocols.

[4] $\Omega(\log n)$ is a lower bound on most interesting computation: by a coupon collector argument, this is the expected time for each agent to have at least one interaction.

[5] The first leaderless phase clock for population protocols was proposed in [3]. Ours is different, based on [38]. Both are nonuniform, relying on an estimate of $\log n$.

probability 1, and the estimate of $\log n$ is used only to bound the counter (hence also the number of states) below $\log n$. Nevertheless, it is not obvious how to modify that protocol to be uniform and have a bounded number of states with high probability.

Self-stabilizing leader election and exact size counting. Cai, Izumi, and Wada [19] (using different terminology) show an impossibility result for uniform population protocols, that no protocol electing a leader can be uniform if it is also required to be *self-stabilizing*: correct with probability 1 from *any* initial configuration. In fact, it must be nonuniform in a very strong way: the *exact* population size must be encoded into each agent. Self-stabilizing exact size computing has also been shown to be possible with a leader [12] in $O(n \log n)$ time and $O(n)$ states for the leader and 2 states for the other agents, all asymptotically optimal parameters in the self-stabilizing setting [10].

Exact size counting. In the less restrictive setting where all agents start from a pre-determined state, Michail [32] proposed a uniform *terminating* protocol (where agents "know" when they have converged) in which a pre-elected leader computes the exact population size n in $O(n \log n)$ time with high probability. Going from the terminating to the less restrictive *converging* criterion (where agents eventually converge on the correct size, but do not know when this occurs), exact size counting is possible in $O(\log n \log\log n)$ time and $O(n^{60})$ states [25], *without* an initial leader.

Approximate size estimation. Alistarh, Aspnes, Eisenstat, Gelashvili, and Rivest [2] show a uniform protocol that in $O(\log n)$ expected time and states converges to an approximation n' of the population size n, computing an integer k such that with high probability $\frac{1}{2}\log n \le k \le 9 \log n$, i.e., $\sqrt{n} \le 2^k \le n^9$. Each agent generates (an approximation of) a geometric random variable, letting k be their maximum. We use their protocol as the first step of ours. The analysis of [2] is based on synthetic coins with a deterministic transition function, which have bias complicating the analysis. Our randomized model assumes access to perfectly random bits, so a simpler analysis (Corollary 3.7) shows that $\log n - \log\ln n \le k \le 2 \log n$ w.h.p. The remainder of our protocol improves this from a constant multiplicative error in approximating $\log n$ to a constant *additive* error. In other words we estimate the population size to within a constant multiplicative factor (instead of a polynomial factor as in [2]), but use $O(\log^2 n)$ time and $O(\log^4 n)$ states.

Berenbrink, Kaaser, and Radzik [16] independently studied the same size estimation problem as ours, obtaining stronger bounds on additive error and number of states: computing the value $\lfloor \log n \rfloor$ or $\lceil \log n \rceil$ (i.e., additive error < 1) with high probability, using $O(\log^2 n)$ time and $O(\log n \log\log n)$ states. They also show a protocol with probability 1 of correctness, using $O(\log^2 n)$ time and $O(\log^2 n \log\log n)$ states.

2 MODEL

To formally define *uniform* computation in population protocols, the agents' transition algorithm is modeled as a 2-tape deterministic Turing machine (TM) with the read only "input tape" as tape 1 (for reading the other agent's state) and read-write "working tape" as tape 2 (for storing this agent's state).[6]

Our protocol describes a constant number of integer fields comprising each agent's state, which could all be stored in the working tape and separated by a special symbol. An agent's working tape is identical to what it was on the conclusion of the previous interaction. When two agents interact, each copies the content of the other's tape 2 its own tape 1, and then each of their TM states is reset from a halting TM state to the start TM state. The space usage (in bits) s is defined as normal for TMs: the maximum number of tape cells that are written during the computation on the read/write working tape. The number of possible agent states (working tape contents) is then c^s, where s is the maximum space usage of any agent during an execution of the protocol and c is the size of the tape alphabet. For ease of understanding, we will use standard population protocol terminology and not refer explicitly to details of the TM definition except where needed. A *state* $s \in \Lambda$ always refers to the TM working tape content of an agent (leaving out TM state and tape head positions since these are identical in all initial configurations), where Λ is the set of all agent states. A *configuration* $\vec{c} \in \mathbb{N}^\Lambda$ is a vector indexed by a state, where $\vec{c}(s)$ is the *count* of state s in the population. We set the output of our protocol the value stored in a special field labeled "output". Some definitions allow the output to be a function of the fields stored in an agent's memory, without the output itself counting against the memory usage. Our protocol reuses a field for the output that is used prior in the protocol, so our memory usage is the same under either definition.

We furthermore assume that each agent has access to independent uniformly random bits, assumed to be pre-written on a special read-only tape (this allows the TM to be deterministic even though it is computing a nondeterministic relation). This is different from the traditional definition of population protocols, which assumes a deterministic transition function. In our case, we have a transition *relation* $\delta \subseteq \Lambda^4$. Several papers [2, 15] indicate how to use the randomness built into the interaction scheduler to provide nearly uniform random bits to the agents, using various *synthetic coin* techniques, showing that the deterministic model can effectively simulate the randomized model. In the interest of brevity and simplicity of presentation, we will simply assume in the model that each agent has access to a source of uniformly random bits. A variant of our protocol using the sender/receiver choice to simulate uniformly random bits with a deterministic transition function, with the same time, state, and error bounds, is available in [24].

Throughout this paper, n denotes the number of agents in the population. Repeatedly, a pair of agents is selected uniformly at random to interact, where they run the transition algorithm on the pair of states they were in prior to the interaction, and storing the output states for their next interactions. The *time* until some event is measured as the number of interactions until the event occurs, divided by n, also known as parallel time. This represents a natural model of time complexity in which we expect each agent to have $O(1)$ interactions per unit of time, hence across the whole population, $\Theta(n)$ total interactions occur per unit time. All references to "time" in this paper refer to parallel time. An *execution* is a sequence of configurations $\vec{c}_0, \vec{c}_1, \ldots$ such that for all i, applying a transition

[6] Our model generalizes the original constant-state model [8] by allowing the memory potentially to grow with n; however, constant-state protocols can be implemented with our model. It is worth distinguishing four ways for memory to increase with n: 1) not at all (constant-state), 2) increasing with n but, for each n, bounded by a constant depending on n (most non-uniform protocols), and 3) possibly unbounded but bounded with probability 1 (this paper), 4) unbounded with positive probability.

to $\vec{c}_i$ results in $\vec{c}_{i+1}$. $\log n$ is the base-2 logarithm of n, and $\ln n$ is the base-e logarithm of n.

