Polylogarithmic-Time Leader Election in Population Protocols

Dan Alistarh[1](✉) and Rati Gelashvili[2]

[1] Microsoft Research, Cambridge, UK
dan.alistarh@microsoft.com
[2] MIT, Cambridge, MA, USA
gelash@mit.edu

Abstract. Population protocols are networks of finite-state agents, interacting randomly, and updating their states using simple rules. Despite their extreme simplicity, these systems have been shown to cooperatively perform complex computational tasks, such as simulating register machines to compute standard arithmetic functions. The election of a unique *leader agent* is a key requirement in such computational constructions. Yet, the fastest currently known population protocol for electing a leader only has *linear* convergence time, and it has recently been shown that no population protocol using a *constant* number of states per node may overcome this linear bound.

In this paper, we give the first population protocol for leader election with *polylogarithmic* convergence time, using polylogarithmic memory states per node. The protocol structure is quite simple: each node has an associated value, and is either a *leader* (still in contention) or a *minion* (following some leader). A leader keeps incrementing its value and "defeats" other leaders in one-to-one interactions, and will drop from contention and become a minion if it meets a leader with higher value. Importantly, a leader also drops out if it meets a *minion* with higher absolute value. While these rules are quite simple, the proof that this algorithm achieves polylogarithmic convergence time is non-trivial. In particular, the argument combines careful use of concentration inequalities with anti-concentration bounds, showing that the leaders' values become spread apart as the execution progresses, which in turn implies that straggling leaders get quickly eliminated. We complement our analysis with empirical results, showing that our protocol converges extremely fast, even for large network sizes.

1 Introduction

Recently, there has been significant interest in modeling and analyzing interactions arising in biological or bio-chemical systems through an algorithmic lens. In particular, the population protocol model [AAD+06], which is the focus of this paper, consists of a set of n finite-state nodes interacting in pairs, where each interaction may update the states of both participants. The goal is to have all nodes converge on an output value, which represents the result of the

M.M. Halldórsson et al. (Eds.): ICALP 2015, Part II, LNCS 9135, pp. 479–491, 2015.
DOI: 10.1007/978-3-662-47666-6_38

computation, usually a predicate on the initial state of the nodes. The set of interactions occurring at each step is assumed to be decided by an adversarial scheduler, which is usually subject to some fairness conditions. The standard scheduler when computing convergence bounds is the *probabilistic (uniform random) scheduler*, e.g., [AAE08b,PVV09,DV12], which picks the next pair to interact uniformly at random in each step. We adopt this probabilistic scheduler model in this paper. (Some references refer to this model as the *probabilistic* population model.) The fundamental measure of convergence is *parallel time*, defined as the number of scheduler steps until convergence, divided by n.[1]

The class of predicates computable by population protocols is now well-understood [AAD$^+$06,AAE06,AAER07] to consist precisely of *semilinear predicates*, i.e. predicates definable in first-order Presburger arithmetic. The first such construction was given in [AAD$^+$06], and later improved in terms of convergence time in [AAE06]. A parallel line of research studied the computability of deterministic functions in chemical reaction networks, which are also instances of population protocols [CDS14]. All three constructions fundamentally rely on the election of a single initial *leader* node, which co-ordinates phases of computation.

Reference [AAD$^+$06] gives a simple protocol for electing a leader from a uniform population, based on the natural idea of having leaders eliminate each other directly through symmetry breaking. Unfortunately, this strategy takes at least linear parallel time in the number of nodes n: for instance, once this algorithm reaches *two* surviving leaders, it will require $\Omega(n^2)$ additional interactions for these two leaders to meet. Reference [AAE08a] proposes a significantly more complex protocol, conjectured to be sub-linear, and whose convergence is only studied experimentally. This reference posits the existence of a sublinear-time population protocol for leader election as a "pressing" open problem. In fact, the existence of a poly-logarithmic leader election protocol would imply that *any semilinear predicate* is computable in poly-logarithmic time by a *uniform population* [AAE06].

Recently, Doty and Soloveichik [DS15] showed that $\Omega(n^2)$ expected interactions are *necessary* for electing a leader in the classic probabilistic protocol model in which each node only has *constant* number of memory states (with respect to n). This negative result implies that computing semilinear predicates in leader-based frameworks is subject to the same lower bound. In turn, this motivates the question of whether faster computation is possible if the amount of memory per node is allowed to be a function of n.

Contribution. In this paper, we solve this problem by proposing a new population protocol for leader election, which converges in $O(\log^3 n)$ expected parallel time, using $O(\log^3 n)$ memory states per node. Our protocol, called *LM* for *Leader-Minion*, roughly works as follows. Throughout the execution, each node is either a *leader*, meaning that it can still win, or a *minion*, following some leader. Each node state is associated to some *absolute value*, which is a positive

[1] An alternative definition is when reactions occur in parallel according to a Poisson process [PVV09,DV12].

integer, and with a *sign*, positive if the node is still in contention, and negative if the node has become a minion.

If two leaders meet, the one with the larger absolute value survives, and increments its value, while the other drops out, becoming a minion, and adopting the other node's value, but with a negative sign. (If both leaders have the same value, they both increment it and continue.) If a leader meets a minion with *smaller* absolute value than its own, it increments its value, while the minion simply adopts the leader's value, but keeps the negative sign. Conversely, if a leader meets a minion with *larger* absolute value than its own, then the leader drops out of contention, adopting the minion's value, with negative sign. Finally, if two minions meet, they update their values to the maximum absolute value between them, but with a *negative* sign.

These rules ensure that, eventually, a single leader survives. While the protocol is relatively simple, the proof of poly-logarithmic time convergence is non-trivial. In particular, the efficiency of the algorithm hinges on the minion mechanism, which ensures that a leader with high absolute value can eliminate other contenders in the system, without having to directly interact with them.

Roughly, the argument is based on two technical insights. First, consider two leaders at a given time T, whose (positive) values are at least $\Theta(\log n)$ apart. Then, we show that, within $O(\log n)$ parallel time from T, the node holding the smaller value has become a minion, with constant probability. Intuitively, this holds since 1) this node will probably meet either the other leader or one of its minions within this time interval, and 2) it cannot increase its count fast enough to avoid defeat. For the second part of the argument, we show via anti-concentration that, after parallel time $\Theta(\log^2 n)$ in the execution, the values corresponding to an arbitrary pair of nodes will be separated by at least $\Omega(\log n)$.

We ensure that the values of nodes cannot grow beyond a certain threshold, and set the threshold in such a way that the total number of states is $\Theta(\log^3 n)$. We show that with high probability the leader will be elected before the values of the nodes reach the threshold. In the other case, remaining leaders with threshold values engage in a *backup* dynamics where minions are irrelevant and leaders defeat each other when they meet based on random binary indicators which are set using the randomness of the scheduler. This process is slower but deterministically correct, and only happens with very low probability, allowing to conclude that the algorithm converges to a single leader within $O(\log^3 n)$ parallel time, both with high probability and in expectation, using $O(\log^3 n)$ states.

In population protocols, in every interaction, one node is said to be the *initiator*, the other is the *responder*, and the state update rules can use this distinction. In our protocol, this would allow a leader (the initiator in the interaction) to defeat another leader with the same value (the responder), and could also simplify the backup dynamics of our algorithm. However, our algorithm has the nice property that the state update rules can be made completely symmetric with regards to the initiator and responder roles. (For this reason, LM works for $n > 2$ nodes, because to elect a leader among two nodes it is necessary to rely on the initiator-responder role distinction.)

Summing up, we give the first poly-logarithmic time protocol for electing a leader from a uniform population. We note that $\Omega(n \log n)$ interactions seem intuitively necessary for leader election, as this number is required to allow each node to interact at least once. However, this idea fails to cover all possible reaction strategies if nodes are allowed to have arbitrarily many states.

We complement our analysis with empirical data, suggesting that the convergence time of our protocol is close to logarithmic, and that in fact the asymptotic constants are small, both in the convergence bound, and in the upper bound on the number of states the protocol employs.

Related Work. We restrict our attention to work in the population model. The framework of population protocols was formally introduced in reference [AAD+06], to model interactions arising in biological, chemical, or sensor networks. It sparked research into its computational power [AAD+06, AAE06, AAER07], and into the time complexity of fundamental tasks such as majority [AAE08b, PVV09, DV12], and leader election [AAD+06, AAE08a].[2] References interested in *computability* consider an adversarial scheduler which is restricted to be *fair*, e.g., where each agent interacts with every other agent infinitely many times. For complexity bounds, the standard scheduler is *uniform*, scheduling each pair uniformly at random at each step, e.g., [AAE08b, PVV09, DV12]. This model is also known as the *probabilistic* population model.

To the best of our knowledge, no population protocol for electing a leader with sub-linear convergence time was known before our work. References [AAD+06, AAE06, CDS14] present leader-based frameworks for population computations, assuming the existence of such a node. The existence of such a sub-linear protocol is stated as an open problem in [AAD+06, AAE08a]. Reference [DH13] proposes a *leader-less* framework for population computation.