2.1 Definition of correctness and time

The notion that a protocol's configuration "has the correct answer" is problem-specific. For leader election, it means there is a single leader agent. For predicate computation, it means all agents have the correct Boolean output. In this paper, since our goal is to approximate $\log n$ within additive factor 5.7, we say a configuration is *correct* if the `output` field of each agent is within 5.7 of $\log n$.[7]

The following definitions match those used in the literature, when other notions of "correct" are substituted. Let $\mathcal{E} = (\vec{c}_0, \vec{c}_1, \ldots)$ be an infinite execution. A configuration $\vec{c}$ is *stably correct* if every configuration reachable from $\vec{c}$ is correct.[8] We say $\mathcal{E}$ *converges* at interaction i if $\vec{c}_i$ is not correct and for all $j > i$, $\vec{c}_j$ is correct. We say $\mathcal{E}$ *stabilizes* at interaction i if $\vec{c}_i$ is not stably correct and for all $j > i$, $\vec{c}_j$ is stably correct. A protocol can converge and/or stabilize with probability 1 or a smaller probability. However, if the set of reachable configurations is bounded with probability 1 (which is the case for the protocols discussed in this paper), then for any $p \in [0, 1]$, a protocol converges with probability p if and only if it stabilizes with probability p.[9] For a computational task T equipped with some definition of "correct", we say that a protocol $\mathcal{P}$ *stably computes T with probability p* if, with probability p, it stabilizes (equivalently, converges). If p is omitted, it is assumed $p = 1$. However, when measuring time complexity, convergence and stabilization may be much different. We say that $\mathcal{P}$ converges (respectively, stabilizes) in (parallel) time $t(n)$ with probability p if, with probability p, it produces an execution that converges (resp., stabilizes) by interaction i, where $i/n \leq t(n)$. Many protocols converge much faster than they stabilize, such as those that combine a fast, error-prone subprotocol with a slow, error-free protocol, e.g., [9, 20, 31]. However, for the protocol of this paper, convergence and stabilization coincide. We use the term "converge" throughout the paper to refer to this event.

Many papers separately measure high-probability time convergence and expected time to converge. Our protocol has positive probability of error, but we argue that expected time is a meaningful notion only with error probability 0, which is why we do not measure expected time. The only reasonable definition "time until correctness" on a non-converging execution is ∞. So with positive error probability, the expected convergence time is $\mathrm{E}[\text{time}|\text{converges}] + \mathrm{E}[\text{time}|\text{doesn't converge}] = \mathrm{E}[\text{time}|\text{converges}] + \infty = \infty$. One could imagine measuring only $\mathrm{E}[\text{time}|\text{converges}]$. However, conditioning can artificially "speed up" the process.[10]

3 FAST PROTOCOL FOR ESTIMATING $\log n$ WITHIN $O(1)$ ADDITIVE ERROR

In this section we describe a uniform protocol for computing the value of $\log n$ with an additive error, i.e., estimating the population size to within a constant multiplicative factor. We say a population protocol is *leaderless* if all agents start in the same state.

Theorem 3.1. *There is a uniform leaderless population protocol that converges in time $O(\log^2 n)$ with probability $\geq 1 - 1/n^2$, uses $O(\log^4 n)$ states with probability $\geq 1 - O(\log n)/n$, and stores in each agent an integer k such that $|k - \log n| \leq 5.7$ with probability $\geq 1 - 9/n$.*

We note that the protocol has a positive probability of error. It is open to find a protocol using polylog(n) time/states computing $\log(n) \pm O(1)$ with probability 1.

The protocol is described and its time and state complexity analyzed in Subsection 3.2. Much of the analysis of the approximation involves proving a bound on the moment-generating function of a maximum of geometric random variables, enabling the Chernoff technique can be applied to sums of such variables. This is quite nontrivial; see appendix in [24] .

3.1 Intuition

Alistarh et al. [2] describe a protocol for estimating $\log n$ within a constant multiplicative factor. A $\frac{1}{2}$-geometric random variable is the number of flips needed to get one head when flipping a fair coin. In their protocol, each agent generates an independent geometric random variable $\mathbf{G}_i$, then propagates the maximum $\mathbf{M} = \max_{1 \leq i \leq n} \mathbf{G}_i$ by *epidemic*: transitions of the form $i, j \to j, j$ for $i \leq j$, which in $O(\log n)$ time "infect" all agents with the maximum. It is known that $\mathrm{E}[\mathbf{M}] \approx \log n$ [28], and $\log n - \log\ln n \leq \mathbf{M} \leq 2 \log n$ with probability $\geq 1 - O(1)/n$ (Lemma C.7 [24]).

We take the obvious extension of this approach: do this K times and take an average. The estimated average is within $O(1)$ of $\log n$ so long as $K = \Omega(\log n)$ [24]. One problem to solve first is how to calculate K; after all, $K = \Theta(\log n)$ scales with n, so with a uniform protocol it cannot be encoded into the agents at the start. The agents estimate it using the protocol of [2]. Since that protocol is converging but not terminating (provably it cannot be made terminating by Theorem 4.1), each time an agent updates its value of K, it reinitializes the remainder of its state.

However, a trickier problem remains: a naïve approach to implement "averaging of K numbers" requires storing $K = \Theta(\log n)$ numbers in each agent, each having value $\Theta(\log n)$, implying the number of states is $\Theta((\log n)^{\log n}) = \Theta(n^{\log\log n})$. This is even more

[7] We note that our notion of function approximation differs from that of Belleville, Doty, and Soloveichik [14]. They use a *distributed* output convention, where the output of a function $f : \mathbb{N}^d \to \mathbb{N}$ is encoded as the population count of agents in a special output state y. Thus one must examine the entire population to know the output. In our *local* output convention, each agent has a field encoding a value from the function's range. The output is undefined if some agents have different values, and defined to be their common value otherwise. This is similar to how Boolean predicate output with range $\{0, 1\}$ is encoded in population protocols [7].