Recent work by Doty and Soloveichik [DS15] showed an $\Omega(n^2)$ lower bound on the number of interactions necessary for electing a leader in the classic probabilistic protocol model in which each node only has *constant* number of memory states with respect to the number of nodes n [AAER07]. The proof of this result is quite complex, and makes use of the limitation that the number of states remains constant even as the number of nodes n is taken to tend to infinity.

Thus, our algorithm provides a complexity separation between population protocols which may only use constant memory per node, and protocols where the number of states is allowed to be a function of n. We note that, historically, the classic population protocol model [AAD+06] only allowed a constant number of states per node, while later references relaxed this assumption.

A parallel line of research studied *self-stabilizing* population protocols, e.g., [AAFJ06, FJ06, SNY+10], that is, protocols which can converge to a correct solution from an *arbitrary* initial state. It is known that stable leader election is *impossible* from an arbitrary initial state [AAFJ06]. References [FJ06, SNY+10] circumvent this impossibility by relaxing the problem semantics. Our algorithm is not affected by this impossiblity result since it is not self-stabilizing.

[2] Leader election and majority are complementary tasks, and no complexity-preserving transformations exist, to our knowledge.

2 Preliminaries

Population Protocols. We assume a population consisting of n agents, or nodes, each executing as a deterministic state machine with states from a finite set Q, with a finite set of input symbols $X \subseteq Q$, a finite set of output symbols Y, a transition function $\delta : Q \times Q \rightarrow Q \times Q$, and an output function $\gamma : Q \rightarrow Y$. Initially, each agent starts with an input from the set X, and proceeds to update its state following interactions with other agents, according to the transition function δ. For simplicity of exposition, we assume that agents have identifiers from the set $V = \{1, 2, \ldots, n\}$, although these identifiers are not known to agents, and not used by the protocol.

The agents' interactions proceed according to a directed *interaction graph* G without self-loops, whose edges indicate possible agent interactions. Usually, the graph G is considered to be the complete directed graph on n vertices, a convention we also adopt in this paper.

The execution proceeds in *steps*, or *rounds*, where in each step a new edge (u, w) is chosen uniformly at random from the set of edges of G. Each of the two chosen agents updates its state according to function δ.

Parallel Time. The above setup considers sequential interactions; however, in general, interactions between pairs of distinct agents are independent, and are usually considered as occurring in parallel. In particular, it is customary to define one unit of *parallel time* as n consecutive steps of the protocol.

The Leader Election Problem. In the *leader election* problem, all agents start in the same initial state A, i.e. the only state in the input set $X = \{A\}$. The output set is $Y = \{\mathit{Win}, \mathit{Lose}\}$.

A population protocol solves leader election within ℓ steps with probability $1 - \phi$, if it holds with probability $1 - \phi$ that for any configuration $c : V \rightarrow Q$ reachable by the protocol after $\geq \ell$ steps, there exists a unique agent i such that, (1) for the agent i, $\gamma(c(i)) = \mathit{Win}$, and, (2) for any agent $j \neq i$, $\gamma(c(j)) = \mathit{Lose}$.

3 The Leader Election Algorithm

In this section, we describe the *LM* leader election algorithm. The algorithm has an integer parameter $m > 0$, which we set to $\Theta(\log^3 n)$. Each state corresponds to an integer value from the set $\{-m, -m+1, \ldots, -2, -1, 1, 2, m-1, m, m+1\}$. Respectively, there are $2m+1$ different states. We will refer to states and values interchangeably. All nodes start in the same state corresponding to value 1.

The algorithm, specified in Figure 1, consists of a set of simple deterministic update rules for the node state. In the pseudocode, the node states before an interaction are denoted by x and y, while their new states are given by x' and y'. All nodes start with value 1 and continue to interact according to these simple rules. We prove that all nodes except one will converge to negative values, and

Parameters:
m, an integer > 0, set to $\Theta(\log^3 n)$
State Space:
$\mathit{LeaderStates} = \{1, 2, \ldots, m-1, m, m+1\}$,
$\mathit{MinionStates} = \{-1, -2, \ldots, -m+1, -m\}$,
Input: States of two nodes, x and y
Output: Updated states x' and y'
Auxiliary Procedures:

$$\textit{is-contender}(x) = \begin{cases} \mathsf{true} & \text{if } x \in \mathit{LeaderStates}; \\ \mathsf{false} & \text{otherwise.} \end{cases}$$

$$\textit{contend-priority}(x, y) = \begin{cases} m & \text{if } \max(|x|, |y|) = m+1; \\ \max(|x|, |y|) + 1 & \text{otherwise.} \end{cases}$$

$$\textit{minion-priority}(x, y) = \begin{cases} -m & \text{if } \max(|x|, |y|) = m+1; \\ -\max(|x|, |y|) & \text{otherwise.} \end{cases}$$

1 **procedure** update$\langle x, y \rangle$
2   **if** $\textit{is-contender}(x)$ ***and*** $|x| \geq |y|$ **then**
3         $x' \leftarrow \textit{contend-priority}(x, y)$
4   **else** $x' \leftarrow \textit{minion-priority}(x, y)$
5   **if** $\textit{is-contender}(y)$ ***and*** $|y| \geq |x|$ **then**
6         $y' \leftarrow \textit{contend-priority}(x, y)$
7   **else** $y' \leftarrow \textit{minion-priority}(x, y)$

Fig. 1. The state update rules for the *LM* algorithm

that convergence is fast with high probability. This solves the leader election problem since we can define γ as mapping only positive states to *Win* (a leader).[3]

Since positive states translate to being a leader according to γ, we call a node a *contender* if it has a positive value, and a *minion* otherwise. We present the algorithm in detail below.

The state updates (i.e. the transition function δ) of the *LM* algorithm are completely symmetric, that is, the new state x' depends on x and y (lines 2-4) exactly as y' depends on y and x (lines 5-7).

If a node is a contender and has absolute value not less than the absolute value of the interaction partner, then the node remains a contender and updates its value using the *contend-priority* function (lines 3 and 6). The new value will be one larger than the previous value except when the previous value was $m+1$, in which case the new value will be m.

If a node had a smaller absolute value than its interaction partner, or was a minion already, then the node will be a minion after the interaction. It will set its value using the *minion-priority* function, to either $-\max(|x|, |y|)$, or $-m$ if the maximum was $m+1$ (lines 4 and 7).

Values $m+1$ and m are treated the same way if the node is a minion (essentially corresponding to $-m$). These values serve as a binary tie-breaker among the contenders that reach the value m, as will become clear from the analysis.

[3] Alternatively, γ that maps states with values m and $m+1$ to *WIN* would also work, but we will work with positive "leader" states for the simplicity of presentation.

4 Analysis

In this section, we provide a complete analysis of our leader election algorithm.

Notation. Throughout this proof, we denote the set of n nodes executing the protocol by V. We measure execution time in discrete steps (rounds), where each step corresponds to an interaction. The *configuration* at a given time t is a function $c : V \to Q$, where $c(v)$ is the state of the node v at time t. (We omit time t when clear from the context.) We call a node *contender* when the value associated with its state is positive, and a *minion* when the value is negative. As previously discussed, we assume $n > 2$. Also, for presentation purposes, consider n to be a power of two. We first prove that the algorithm never eliminates all contenders and that having a single contender means that a leader is elected.

Lemma 1. *There is always at least one contender in the system. After an execution reaches a configuration with only a single node v being a contender, then from this point, v will have $c(v) > 0$ (mapped to WIN by γ) in every reachable future configuration c, and there may never be another contender.*

Proof. By the structure of the algorithm, a node starts as a contender and may become a minion during an execution, but a minion may never become a contender. Moreover, an absolute value associated with the state of a minion node can only increase to an absolute value of an interaction partner. Finally, an absolute value may never decrease except from $m + 1$ to m.

Let us assume for contradiction that an execution reaches a configuration where all nodes are minions. Consider such a time point T_0 and let the maximum absolute value of the nodes at T_0 be u. Because the minions cannot increase the maximum absolute value in the system, there must have been a contender node v and a time $T_1 < T$ such that v had value u at time T_1. In order for this contender to have become a minion by time T_0, it must have interacted with another node with an absolute value strictly larger than u, after time T_1. However, the absolute value of a node never decreases except from $m+1$ to m, and despite the existence of an absolute value larger than u before time T_0, u is the largest absolute value at time T_0. The only way this can occur is if $u = m$ and the node v interacted with a node v' with value $m + 1$. But after such an interaction the node v' remains a contender with value m. In order for v' to become a minion by time T_0, it must have interacted with yet another node v'' of value $m + 1$ at some time T_2 between T_1 and T_0. But then this node v'' is left as a contender with value m, and the same reasoning applies to it. By infinite descent, we obtain a contradiction with the initial assumption that all nodes are minions.

Consequently, whenever there is a single contender in the system, it must have the largest absolute value. Otherwise, it could interact with a node with a larger absolute value and become a minion itself, contradicting the invariant that not all nodes can be minions at the same time.

Now we turn our attention to the convergence speed of the *LM* algorithm. Our goal is bound the number of rounds necessary to eliminate all except a single contender. In order for a contender to get eliminated, it must come across a larger value of another contender, the value possibly conducted through a chain of multiple minions via multiple interactions.