[8] Belleville, Doty, and Soloveichik [14] also consider function approximation, but define a configuration to be stable if the output *cannot* change, whereas we allow it to change within a small interval around the correct value. The time lower bound techniques of [14] do not apply to our more relaxed notion of stability.

[9] Let C and S respectively be the set of stabilizing and converging executions. Clearly $S \subseteq C$. Although $S \subsetneq C$ is possible, we argue that $\Pr[C \setminus S] = 0$. Suppose a protocol converges in an execution $(\vec{c}_0, \vec{c}_1, \ldots)$ at interaction i (so $\vec{c}_j$ is correct for all $j > i$). If did not stabilize, then for all $j > i$, some incorrect configuration $\vec{d}_j$ would be reachable from $\vec{c}_j$. Let $p_j > 0$ denote the probability of reaching $\vec{d}_j$ from $\vec{c}_j$. The set of reachable configurations is bounded with probability 1, so $\min_{j>i} p_j$ is well-defined and positive. The probability of never reaching any $\vec{d}_j$ is then 0.

[10] Consider a hypothetical protocol that runs a parallel subprotocol S that completes quickly and, upon completion, somehow prevents the main protocol M from converging. The main protocol, on the other hand, may somehow detect if it completes before S does, and if so, M then shuts S down. Many executions will not converge, but those that do must be very fast in order to converge before S completes. Thus conditioning on convergence "anthropically speeds up" convergence [1]. This is an extreme example that has the property that the probability of correctness is reduced by S, but it nevertheless shows that measuring conditional expected time can be problematic.

than the $O(n^{60})$ sufficient to quickly compute *exactly* n [25]. To overcome this problem, we use a "leaderless phase clock" similar to those of [3, 34, 38], but uniform. Unlike the phase clock used by [3, 34], our leaderless phase clock simply increments a counter on every interaction. This simultaneously gives an elegant way to compose our protocols with downstream protocols requiring the size estimate. Agents count their number of interactions and compare it with a threshold value $\Theta(\log n)$. Whenever their number of interaction passes the threshold they will move to the next round similar to the protocol described above (the population with a leader). The threshold is calculated in the following way. In our protocol, agents start generate a geometric random variable called `logSize2` and propagate the maximum `logSize2` among themselves. After agents agree on the `logSize2` variable, a constant multiple 95.`logSize2` is the threshold in their leaderless phase clock. This lets the agents synchronize epochs of the algorithm, each taking $O(\log n)$ time, and prevent the next epoch from starting until the previous has concluded.

The probabilistic clock inside agents might go off very soon at the very beginning of the protocol, but after $O(\log n)$ time all agents will store the maximum generated `logSize2` and their leaderless phase clock will eventually converge to a stable one which goes off after completion of a predefined constant factor of $\log n$; to handle this, each time an agent updates its value of `logSize2`, the remainder of its state is reset and it begins the rest of the protocol anew. Restarting the downstream protocol is a known technique in population protocols also used in [29] to compose two leader elimination subprotocols. The agents then generate K additional geometric random variables in sequence, taking their sum. Upon completing the generation and propagation of the K'th number, the agent divides the sum by K and stores the result in their output field. Composition with a downstream protocol is as simple as letting that protocol be the last phase. However, since our protocol has a positive probability of failure, this would translate to the downstream protocol as well.

The time is $O(\log^2 n)$ by the following rough analysis (details follow). We propagate K numbers one after each other and for each epidemic $O(\log n)$ time is required. Since we set $k = O(\log n)$ then the protocol will take $O(\log^2 n)$ total time to complete.

3.2 Formal specification of protocol

Our protocol uses uniform random bits in multiple places. We assume agents have access to independent uniformly random bits. In the protocol, agents start by dividing in two groups of `S` and `A`. `A` agents are responsible for the most part of the algorithm including generating geometric random variables and propagating their maximums while the `S` agents only provide memory to store the sum of K maximum geometric random variables. We split the state space such that `A` agents and `S` agents are responsible to store different variables. The space multiplexing is a common approach used in population protocols to reduce the space complexity of the protocols [5].

Agents initially have no role (`X`), and partition into roles via $\mathtt{X}, \mathtt{X} \to \mathtt{A}, \mathtt{S}$. Since this takes $\Theta(n)$ time to complete, we add transitions $\mathtt{A}, \mathtt{X} \to \mathtt{A}, \mathtt{S}$ and $\mathtt{S}, \mathtt{X} \to \mathtt{S}, \mathtt{A}$, converging in $O(\log n)$ time, with the price of deviating from $\frac{n}{2}$ for each role. It is proven in [24] this deviation is $O(\sqrt{n \ln n})$, increasing the size estimation error by merely a constant additive factor.[11] All agents start at `epoch` $= 0$. The `A` agents generate one geometric random variable (called `logSize2`) and continue by propagating the maximum among the whole population. Since we use this `logSize2` value for all early estimation of $\log n$, each time an agent finds out there was a greater value for the `logSize2` than its own, it will reset all other computations that might have happened.

The maximum `logSize2` amongst the population is a 2 factor estimation of $\log n$. When any agent updates its `logSize2` with a new maximum, it restarts the entire downstream protocol via Restart. Once the maximum `logSize2` value is generated in the population, it propagates (triggering Restart) by epidemic in $O(\log n)$ time. The `logSize2` variable could be used to estimate K, which is the number of independent additional geometric random variables each agent will generate. We also use `logSize2` to set the leaderless phase clock inside each agent. In each epoch, the `A` agents will generate one new geometric random variable and propagate its maximum. They count their number of interactions in each epoch using the `time` variable. At the end of an epoch, when `time` reaches 95.`logSize2`, the `A` agents accumulate the value of the maximum `gr` into the `sum` of a `S` agent. The `A` agents increase their `epoch` variable by one and set `time` $= 0$ after either passing the geometric random variable to a `S` agent or interacting with a `S` agent in a higher phase. Separately, `S` agents are responsible to propagate the maximum `sum` and maximum `epoch` among themselves.