We first show by a rumor spreading argument that if the difference between the values of two contenders is large enough, then the contender with the smaller value will become a minion within the next $O(n \log n)$ rounds, with constant probability. Then using anti-concentration bounds we establish that for any two contenders, if no absolute value in the system reaches m, after $O(n \log^2 n)$ rounds the difference between their values is large enough with constant probability.

Lemma 2. *Consider two contender nodes with values u_1 and u_2, where $u_1 - u_2 \geq 4\xi \log n$ at time T for $\xi \geq 8$. Then, after $\xi n \log n$ rounds from T, the node that initially held the value u_2 will be a minion with probability at least $1/24$, independent of the history of previous interactions.*

Proof. Call a node that has an absolute value of at least u_1 an *up-to-date* node, and *out-of-date* otherwise. At time T, at least one node is up-to-date. Before an arbitrary round where we have x up-to-date nodes, the probability that an out-of-date node interacts with an up-to-date node, increasing the number of up-to-date nodes to $x+1$, is $\frac{2x(n-x)}{n(n-1)}$. By a Coupon Collector argument, the expected number of rounds until every node is up-to-date is then $\sum_{x=1}^{n-1} \frac{n(n-1)}{2x(n-x)} \leq \frac{(n-1)}{2} \sum_{x=1}^{n-1} \left(\frac{1}{x} + \frac{1}{n-x} \right) \leq 2n \log n$.

By Markov's inequality, the probability that not all nodes are up-to-date after $\xi n \log n$ communication rounds is at most $2/\xi$. Let Y denote the number of up-to-date nodes at some given time after T. It follows that, after $\xi n \log n$ rounds, $\mathbb{E}[Y] \geq \frac{n(\xi-2)}{\xi}$. Let q be the probability of having at least $\frac{n}{3}+1$ nodes after $\xi n \log n$ communication rounds. Then we have $qn + (1-q)(\frac{n}{3}+1) \geq \mathbb{E}[Y] \geq \frac{n(\xi-2)}{\xi}$, which implies that $q \geq \frac{1}{4}$ for $n > 2$ and $\xi \geq 8$.

Hence, with probability at least $1/4$, at least $n/3+1$ are nodes are up to date after $\xi n \log n$ rounds. By symmetry, the $n/3$ up-to-date nodes except the original node are uniformly random among the other $n-1$ nodes. Therefore, any given node, in particular the node that had value u_2 at time T, has probability at least $1/4 \cdot 1/3 = 1/12$ to be up-to-date after $\xi n \log n$ rounds from T.

Let v_2 be the node that had value u_2 at time T. We now wish to bound the probability that v_2 is still a contender once it becomes up-to-date. The only way in which this can happen is if it increments its value at least $4\xi \log n$ times (so that its value can reach u_1) during the first $\xi n \log n$ rounds after T. We will show that the probability of this event is at most $1/24$.

In each round, the probability to select node v_2 is $2/n$ (selecting $n-1$ out of $n(n-1)/2$ possible pairs). Let us describe the number of times it is selected in $\xi n \log n$ rounds by considering a random variable $Z \sim \text{Bin}(\xi n \log n, 2/n)$. By a Chernoff Bound, the probability of being selected at least $4\xi \log n$ times in these rounds is at most $\Pr\left[Z \geq 4\xi \log n\right] \leq \exp\left(-2\xi \log n/3\right) \leq 1/n^{2\xi/3} \leq 1/24$.

The next Lemma shows that, after $\Theta(n \log^2 n)$ rounds, the difference between the values of any two given contenders is high, with reasonable probability.

Lemma 3. *Fix an arbitrary time T, and a constant $\xi \geq 1$. Consider any two contender nodes at time T, and time T_1 which is $32\xi^2 n \log^2 n$ rounds after T.*

If no absolute value of any node reaches m at any time until T_1, then, with probability at least $\frac{1}{24} - \frac{1}{n^{8\xi}}$, at time T_1, either at least one of the two nodes has become a minion, or the absolute value of the difference of the two nodes' values is at least $4\xi \log n$.

Proof. We will assume that no absolute value reaches m at any point until time T_1 and that the two nodes are still contenders at T_1. We should now prove that the difference of values is large enough.