In the Log-Size-Estimation protocol, all agents in role `A` will finally generate $K = 5 \cdot$ `logSize2` geometric random variable and let the `S` agents to store a sum of maximum one generated for each phase. Once all agents reach `epoch` $= 5 \cdot$ `logSize2` they set `protocolDone` $=$ True and `output` $= \frac{\mathtt{sum}}{\mathtt{epoch}} + 1$. We use $|\mathtt{A}|$, $|\mathtt{S}|$ for the cardinality of `A` and `S` agents respectively.

Corollary 3.2. *In the Log-Size-Estimation protocol the cardinality of agents with* `A` *role is in the interval of* $\left[\frac{n}{3}, \frac{2n}{3}\right]$ *with probability* $\geq 1 - e^{-n/18}$.

In each epoch, one geometric random variable (in the first epoch `logSize2` and in the subsequent epochs `gr`) is generated and its maximum will be propagated by epidemic among the population. We set the time of each epoch equal to the required time of generating one plus the time for completion of an epidemic. To analyze the time complexity of our protocol, we require the time bounds for completing an epidemic from the paper [9]. The current form is taken from [25]. For all $n \in \mathbb{N}^+$, let $H_n = \sum_{k=1}^{n} \frac{1}{k}$ denote the n'th harmonic number. Note that $\ln n \leq \frac{n-1}{n} H_{n-1} \leq 1 + \ln n$.

Lemma 3.3 ([9]). *Let T denote the time to complete an epidemic. Then* $\mathrm{E}[T] = \frac{n-1}{n} H_{n-1}$, $\Pr\left[T < \frac{1}{4} \ln n\right] < 2e^{-\sqrt{n}}$, *and for any* $\alpha_u > 0$, $\Pr[T > \alpha_u \ln n] < 4n^{-\alpha_u/4+1}$.

The following corollary describes an epidemic in a subpopulation. This refers to some subset S of the population executing epidemic transitions only among themselves, which slows down the epidemic by only a constant factor if $|S| = \Omega(n)$.

[11] This mechanism of splitting the population approximately in two works for our protocol, because the number of `A` agents is likely to be so close to $n/2$ that our estimate of $\log n$ is reduced by an additive factor likely to be very close to -1.

Protocol 1 Log-Size-Estimation(rec, sen)

▷ initial state of agent:
role = X,
time = 0, sum = 0, epoch = 0,
gr = 1, logSize2 = 1,
protocolDone = False
Partition-Into-A/S(rec, sen)
if rec.role = A **then**
rec.time ← rec.time + 1
Check-if-Timer-Done-and-Increment-Epoch(rec)
if sen.role = A **then**
sen.time ← sen.time + 1
Check-if-Timer-Done-and-Increment-Epoch(sen)
Propagate-Max-Clock-Value(rec, sen)
Propagate-Incremented-Epoch(rec, sen)
if one agent have role = S and one have role = A **then**
Update-Sum(rec, sen)
if both agents have role = A **then**
Propagate-Max-G.R.V.(rec, sen)
if sen.protocolDone **then**
output ← $\frac{\text{sum}}{\text{epoch}} + 1$

Subprotocol 2 Partition-Into-A/S(rec, sen)

▷ Partition the population in two almost equal size subpopulations.
if sen.role = X, rec.role = X **then**
sen.role ← A
sen.logSize2 ← one geometric random variable
rec.role ← S
else if sen.role = A, rec.role = X **then**
rec.role ← S
else if sen.role = S, rec.role = X **then**
rec.role ← A
rec.logSize2 ← one geometric random variable

Subprotocol 3 Propagate-Max-Clock-Value(agent1, agent2)

▷ Maximum generated geometric variable for logSize2 will be propagated.
if agent1.logSize2 < agent2.logSize2 **then**
agent1.logSize2 ← agent2.logSize2
Restart(agent1)
else if agent2.logSize2 < agent1.logSize2 **then**
agent2.logSize2 ← agent1.logSize2
Restart(agent2)

Subprotocol 4 Restart(agent)

time ← 0, sum ← 0, epoch ← 0
gr ← one geometric random variable
protocolDone ← False

Setting $c = 3$ and $\alpha_u = 24$ in Corollary A.3 of [24] gives the following.

Subprotocol 5 Propagate-Max-G.R.V.(agent1, agent2)

▷ Maximum generated geometric variable for gr will be propagated.
if agent1.epoch = agent2.epoch **then**
if agent1.gr < agent2.gr **then**
agent1.gr ← agent2.gr
else if agent2.gr < agent1.gr **then**
agent2.gr ← agent1.gr

Subprotocol 6 Check-if-Timer-Done-and-Increment-Epoch(agent)

▷ Agents compare their time value to the specified threshold.
if agent.time = 95 × agent.logSize2, agent.protocolDone = False, agent.updatedSUM = True **then**
agent.epoch ← agent.epoch + 1
Move-to-Next-G.R.V(agent)
if agent.epoch = 5 × agent.logSize2 **then**
agent.protocolDone ← True

Subprotocol 7 Propagate-Incremented-Epoch(agent1, agent2)

▷ The maximum epoch will be propagated.
if both agents have role = A **then**
if agent1.epoch < agent2.epoch **then**
agent1.epoch ← agent2.epoch
Move-to-Next-G.R.V(agent1)
else if agent2.epoch < agent1.epoch **then**
agent2.epoch ← agent1.epoch
Move-to-Next-G.R.V(agent2)
else if both agents have role = S **then**
if agent1.epoch < agent2.epoch **then**
agent1.epoch ← agent2.epoch
agent1.sum ← agent2.sum
else if agent2.epoch < agent1.epoch **then**
agent2.epoch ← agent1.epoch
agent2.sum ← agent1.sum

Subprotocol 8 Move-to-Next-G.R.V(agent)

▷ The agent move to the next epoch:
agent.time ← 0
agent.gr ← one geometric random variable
agent.updatedSUM = False

Corollary 3.4. *Suppose an epidemic happens among a subpopulation of $n/3$ agents with time* $\mathbf{T}$. *Then* $\Pr[\mathbf{T} > 24 \ln n] < 27n^{-3}$.

The next lemma bounds the number of interactions an agent has in a given time, and it is the basis of the leaderless phase clock we use. It is proven in [24]. It follows from a simple Chernoff bound on the number of interactions involving a single agent in a given window of time.

Lemma 3.5. *Let $C \geq 3$ and $D = 2C + \sqrt{12C}$. In time $C \ln n$, with probability $\geq 1 - 1/n$, each agent has at most $D \ln n$ interactions.*

Subprotocol 9 UPDATE-SUM(agent1, agent2)

▷ The agent accumulates the current value of gr in sum:
a ← agent with role = A
s ← agent with role = S
if a.epoch = s.epoch, a.time ≥ 95· a.logSize2, and a.protocolDone = FALSE **then**
  s.epoch ← s.epoch + 1
  s.sum ← s.sum+ a.gr
  a.updatedSUM = TRUE
else if a.epoch < s.epoch **then**
  a.updatedSUM = TRUE

COROLLARY 3.6. *Each agent has* $\geq 65 \ln n$ *interactions in time* $24 \ln n$ *with probability* $\leq 1/n$.

By Lemma 3.5 each agent has at most $\left(2 \cdot 24 + \sqrt{12 \cdot 24}\right) \ln n \leq 65 \ln n \leq 94 \log n$ interactions in the time that it takes to generate and propagate maximum of one geometric random variable. Thus, each agent should count up to $94 \log n$ for its leaderless phase clock, to ensure that with high probability none reaches that count until the maximum geometric random variable is known to all agents. However, agents are not aware of any prior approximation of $\log n$. In the LOG-SIZE-ESTIMATION protocol, agents use their logSize2 variable for this approximation. As mentioned, all the agents in role A start by generating one geometric random variable logSize2. The maximum in the population is used as a weak (constant factor) approximation of $\log n$. Their statement and proofs appear in [24].

COROLLARY 3.7. *The* logSize2 *(*gr*) value generated by GENERATE-CLOCK (GENERATE-G.R.V) is in the interval of* $[\log n - \log \ln n - 2, 2 \log n - 1]$ *with probability at least* $1 - 1/n - e^{-n/18}$.

COROLLARY 3.8. *The number of interactions in each epoch in the LOG-SIZE-ESTIMATION is in the interval* $[95 \log n - 95 \log \ln n, 189 \log n]$ *with probability* $\geq 1 - 1/n - e^{-n/18}$.

PROOF. By Corollary 3.6, agents should count up to $96 \ln \leq 139 \log n$ before moving to the next epoch. if we set the threshold of the time to $95 \cdot$ logSize2, $95 \log n - 95 \log \ln n \geq 93 \log n$ then the time variable will be in the interval of $[95 \log n - 95 \log \ln n, 188 \log n + 95]$ with high probability ($188 \log n + 95 \leq 189 \log n$ for $n \geq 2$). □

COROLLARY 3.9. *The number of epochs in the LOG-SIZE-ESTIMATION is in the interval* $[5 \log n - 5 \log \ln n, 11 \log n]$ *with probability* $\geq 1 - 1/n - e^{-n/18}$.

PROOF. By Corollary C.10 in [24], to achieve the additive error of 4.7 for our protocol the number of geometric random variables should be $\geq 4 \log n$. By setting the threshold of the number of phases to $5 \times$ logSize2, for $n \geq 200$, $5 \log n - 5 \log \ln n \geq 4 \log n$. The number of phases will be in the interval of $[5 \log n - 5 \log \ln n, 10 \log n + 5]$ with high probability ($10 \log n + 5 \leq 11 \log n$ for $n \geq 2$). □

The next Lemma bounds the space complexity of our main protocol by counting the likely range taken by the variables in LOG-SIZE-ESTIMATION. It is proven in [24].

LEMMA 3.10. *LOG-SIZE-ESTIMATION uses* $O(\log^4 n)$ *states with probability* $\geq 1 - O(\log n)/n$.

The next corollary bounds the time complexity of protocol LOG-SIZE-ESTIMATION; the main component of the time complexity is that $\Theta(\log n)$ geometric random variables must be generated and propagated by epidemic among the population, each epidemic taking $\Theta(\log n)$ time. A proof appears in [24].

COROLLARY 3.11. *The LOG-SIZE-ESTIMATION protocol converges in* $O(\log^2 n)$ *time with probability at least* $1 - 1/n^2$.

The following result is a Chernoff bound on sums of random variables, each of which is the maximum of independent geometric random variables (with probability of success $\frac{1}{2}$). It is a corollary of a similar Chernoff bound proven in the appendix of [24].

LEMMA 3.12. *Let* $K \geq 4 \log n$ *and a be a number in the interval of* $[n/2 - \sqrt{n \ln n}, n/2 + \sqrt{n \ln n}]$. *Let* sum$/K$ *be the average of* K $\frac{1}{2}$*-geometric random variables. Then* $\Pr\left[\left|\frac{\text{sum}}{K} + 1 - \log n\right| \geq 5.7\right] \leq \frac{6}{n}$.

LEMMA 3.13. *In the LOG-SIZE-ESTIMATION protocol, with probability* 1*, all agents converge to the same value* C *in their* output *field. Furthermore,* $\Pr[|C - \log n| \geq 5.7] \leq 9/n$.

Termination with a leader and guaranteed size upper bound. Two other results are discussed in more detail in [24]. The first is that we can make the size-estimation protocol terminating with high probability using an initial leader. Intuitively, the leader can be used to trigger an epidemic-based phase clock used to count to $\Omega(\log^2 n)$, enough time for the protocol to probably have converged. The second discusses the possibility of transforming the size estimation, which has a small probability of being much larger *or* much smaller than the actual size, into a *guaranteed upper bound* on the population size.

Reducing the space complexity. In our protocol, we used space multiplexing, a known technique in population protocols that split the state space such that different agents are responsible to store different variables. Although this technique reduces the number of states per agent, we cannot push it further with the current scheme. Our protocol is dependent on *all* agents agreeing on the values of logSize2 and epoch to stay synchronized. Thus, if an agent participates in the protocol it is required to stores the updated value of both logSize2 and epoch.

4 TERMINATION

The concept of termination has been referenced and studied in population protocols [11, 32, 33], but to our knowledge no formal definition exists. We give an abstract definition capturing the behavior of most protocols that "perform a computational task".