Consider $32\xi^2 n \log^2 n$ rounds following time T. If a round involves an interaction with exactly one of the two fixed nodes we call it a *spreading* round. A round is spreading with probability $\frac{4(n-2)}{n(n-1)}$, which for $n > 2$ is at least $2/n$. So, we can describe the number of spreading rounds among the $32\xi^2 n \log^2 n$ rounds by a random variable $X \sim \text{Bin}(32\xi^2 n \log^2 n, 2/n)$. Then, by Chernoff Bound, the probability of having at most $32\xi^2 \log^2 n$ spreading rounds is at most

$$\Pr\left[X \leq 32\xi^2 \log^2 n\right] \leq \exp\left(-\frac{64\xi^2 \log^2 n}{2^2 \cdot 2}\right) \leq 2^{-8\xi^2 \log^2 n} < \frac{1}{n^{8\xi}},$$

Let us from now on focus on the high probability event that there are at least $32\xi^2 \log^2 n$ spreading rounds between times T and T_1, and prove that the desired difference will be large enough with probability $\frac{1}{24}$. This implies the claim by Union Bound with the above event (note that for $n > 2$, $\frac{1}{n^{8\xi}} < \frac{1}{24}$ holds).

We assumed that both nodes remain contenders during the whole time, hence in each spreading round, a value of exactly one of them, with probability 1/2 each, increases by one. Without loss of generality assume that at time T, the value of the first node was larger than or equal to the value of the second node. Let us now focus on the sum S of k independent uniformly distributed ± 1 Bernoulli trials x_i where $1 \leq i \leq k$, where each trial corresponds to a spreading round and outcome $+1$ means that the value of the first node increased, while -1 means that the value of the second node increased. In this terminology, we are done if we show that $\Pr[S \geq 4\xi \log n] \geq \frac{1}{24}$ for $k \geq 32\xi^2 \log^2 n$ trials.

However, we have that:

$$\Pr[S \geq 4\xi \log n] \geq \Pr[|S| \geq 4\xi \log n]/2 = \Pr[|S^2| \geq 16\xi^2 \log^2 n]/2 \tag{1}$$

$$\geq \Pr[|S^2| \geq k/2]/2 = \Pr[|S^2| \geq \mathbb{E}[S^2]/2]/2 \tag{2}$$

$$\geq \frac{1}{2^2 \cdot 2} \frac{\mathbb{E}[S^2]^2}{\mathbb{E}[S^4]} \geq 1/24 \tag{3}$$

where (1) follows from the symmetry of the sum with regards to the sign. For (2) we have used that $k \geq 32\xi^2 \log^2 n$ and $\mathbb{E}[S^2] = k$. Finally, to get (3) we use the Paley-Zygmund inequality and the fact that $\mathbb{E}[S^4] = 3k(k-1) + k \leq 3k^2$. Evaluating $\mathbb{E}[S^2]$ and $\mathbb{E}[S^4]$ is simple by using the definition of S and the linearity of expectation. The expectation of each term then is either 0 or 1 and it suffices to count the number of terms with expectation 1, which are exactly the terms where each multiplier is raised to an even power.

Now we are ready to prove the bound on convergence speed.

Theorem 1. *There exists a constant α, such that for any constant $\beta \geq 3$ following holds: If we set $m = \alpha\beta \log^3 n = \Theta(\log^3 n)$, the algorithm elects a leader (i.e. reaches a configuration with a single contender) in at most $O(n \log^3 n)$ rounds (i.e. parallel time $O(\log^3 n)$) with probability at least $1 - 1/n^{\beta}$.*

Proof. Let us fix constants $0 < p < 1$ and $\xi \geq 8$ large enough such that

$$1/24 \cdot \left(1/24 - 1/n^{8\xi}\right) \geq p. \tag{4}$$

Let β be any constant ≥ 3 and take $\alpha = 16(33\xi^2)/p$. We set $m = \alpha\beta \log^3 n$ and consider the first $\alpha\beta n \log^3 n/4$ rounds of the algorithm's execution. For a fixed node, the probability that it interacts in each round is $2/n$. Let us describe the number of times a given node interacts within the first $\alpha\beta n \log^3 n/4$ rounds by a random variable $B \sim \text{Bin}(\alpha\beta n \log^3 n/4, 2/n)$. By the Chernoff Bound, the probability of being selected more than m times during these rounds is at most:

$$\Pr[B \geq m] \leq \exp\left(-\alpha\beta \log^3 n/6\right) \leq 2^{-\frac{\alpha\beta}{6} \log^3 n} \leq 1/n^{\alpha\beta/6}.$$

Taking the Union Bound over all n nodes, with probability at least $1-(n/n^{\alpha\beta/6})$, all nodes interact strictly less than m times during the first $\alpha\beta n \log^3 n/4$ rounds.