Let P be a protocol with a set I of "valid" initial configurations, where each agent's memory has a Boolean field terminated set to FALSE in every configuration in I. A configuration $\vec{c}$ of P is *terminated* if at least one agent in $\vec{c}$ has terminated = TRUE. (Note the distinction with a *silent* configuration, where no transition can change any agent's state [13].) Let $\kappa > 0$ and $t : \mathbb{N} \to \mathbb{N}$. P is κ*-*t*-terminating* if, for all $\vec{i} \in I$, with probability $\geq \kappa$, P reaches from $\vec{i}$ to a terminated configuration $\vec{c}$, but takes time $\geq t(n)$ to do so.

This definition leaves totally abstract which particular task (e.g., leader election) is assumed to have terminated. The idea is that if the

task will not be complete before time $t(n)$ with high probability, then no agent should set terminated = True until time $\geq t(n)$ with high probability. So proving an upper bound on $t(n)$ in the definition of terminating implies that no protocol can be terminating if it requires time $> t(n)$ to converge.

The definition is applicable beyond the narrow goal of terminating a population protocol. It says more generally that a "signal" is produced after some amount of time. This signal may be used to terminate a protocol, move it from one "stage" to another, or it may be some specific Boolean value relevant to a specific protocol, where in any case the value will start False for all agents and eventually be set to True for at least one agent.

Let $\alpha > 0$. We say a configuration $\vec{c}$ is *α-dense* if, for all $s \in \Lambda$, $\vec{c}(s) > 0 \implies \vec{c}(s) \geq \alpha n$. (Recall $n = \|\vec{c}\|$.) In other words, every state present occupies at least fraction α of the population. We say protocol P with valid initial configuration set I is *i.o.-dense* if there exists $\alpha > 0$ such that infinitely many $\vec{i} \in I$ are α-dense. In particular, an i.o.-dense protocol does not always have an initial *leader*: a state present in count 1 in every $\vec{i} \in I$.

The next theorem, our second main result, shows that termination is impossible for uniform i.o.-dense protocols that require more than constant time, no matter the space allowed.

Theorem 4.1. *Let $\kappa > 0$ and $t : \mathbb{N} \to \mathbb{N}$, and let P be a uniform i.o.-dense population protocol. If P is κ-t-terminating, then $t(n) = O(1)$.*

Let Λ be the (possibly infinite) set of all states of a population protocol. Recall the definition of randomized transitions from Section 2; We now introduce extra notation that will be useful in this Section. We consider a transition *relation* $\Delta \subseteq \Lambda^4$, writing $a, b \to c, d$ to denote that $(a, b, c, d) \in \Delta$ (i.e., if agents in states a and b interact, then one of the possible random outcomes is to change to states c and d). For $\rho \in (0, 1]$, we write $a, b \xrightarrow{\rho} c, d$ to denote that when states a and b interact, with probability ρ they transition to c and d. Say that ρ is the *rate constant* of transition $a, b \xrightarrow{\rho} c, d$. If there exist $a, b \in \Lambda$ and $\rho' \geq \rho$ such that $a, b \xrightarrow{\rho'} c, d$, we write $c \in \mathrm{PROD}_\rho(a, b)$ and $d \in \mathrm{PROD}_\rho(a, b)$. (In other words, $c \in \mathrm{PROD}_\rho(a, b)$ if c is produced with probability at least ρ whenever a and b interact). For any $\Gamma \subseteq \Lambda$ and $\rho \in [0, 1]$, define $\mathrm{PROD}_\rho(\Gamma) = \{s \in \Lambda \mid (\exists a, b \in \Gamma)\ s \in \mathrm{PROD}_\rho(a, b)\}$.

Let $\Lambda^0 \subseteq \Lambda$. For $i \in \mathbb{N}^+$, define $\Lambda_\rho^i = \Lambda_\rho^{i-1} \cup \mathrm{PROD}_\rho(\Lambda_\rho^{i-1})$. For $m \in \mathbb{N}$, if $s \in \Lambda_\rho^m$, we say s is *m-ρ-producible from Λ^0*. For configuration $\vec{c}$, we say s is *m-ρ-producible from $\vec{c}$* if s is m-ρ-producible from $\Lambda^0 = \{s \in \Lambda \mid \vec{c}(s) > 0\}$, the states present in $\vec{c}$.

Our main technical tool is the following lemma, a variant of the "timer/density lemma" of [23] (see also [2]). The original lemma states that in a protocol with $O(1)$ states, from any sufficiently large α-dense configuration, in $O(1)$ time all states appear with δ-density (for some $0 < \delta < \alpha$). The proof is similar to that of [23], but is retooled to apply to protocols with a non-constant set of states (also to use the discrete-time model of population protocols, instead of the continuous-time model of chemical reaction networks).[12] The key new idea is that, even if a protocol has infinitely many states (of which only finitely many can be produced in finite time), for any subset of states Λ_ρ^m "producible via only m transitions, each having rate constant at least ρ", all states in Λ_ρ^m are produced in constant time with high probability from sufficiently large configurations.

Lemma 4.2. *Let $\alpha > 0$, $m \in \mathbb{N}^+$, $\rho \in (0, 1]$, and P be a population protocol. Then there are constants $\epsilon, \delta, n_0 > 0$ such that, for all $n \geq n_0$, for all α-dense configurations $\vec{c}$ of P with $n = \|\vec{c}\|$, the following holds. Let Λ_ρ^m be the set of states m-ρ-producible from $\vec{c}$. For $s \in \Lambda$ and $t > 0$, let $\mathbf{C}_{t,s}$ be the random variable denoting the count of s at time t, assuming at time 0 the configuration is $\vec{c}$. Then* $\Pr\left[(\forall s \in \Lambda_\rho^m)\ \mathbf{C}_{1,s} \geq \delta n\right] \geq 1 - 2^{-\epsilon n}$.

A self-contained proof is in [24]. Intuitively, Lemma 4.2 can be used to prove Theorem 4.1 in the following way. In some "small" population size n_0, the terminal signal appears. The set of states $\Lambda' \subseteq \Lambda$ appearing with the terminal signal is constant size. Lemma 4.2 states that for any constant-size $\Lambda' \subseteq \Lambda$, in all sufficiently large population sizes, all states in Λ' appear in constant time with high probability, so the termination signal appears prematurely in larger populations. This is fairly straightforward for deterministic transition functions, but it requires some care to handle a randomized protocol.

We now use Lemma 4.2 to formally prove Theorem 4.1.

Proof of Theorem 4.1. Assume P is κ-t-terminating; we will show $t(n) = O(1)$. Let $(\vec{c}_i)_{i=1}^\infty$ be an infinite sequence of α-dense initial configurations in I. Dickson's Lemma [22] states that every infinite sequence in $\mathbb{N}^k$ has an infinite nondecreasing subsequence, so assume without loss of generality that $\vec{c}_i \leq \vec{c}_{i+1}$ for all $i \in \mathbb{N}$. Let $\Lambda^0 = \{s \in \Lambda \mid \vec{c}_0(s) > 0\}$ be the set of states present in $\vec{c}_0$.

By hypothesis $\Pr[P \text{ terminates from } \vec{c}_0] \geq \kappa > 0$. Thus there is at least one finite execution $\mathcal{E}$ starting with $\vec{c}_0$ and ending in a terminated configuration. Let $m = |\mathcal{E}|$ be the length of this execution. Let ρ be the minimum rate constant of any transition in $\mathcal{E}$. Then every state appearing in configurations in $\mathcal{E}$ is m-ρ-producible from $\vec{c}_0$, i.e., is in Λ_ρ^m where $\Lambda^0 = \{s \in \Lambda \mid \vec{c}_0(s) > 0\}$ is the set of states present in $\vec{c}_0$.

For any $\ell \geq 1$, since $\vec{c}_0 \leq \vec{c}_\ell$, all states in Λ_ρ^m are m-ρ-producible from $\vec{c}_\ell$ as well. By Lemma 4.2, there are constants $\epsilon, \delta, n_0 > 0$ such that, for all $\ell \in \mathbb{N}$ such that $n = \|\vec{c}_\ell\| \geq n_0$, letting $\mathbf{C}_{1,s}$ be the random variable denoting the count of s at time 1, assuming at time 0 the configuration is $\vec{c}_\ell$, $\Pr\left[\left(\forall s \in \Lambda_\rho^m\right)\ \mathbf{C}_{1,s} \geq \delta n\right] \geq 1 - 2^{-\epsilon n}$.

However, Λ_ρ^m contains terminated states, so for all $\vec{c}_\ell$ with $\|\vec{c}_\ell\| \geq n_0$, with probability $\geq 1 - 2^{-\epsilon n}$, P terminates within time 1. Since $2^{-\epsilon n} < \kappa$ for sufficiently large n, this implies that if P is κ-t-terminating, then $t(n) \leq 1$ for sufficiently large n. Thus $t(n) = O(1)$. □

Observe how the assumption of uniformity is used in the proof: we take a set of transitions used on the population $\vec{c}_0$ and apply it to a larger population $\vec{c}_\ell$. In a nonuniform protocol, the transitions may not be legal to apply to $\vec{c}_\ell$. As a concrete example, in a nonuniform protocol, an agent increments a counter using transitions such as $c_7, x \to c_8, y$ until the counter exceeds $\log n$, then produces a termination signal t via a transition $c_8, x \to t, y$. The transition

[12] Alistarh et al. [2] also prove a variant applying to protocols with $\omega(1)$ states, but for a different purpose: to show that all states in Λ appear as long as $|\Lambda| \leq \frac{1}{2} \log \log n$. However, beyond that bound, the lemma does not hold [29]. In our case, we are not trying to show that all states in Λ appear, only those in some constant size subset of states, all of which are m-ρ-producible from the initial configuration.

$c_8, x \to t, y$ producing this signal is not legal in a population larger than twice n, since the value $\log n$ is at least 1 larger in such a protocol. In this example, the transition of the larger protocol with the same input states simply increments the counter without producing a termination signal: $c_8, x \to c_9, y$.

Acknowledgements. We are grateful to Eric Severson for helpful comments and anonymous reviewers for their suggestions, which vastly improved the paper. The second author thanks James Aspnes for discussions that stimulated a key idea used in the main protocol.

REFERENCES

[1] Scott Aaronson. 2006. Computational Complexity and the Anthropic Principle. https://www.scottaaronson.com/talks/anthropic.html.

[2] Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L Rivest. 2017. Time-space trade-offs in population protocols. In *SODA 2017: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2560–2579.

[3] Dan Alistarh, James Aspnes, and Rati Gelashvili. 2018. Space-optimal majority in population protocols. In *SODA 2018: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2221–2239.

[4] Dan Alistarh and Rati Gelashvili. 2015. Polylogarithmic-Time Leader Election in Population Protocols. In *42nd International Colloquium on Automata, Languages, and Programming (ICALP) (Lecture Notes in Computer Science)*, Vol. 9135. Springer, Berlin, Heidelberg, 479 – 491.

[5] Dan Alistarh and Rati Gelashvili. 2018. Recent Algorithmic Advances in Population Protocols. *SIGACT News* 49, 3 (Oct. 2018), 63–73. https://doi.org/10.1145/3289137.3289150

[6] Dan Alistarh, Rati Gelashvili, and Milan Vojnović. 2015. Fast and Exact Majority in Population Protocols. In *PODC 2015: Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*. ACM, 47–56.

[7] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. 2006. Computation in networks of passively mobile finite-state sensors. *Distributed Computing* 18, 4 (March 2006), 235–253.

[8] Dana Angluin, James Aspnes, and David Eisenstat. 2006. Stably computable predicates are semilinear. In *25th annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM Press, New York, NY, USA, 292–299. https://doi.org/10.1145/1146381.1146425

[9] Dana Angluin, James Aspnes, and David Eisenstat. 2008. Fast computation by population protocols with a leader. *Distributed Computing* 21, 3 (September 2008), 183–199.

[10] James Aspnes, Joffroy Beauquier, Janna Burman, and Devan Sohier. 2017. Time and Space Optimal Counting in Population Protocols. In *20th International Conference on Principles of Distributed Systems (OPODIS 2016)*, Vol. 70. 13:1–13:17.

[11] James Aspnes and Eric Ruppert. 2007. An introduction to population protocols. *Bulletin of the European Association for Theoretical Computer Science* 93 (October 2007), 98–117.