Next, let us focus on the high probability event above, meaning that all absolute values are strictly less than m during the first $\frac{\alpha\beta n \log^3 n}{4} = \frac{4\beta}{p}(33\xi^2) n \log^3 n$ rounds. For a fixed pair of nodes, this allows us to apply Lemma 3 followed by Lemma 2 (with parameter ξ) $\frac{4\beta(33\xi^2) n \log^3 n}{p(32\xi^2 n \log^2 n + \xi n \log n)} \geq \frac{4\beta \log n}{p}$ times. Each time, by Lemma 3, after $32\xi^2 n \log^2 n$ rounds with probability at least $1/24 - 1/n^{8\xi}$ the nodes get values at least $4\xi \log n$ apart. Then, after the next $\xi n \log n$ rounds, by Lemma 2, one of the nodes becomes a minion with probability at least $1/24$. Since Lemma 2 is independent from the interactions that precede it, by (4), each of the $\frac{4\beta \log n}{p}$ times if both nodes are contenders, we get probability at least p that one of the nodes becomes a minion. Consider a random variable $W \sim \text{Bin}\left(4\beta \log n/p, p\right)$. By Chernoff bound the probability that both nodes in a given pair are still contenders after $\frac{\alpha\beta n \log^3 n}{4}$ rounds is at most:

$$\Pr[W \leq 0] = \Pr[W \leq 4\beta \log n\,(1-1)] \leq \exp\left(-\frac{4\beta \log n}{2}\right) \leq 2^{-2\beta \log n} < \frac{1}{n^{2\beta}},$$

By a Union Bound over all $< n^2$ pairs, for every pair of nodes, one of them is a minion after $\frac{\alpha\beta n \log^3 n}{4}$ communication rounds with probability at least $1 - \frac{n^2}{n^{2\beta}}$. Hence, with this probability, there will be only one contender.

Finally, combining with the conditioned event that none of the nodes interact m or more times gives that after the first $\frac{\alpha\beta n \log^3 n}{4} = O(n \log^3 n)$ rounds there must be a single contender with probability at least $1 - \frac{n^2}{n^{2\beta}} - \frac{n}{n^{\alpha\beta/6}} \geq 1 - \frac{1}{n^{\beta}}$ for $\beta \geq 3$. A single contender means that leader is elected by Lemma 1.

Finally, we can prove the expected convergence bound.

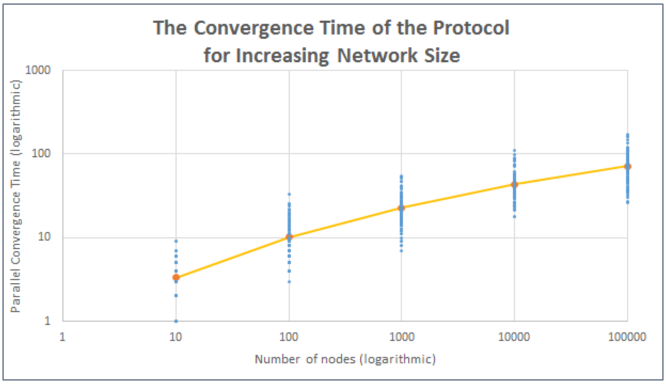

Fig. 2. The performance of the *LM* protocol. Both axes are logarithmic. The dots represent the results of individual experiments (100 for each network size), while the solid line represents the mean value for each network size.

Theorem 2. *There is a setting of parameter m of the algorithm such that $m = \Theta(\log^3 n)$, and the algorithm elects the leader in expected $O(n \log^3 n)$ rounds of communication (i.e. parallel time $O(\log^3 n)$).*

Proof. Let us prove that from any configuration, the algorithm elects a leader in expected $O(n \log^3 n)$ rounds. By Lemma 1, there is always a contender in the system and if there is only a single contender, then a leader is already elected. Now in a configuration with at least two contenders consider any two of them. If their values differ, then with probability at least $1/n^2$ these two contenders will interact in the next round and the one with the lower value will become a minion (after which it may never be a contender again). If the values are the same, then with probability at least $1/n$, one of these nodes will interact with one of the other nodes in the next round, leading to a configuration where the values of our two nodes differ[4], from where in the next round, independently, with probability at least $1/n^2$ these nodes meet and one of them again becomes a minion. Hence, unless a leader is already elected, in any case, in every two rounds, with probability at least $1/n^3$ the number of contenders decreases by 1.

Thus the expected number of rounds until the number of contenders decreases by 1 is at most $2n^3$. In any configuration there can be at most n contenders, thus the expected number of rounds until reaching a configuration with only a single contender is at most $2(n-1)n^3 \leq 2n^4$ from any configuration.