[12] Joffroy Beauquier, Janna Burman, Simon Claviere, and Devan Sohier. 2015. Space-optimal counting in population protocols. In *DISC 2015: International Symposium on Distributed Computing*. Springer, 631–646.

[13] Joffroy Beauquier, Maria Gradinariu, and Colette Johnen. 1999. Memory Space Requirements for Self-stabilizing Leader Election Protocols. In *Proceedings of the 18th Annual ACM Symposium on Principles of Distributed Computing (PODC 1999)*. ACM, 199–207. https://doi.org/10.1145/301308.301358

[14] Amanda Belleville, David Doty, and David Soloveichik. 2017. Hardness of computing and approximating predicates and functions with leaderless population protocols. In *ICALP 2017: 44th International Colloquium on Automata, Languages, and Programming (LIPIcs)*, Vol. 80. 141:1–141:14.

[15] Petra Berenbrink, Dominik Kaaser, Peter Kling, and Lena Otterbach. 2018. Simple and Efficient Leader Election. In *1st Symposium on Simplicity in Algorithms (SOSA 2018)*, Vol. 61. 9:1–9:11.

[16] Petra Berenbrink, Dominik Kaaser, and Tomasz Radzik. 2019. On counting the population size. In *Proceedings of the 38th Annual ACM Symposium on Principles of Distributed Computing (PODC 2019)*. to appear.

[17] Andreas Bilke, Colin Cooper, Robert Elsässer, and Tomasz Radzik. 2017. 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, 451–453.

[18] James M Bower and Hamid Bolouri. 2004. *Computational modeling of genetic and biochemical networks*. MIT press.

[19] Shukai Cai, Taisuke Izumi, and Koichi Wada. 2010. Space Complexity of Self-stabilizing Leader Election in Passively-Mobile Anonymous Agents. In *Structural Information and Communication Complexity*, Shay Kutten and Janez Žerovnik (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 113–125.

[20] Ho-Lin Chen, David Doty, and David Soloveichik. 2013. Deterministic Function Computation with Chemical Reaction Networks. *Natural Computing* 13, 4 (2013), 517–534. Special issue of invited papers from DNA 2012.

[21] Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. 2013. Programmable chemical controllers made from DNA. *Nature Nanotechnology* 8, 10 (2013), 755–762.

[22] Leonard E. Dickson. 1913. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics* 35 (1913), 413–422. Issue 4. https://doi.org/10.2307/2370405

[23] David Doty. 2014. Timing in chemical reaction networks. In *SODA 2014: Proc. of the 25th Annual ACM-SIAM Symp. on Discrete Algorithms*. 772–784.

[24] David Doty and Mahsa Eftekhari. 2018. Efficient size estimation and impossibility of termination in uniform dense population protocols. *CoRR* abs/1808.08913 (2018). arXiv:1808.08913 http://arxiv.org/abs/1808.08913

[25] David Doty, Mahsa Eftekhari, Othon Michail, Paul G. Spirakis, and Michail Theofilatos. 2018. Brief announcement: Exact size counting in uniform population protocols in nearly logarithmic time. In *32nd International Symposium on Distributed Computing (DISC 2018)*. 46:1–46:3.

[26] David Doty and Monir Hajiaghayi. 2015. Leaderless Deterministic Chemical Reaction Networks. *Natural Computing* 14, 2 (2015), 213–223. Preliminary version appeared in DNA 2013.

[27] David Doty and David Soloveichik. 2018. Stable leader election in population protocols requires linear time. *Distributed Computing* 31, 4 (2018), 257–271. Special issue of invited papers from DISC 2015.

[28] Bennett Eisenberg. 2008. On the expectation of the maximum of IID geometric random variables. *Statistics & Probability Letters* 78, 2 (2008), 135 – 143. https://doi.org/10.1016/j.spl.2007.05.011

[29] Leszek Gasieniec and Grzegorz Stachowiak. 2018. Fast Space Optimal Leader Election in Population Protocols. In *SODA 2018: ACM-SIAM Symposium on Discrete Algorithms*. to appear.

[30] Leszek Gasieniec, Grzegorz Stachowiak, and Przemyslaw Uznanski. 2018. *Almost logarithmic-time space optimal leader election in population protocols*. Technical Report. arXiv:1802.06867 http://arxiv.org/abs/1802.06867

[31] Adrian Kosowski and Przemyslaw Uznanski. 2018. Population Protocols Are Fast. *CoRR* abs/1802.06872 (2018). arXiv:1802.06872 http://arxiv.org/abs/1802.06872

[32] Othon Michail. 2015. Terminating Distributed Construction of Shapes and Patterns in a Fair Solution of Automata. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*. 37–46. Also in *Distributed Computing*, 2017.

[33] Othon Michail, Ioannis Chatzigiannakis, and Paul G Spirakis. 2012. Terminating population protocols via some minimal global knowledge assumptions. In *Stabilization, Safety, and Security of Distributed Systems (SSS)*. Springer, 77–89.

[34] Yves Mocquard, Bruno Sericola, and Emmanuelle Anceaume. 2018. Population protocols with convergence detection. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. IEEE, 1–8.

[35] Philippe Rigollet. 2015. Lecture Notes for MIT course 18.S997: High Dimensional Statistics. URL: https://ocw.mit.edu/courses/mathematics/18-s997-high-dimensional-statistics-spring-2015/lecture-notes/.

[36] David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. 2008. Computation with Finite Stochastic Chemical Reaction Networks. *Natural Computing* 7, 4 (2008), 615–633. http://dx.doi.org/10.1007/s11047-008-9067-y

[37] Niranjan Srinivas, James Parkin, Georg Seelig, Erik Winfree, and David Soloveichik. 2017. Enzyme-free nucleic acid dynamical systems. *Science* 358, 6369 (2017), eaal2052.

[38] Yuichi Sudo, Fukuhito Ooshita, Taisuke Izumi, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. 2019. Brief Announcement: Logarithmic Expected-Time Leader Election in Population Protocol Model. In *Proceedings of the 38th Annual ACM Symposium on Principles of Distributed Computing (PODC 2019)*. to appear.

[39] Vito Volterra. 1926. Variazioni e fluttuazioni del numero dâĂŹindividui in specie animali conviventi. *Mem. Acad. Lincei Roma* 2 (1926), 31–113.