Now using Theorem 1 with $\beta = 4$ we get that with probability at least $1-1/n^4$ the algorithm converges after $O(n \log^3 n)$ rounds. Otherwise, with probability at most $1/n^4$ it ends up in some configuration from where it takes at most $2n^4$ expected rounds to elect a leader. The total expected number of rounds is therefore also $O(n \log^3 n) + O(1) = O(n \log^3 n)$, i.e. parallel time $O(\log^3 n)$.

[4] This is always true, even when the new value is not larger, for instance when the values were equal to $m + 1$, the new value of one of the nodes will be $m \neq m + 1$.

5 Experiments and Discussion

Empirical Data. We have also measured the convergence time of our protocol for different network sizes. (Figure 2 presents the results in the form of a log-log plot.) The protocol converges to a single leader quite fast, e.g., in less than 100 units of parallel time for a network of size 10^5. This suggests that the constants hidden in the asymptotic analysis are small. The shape of the curve confirms the poly-logarithmic behavior of the protocol.

Discussion. We have given the first population protocol to solve leader election in poly-logarithmic time, using a poly-logarithmic number of states per node. Together with [AAE06], the existence of our protocol implies that population protocols can compute any semi-linear predicate on their input in time $O(n \log^5 n)$, with high probability, as long as memory per node is poly-logarithmic.

Our result opens several avenues for future research. The first concerns *lower bounds*. We conjecture that the lower bound for leader election in population protocols is $\Omega(\log n)$, irrespective of the number of states. Further, empirical data suggests that the analysis of our algorithm can be tightened, cutting logarithmic factors. It would also be interesting to prove a tight a trade-off between the amount of memory available per node and the running time of the protocol.

Acknowledgments. Support is gratefully acknowledged from the National Science Foundation under grants CCF-1217921, CCF-1301926, and IIS-1447786, the Department of Energy under grant ER26116/DE-SC0008923, and the Oracle and Intel corporations."

References

[AAD+06] Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Distributed computing **18**(4), 235–253 (2006)

[AAE06] Angluin, D., Aspnes, J., Eisenstat, D.: Stably computable predicates are semilinear. In: Proceedings of PODC 2006, pp. 292–299 (2006)

[AAE08a] Angluin, D., Aspnes, J., Eisenstat, D.: Fast computation by population protocols with a leader. Distributed Computing **21**(3), 183–199 (2008)

[AAE08b] Angluin, D., Aspnes, J., Eisenstat, D.: A simple population protocol for fast robust approximate majority. Distributed Computing **21**(2), 87–102 (2008)

[AAER07] Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: The computational power of population protocols. Distributed Computing **20**(4), 279–304 (2007)

[AAFJ06] Angluin, D., Aspnes, J., Fischer, M.J., Jiang, H.: Self-stabilizing population protocols. In: Anderson, J.H., Prencipe, G., Wattenhofer, R. (eds.) OPODIS 2005. LNCS, vol. 3974, pp. 103–117. Springer, Heidelberg (2006)

[CDS14] Chen, H.-L., Doty, D., Soloveichik, D.: Deterministic function computation with chemical reaction networks. Natural computing **13**(4), 517–534 (2014)

[DH13] Doty, D., Hajiaghayi, M.: Leaderless deterministic chemical reaction networks. In: Soloveichik, D., Yurke, B. (eds.) DNA 2013. LNCS, vol. 8141, pp. 46–60. Springer, Heidelberg (2013)

[DS15] Doty, D., Soloveichik, D.: Stable leader election in population protocols requires linear time (2015). ArXiv preprint. http://arxiv.org/abs/1502.04246

[DV12] Draief, M., Vojnovic, M.: Convergence speed of binary interval consensus. SIAM Journal on Control and Optimization **50**(3), 1087–1109 (2012)

[FJ06] Fischer, M., Jiang, H.: Self-stabilizing leader election in networks of finite-state anonymous agents. In: Shvartsman, M.M.A.A. (ed.) OPODIS 2006. LNCS, vol. 4305, pp. 395–409. Springer, Heidelberg (2006)

[PVV09] Perron, E., Vasudevan, D., Vojnovic, M.: Using three states for binary consensus on complete graphs. In: IEEE INFOCOM 2009, pp. 2527–2535. IEEE (2009)

[SNY+10] Sudo, Y., Nakamura, J., Yamauchi, Y., Ooshita, F., Kakugawa, H., Masuzawa, T.: Loosely-stabilizing leader election in population protocol model. In: Kutten, S., Žerovnik, J. (eds.) SIROCCO 2009. LNCS, vol. 5869, pp. 295–308. Springer, Heidelberg (2010)